A SYSTEM FOR PROCESSING DIGITAL PICTURES

W.A. Davis, R.N. McPherson, M.W. Smith & N. Tsang

University of Alberta

ABSTRACT

This paper describes a Digital Picture Processing System (DIPPS) whose purpose is to quantize, store, manipulate and display images for picture processing operations. The hardware includes a TV camera, a quantizer, a PDP-9 computer and a storage scope. To facilitate the use of DIPPS a Picture Operating System is being developed for the computer. This will allow the hardware to be easily used by even the novice experimenter. The system has been used to compare various edge detection algorithms with some success.

ABRÉGÉ

Dans cette communication on décrit un système pour le traitement des images en forme numérique (DIPPS). Le système quantifie, stocke, manie et représente des images pour le traitement par ordinateur. L'équipement consiste en une caméra de télévision, un convertisseur numérique, un ordinateur PDP9, et un oscilloscope à stockage. Un système d'exploitation pour images est en cours de réalisation afin de faciliter l'utilisation de ce système (DIPPS) de traitement d'images. Ainsi, l'experimenteur débutant pourra utiliser facilement le système. On l'a déjà employé avec quelque succès pour faire un comparaison entre divers algorithmes pour la détection de bords.

A SYSTEM FOR PROCESSING DIGITAL PICTURES

W.A. Davis, R.N. McPherson, M.W. Smith & N. Tsang
University of Alberta

INTRODUCTION

Basically DIPPS should provide a facility for experimenting with picture processing algorithms and techniques. In other words, it should be able to put pictures into a computer, do whatever processing is necessary and at any stage in that process be able to display them. This meant that DIPPS should have the following features:

1) be able to quantize, store, manipulate, and display pictures,
2) be quick and easy to build,
3) be inexpensive,
4) provide reasonable accuracy,
5) be easy and convenient to use,
6) be flexible and easy to change should the need arise.

HARDWARE

After some consideration it was decided to digitize the output of a standard TV camera for the quantizer requirement, to use an existing PDP-9 computer to store and manipulate the quantized images, and to use a binary storage scope for display purposes.

The camera used is a standard SONY DXC2000A TV camera, the output of which is sampled and converted to digital by the video quantizer. Because of the slow speed of the A to D converters, it is only possible to sample one point per line. Therefore, the time taken to quantize a complete picture is on the order of k/30 seconds, where k is the number of points to be quantized in each line. While this is relatively slow, the speed was not considered to be important since with faster converters and a few

changes in the quantizer control circuitry the time could be decreased. It is possible to manually control the size of the picture to be sampled by setting window switches. The vertical resolution is determined by the TV camera and is 525 lines, while the horizontal resolution is 500 points per line which may be changed by selecting every ith point where i = 1,2,...,7 which is also determined by a consol switch.

The central control unit of DIPPS is a PDP-9 computer, which has 8K words of 18 bits each, and a cycle time of 1 microsec. For secondary storage a 256K-words fixed head disk is used, having a cycle time of 16 microseconds. Tertiary storage is accomplished with a cassette tape recorder having a read/write speed of approximately 5 msec/8-bit character.

Picture input requires the camera, the video quantizer, and the computer. The process is initiated, by pressing a switch located on the quantizer console. The video signals are sampled and digitized by the quantizer and the resulting data is placed in the output buffer. When the buffer is filled, the computer is notified by setting a flag. The computer then packs the data and stores it in the output buffers. The contents of these buffers are then copied onto the disk. The process continues until the whole picture, whose size is defined by the quantizer window switches, has been digitized. The whole process is asynchronously controlled by the quantizer.

The display unit used is a Tektronix 611 Storage Scope. It has a 16cm x 16cm display area consisting of 1024 x 1024 points each of which are addressable. Although the operation of the scope is geared towards the storage feature, it may be operated in non-store mode. The display interface has three interesting features:

1) With two consol switches the picture to be displayed can be reversed either left to right or top to bottom (or both). In other words, the origin can be placed in any corner.

2) With another consol switch one of the four quadrants of the picture can be selected for display on the whole screen and this quadrant can be further enlarged by yet another switch which selects a subquadrant.

3) The third feature is that whenever the beam is moved to display successive points, the delay time between the illumination of the points is a function of the distance that the beam has to move.

Other than the four switches mentioned in 1) and 2) all other functions are under program control. Bulk picture storage is provided by a cassette tape unit.

Besides the slow quantizer speed the hardware has two basic faults. The first is the presence of lines, both horizontal and

vertical in the display which appear as a superimposed grid. This is due to the use of low quality D/A converters which produce transients at regular intervals. This will be rectified by the addition of high quality converters. The second fault is due to the fact that the horizontal resolution is not the same as the vertical and the fact that a video picture has a 4 to 3 aspect ratio. During display it is assumed that any four adjacent points form a square which they do not. This feature tends to squeeze pictures together horizontally, and will be rectified by changing the frequency of the horizontal oscillator.

SOFTWARE

The software portion of DIPPS is referred to as the Picture Operating System (POPS). The purpose of POPS is to provide the user with a simple interactive means of initiating an operation, which may be comprised of many subroutines, and the facility to move from one operation to another at one level, that of POPS. To effect these requirements it has been necessary to construct a system which allows the composition, under one command word, of sets of routines and to effect the operations associated with these routines under control of POPS. POPS has done this; not as a totally new system to supplant the existing Keyboard Monitor System, but as a system program which runs under the existing System and usurps some of the control of that system. Prior to the inception of POPS, all operations performed under DIPPS had to be specified using the facilities of the existing system. The cumbersome nature of this approach, in addition to a number of other problems, not only led to the development of POPS but identified many of desired features, which have been included in the following set of requirements:

1) In order to maximize memory space available for picture data, not all POPS routines should be simultaneously resident in core. Therefore POPS should be able to link, load and execute logical subsets of these routines.

2) In the system there should be only one copy of any subroutine, which could be linked and loaded as required.

3) It should be possible to specify a complete operation with a single command word.

4) Parameters could be specified as a part of the command string.

5) All systems routines and device handlers should be available to the user's programs in the usual manner. That is, other than identifying return destinations as either POPS or the Monitor, the user would have no changes to make in his programming and would have available to him all system facilities.

6) All routines associated with POPS should be available to the user under the Keyboard Monitor operation alone (i.e. no subroutines would be specifically for POPS by

virtue of their structure).

7) It should be possible to define new commands, delete old ones, etc. with minimal difficulty.

8) It should be easy to move between POPS and the Keyboard Monitor so as to have access to all the existing system programs without an involved regeneration of the keyboard system.

9) Changes to the existing Keyboard Monitor System to allow the implementation of POPS should be minimal.

10) It should be possible for successive operations to communicate various data.

Implementation of POPS, to satisfy these requirements, implicitly required dynamic access to a linking loader (preferably non-resident to preserve core). Since the existing linking loader could only be invoked by the Monitor, and writing a new loader was considered undesirable, a means was sought whereby the system could be designed to have the Monitor invoke the existing linking loader, while allowing POPS to maintain control. The result was the implementation of POPS to operate as follows:

1) For a cold start, a BOOTSTRAP first loads the Keyboard Monitor.

2) The System command 'POPS' causes POPS to be loaded.

3) POPS accepts and interprets command strings, stores info on disk, then reloads the Monitor, but with changes made dynamically so as to cause the Keyboard Monitor to communicate with POPS.

4) The Monitor invokes the Linking Loader but again with communication directed to POPS as opposed to the Keyboard.

5) POPS reloads command information, then communicates the string of programs to be loaded and establishes a new loader address below the resident portion of POPS as opposed to the standard address below the BOOTSTRAP.

6) The Linking Loader now performs its function in exactly the same manner as if the string had been specified from the keyboard, except for the new loading address.

7) Upon completion, the user program can return to the Monitor or to the resident portion of POPS which then reloads itself.

This implementation of POPS required no changes other than those dynamically effected by POPS to the existing Monitor System, and to include "POPS" as a system command. Further, by using the system facilities in this way, requirements 2,5, and 6 were easily satisfied. The fulfillment of the other requirements of POPS were all realized through the command feature of POPS

At the time of writing, POPS has been developed and implemented as described in the text, with capabilities to maintain its command file, take and store pictures, display pictures, perform some operations on pictures (e.g. edge operators), save and restore pictures on cassette, window and a few other operations. Due to the ease of implementing programs under POPS as a single command it is anticipated that not only will POPS have its picture processing repertoire developed, but also other routines associated with the 611 Scope. One of these already implemented is a command called *DRAW which allows the production of line drawings from combinations of lines, circles, etc.

A number of possible improvements and extensions to the basic POPS systems as well as new commands have been identified and it is expected that most of these changes will be implemented in the near future. Some of these are:

1) Allow invoking other system programs directly from POPS (e.g. assembler, editor, etc.)
2) Allow specification of loader type strings instead of just commands so as to facilitate development of programs.
3) Allow the user to specify an additional file of his own commands to be searched in addition to the POPS master file.
4) Allow CUP to maintain these files.
5) Develop other methods of displaying grey level pictures.
6) Develop procedures for restructuring pictures from their current form as contiguous regions on disk to file type structures.
7) Allow various other of the Keyboard Commands to be specified as input to POPS rather than having to return to the Keyboard Monitor. (e.g. assignment of device handlers).

APPLICATIONS

Currently, DIPPS is simplifying the evaluation and functional testing of several edge detecting algorithms. Photographic results of the scope images and their edge transforms are shown in Fig. 1. It is now apparent that almost any picture manipulation function can be implemented as a permanent or temporary DIPPS subroutine, and can be put to use quickly and accurately. Aside from the six edge detecting routines, there are proven programs that measure certain statistics of disc-stored images, and experimental programs whose intent is to evaluate the merit of edge transformations.

Progressing from this basis, it will be a simple coding

whereby the user could, after invoking the system, specify a command of the form

*Command PAR=parameters

which would then be interpreted and a corresponding set of routines loaded and executed.

Most of the remaining software which has or is being developed for POPS takes the form of subroutines, to be combined as necessary, for the various commands of POPS. The Command Utility Program (CUP) in some sense overlaps the two areas, and a discussion of its function is a logical prelude to the discussion of other features of POPS. CUP is used to maintain a file of commands searched by POPS during command interpretation. This file is a sequential set of records consisting of command words with their corresponding loader strings and has the following performed on it by CUP:

1) ADD a command
2) DELETE a command
3) CHANGE a command
4) LIST one or all commands
5) COPY the file to a hard copy backup
6) RESTORE the file from a hard copy backup

CUP itself is invoked through POPS by means of the command *CUP. Some other POPS programs which have been incorporated as commands in the system are:

*SNAP-- essentially takes a picture from the quantizer and stores it on disk according to a number of specified and/or defaulted parameters.

*SHOW-- displays a stored picture.

*WINDOW- allows windowing an existing display and thereafter the window can be displayed in expanded form (this utilizes the "communication area" to be discussed at the end of this section).

*MON-- returns to the Monitor.

The implementation of these commands is effected by defining through CUP each command as a combination of programs such as parameter interpreter, packer, unpacker, grey level generator, etc. These routines must all then be linked and loaded but the user need only specify a single command word.

To satisfy the final requirement that successive operations be able to pass on information, a "communications area" was set up in the resident portion of POPS. The present use being made of this area is to retain information on the "last picture" (snapped or shown) and "window" data from the windowing routine.

task to implement the Fourier and Hadamard transforms in their fast version, and begin picture transformation experiments. Although much previous work on picture compression has been done with these transforms (at least as far as the Fourier is concerned), it will be a goal to use these algorithms in a pre-processing role to prepare images for pattern recognition purposes. In particular, the Hadamard transform would appear to be quite useful in this regard, since the transform matrix is clearly related to the number and strength of edges in the original image. Thus, work directed towards determination of edges in images should find the Hadamard quite useful. It also seems plausible to develop PR routines that are able to work with the transform matrix rather than with the original image.

Another interesting application is "histogram equalization", where linear operations are performed on each pixel of a picture matrix, as a result of a histogramic analysis of a stored picture. The stored picture is then transformed to produce an image that 'appears' better than the original. This technique will be investigated and evaluated within DIPPS.

With all this, one must keep in mind that DIPPS was designed primarily to move pictures. Specific picture processing tasks are, however, easily attached, and removed, from the system.

Acknowledgement

Financial support provided by National Research Council grants A7634, E3227 and C0304 is acknowledged. The assistance of Wayne Sherrard and Martin de Leeuw of Technical Services in construction of the interfaces is appreciated.

Bibliography

1) Digital Equipment, PDP-9 Advanced Software System Monitors, DEC-9A-MADO-D.

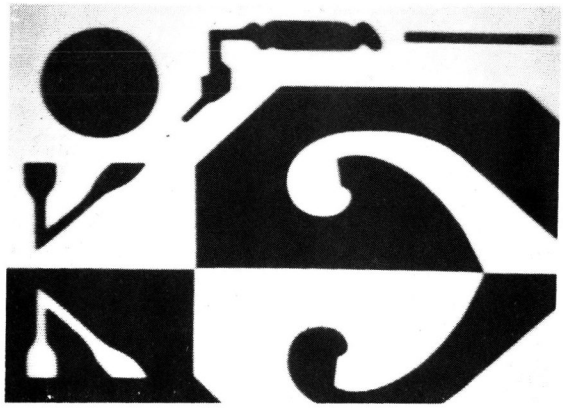
2) DEC, PDP-9 User Handbook.

3) DEC, PDP-9 Keyboard Monitor Guide, DEC-9A-NGBA-D.

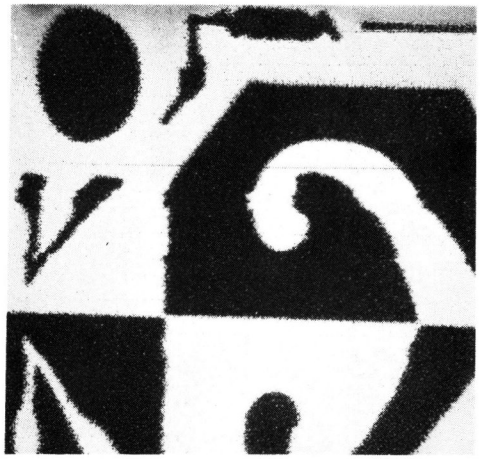
4) DEC, RF09/RS09 Decdisk System V1, DEC-09-H9ZA-D.

5) DEC, PDP-9 Utility Programs, Advanced Software Systems, DEC-9A-GUAB-D.

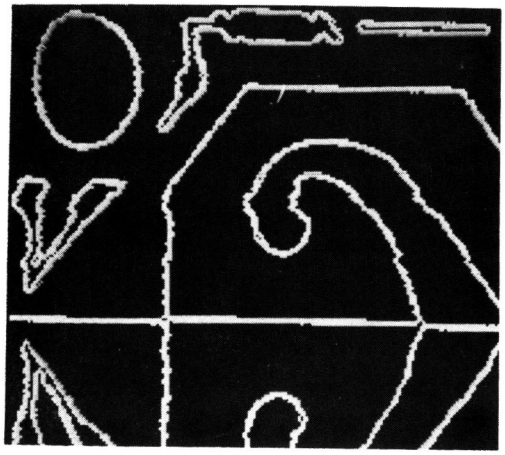
6) Video Quantizer, Operation and Maintenance Manual, Technical Services, University of Alberta, May 1972.



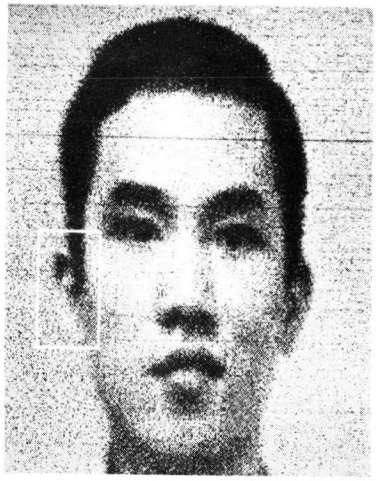
a) Output on TV monitor.



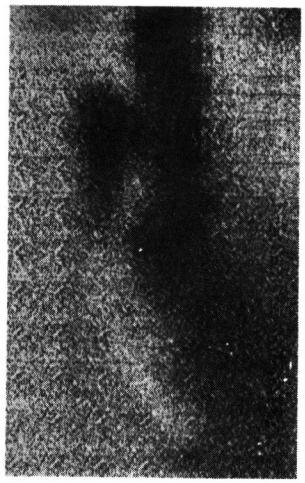
b) Digitized, output on 611.



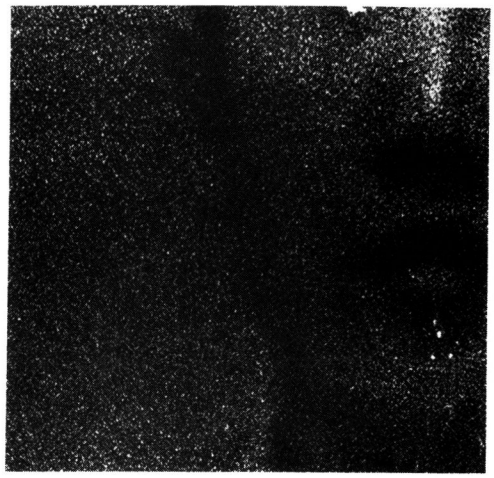
c) Edge detected, output on 611.



d) Face with window outline.



e) Window area software englarged.



f) One quadrant hardware enlarged.

Figure 1: Picture Examples