INTERNATIONAL ATOMIC ENERGY AGENCY
UNITED NATIONS EDUCATIONAL, SCIENTIFIC AND CULTURAL ORGANIZATION
INTERNATIONAL CENTRE FOR THEORETICAL PHYSICS
I.C.T.P., P.O. BOX 586, 34100 TRIESTE, ITALY, CABLE: **CENTRATOM TRIESTE**

H4.SMR/773-19

College on Medical Physics:
Radiation Protection and Imaging Techniques

5 - 23 September 1994

Fundamentals on Image Processing, Hardware and Practices Using a Programmable Image Processing System

P.E. Cruvinel

Empresa Brasileira de
Pesquisa Agropecuaria
Sao Carlos, Brazil

FUNDAMENTALS ON IMAGE PROCESSING, HARDWARE AND PRACTICES USING A PIP (PROGRAMMABLE IMAGE PROCESSING) SYSTEM

Paulo E. Cruvinel
EMBRAPA/CNPDIA
P.O.Box 741
13560-970-São Carlos-Brazil
cruvinel@npdia.embrapa.ansp.br

INTRODUCTION

Image processing is concerned with the manipulation and analysis of images by computer. Its majors subareas include:

a) Digitization and compression: Converting images to discrete or digital form, and approximation of images so as to save storage space or channel capacity;

b) Enhancement, restoration, and reconstruction: Improving degraded (low-contrast, blurred, noisy) images, and reconstructing pictures from sets of projections;

c) Matching, description, and recognition: Comparing and registering images to one another, segmenting pictures into parts, measuring properties of and relationships among the parts, and comparing the resulting descriptions to models that define classes of images.

When a scene is viewed from a given point, the light received by the observer varies in brightness and color as a function of direction. Thus the information received from the scene can be expressed as a function of two variables, i.e., of two angular coordinates that determine a direction. The scene brightness and color themselves are resultants of the illumination, reflectivity, and geometry of the scene.

In an optical image of the scene, if produced by a lens, light rays from each scene point in the field of view are collected by the lens and brought together at the corresponding point of the image. Scene points at different distances from the lens give rise to image points at different distances. Thus

the imaging process converts the scene information into an illumination pattern in the image plane, which is a function of two variables, but they are coordinates in the plane. It is possible now record or measure the pattern of light from the scene by placing some type of sensor in the image plane. Any given sensor has a characteristics spectral sensitivity, i.e., its response varies either with the color of the light or the signal type, and its total response to the signal at a given point can be expressed by an integral of the form:

$$\int S(\lambda)I(\lambda)d\lambda \qquad (1)$$

where $I(\lambda)$ is the light intensity and $S(\lambda)$ is sensitivity as functions of wavelength. This means that if just one sensor is used, it is possible to measure only light intensity. To measure color one must uses several sensors having different spectral responses, or split the light into a set of spectral bands, using color filters, and measure the light intensity in each band. In other words when using only one sensor, the information is representing by a scalar-valued function of position in the image, representing scene brightness. To represent color, it is necessary to use a k-tuple, usually a triple, of such functions, or equivalently, a vector-valued function, representing the brightness in a set of spectral bands.

In general the light received from a scene by an optical system produces a two-dimensional image. this image can be directly converted into electrical signal form by a sensor, or it can be recorded photographically as a picture and subsequently converted. Mathematically an image is defined by a function f(x,y) of two variables, i.e., coordinates in the plane, corresponding to spatial directions. The function values are brightness, or k-tuples of brightness values in several spectral bands. In the black-and-white case, the values will be called gray levels. These values are real, non-negative, and bounded. They are also assumed to be zero outside a finite region, since an optical system has a bounded field of view, so that the image is of finite size i.e., they have inverse Fourier transforms.

When an image is digitized, a sampling process is used to extract from the image a discrete set of real numbers. These samples are usually the gray levels at a regularly spaced array of points, or in general the average gray levels taken over a small neighborhoods of such points. The array is almost

always taken to be Cartesian or rectangular. Normally the set of points are in the form (md,nd), where m and n are integers and d is some unit distance. The image samples are usually quantized to a set of discrete gray level values, which are often taken to be equally spaced. The gray scale is divided into equal intervals, i.e., from I_0 to I_K and the gray level f(x,y) of each sample is changed into the level of the mid-point of the interval I_i in which f(x,y) falls. The resulting quantized gray levels an be represented by their interval numbers 0,..., K, i.e., they can be regarded as integers.

The result of sampling and quantizing is a digital image. Then it is possible to assume a digital image as a rectangular array of integer values. An element of a digital image is called a picture element, which id often abbreviated to pixel or pel. If an image has just two values, in terms of gray level, i.e., black and white, it is called a binary image.

The Fourier transform is a powerful tool for signal and image analysis, which can be extended to either continuous or discrete applications from one to two dimensions. The discrete Fourier transform in two dimensions is the form typically used in image processing i.e., the classical definitions establishes the two-dimensional discrete Fourier transform of f(k,l) as:

$$F(h,i) = \frac{1}{n}\sum_{k=0}^{n-1}\sum_{l=0}^{n-1} f(k,l)e^{-j2\pi(kh+li)/n} \qquad 0 \le h,i \le n-1 \qquad (2)$$

and the inverse transform as:

$$f(k,l) = \frac{1}{n}\sum_{h=0}^{n-1}\sum_{i=0}^{n-1} F(h,i)e^{j2\pi(kh+li)/n} \qquad 0 \le k,l \le n-1 \qquad (3)$$

where n is the number of pixels on the image. There are limitations on the function f(k,l) for its transform to exist. If it is a continuous function it must be a piece-wise with left and right-hand derivatives at every point. A completely study of Fourier transform, its properties, and particularities may

be found in Young and Sun Fu (1986). Many imaging and optical systems can be analyzed by using Fourier methods. In addition, due the computational efficiency, there is a natural choice for its use in applications with need of both image filtering and pattern matching. In this case, several applications arise in which it is desired to know where a pattern image best fits within another. Moreover, because of the convolution theorem, it is possible to make the correlation in the spatial domain or in the Fourier domain. In the digital image processing, for the digital case, the convolution and the correlation may be respectively given by:

$$g(m,n) = \sum_i \sum_j f(i,j)h(m-i,n-j) \tag{4}$$

and

$$g(m,n) = \sum_i \sum_j f(i,j)h(i-m,j-n) \tag{5}$$

where g(m,n) is the output image. It is possible to observe that if h is symmetric both operations are identical. In terms of convolution the interest is in the complete output image. On the other hand, when speaking of correlation, the interest is usually in the values of m and n that lead to the maximum of the summation.

Furthermore, when computing convolutions all the values as computed directly by the summation presented in equation (4) are used, but for correlation both f and h are typically normalized so that the results are in the range (-1, +1).

IMAGE PROCESSING SYSTEM

A typical image processing system consists of an image acquisition device, an image memory, a computer that can access this memory and a monitor that can display the contents of memory. Figure 1 shows a minimal image processing system.

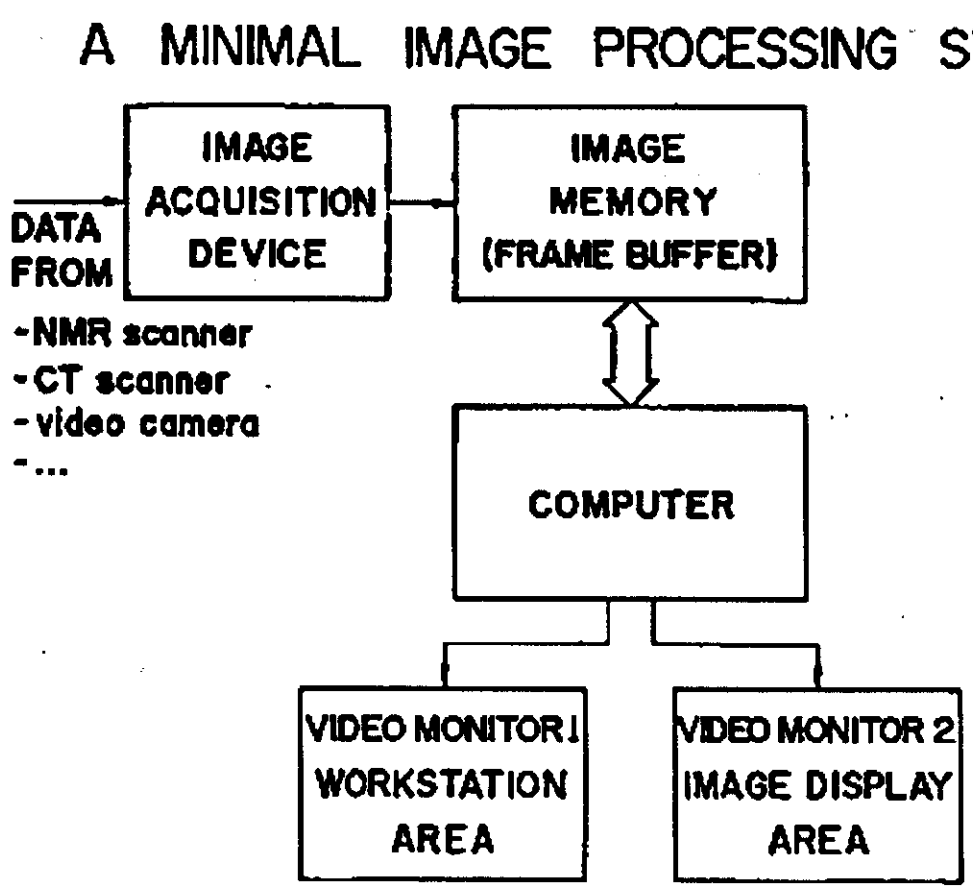

Figure 1 - A minimal image processing system.

This basic system can acquire, processes and display monochrome images (if the monitor is RGB it is possible display image memory values in color by adding output lookup tables).

The image acquisition device puts an image into the image memory. This operation involves digitizing scanning a continuous image and breaking it into an array of digital intensity values (those values called pixels). In general the signal from the video camera is converter by an A/D converter into a pixel array in the image memory.

The acquisition device can write to the image memory, which can be read and written to by the computer's CPU, and read by the display device.

If the image memory stores an entire video image, it is called a frame buffer or frame store. For acceptable intensity and detail resolution, an average monochrome image must be represented by an array of at least 256

by 256 pixels and each pixel must have at least 6 bits of value. Gray level intensities in the pixel is a function of the N-bit byte.

The display device converts the processed pixels back into a spatially organized image intensities. The display device is usually coupled to a D/A converter that drives either a monochrome or RGB TV monitor.

A lookup table (LUT) hangs a pixel's value based on the values in a table. This hardware consists of a memory that has a storage location for each possible pixel value. An input pixel value, in this case, is used as an address into this memory, and the output is the value at that address. The LUT computes an arbitrary function of one or more variables, with a domain and range limited to the possible pixel values.

An image processing system have three LUTs that map the image memory to the display device. These LUTs output values to the red, green, and blue channels of a color monitor, based on some input pixel values. This lets the display with different gray levels from the monochrome image in various arbitrary colors or pseudocolors.

A typical electronic board for image processing is the PIP 640-B, from Matrox Corporation, which is a plug-in video frame grabber-digitizer board for the IBM PC-compatible architecture. It has a resolution of 640X512 pixels in normal (no zoom) mode with eight bits per pixel and a power consumption of approximately 17W. The PIP 640-B is capable of operating in continuous or single frame grab mode and also has a built-in video keying capabilities. Frames which have been stored in the on board frame buffer can be loaded either into the PC's memory or onto the hard-disk.

Conversely, video data, as well as lookup tables data, can be written to the PIP from the PC-compatible computer.

Pixels can be individually addressed by the PC by using the X and Y address registers or they can be addressed sequentially by using an auto-increment register. It has one input and three output lookup tables, each of which has eight maps to choose. There are three input ports, an IBM pin compatible RGB output, and an internal feedback for diagnostic purposes.

Figure 2 shows the PIP.640-B in block diagram. Input signals are selected, in software, from one of the three input ports and the PIP640-B will lock on to the input signal's sync, as well as use the sync signal to drive the video functions of the board.

Data, after it passes the sync separator, is subject to an adjustable DC offset voltage, and this offset adjustment, with the gain, makes the picture brighter or darker.

- BOTH THE SYSTEM UNIT AND CRT-CONTROLLER HAVE SIMULTANEOUS TRANSPARENT ACCESS TO THE FRAME BUFFER.

- 1024 x 1024 STORAGE AREA AND 640 x 512 DISPLAY AREA.

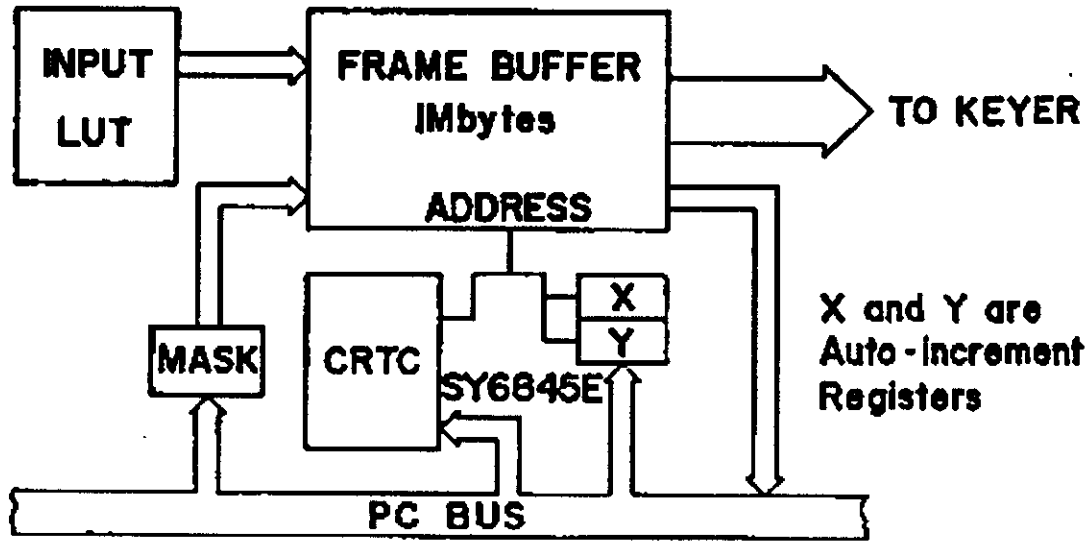

Figure 2 - The PIP.640-B in block diagram.

The video input is digitized in real time in the A/D converter, producing an eight bit number that is sent to the input lookup table.

The PIP.640-B is assembled with 1Mbyte of frame buffer RAM, which is used to store frame grab data. Both the system unit and the CRT-controller have simultaneous transparent access to the frame buffer.

The CRT-controller sends pixel data to the output LUT's. It uses the SY6845E integrated circuit.

This gives the ability to shift the image both vertically and horizontally which is useful in the PIP.640-B with its 1024X1024 storage area and 640X512 display area.

The frame buffer is accessible (read and write) from the system unit using X,Y coordinates. The frame buffer address registers can be set also to automatically increment.

There are two sources of data for output from the PIP 640-B, that are the video keying, or simply the output from either the frame buffer or the input LUT. Figure 3 shows the keying and the output section in block diagram.

- TWO SOURCES OF DATA FOR OUTPUT { FRAME BUFFER / INPUT LUT
- IT HAS ONE INPUT AND THREE OUTPUT LUT.

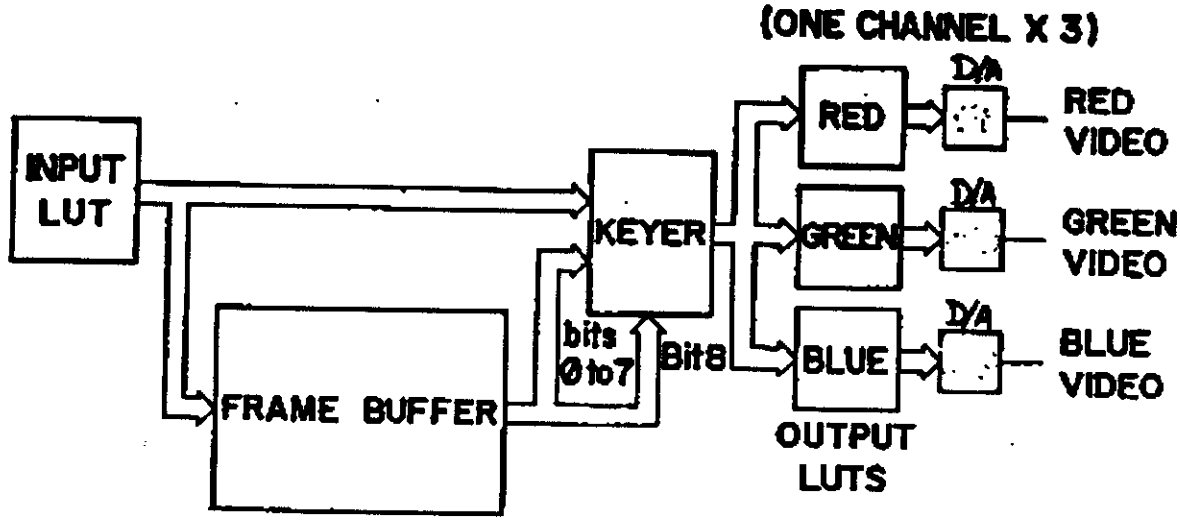

Figure 3 - The keying and the output section in block diagram.

IMAGE PROCESSING OPERATIONS

Image processing includes :

- ENHANCEMENT
- RESTORATION
- MEASUREMENT OF IMAGE ELEMENTS
- CLASSIFICATION OF IMAGE ELEMENTS
- RECOGNITION

The knowledge level used in an algorithm can range from preliminary conception about the physics of image formation to a specific knowledge about sections in a picture-frame, and an adequate diagnostic at medical field

or about pixel structures. In these practices a set of tools will be used for image processing as well as algorithms using the following procedures :

- POINT PROCESSING
- AREA PROCESSING
- GEOMETRIC PROCESSING
- FRAME PROCESSING

In all cases images-based algorithms transform pixel values into other pixel values or locations by using either numerical or logical operations. Pseudocolor will also be used in the experiments.

The image that will be used in these practices is showed in figure 4. It was obtained by using the dual energy CT technique. The object beneath the cross section is a bone mineral phantom with five calibration materials i.e., fat equivalent material, water, 50mg/cc, 100mg/cc, and 200mg/cc K_2HPO_4 respectively.

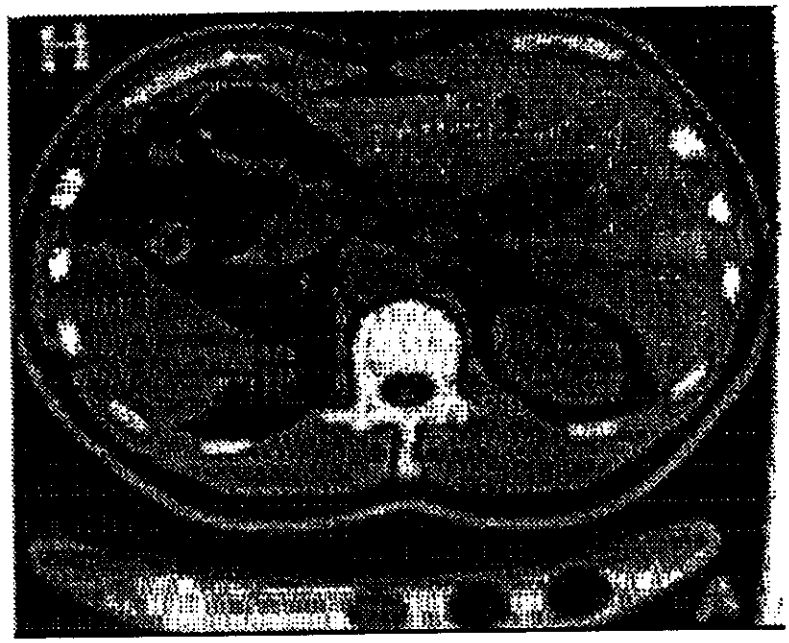

Figure 4 - CT image obtained by using both 140 kVp, 100mA, 10mm thickness, and a scan through the abdominal region. There is also a bone mineral phantom with five calibration materials.

POINT PROCESSING

A point process algorithm scans through the image area and uses the pixel value at each point to compute a new value for the point. This method can be used to enhance or modify pixel values.

If the pixel value and its location are used then it will be useful to correct artifacts or smoothly change pixel values (add or subtraction values) in an image area.

Another possibility with point process algorithm is the use of the histogram since it is an example of image measurement.

An intensity histogram performed on a digitized image results in a graphic interpretation of the number of occurrence of a given pixel intensity value throughout the image. It is possible to create histogram of a gray scale level in a selected current Window. This facility involves two steps:

- COMPUTING THE ARRAY OF VALUES FOR THE HISTOGRAM

- DRAWING THE HISTOGRAM ONTO THE IMAGE BEING DISPLAYED

However, by using point process algorithm, there is only the possibility that it examine a single pixel at a time without changing the pixel's values.

The information provide by the histogram can be used for image enhancement and classification. In other words the histogram represents the relationship between the relative intensity (in general with range from 0 to 255) of the pixel's values and the number of pixels of each relative intensity.

Pseudocoloring of a monochrome image is a point process algorithm. In this case it is necessary to map the values of Xn (the argument, that is, the pixel value is the input argument) onto Yn (the function drives, that is, the three different functions i.e., the red, the green, and the blue) as in the function with the coordinates X as the index to the lookup table, and the Y as the mapped value. In this way the information stored in the frame buffer is manipulated by hanging the values stored in a lookup table (LUT). These output functions drive the red (R), the green (G), and the blue (B) guns of a color monitor.

AREA PROCESSING

The area process algorithm uses neighborhood information to modify pixel values.

These kind of algorithms are typically used for spatial filtering and changing an image'structure. By using area process it is possible to perform the following operations:

- AN IMAGE SHARPENING TRANSFORMATION
- A PIXEL AVERAGING TRANSFORMATION
- A HORIZONTAL EDGE DETECTION TRANSFORMATION
- A VERTICAL EDGE DETECTION TRANSFORMATION
- A LAPLACIAN EDGE DETECTION

In addition, by using area process algorithm, it is also possible to perform non-linear area process, such as SOBEL filter operation.

Convolution is a classic image processing algorithm used for spatial filtering and it is useful to remove noise, smoothing the image, as well as to make edge detections in the image.

The convolution operation replaces a pixel's value with the sun of that pixel's value at its neighbors, each weighted by a factor. The weighting factors are called the convolution kernel.

To convolve an image area with a kernel it is necessary to repeat the operation, i.e., the convolution, at every pixel position in the image.

At each point or pixel, it is necessary to multiply the kernel values by the image values, sun the results and replace the pixel at the enter of the kernel with that value.

If using a 3 by 3 pixel neighborhood and kernel the equation for the convolution operation, i.e., from equation (4), becomes:

$$P(x,y) = \sum_{m,n=0}^{2} K(m,n) * P(x+m, y+n) \qquad (6)$$

where P(x,y) is the image point, and K(m,n) is the kernel. Convolving an area of size X by Y with a kernel of size N by M, it is necessary (X*Y*M*N) multiplications and additions. The kernel is also called filter weights, filter kernel, or filter mask. For reasons of symmetry it is almost always chosen to be of size mXn, where both m and n are odd. Often m=n.

Another important possibility using area process algorithms is that it performs spatial frequency filtering.

Spatial frequency is the number of times per unit distance that a pattern repeats. As with a one dimensional signal, an image can be represented by a series of sine and cossine waves with the spatial frequency and amplitude specified for each component.

Therefore, for the image, transformation must be done both horizontally and vertically because it is bi-dimensional, and thus it has spatial frequencies in both directions. It is possible to detect a certain band of frequencies by using a selected kernel.

In general, quickly changing in image intensities, are represented by high spatial frequencies, while slowly changing intensities are represented by lower spatial frequencies. Thus, if a high spatial frequencies kernel is selected, then edges will be detected. On the other hand if a kernel matches lower spatial frequencies it will blur the image.

The SOBEL filter, as well as, the median filter are two particular cases of area processes algorithm because they are a non-linear operation. The SOBEL filter, for instance, provides a better signal to noise rate, and it is used to detect images elements or features with less computation. The SOBEL filter compares the results of two convolutions to estimate the strength and orientation of edges in the image. For instance, if two kernels are represented by k_1 and k_2, then the edge strength and orientation will be represented by:

$$STRENGTH = (k_1 * k_1 + k_2 * k_2)^{\frac{1}{2}} \qquad (7)$$

$$ORIENTATION = \arctan\left(\frac{k_2}{k_1}\right) \qquad (8)$$

The median filter replaces the pixel at the center of a neighborhood of pixels with the median of the pixel value. The neighborhood values, also the

center pixel, are sorted into ascending order and the median value is used to replace the center pixel. In this way it is possible to remove spot noise.

GEOMETRIC PROCESSING

The geometric process algorithm changes the spatial arrangement of pixels. By using geometric processing it is possible to rotate, stretch, translate the image position, dilate or erode, as well as to warp the image.

Geometric process algorithm an be used to improve accuracy, as well as to compensate rectangular pixels by adjusting the transformation equations.

FRAME PROCESSING

All algorithms that uses more than one image are called frame process. For differential CT-scanner analysis this method is very useful i.e., it can be used to subtract one image from another. Also it can be used to improve image quality and to detect motion. In addition, this method is used for microscopic image analysis, that means, by using N successive images frames it is possible to reduce noise i.e., by making an average.

In order to detect motion the frame processing algorithm is adequate but the hardware must be adequate to subtract the images in real time.

TO OPERATE THE VIDEO DIGITIZER BOARD PIP.640-B

To operate the video digitizer board a MS DOS software library is provided. It is possible to use the software library with Microsoft C (Vers. 4.0 or higher) and Microsoft FORTRAN (Vers. 3.31 or higher), as well as Microsoft PASCAL. In addition to the software library an interpreter program is also provided.

A brief overview of the available commands for the video digitizer board are shown below:

a) Initialization Commands

Name	Description
init	Initialises the board to a known state
inifmt	Initialises the board to a specified video format
inisys	Initialises system variables only
format	Selects a video format to use
dfmt	Defines a video format

b) Hardware Commands

Name	Description
cgrab	Enables or disables continuous frame grabbing
chan	Selects active input channel
clear	Clears the screen to the current draw index
dquad	Selects quadrant on PIP-1024 when in 512 × 512 mode
exit	Resets system unit hardware and exits library
gain	Sets the gain
key	Enable or disable keying
lutm	Selects active map in the LUT
mask	Sets the write protect mask
mode	Tells software about hardware parameters
offset	Sets the offset
outv	Enables or disables video output
pan	Sets the horizontal pan position
panrel	Moves the horizontal pan position by an offset
quadm	Sets the display mode on a PIP-1024
sboard	Selects default board
sbuf	Selects between incoming video and frame buffer
scroll	Sets the vertical pan position
scrorel	Moves the vertical pan position by an offset
setxy	Sets the x and y position for auto-incrementing
snapshot	Takes a snapshot
snapall	Takes a synchronised snapshot using a group of PIPs
statr	Saves the selected board's status
statw	Loads a board's status
sync	Selects the sync source (external or internal)
video	Selects video standard (American or European)
vwait	Waits for vertical blanks
winmode	Sets workspace configuration
zoom	Enables zoom mode

c) Graphic Commands

Name	Description
border	Draws a border
circ	Draws a circle
circf	Draws a filled circle
clear	Clears and sets the frame buffer to current index
elips	Draws an ellipse
elipf	Draws a filled ellipse
grid	Draws a variable sized grid
linerel	Draws a line to a relative point
lineto	Draws a line to a specified point
moverel	Moves the current point to a relative point
moveto	Moves the current point to a specified point
rect	Draws a rectangle
rectf	Draws a filled rectangle
setind	Specifies the drawing index
tchar	Writes a character
text	Writes a string
tfont	Selects a character font

d) I/O Commands

Name	Description
btransfer	Transfers data between a PIP and the system
colr	Reads a column from the frame buffer
colw	Writes a column to the frame buffer
cpws	copies between workspaces
decode	Decodes a file and write it to the window
encode	Encodes a window and write it to a file
frdisk	Copies a DOS file into the frame buffer
frmem	Copies from system memory to frame buffer
igetb	Reads a byte from a buffer
inp	Reads an I/O port
outp	Writes to an I/O port
pixr	Reads a pixel
pixw	Writes a pixel
putbch	Writes a byte to a buffer
rowr	Reads a row from the frame buffer
roww	Writes a row to the frame buffer
setwindow	Defines the current window
setxy	Sets position in frame buffer
todisk	Copies the frame buffer to a DOS file
tomem	Copies from frame buffer to system memory
winfrdisk	Copies to window from disk
winr	Copies the current window to a buffer
wintodisk	Copies from widow to disk
winw	Copies a buffer to the current window

e) LUT Commands

Name	Description
dhisto	Draws a histogram in the frame buffer
histo	Calculates a histogram from the current window
initlut	Initializes LUT to grey scale ramp
lutd	Initializes a LUT
luts	Initializes a LUT to a ramp
modhist	Modifies a histogram through mapping
scaling	Defines a mapping function

f) Imaging Commands

Name	Description
average	Pixel averaging using a 3 × 3 kernel
con3	General convolution around a 3 × 3 kernel
dilate	Dilation routine
erode	Erosion routine
fil3	Convolution around a 3 × 3 kernel
filtype	Select the type of normalization in filtering
horfilter	Horizontal edge detection using a 3 × 3 kernel
ker3	Defines a arbitrary 3 × 3 kernel
lp1 & lp2	Laplacian edge detection using a 3 × 3 kernel
map	Transforms the frame buffer using a software LUT
median	3 x 3 median filter
prewitt	Prewitt edge detection using 3 × 3 kernel
sh1 & sh2	Image sharpening using a 3 × 3 kernel
sobel	Sobel edge detection using 3 × 3 kernel
verfilter	Vertical edge detection using a 3 × 3 kernel

g) Interpreter specific commands

Name	Key	Description
	F1	Gives context sensitive help
	F2	Lists available commands
	F3	Recalls previous command
	F4	Recalls stored line
	F6	Gives detailed command descriptions
	F7	Gives the display work buffer address
	F9	Reads and executes a list file
	F10	Executes DOS comands
	→	Moves cursor right
	←	Moves cursor left
	Home	Moves cursor to beginning of line
	End	Moves cursor to end of line
	Ins	Toggles insert/overstrike mode
	Esc	Cancels current line
	↑	Recalls next stored command
	↓	Recalls previous stored command
	Pg Up	Displays list of stored commands
	CTRL Fn	Displays specified macro

Practice 1:

a) How to initialize the PIP640-B in the system unit to a known state and declare, store, and execute, a macro by using the interpreter program ?

b) How to copy an image file from the disk to the current display Window ?

c) How to set a pre-defined Window, as well as to read and write the value of a pixel in the frame buffer ?

Commands of the interpreter program to be used in this practice, section (a):

macro macro_number
This command is used to declare a macro. The *macro_number* is a parameter used to identify the number of the macro. It is a number from 0 to 19 and its purpose is to permit future identification of a particular sequence commands.

save macro file_name<s> macro-number
This command is used to store a macro in the disk file. The *file_name<s>* is the name of the file in which the macro will be stored. The *macro_number* is the same parameter as described above.

getm(acro) file_name<s>macro_number
This command is used t transfer a macro stored in a file to the interpreter. The *file_name<s>* is the name of the file which contains the macro, and the *macro_number* is used to specify the macro to be transferred.

exe(cute) macro_number
This command is used to execute a sequence of commands define in a macro. The *macro_number* specifies the macro to be executed.

inif(mt)base_addr, mode, speed, class, vid_type, zoom
This command initializes the PIP640-B board in the system unit to a known state. The *base_addr* parameter specifies the offset of the I/O address used. The hexadecimal value $(26c)_H$ must be entered if the board has an offset of zero at any of the base addresses. The parameter *mode* tells the software whether the PIP640-B card is strapped to allow zoom resolution or not, i.e., if it is non-zero the zoom resolution is allowed. The parameter *speed* specifies the clock speed and must be non-zero to indicate that the PIP640-B is

installed. The parameter *class* specifies the video format. A value from 0 to 6 must be used, as indicated in the table below:

class	American	European	Resolution	PIP type
0	512X480	512X512	zoom or full	all
1	640X480	640X512	zoom or full	640 only
2	not allowed	512X576	zoom or full	640 or 1024
3	not allowed	640X576	zoom or full	640
4	1024X1024	1024X1024	full only	640 or 1024
5	user defined	user defined	zoom or full	all
6	user defined	user defined	zoom or full	all

The *vid_type* parameter specifies whether the video format is to be a 50Hz or 60Hz format i.e., non-zero is an European format, and a zero is for an American standard video format. The *zoom* parameter specifies whether the format is to use full resolution or zoom resolution i.e., zero is used for full resolution, and non-zero is used for zoom resolution.

setw(indow) x1 y1 x2 y2
This command specifies the lower left and upper right corners of the Window. The parameters *x1* and *x2* indicate the *x* coordinates of corners 1 and 2 of the display Window. The parameters *y1* and *y2* indicate the y coordinates of the corners 1 and 2 of the display Window.

cl(ear) smap umap
This command clears the screen by setting one of the unused maps of the input LUT to the current draw index and then taking a snapshot. If none of the input LUT's maps are available, the screen may also be cleared by drawing a full-screen filled rectangle. The parameters *smap* and *umap* select one of the 8 input LUT maps to be active map following the clear operation, and the current draw index respectively. Values from 0 to 7 are used.

pause
This command is used to halt the interpreter's operation until a key from the microcomputer' keyboard is strike.

Commands of the interpreter program to be used in this practice, section (b):

winfrdisk bsize<x> file<s> workbuffer<x> seg<x>
This command is used to copy an image file from the disk to the current display Window. The parameter *bsize<x>* specifies the size of an intermediate buffer that MS-DOS requires to make the transfer i.e., the optimal buffer size is 4096 bytes. The parameter *file<s>* provides the image file name to be transfer. The *workbuffer<x>* parameter gives the offset of the intermediate buffer within segment, i.e., the offset value 506a is normally used. The *seg<x>* parameter specifies the segment that contains the intermediate buffer. The value -1 is normally used.

Commands of the interpreter program to be used in this practice, section (c):

winr(ead) workbuffer<x>seg<x>
This command is used to copy the contents of the current display to a buffer. The parameter *workbuffer<x>* specifies the address of the buffer where the contents of the Window will be stored. The parameter *seg<x>* specifies the segment portion of the memory buffer address. This command will return the number of bytes that have been transferred.

putfile file_name<s> address<x> count
This command is used to copy the content of a buffer located in the system's memory into a disk file. The *file_name<s>* is the name of the file into which the contents of the buffer is to be copied. The *address<x>* specifies the address of the buffer to be copied. The parameter *count* specifies the number of bytes to be transferred from the buffer to the file.

cm(emory) address<x> count
This command is used to clear to $(00)_H$ the contents of a portion of the interpreter's 16K buffer. The parameter *address<x>* specifies the address of the initial memory location to be cleared. The parameter *count* specifies how many memory locations will be cleared starting at *address<x>*.

getf(ile) file_name<s> address<x>
This command is used to copy the contents of a file into a specified buffer i.e., it is too important that the buffer contains enough space for the contents of the file. The *file_name<s>* specifies the name of the file to be copied. The

address<x> parameter specifies the address of the buffer where the content of the file is to be copied. This command will return the number of the bytes transferred from the file.

winw(rite) workbuffer<x>seg<x>
This command is used to copy the contents of a buffer to the current display Window. The *workbuffer<x>* parameter contains the address of the buffer, containing the data to be transferred to the current display Window. The *seg<x>* parameter specifies the segment portion of the memory buffer address. This command will return the number of bytes that has been transferred.

gr(id) incx incy
This command is used to draw a grid by using the current index, inside the current Window. The parameters *incx* and *incy* are respectively the number of pixels between the lines of the grid in the *x* and *y* direction.

pixr(ead) x y
This command is used to read the value of a pixel in the frame buffer. The parameters *x* and *y* are coordinates of he pixel to be read.

pixw(rite) x y value
This command is used to write a pixel to the frame buffer. The parameters *x* and *y* are coordinates of the pixel to be written. The parameter *value* is the value to be written to the frame buffer i.e., from 0 to 255, a decimal value.

Practice 2:

How to interface a video camera with the PIP640-B and save the image from the frame buffer on the disk ?

Commands of the interpreter program to be used in this practice:

ch(annel) channel
This command selects the active video input port. The parameter *channel* selects the input channel to be used as in the table below:

channel	action
0	selects channel 0
1	selects channel 1
2	selects channel 2
3	selects channel 3

sy(nc) mode
This command is used to select the source of the sync signal. If the parameter *mode* is equal to 0 the internal sync is used. However, if the parameter *mode* is equal to 1 the external sync is used.

sbu(f) mode
This command is used to select the video output source. If the parameter *mode* is equal to 0 the output of the input LUT is displayed, and the keying is disabled. However, if *mode* is equal to 1 the output of the frame buffer is displayed, and the keying is allowed.

wint(ordisk)bsize<x>file<s>off<x>seg<x>
This command is used to copy the contents of the current display Window to disk. The parameter *bsize<x>* specifies the size of an intermediate buffer that MS-DOS requires to make the transfer i.e., the optimal buffer size is 4096 bytes. The parameter *file<s>* provides a file name for the transfer i.e., will be the name of the image file to be copy. The parameter *off<x>* gives the offset of the intermediate buffer within segment i.e., the value 506a is normally used. The parameter *seg<x>* specifies the segment that contains the intermediate buffer i.e., the value -1 is normally used.

Practice 3:

How to calculate a histogram of the gray level in a selected current Window ?

Commands of the interpreter program to be used in this practice:

hi(stogram) buffer<x>
This command is used to calculate a histogram of the gray level in the current Window. The parameter *buffer<x>* specifies the array of 256 elements which is used to store the histogram value. To calculate a histogram of a gray levels using different Windows at the same image, it will be always necessary, first, to use the *setw(indow) x1 y1 x2 y2* command.

dh(isto)max<x> x y scale gcol tcol buf<x>
This command draws a histogram in the frame buffer. The parameter *max<x>* specifies the count of the most frequently occurring pixel value in the histogram. The parameters *x*, and *y* are used to specifies the *x* and *y* position of the histogram. The *scale* parameter specifies the height in pixels of they highest bar of the histogram. All other values are drawn relative to this height. The parameters *gcol*, and *tcol* specifies the index i.e., the color, to be used to draw the bar and the labels of the histogram. The parameter *buf<x>* is the address of the buffer where the histogram created by the *hi(stogram)buffer<x>* command is stored.

reta(ngle) x1 y1 x2 y2
This command is used to draw a rectangle having (x1,y1) and (x2,y2) as opposite corners. The parameters *x1*, *x2*, *y1*, and *y2* specifies respectively the first and the second *x* and *y* coordinates.

Practice 4:

How to initialize a lookup table map (the use of pseudocolor) as well as to combine different functions drives R- red, G- green, and B-blue ?

Commands of the interpreter program to be used in this practice:

scaling x1 y1 x2 y2 buffer<x>
This command maps the values of Xn onto Yn, with X as the index to the LUT and Y as the mapped value. On the basis of this command it is possible to emphasize one part of an image at the expense of anther. By making multiple calls to scaling it is possible to break the LUT into several sections. The parameters *x1* and *x2* are the first and the last address in the buffer to be scaled. The parameters *y1* and *y2* are the first and the last value (the low and the upper end of the scale) to be stored at buf[x1] and buf[x2] respectively. The parameter *buffer<x>* is the address of the buffer where the result of scaling function is to be stored.

lutd(define) map color start length buffer<x>
This command, using the values stored in a buffer, initializes a lookup table map. The input LUT is load from the buffer in reverse order with respect to the other LUT's. The parameter *map* specifies which of the 8 maps of any one particular LUT is to be load with the data. The parameter *color* specifies the lookup table which will be affected by the change of data i.e., 0 specifies an input lookup table, 1,2, and 3 specify respectively the blue, the green and the red lookup tables. However, if color equal to 4, then all output lookup tables will be affected. The parameter *start* indicates which of the 256 bytes of the selected map will serve as the starting point to initialization (from 0 to 255 inclusive). The parameter *length* indicates how many bytes will be rewritten using the scaling function (it must be between 1 and 256 inclusive). The parameter *buffer<x>* contains the location of the buffer i.e., containing the values of the initialization.

It is important to observe that the value in the LUT's determine the color intensity, while the difference between the three values control the color. Thus a white pixel of value 255 is the sum of one red pixel of value 255 plus one green pixel of value 255 plus one blue pixel of value 255. In the same way a black pixel of value 0 is the sum of one red pixel of value 0 plus one green pixel of value 0 plus one blue pixel of value 0.

Practice 5:

Convolutions is a special class of image enhancement technique known as spatial filtering.

Spatial filtering is typically used for edge enhancement or noise reduction prior to edge or objet detection.

How to convolve an image present in the current Window with a pre-defined kernel ?

Commands of the interpreter program to be used in this practice:

ker3 k1 k2 k3 k4 k5 k6 k7 k8 k9 buffer<x>

This command is used to generate the 3X3 convolution kernel used in the area process operation. The parameters *k1..k9* specify the values within the kernel as showed below:

$$k = \begin{vmatrix} k1 & k2 & k3 \\ k4 & k5 & k6 \\ k7 & k8 & k9 \end{vmatrix}$$

The parameter *buffer<x>* specifies the storage location for the kernel.

con(3) source dest buf3<x>

This command convolves the image in the current Window in the source workspace and copies the result to an identical Window in the destination workspace. It uses a kernel that is previously stored in a pre-defined buffer, i.e., if *PA...PI* are the pixels in the current Window as showed below:

PA	*PB*	*PC*
PD	*PE*	*PF*
PG	*PH*	*PI*

the result of the operation *PE* is given by:

$$PE = \frac{k1PA + k2PB + ... + k9PI}{(k1 + k2 + ... + k9)} \qquad (9)$$

If *k1, k2,...,k9* are equal to zero the division is not performed and *con(3)source dest buf3<x>* takes the absolute value of the result and truncates it to 255. The *source* parameter selects the workspace in which the convolution is to be performed. The parameter *dest* selects the workspace into which the convoluted image will be copied. The parameter *buf3<x>* is the location of the buffer where the kernel is stored.

As a function of the kernel it is possible to define the processing i.e., low pass filtering, high pass filtering, Laplacian filtering, SOBEL filtering, horizontal or vertical edge detection, or median filter operation.

Low pass filtering of an image produces an output image in which spatial frequency components have been attenuated. It is often used as a smoothing operation to remove visual noise.

High pass filtering of an image produces an output image in which high spatial frequency components are accentuated. It is often used in the enhancement of edges.

Laplacian edges enhancement produces an output image in which high spatial frequency components, such as edges, are sharply attenuated. It is useful for extraction of object edges or boundaries.

The PIP640-B board has also built in two different Laplacian filters, a SOBEL, a horizontal and a vertical edge detection, an average, and a median filter available on the interpreter program, that are:

lp1 source dest
lp2 source dest
The commands perform a Laplacian filter i.e., horizontal and vertical edge detection at the same time. The parameter *source* specifies the workspace for the operation and the *dest* parameter the destination workspace for the operation. The kernels used by those commands are:

$$k(lp1) = \begin{vmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{vmatrix}$$

$$k(lp2) = \begin{vmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{vmatrix}$$

sobel source dest
This command compares the result of the two convolutions to estimate the strength and the orientation of edges in the image. The *source* and the *dest* parameters specify respectively the source workspace and the destination workspace.

ho(rfilter) source dest
ve(rfilter) source dest
These commands perform a horizontal and a vertical edge detection transformation on the current Window respectively. The parameters *source* and *dest* specify respectively the source workspace and the destination workspace. They use a 3X3 convolution by using the kernels showed below:

$$k(h) = \begin{vmatrix} -2 & -2 & -2 \\ 0 & 0 & 0 \\ 2 & 2 & 2 \end{vmatrix}$$

$$k(v) = \begin{vmatrix} -2 & 0 & 2 \\ -2 & 0 & 2 \\ -2 & 0 & 2 \end{vmatrix}$$

average source dest
This command performs a pixel averaging transformation on the current Window. The parameters *source* and *dest* specify respectively the source workspace and the destination workspace. They use a 3X3 convolution by using the kernels showed below:

$$k(av) = \begin{vmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{vmatrix}$$

me(dian) source dest
This command performs a 3X3 median filter operation on the current Window. The parameters *source* and *dest* specify respectively the source workspace and the destination workspace.

Practice 6:

How to perform morphological transformation on the pixel structure into a pre-defined Window ?

Commands of the interpreter program to be used in this practice:

di(late) source dest
This command performs a dilation on the contents of the Window in the source workspace. It use a 3X3 structuring element to perform the dilation. The parameters *source* and *dest* specify respectively the source workspace and the destination workspace.

er(ode) source dest
This command performs an erosion operation on the contents of the Window in the source workspace. It also uses a 3X3 structuring element to perform the erosion. The parameters *source* and *dest* specify respectively the source workspace and the destination workspace.

Practice 7:

How to perform zoom operation on the image structure into a pre-defined Window ?

Commands of the interpreter program to be used in this practice:

zoom mode
This command selects either full resolution or zoom resolution and allows 4 images to be stored in a quadrant. The position of images within a quadrant is set by the *pan(el) offset* and *scror(el) offset* commands. If the parameter *mode* is equal to zero it is set up the standard video format. However, if it is a non-zero the zoomed video format is enabled.

scror(el) offset
This command is used to shift the image relative to its current position on the screen. Scrolling is done in blocks of sixteen pixels. The parameter *offset* specifies the relative scroll of the Window. Both positive and negative values are accepted.

panr(el) offset
This command is used to shift the displayed image relative to its current position on the screen. Panning is performed in blocks of eight pixels. The parameter *offset* specifies the number of pixels by which the image will move. Both positive and negative values are accepted.

ACKNOWLEDGMENT

The author wishes to thank the support from the International Centre for Theoretical Physics, the Department of Computer Science from the Federal University of São Carlos, and the Brazilian Enterprise for Agricultural Research (EMBRAPA) , as well as to acknowledge the helpful interaction with Prof. Dr. Sergio Mascarenhas, Prof. Dr. Roberto Cesareo, and Prof. Dr. Giovani E. Gigante. It is also recognized the help, related to drawing works, received from V. Monzane.

REFERENCES

Bille, J.; Scharfenberg, H.; and Männer, R., *"Biological dosimetry by chromosome aberration scoring with parallel image processing with the Heidelberg polyprocessor system"*, Computer in Biology and Medicine, Vol. 13, 49-79, 1983.

Brunetti A.; Cesareo R.; Gigante,G.E; Appoloni C.R.; De Assis J.T.; Almeida A. de; Cruvinel P.E.; Mascarenhas S.; Castellano A.; Staderini E.M.; Storelli L.; *Sistemi di immagini con raggi X*, In Elaborazione di dati biomedici, Università Degli Studi di Roma "La Sapienza", 65-81, Maggio 1994.

Cruvinel P.E. and Minatel E. R., *Use of proton induced X-ray analysis and image processing for determination of trace-elements distribution in agricultural fields*, Systems Integration'94, IEEE Computer Society Press, Vol. 1, 371-376, August 1994.

Cruvinel P.E.;Cesareo,R.;Crestana,S.;Mascarenhas,S., *X-and Gamma-rays computerized minitomograph scanner for soil science*, IEEE Transactions on Instrumentation and Measurement, Vol. 39, No.5, 745-750, October 1990.

Cruvinel P.E.; S.Mascarenhas; J.Miranda; and R.G.Flochinni, *The use of a Perovskite crystal as a detector for proton beam current*, IEEE Transaction on Nuclear Science, Vol. 39, No.1, 25-28, February 1992.

Duda R.O. and Hart P.E., *"Pattern classification and scene analysis"*, Wiley, New York, 1973.

Fukunaga K., *"The estimation f the Bayes error by the k-nearest neighbor approach, In: Progress in Pattern Recognition"* (L.N. Kanal and A. Rosenfield, eds.), Vol. 2, North-Hollaand Publ., Amsterdam, 1985.

Gonzales, R.C., and Wintz,P., *Digital Image Processing*, Addison-Wesley Pub. Co., 2 ed., 1987.

Huang, H.K. *Elements of digital radiology; A professional handbook and guide*. Englewood Cliffs: Prentice-Hall, 1987, 370p.

MATROX-Electronic Systems. *PIP video digitizer board hardware and the user manual 289-MH-00 Rev 3*. Canada, Feb. 1988, 1v.

Niblack, W., *An introductiom to Digital Image Processing*, Prentice/Hall International (UK) Ltda, 1986.

Preuss, R.D., *"Very fast computation of the Radix-2 discrete Fourier transformation"*, IEEE Transaction on Acoustic, Speech, Signal Processing, ASSP-30, 595-607, 1982.

Young T. Y. and Sun Fu K. *"Handbook of pattern recognition and image processing"*, Academic Press, Inc. San Diego, California, 1986.