

TR 86062

RTM FILE COPY

UNLIMITED

160099

TR 86062

AD-A208 106



ROYAL AIRCRAFT ESTABLISHMENT

Technical Report 86062

September 1986

HANDBOOK OF THE GEMS IMAGE PROCESSING SYSTEM

by

K. H. Bagot

DTIC
ELECTE
MAY 23 1989
S D
Cb D

DISTRIBUTION STATEMENT A

Approved for public release;
Distribution Unlimited

"Original contains color
plates; All DTIC reproductions
will be in black and
white"

Procurement Executive, Ministry of Defence
Farnborough, Hants

UNLIMITED 89 5 22 007

DISCLAIMER NOTICE

**THIS DOCUMENT IS BEST QUALITY
PRACTICABLE. THE COPY FURNISHED
TO DTIC CONTAINED A SIGNIFICANT
NUMBER OF PAGES WHICH DO NOT
REPRODUCE LEGIBLY.**

CONDITIONS OF RELEASE

0036899

BR-100999

DRIC Registrar
BDS(W)
British Embassy
BFPO 2
Washington DC 20008

H

THIS DOCUMENT IS THE PROPERTY OF HER BRITANNIC MAJESTY'S GOVERNMENT, and is issued for the information of such persons only as need to know its contents in the course of their official duties. Any person finding this document should hand it to a British forces unit or to a police station for its safe return to the MINISTRY OF DEFENCE, HQ Security, LONDON SW1A 2HB, with particulars of how and where found. THE UNAUTHORISED RETENTION OR DESTRUCTION OF THE DOCUMENT IS AN OFFENCE UNDER THE OFFICIAL SECRETS ACTS OF 1911-1939. (When released to persons outside Government service, this document is issued on a personal basis and the recipient to whom it is entrusted in confidence, within the provisions of the Official Secrets Acts 1911-1939, is personally responsible for its safe custody and for seeing that its contents are disclosed only to authorised persons.)

U

COPYRIGHT (c)
1988
CONTROLLER
HMSO LONDON

UDC 681.3.056

ROYAL AIRCRAFT ESTABLISHMENT

Technical Report 86062

Received for printing 30 September 1986

HANDBOOK OF THE GEMS IMAGE PROCESSING SYSTEM

by

K. H. Bagot

SUMMARY

This Report describes the operation of the GEMS interactive image processing system and gives details on the background to the functions as well as details of how the operations are controlled by the user. The Report relates to the 'Diamond' version of the software.



Departmental Reference: Space 664

SEARCHED	INDEXED
SERIALIZED	FILED
OCT 1 1986	
A-1	

Copyright
©
Controller HMSO London
1986

LIST OF CONTENTS

	Page
1 INTRODUCTION	3
1.1 Introduction to the operation of GEMS	3
2 GEMS ORGANISATION	4
3 GEMSTONE ORGANISATION	8
4 CONTROL PANEL	10
5 DETAILS OF THE COMMANDS	15
5.1 Introduction to the Commands	15
5.2 Choose an Image	16
5.3 Copy an Image	18
5.4 Contrast Stretch, Histograms, Intensities	22
5.5 Density Slice, Measure Areas	36
5.6 Maths Operations	44
5.7 Linear Filters	76
5.8 Classifiers, Copy Bit Plane	95
5.9 Text and Graphics	111
5.10 Arithmetic	127
5.11 Special	141
5.12 Fourier Transforms	154
5.13 Information	181
Acknowledgements	183
Bibliography	183
Primary Command Index	184
General Index	187
Report documentation page	inside back cover

1 INTRODUCTION

GEMS is an image processing instrument and GEMSTONE is the 'user friendly' software system that controls GEMS. The combination of GEMS and GEMSTONE can cope with a wide variety of images. Initially, GEMSTONE was designed to deal with the data from the American spacecraft Landsat and in particular from its Multi-Spectral Scanner (MSS). Although many of the processes in the system have come from experience of handling Landsat MSS imagery, they have been implemented in a very general way. Thus, whereas Landsat MSS has four spectral bands (ie four co-incident images taken in different parts of the spectrum), the system can cope with 20 images, making it suitable for examining data from the Landsat Thematic Mapper (7 bands), airborne scanner systems (11 typical), and multitemporal images (up to 20), etc. Similarly, in Landsat MSS, one picture point can have an intensity in the range from black to white, or from 0 to 255. The system can cope with an intensity range of -32768 to 32767, making it suitable for the examination of data from the NOAA series of weather satellites (0 - 1023) and radar imaging systems (eg Seasat SAR, 0 - 8191), etc. Floating point numbers are also supported. The processes are thus very general in design, so that current and future spacecraft imagery of all types can be handled, together with aircraft scanner/camera data, and also pictures from a wide variety of terrestrial activities.

1.1 Introduction to the operation of GEMS

GEMSTONE provides easy control of GEMS by means of a 'menu' system. Rather than typing computer commands at a terminal, the user points at commands on the television screen. Unfortunately, a finger cannot be used, but it is easy to move a cursor, under the control of a rolling ball, to the required command written on the screen and then to press a button to execute the command. Perhaps this can best be illustrated by the following short sequence. Little explanation of what is happening is given at this stage, but if these instructions are followed, the user can select and display an image.

Initially, the display on the television screen should have the cursor placed over the box labelled 'Choose an Image'. The 'INPUT' button is then pressed. (See Fig 5 if the button labelling is not obvious.) A list of scenes then appears and the user should move the rolling ball so that the cross of the cursor lies inside the box at the start of the description of one of the scenes. 'INPUT' is then pressed and the cursor now appears over the command 'Copy an Image'. 'INPUT' is again pressed allowing the user to copy an image into GEMS from the computer. A large number of commands are now displayed, but if the user simply presses 'INPUT' until the command 'end' has been executed, then the chosen image will be brought into GEMS and displayed on the screen. All the complexities of the the storage of images in computers, the transfer and display are thus transparent to the user.

Simply following these instructions will enable the user to display a chosen image. However, to proceed further and in a wide variety of directions, an understanding of the background of the GEMS system, the GEMSTONE software and the image processing functions is required.

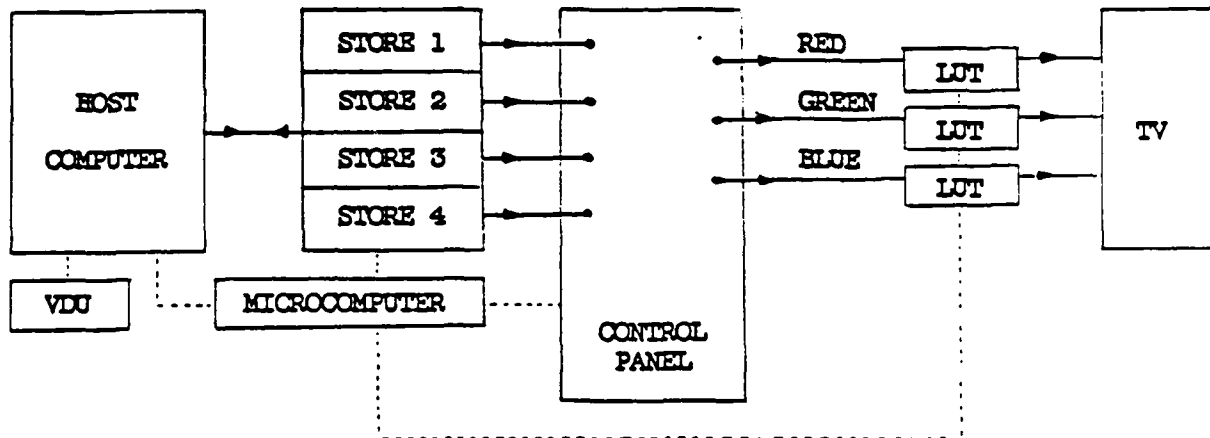
2 GEMS ORGANISATION

Fig 1 Organisation of GEMS

The central component of the GEMS system is the image store shown on the left in Fig 1. A digital image is made from a large number of discrete rectangular picture elements, or pixels for short. A photograph can be considered as a large number of successive lines where each line contains a large number of pixels. In each GEMS store there are 512 lines each containing 512 pixels, making a total of 262,144 pixels in one image. Later versions of the GEMS hardware have stores which can hold 1024 lines of 1024 pixels. GEMSTONE also works with these versions. There are so many pixels that the eye cannot see them as a collection of points, but rather as a continuous picture. Each one of these pixels has an intensity in the range from black to white. For the computer, the number 0 represents black and 255 represents white so that 127 corresponds to mid-grey.

Four of these images can be stored and any one of these can be connected to any of the three colour guns of the television. Thus in Fig 1, inside the box labelled 'CONTROL PANEL', a connection can be made from store 1 to red, simply by pressing a button. This set of three colour guns, red, green and blue can be used to make any other colour. Thus if store 1 were connected to red and green, the picture on the screen would be yellow. Connecting store 1 to blue as well, would result in a black and white picture. However, before the picture is displayed on the television, the data pass through a box labelled 'LUT', which enables the user to change the contrast range (see section 5.4) and to perform a density slice (see section 5.5).

Simple menu requests to the microcomputer can copy images from the host computer into the GEMS stores as has been demonstrated in section 1. For example, a Landsat MSS scene might be copied so that band 4 is in store 1, band 5 in store 2, band 6 in store 3, and band 7 in store 4. The standard Landsat false colour composite can then be created by pressing the buttons to connect store 4 (ie band 7) to red, store 2 to green, and store 1 to blue. (See section 4 for more details of the control panel).

Processing other than density slicing and contrast stretching is done in the host computer and the simple menu requests to the microcomputer can result in a wide range of simple and complex processes being executed by the host computer.

Besides the four stores, there are four 'overlay planes' that allow the user to superimpose information on top of the picture by pressing the buttons on the extreme left of the control panel. These overlays are used to display the menus and classification results etc. (See section 4 for more details of the control panel).

The remainder of this section may be omitted initially.

In order to consider the operation of the stores in more detail, it is necessary to have a basic understanding of binary numbers. With decimal numbers there are ten characters, namely 0,1,2,3,4,5,6,7,8,9 but in binary there are only two, 0 and 1. When counting using decimal numbers, the characters are used in order from 0 to 9. To proceed beyond 9 requires two characters starting at 10 and going to 99, after which three characters are required, etc. Similarly in binary, after counting from 0 to 1, a second character is required, starting at 10 and going to 11, after which three characters are required, etc. This decimal and binary counting procedure is shown in Table 1, where, starting at 0, the number 1 is added to each successive line.

<u>Decimal</u>	<u>Binary</u>
0	0
1	1
2	10
3	11
4	100
5	101
6	110
7	111
8	1000
9	1001
10	1010
11	1011
:	:
:	:
98	1100010
99	1100011
100	1100100
101	1100101
:	:
:	:
254	11111110
255	11111111
256	100000000
:	:

Table 1 Comparison of Decimal and Binary Counting

Thus, to count to 255 requires 8 digits in binary. This binary number is said to have 8 'bits' (short for BInary digITS). The GEMS system is designed to deal with images where the range of intensity for a pixel is from black (level 0) to white (level 255). ie 8 bits are required for each intensity value. Note that the characters in the leftmost column are more important or more significant. Thus in decimal, the numbers in the hundreds column carry more weight than the numbers in the tens column. In binary, the bit on the left is called the 'most significant bit' or MSB and the one on the right is the 'least significant bit' or LSB. The MSB is generally called bit 1, and the LSB, bit 8.

An electronic switch can be used to represent a single bit. A switch which is OFF represents 0, and ON represents 1. An 8 bit number therefore needs 8 switches. A simple example of a digital image is shown in Fig 2.

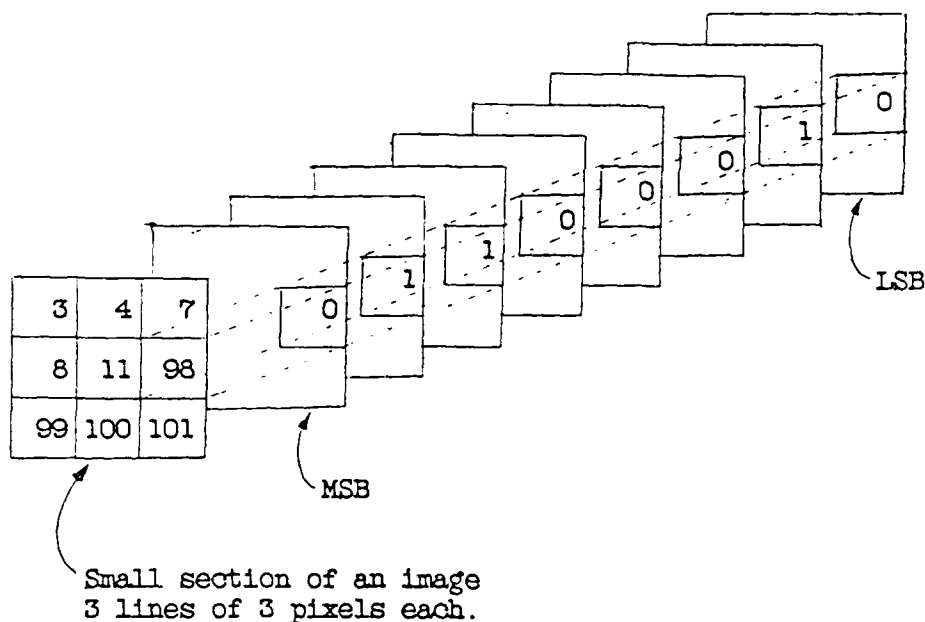


Fig 2 Storage of pixels in a computer memory

One pixel has the value 98. From Table 1, the binary equivalent of this is 01100010. The eight switches required to hold this number are shown. Each pixel has a similar set of eight switches and the result is that all the MSB switches are collected in a plane, and likewise for the other seven bits. These so-called 'bit planes' are very useful in GEMS. For example, suppose a scene has been classified into water and non-water (see section 5.8 for details of the classifier). Wherever there is a 'water pixel' a switch in a bit plane can be turned ON. Wherever there is a land pixel it can be turned OFF. This classification result can be stored in one bit plane, therefore allowing eight classification results to be held in one image store.

The 'overlay planes' mentioned above are single bit planes which could hold a classification result. The buttons on the extreme left of the control panel allow this information to be superimposed on top of the displayed image.

Within the various chapters of GEMSTONE it will be noticed that 20 stores are listed and not 4. Only the top 4 labelled D1,D2,D3,D4 are real physical stores in the GEMS hardware. The others are 'virtual' stores in the host computer memory system and are used to hold intermediate results, classifications, other extracts, other images, images with more than 4 bands etc. These 'working' stores can be treated exactly as if they were real GEMS stores, the only difference being that they cannot be viewed at the press of a button. To display an image in a working store, it is simply copied into a display store (see section 5.3). With a little experience, great use can be made of these working stores. Initially it can be likened to a desk on which there are 20 photographic prints. Each one can be brought into the middle of the desk for close examination, and put on the outside edge when not currently needed. Thus an image can be moved quickly into one of the top 4 stores (ie the middle of the desk) for examination, and then put in one of the lower 16 stores when the examination is complete (ie the outside of the desk). However, the computer does not have the limitation of the human in looking at images. Thus the human at his desk can only look at one photograph at one time, whereas the computer can work on all 20 images at the same time.

These bottom 16 stores or working stores have one extra capability. Instead of having 8 bit planes, they have 16, enabling one to store images with a much wider dynamic range (ie -32768 to 32767). The processes in the system are designed to handle such data, and when it is required to display such an image, the facilities within the arithmetic package (see section 5.10) can be used to compress or truncate the data to 8 bits.

For operations such as Fourier transforms, even the dynamic range of a 16 bit integer number is too small. Hence there is a floating point format which can be used in the lower 16 stores. Bit 1 (the MSB) is the sign of the mantissa. The next eight bits (ie bits 2 to 9 inclusive) are for the exponent giving a range of 2 to the power -127 to +128 (equivalent to 5.9×10^{-39} to $3.4 \times 10^{+38}$). The last 7 bits (bits 10 to 16 inclusive) are for the mantissa, which is normally stored in the form 0.xxxxxxx, with the 7 bits after the decimal point. As an example, the binary number 10111.01101 could be stored as $0.0001011 \times 2^{+8}$, but there are only four significant digits in the mantissa. A better way to store this would be as $0.1011101 \times 2^{+5}$, ie with the significant digits left justified to the decimal point. However, every number would start with 0.1. Thus the best way to store the number is as $1.0111011 \times 2^{+4}$ and only the bits after the decimal point are stored. This effectively provides a mantissa resolution of 8 bits plus the sign bit.

3 GEMSTONE ORGANISATION

This section describes the general way in which GEMSTONE operates. There is no information in this section about what a particular command does, but it is about how to execute commands in general.

The first page on the screen is the 'Contents Page', and rather like a book, it has a list of 'Chapter Headings'. There is a cursor which can be moved using a rolling ball. If the cross of the cursor is placed inside a box containing a chapter heading, and the INPUT button pressed, the chapter itself is displayed on the screen. Models of GEMS with fixed overlay planes have a coloured patch rather than a cursor, which is moved in exactly the same manner, to the desired command. This operation is similar to opening a book and turning the pages to the desired chapter. Just as in a book, when the user has finished with a chapter he 'returns' to the contents page and selects the next chapter. This sequence of turning pages in a book is illustrated in Fig 3 and in the following example.

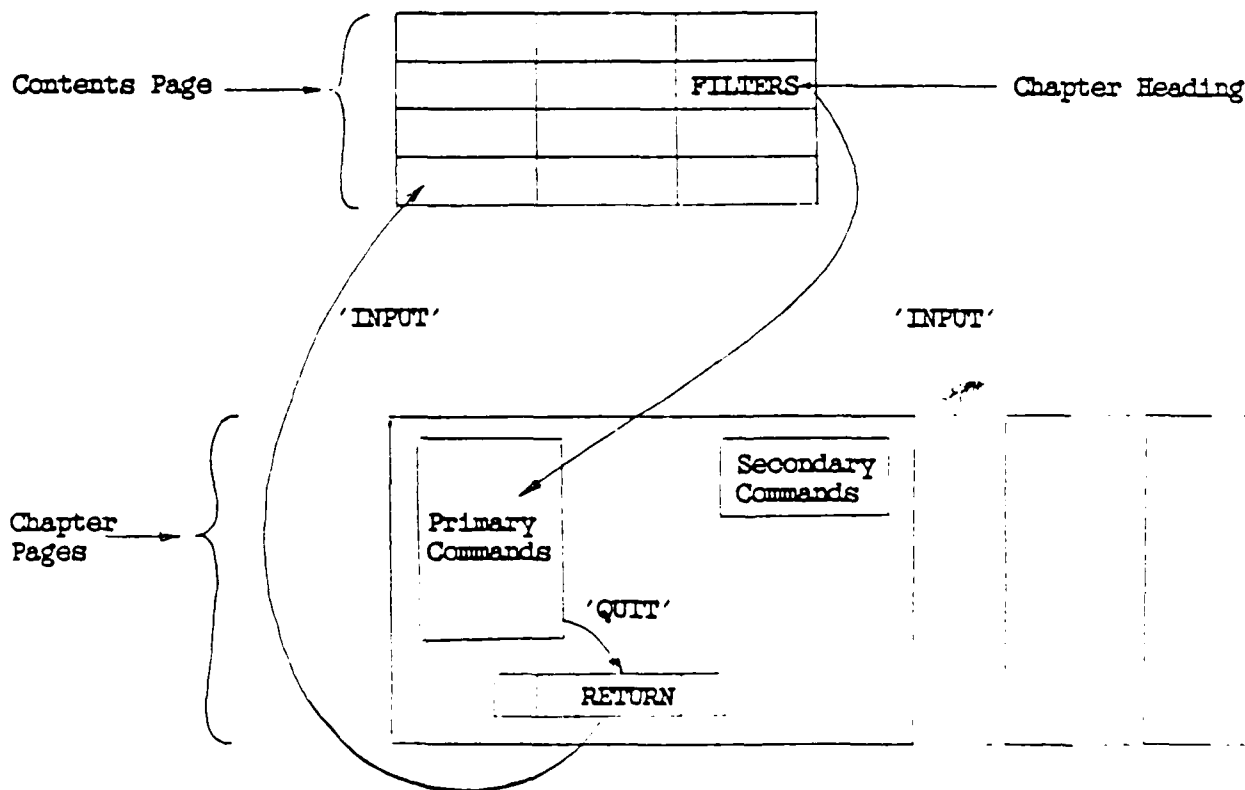


Fig 3 Operation of the GEMS menu

Suppose it is required to perform a density slice (see section 5.5). The cursor is placed over the box containing the chapter heading 'density slice' in the contents page and the INPUT button is pressed. This commands the pages to be turned to the chapter on density slicing. The cursor appears over a set of instructions on the left hand side of the screen, and these are the 'Primary Commands'. As before, a command is executed by moving the cursor over the required operation and pressing the INPUT button. In general, further instructions are needed, and the cursor is automatically moved to groups of 'Secondary Commands', where the user is invited to choose from various options and press INPUT. Text appears at the bottom of the screen to help the user decide what information the system requires. This sequence of secondary commands continues until the system has enough information to do the required processing.

When the process is completed, the system may return to the primary commands to await the next instruction, or it may display the result of the processing. To proceed from the latter case, the user simply presses INPUT to get back to the primary commands.

When the user wishes to perform another process not listed in the chapter, (ie to leave or quit the chapter), the QUIT button (top right of control panel. See section 4) is pressed. This command moves the cursor down to the set of commands at the bottom of every chapter, as shown in Fig 4.

STATUS	HELP	RETURN	IMAGE
--------	------	--------	-------

Fig 4 Commands at the bottom of every page

Execution of these commands is exactly the same as for all the other commands, ie press INPUT when the command has been selected. The cross of the cursor will appear over the 'Return' box. Simply pressing INPUT returns the user to the contents page, where another chapter can be selected. In the contents page 'Return' is replaced by 'Stop', which on execution causes the program to terminate. This is only used at the end of a session since it requires the system to be restarted from the computer terminal.

'Help' provides the user with basic information on the terminal relating to the commands on the current page only.

'Status' provides the user with information on the terminal relating to the status of the system with regard to the particular commands on the current page. Thus the status of contrast stretches only appears in the page on contrast stretching.

'Image' simply removes the menu overlay so that the image can be seen. To restore the menu, INPUT is pressed.

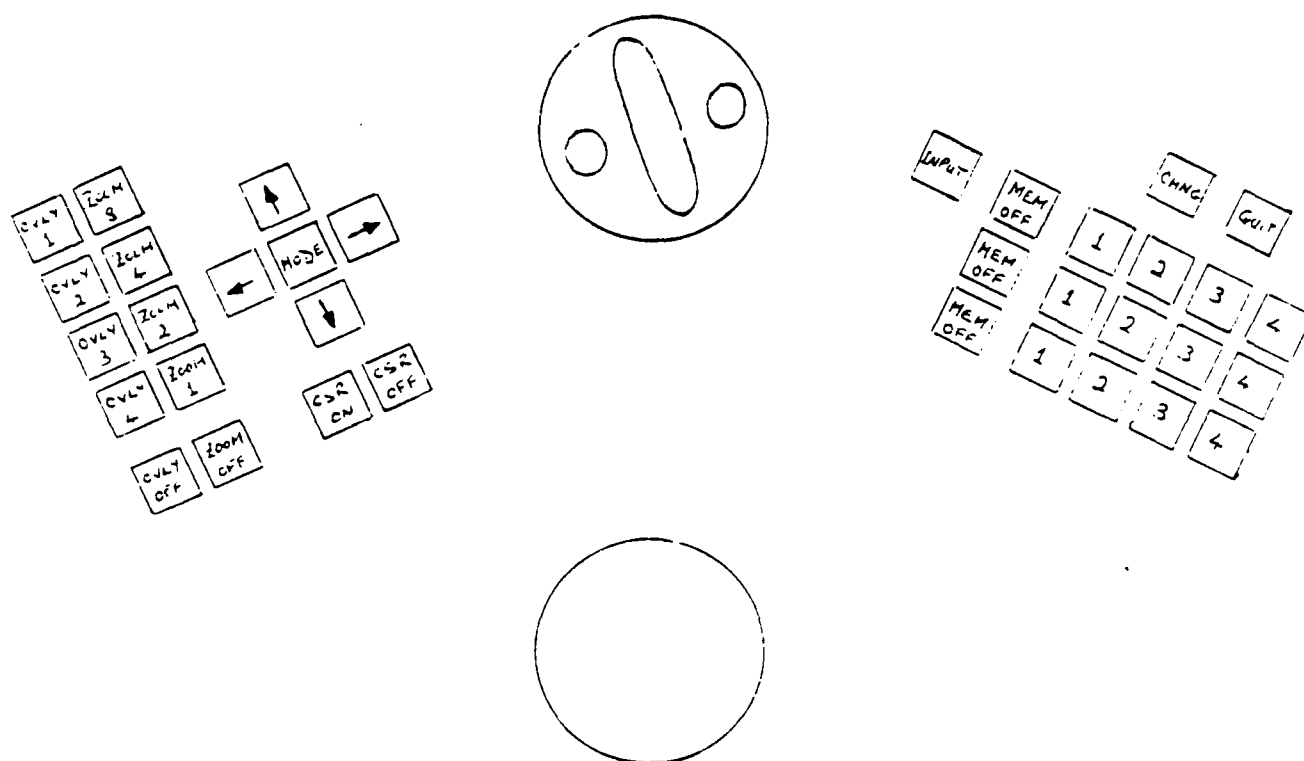


Fig 5. Layout of the Control Panel.

4 CONTROL PANEL

Since in designing the control panel there were two conflicting requirements, namely, to keep a fairly simple layout and yet to provide the user with a wide range of functions, many of the buttons have two modes. The key to this mode complexity is the white button labelled MODE in Fig 5. If the light is on under this switch, then the system is in mode 0. This is the initial mode when all the buttons on the panel execute the function written on them. In mode 1, there is no light under the MODE button.

(a) Mode 0 functions

Starting from the left hand side, the first set of buttons are labelled, OVLY 1, OVLY 2, OVLY 3, OVLY 4, OVLY OFF. Pressing OVLY 1 will cause the first overlay plane to be displayed on top of the image. Pressing the button again will cause the overlay to be removed. The other three overlay planes operate in the same manner. The last button in the group, OVLY OFF simply removes all overlays from the image. In normal operation, it will be found that the menu pages reside in the first overlay plane. If the GEMS model has fixed overlay planes, one of these will be used for the menus. GEMSTONE uses overlay planes two and three to display information to the user. It does not write into overlay plane four, which is kept free for the user to insert text, outlines, patches, etc which may also be used as inputs to some of the GEMSTONE routines.

The next set of buttons are labelled, ZOOM 8, ZOOM 4, ZOOM 2, ZOOM 1, ZOOM OFF. These buttons allow the user to enlarge part of an image by factors of 2, 4 or 8. Pressing a zoom button causes that zoom factor to be applied. To change the zoom, another zoom button can be pressed, or the zoom may be disabled by pressing ZOOM OFF. While in the zoom condition, it is possible to 'pan' around the image using the rolling ball. Hence the reason for a ZOOM 1 button, since this enables the user to pan an image with no zoom. The way in which this function operates, for a zoom of 2, is to display a pixel from the store on two television picture points, and duplicate the line. Similarly for zooms of 4 and 8, where one pixel is repeated 8 times along a line and on 8 lines, making 64 points in all. The zooming operation is centred on the cursor cross. Thus if the cursor is positioned on top of the area of interest in the scene, this area will remain on the television screen when the zoom is changed. For a zoom other than one, about 850 pixels and 580 lines can be displayed. Thus when zoomed by 2, 850 pixels out of the 1024 can be displayed and 580 lines out of 1024. This amounts to about 1/2 of the image. About 12% can be seen at a zoom of 4 and about 3% at a zoom of 8.

The next group of buttons are round the MODE button and they have directional arrows on them. They are used to move the cross of the cursor by one step in the direction of the arrow. It should be noted that when rectangles are on the screen, they are not versions of the cursor and cannot be moved by these buttons.

Below this group of cursor control buttons, are two buttons, CSR ON, and CSR OFF, that allow the user to display the cursor or to remove it.

In the middle is a large black ball that rolls when pushed sideways with the fingers. This is used to move the cursor, move rectangles, change the size of rectangles and provide the input to many of the processes which require user interaction.

Immediately above the black ball, is a knob which when rotated clockwise brightens the lamps under the buttons on the control panel.

The next button labelled INPUT is the one used to execute all processes in the system. No command operates until this button is pressed. It should be noted that the button only operates when it is illuminated. Thus the user is 'invited' to execute a command when it is lit.

The next group of buttons are the coloured ones labelled MEM OFF, 1, 2, 3, 4. If the red '1' button is pressed, then store 1 is connected to the red gun of the television. If green '1' and blue '1' are also pressed, then store 1 is connected to all three guns and a black and white picture results. Whereas one store can be connected to more than one television gun, only one gun can be connected to one store. So if red '1' is pressed followed by red '2', store 1 will be disconnected. The buttons MEM OFF allow the user to disconnect the store from the colour gun. Note that the illumination under these buttons inform the user which store is connected to which colour gun.

Above the coloured switches are the final two buttons. The one on the left labelled 'CHNG', allows multiple operations of the rolling ball. For example, if a rectangle is to be positioned inside a small area, then it must be moved using the rolling ball, but its size must also be changed. This button when pressed allows the user to change the size of the rectangle. A further press returns the user to moving the rectangle. In other words, the button is cyclic. If there are three operations for the rolling ball, then they are implemented in turn. When the contents page or the various chapter pages are being examined, this button serves another purpose. The cursor will be over a primary or secondary command and to move to another, the rolling ball will normally be used. Pressing the 'CHNG' button moves the cursor to the next command in the sequence (or to the top if currently at the last command). These two ways of moving the cursor are interchangeable and can be employed at any time.

The last button at the top right is labelled QUIT. It is used to quit or leave a chapter (see section 3). If one makes a mistake when supplying details for a process, pressing the QUIT button twice returns the user to the primary commands so that a fresh start can be made. It can only be operated when it is lit and in particular, if the user has started execution of a process, it cannot be stopped using this button. Care should therefore be exercised before finally executing a command that may take a long time or may destroy some previous work.

(b) Mode 1 functions

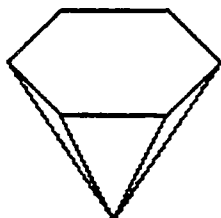
As mentioned above mode 1 is operative when the light under the MODE button is off. Mode 1 only affects the buttons on the left side of the panel. The four overlay buttons OVLY 1, OVLY 2, OVLY 3, OVLY 4, can now be used to change the colour of the overlay. Pressing one button several times will cause the colour of the corresponding overlay to change through a fixed, short sequence of colours (white, black, red, green, yellow, blue, magenta, and cyan). This is a useful facility when the background of the image is close to the default colour of the overlay.

For the zoom buttons, even in mode 1, operation continues as in mode 0 as long as the light under ZOOM OFF is unlit. When this button is pressed and therefore illuminated, the user will find that if the image is already zoomed, it stays zoomed and that the scene cannot be panned underneath the fixed cursor. In this case 'ZOOM OFF' is much more like 'pan off'. On returning now to mode 0 (ie light under mode button on), the rolling ball controls the cursor, which can now be moved over the fixed, zoomed image. Pressing ZOOM OFF once more causes the rolling ball to revert to controlling the pan around the image.

In the group around the MODE button, if the button with the arrow pointing to the left is depressed, the button will be illuminated and all operations in that direction (along the x-axis) will be frozen. Thus if the cursor is to be moved in this situation, it can only be moved vertically. This allows the user to make precise vertical movements. If a ZOOM button is pressed now, there will only be a zoom in the vertical direction. Similarly the button with the arrow pointing down (towards CSR OFF/CSR ON buttons) freezes operations in the vertical direction (or along the y-axis).

Within the 'Classifiers, Copy Bit Plane' and 'Text and Graphics' chapters are facilities to 'paint' an area (see sections 5.8 and 5.9). In mode 0, the rolling ball can move the 'paint brush' around the screen and in mode 1, the size of the brush can be varied using the rolling ball.

DIAMOND VERSION



WELCOME TO GEMS

CHOOSE AN IMAGE	COPY AN IMAGE	CONTRAST STRETCH HISTOGRAMS INTENSITIES
DENSITY SLICE MEASURE AREAS	MATHS OPERATIONS	LINEAR FILTERS
CLASSIFIERS COPY BITPLANE	TEXT AND GRAPHICS	ARITHMETIC
SPECIAL	FOURIER TRANSFORMS	INFOR- MATION

STATUS	HELP	STOP	IMAGE
--------	------	------	-------

INPUT FOR OPERATIONS OR QUIT

Fig 6 Layout of the contents page

5 DETAILS OF THE COMMANDS

5.1 Introduction to the commands

Each of the following sections deals with one of the chapters shown in the contents page illustrated in Fig 6. It is now assumed that the user can access and quit each chapter (see section 3). For each section following, there is a figure showing the chapter page as it appears on the screen. All the secondary commands also appear. On first accessing a page on GEMS, the user may find some of the secondary commands missing. They will appear when the user initiates a command that requires them.

All the sections from 5.3 to 5.12 are laid out in the same style as follows.

'Command Name'

SUMMARY

'Brief description of the command'

COMMAND IN DETAIL

'Full description of the background and operation of the command'

	BANDS	SIZE	TYPE	DESCRIPTION...
☐	4	3349	2286	[1] TEST
☐	1	3500	2944	[1] SE
☐	1	1024	1024	[1] IOW
☐	3	254	254	[1] FLEET ENHANCED
☐	3	256	256	[1] FLEET
☐	1	698	2891	[1] AERIAL PICTURE
☐	4	512	512	[1] PRESTON
☐	4	1024	1024	[1] VENICE HARBOUR EXTRACT 1024*1024
☐	3	1024	1024	[1] THEMATIC MAPPER MISSISSIPPI
☐	4	1700	1500	[1] DYFED
☐	3	1501	1261	[1] GRAMPIAN

CHOOSE IMAGE

Fig 7 The 'Choose an Image' page

5.2 Choose an Image

A list of all the images available to the user is displayed. An example of this is shown in Fig 7. To select an image, the cursor is moved into the box at the end of the required image, and the INPUT button pressed. On each line, there is a short description of the scene, the number of spectral bands, the size of the image, and, under 'Type', whether the intensity of a pixel is represented by 8 bits (ie I1) or by 16 bits (I2). If, as in most cases, the image is I1, the user need concern himself no further, since all the GEMSTONE functions will operate without any problems. If the image is I2, as for example in some radar imagery, the working stores which are 16 bit, must be used for the original data. The arithmetic package (see section 5.10) can be used to manipulate such images and to compress or truncate them for presentation on the television from the display stores.

INPUT FOR COMMAND OR QUIT

Fig 8 The 'Copy an Image' page

5.3 Copy an Image

This chapter contains the following commands.

COPY IMAGE TO STORE

SUMMARY

The image selected in 'Choose an Image' (pl7), can be copied from the host computer into any of the GEMS display or working stores. The command is also used to copy an extract, specified by 'DEFINE EXTRACT' (see below).

COMMAND IN DETAIL

The first time the command is executed very few secondary commands are required. GEMSTONE calculates how to sample the original scene so that it fits on the screen. For example, if there is an image with 2700 pixels per line and 2000 lines, then GEMSTONE will only pick one pixel in six, and one line in four. This image which now contains 450 pixels and 500 lines is the largest one which is still less than the limit of 512 by 512. Only integer sampling is used in this process. Note also that, since the picture does not cover the whole screen, but is only 450 pixels by 500 lines in this example, the area of the screen that GEMS will process when using other functions in the menu, is also 450 by 500. For most operations this is exactly what is wanted, since account is not taken of areas with no real picture information. Occasionally it is required to process outside this limited area, and this can be done by copying one image into any store such that all 512 by 512 pixels are filled. As an aside, it should be noted, that execution of the STATUS command results in the display on the terminal of scene details including the sampling used. Thus the sampling which GEMSTONE has selected for the original scene can be found if required.

The user is asked next which spectral band is to be copied into which GEMS store. For the first execution of this command, it is suggested by GEMSTONE that only the last band be copied. This is because the computer has to do a lot of work in the sampling just described, and therefore the copy process is a little slow. If the next requirement of the user is to define an extract (see below), then a single band is all that is necessary to find the area of interest in the scene. Of course, this suggestion can be overruled and all the bands can be copied. Since the system has to know when all the bands and stores have been selected, the last secondary command is 'END'. This causes the process to execute, copying the image into the GEMS stores.

When the chapter page is accessed next, it will be noticed that there is a yellow box beside the store number, where previously there were four question marks, indicating that the store no longer has unknown data, but has been loaded with an image.

On subsequent executions, a further command is used. There may be one or more extracts defined (see below) in which case the extract (or the original) to be copied, has to be specified.

DEFINE EXTRACT

SUMMARY

As was stated in the 'COPY IMAGE TO STORE' command (p19), GEMSTONE normally samples the image to make it fit the television screen (ie the 512 by 512 store). This command is used to define smaller areas for extraction and subsequent display with finer sampling, down to '1', ie every pixel can be used.

COMMAND IN DETAIL

The first question asked is the number by which the extract is to be known. Up to 8 extracts can be defined and are called extract number 1, extract number 2, etc. When one (or more) extract has already been defined, it is possible to define the new extract from the original scene or from an existing extract. The source for the extract therefore has to be decided. Next, there is the option of displaying the extract over the whole screen or in one of the four quadrants.

If the original scene was larger than 512 by 512 pixels, then it will have been sampled to fit the screen. The user is now asked how the extract should be sampled in both the x direction (ie horizontally), and the y direction (ie vertically). A rectangle appears on the image and the chapter page is removed. Using the rolling ball, the user can select the area of interest. The co-ordinates of the top left corner of the rectangle relative to the top left of the original image, are given at the bottom of the display. This whole process is so arranged that the area inside the rectangle will just fill the screen (or store) when sampled as the user directed. It is possible to reduce the size of the rectangle by pressing the 'CHNG' button and moving the rolling ball. In this case, the size of the extract is displayed at the bottom of the screen. When the image is subsequently transferred, it will not fill the whole screen and as noted in 'COPY IMAGE TO STORE', only this area will be used for processing in subsequent commands. If the rectangle does not include all of the area of interest, then either it has to be examined in pieces, or a coarser sampling must be used. In this latter case, the INPUT button can be pressed to return the user to the Primary commands, and the same extract can be redefined with a different sampling. Note that one of the sampling options is 'AUTO'. Here, just as in the initial copy, the system chooses the best sampling to almost fill the screen with the whole image.

On setting up second and subsequent extracts, the position of the previous extracts are shown to help the user locate the required area. It is often helpful to keep one band of the original image in one of the display stores (ie stores 1 to 4), to further assist in the location of extracts.

COPY STORE TO STORE

SUMMARY

Using this command, images can be moved quickly from one store to another. This copy is many times faster than copying from the original image.

COMMAND IN DETAIL

Having commanded GEMS to copy the whole image or a quadrant, the source store and the store into which the image is to be copied, are specified. The last command instructs the system to apply no contrast stretch to the data during the copy process, or to apply one from the colour LUTs (see section 5.4). This option allows the user to contrast stretch (or slice) an image and to store the image after stretching.

ERASE STORE

SUMMARY

This command simply allows the user to erase all or part of a store.

COMMAND IN DETAIL

The system is first commanded to erase the whole store or a quadrant and the store is then specified. When the task is completed, the yellow box (or the question marks) beside the store number on the chapter page, is erased.

ADD WEDGE

SUMMARY

When performing a contrast stretch (section 5.4) or a density slice (section 5.5), it is sometimes useful to have a grey wedge (ie a strip of image which is black at one end and white at the other) inserted into the image.

COMMAND IN DETAIL

The only choice, is the location of the wedge, along any of the four edges of the image. The wedge is applied to all four display stores and the data that the wedge replaces are held elsewhere in the computer.

REMOVE WEDGE

SUMMARY

This command restores the original data removed by 'ADD WEDGE'.

5.4 Contrast stretch, Histograms, Intensities

Adjustment of contrast has been done to pictures since the birth of photography, but the problem of contrast stretching is much more extreme in digital images. The human eye can distinguish between 10 and 20 separable grey levels and so photographic pictures which hold 20 or 30 levels, are well matched to the eye. However, it is very difficult to display the subtle intensity detail in a digital image of say 200 grey levels, when the eye can only see 10 to 20. Contrast stretching simply to brighten up a dark image can make the picture more interpretable, but it may not highlight a great deal of detail that could be useful. This chapter contains four automatic contrast stretches and three manual ones to help the user make the most of the data, visually.

Much use is made in this chapter, of histograms of intensity. These are charts or graphs where the horizontal axis contains all 256 possible intensity levels, and the vertical axis shows the number of pixels from the scene at each intensity. Fig 9 shows a typical histogram.

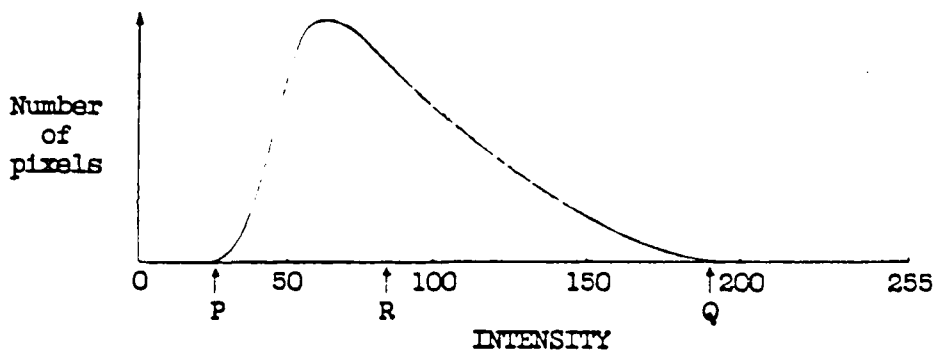


Fig 9 Typical histogram of intensity in an image

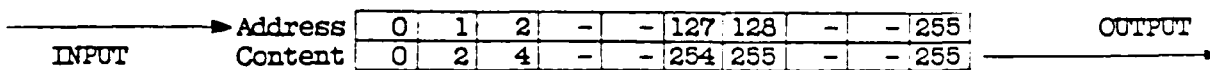


Fig 10 Block diagram of the operation of a look-up table (LUT)

As was mentioned in section 2, the data from an image pass through a box labelled 'LUT' (ie look-up table), before being presented on the television screen. This is shown in Fig 10, where a pixel at the input is treated as an address in the LUT. In this example, if a pixel of intensity 2 appeared at the input, location 2 would be examined, and the output would have a value equal to the contents of location 2 ie a value of 4. This simple LUT can thus multiply the intensity of any pixel by 2 up to a maximum input value of 127 after which all the output values are 255. The microcomputer in GEMS, although not fast enough to cope with the high data rates required by TV, can still load the 256 numbers into the locations in a fraction of a second, and the LUT itself can then run fast enough for the TV. Different contrast stretches are produced by the microcomputer changing the numbers in the LUTs.

The graph in Fig 11(A) shows what happens to the data when location x in the LUT has the contents x . Data entering the LUT at level 80, leave the other side at level 80 as well. This graph is called a 'transfer function' since it shows how the input data are transferred to the output. Fig 11(A) is the 'unity' transfer function, since it does not change the data. In Fig 11(B), the transfer function has been changed so that data at level 80 leave the LUT at level 85. However data at 30 or below all emerge at level 0, and data at 180 and above emerge at level 255. This would be a suitable contrast stretch for the data shown in the histogram in Fig 9, since there is virtually no information below 30 or above 180. Thus the range from 30 to 180 has been stretched to the range of 0 to 255. Hence the name for the process. Since the graph is a straight line, this is called a linear contrast stretch.

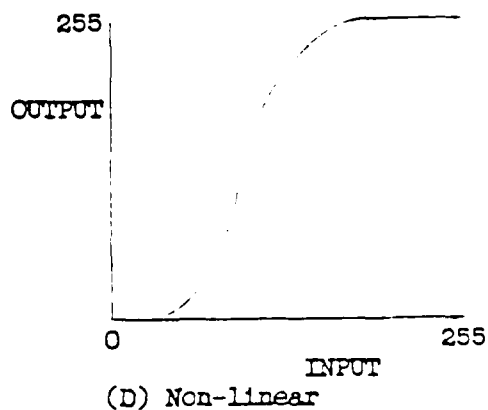
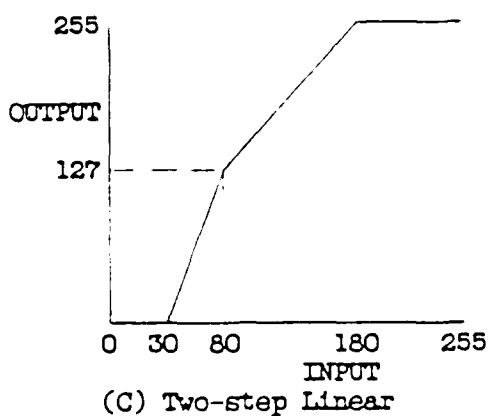
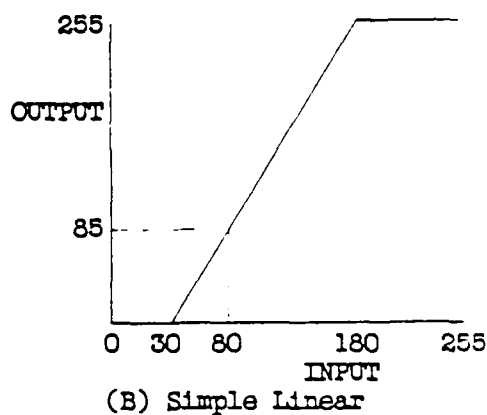
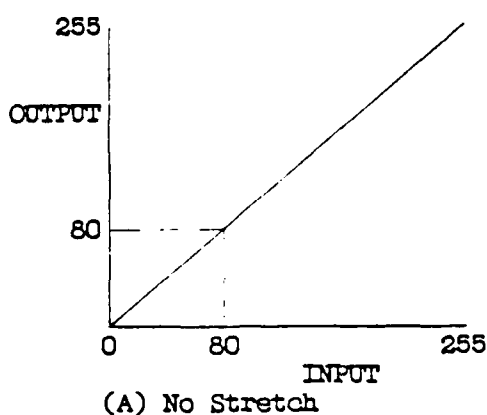


Fig 11 Typical transfer functions for contrast stretching

The histogram shown in Fig 9 was skewed. In fact the median of the data (ie the level at which half the pixels are below and half are above) was at 80 (R in Fig 9). The linear contrast stretch moved this to level 85 as shown in Fig 11(B). Most pixels would therefore leave the LUT with an intensity of less than 127, or mid-grey. Thus the picture will appear fairly dark and difficult to interpret. The next stage in this sequence of contrast stretching is a two step linear stretch as shown in Fig 11(C). Here the median data emerge at level 127, ie mid-grey. There are as many pixels above this level as below giving the picture a good balance. The next stage in the sequence of contrast stretching leads to much more complicated transfer functions as illustrated in Fig 11(D) and they are non-linear.

Fig 12 illustrates the effect of the four transfer functions in Figs 11 (A), (B), (C), and (D). It is difficult to see the image in Fig 12 (A) since it is so dark, and this is illustrated in the histogram shown below the image. The scene is of the Portsmouth area in England and was taken by the Thematic Mapper (Band 4) on Landsat 4 on 4-Feb-83. A mid morning winter image of the UK is bound to be dark, and this picture demonstrates the need for contrast stretching.

The simple linear contrast stretch shown in Fig 12 (B) (corresponding approximately to the transfer function in Fig 11 (B)), produces a satisfactory result. The histogram of the stretched image shows a good spread of the information over the whole display range of 0 to 255. Note the distribution appears to be bi-modal, the left hand peak corresponding to the water area, and the right hand peak, to the land. The image might still be thought a little dark and the median of 82.9 (shown to the right of the histogram in Fig 12 (B)), is well below mid-grey level of 127. The result of a two-step linear contrast stretch (of the type in Fig 11 (C)) is shown in Fig 12 (C), where the median of the land data has been moved to mid-grey, although the overall median is at 120.9 due to the large number of dark sea pixels contributing to the statistics. Finally, in Fig 12 (D), a non-linear contrast stretch has been applied on top of the stretch in Fig 12 (C). The effect of this is to broaden the distribution (note that the standard deviation has increased from 52 in (C) to 61 in (D)), or to make the image look more contrasty. It may be easier for the eye to see certain features in such an image, but others will be lost in the highlights and shadows. This demonstrates that a single contrast stretch for all applications is a compromise and that the human eye can extract most information for a particular application by using an interactive image processing machine to produce a specific stretch.

It should be noted that when performing contrast stretches, the contents of the stores are not altered. It is only the LUTs that are changed. To have a stretched image in a store, requires the original store to be copied to itself or elsewhere, through the LUTs, using the 'COPY STORE TO STORE' command (p21).

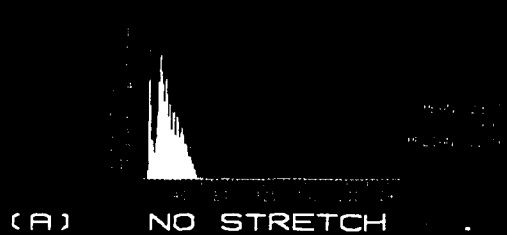


FIG 12 FOUR EXAMPLES OF CONTRAST STRETCHING

<table border="1"> <tr> <td>AUTO LINEAR</td> <td>AUTO 2 LINEAR</td> </tr> <tr> <td>AUTO EQUALISE</td> <td>AUTO GAUSSIAN</td> </tr> <tr> <td>MANUAL</td> <td>NEGATE STRETCH</td> </tr> <tr> <td>SAVE STRETCH</td> <td>RESTORE STRETCH</td> </tr> <tr> <td>PIXEL VALUE</td> <td>TRANSECT GRAPH</td> </tr> <tr> <td>HISTOGRAM</td> <td>OVERLAY 4 HISTOGRAM</td> </tr> </table>		AUTO LINEAR	AUTO 2 LINEAR	AUTO EQUALISE	AUTO GAUSSIAN	MANUAL	NEGATE STRETCH	SAVE STRETCH	RESTORE STRETCH	PIXEL VALUE	TRANSECT GRAPH	HISTOGRAM	OVERLAY 4 HISTOGRAM	<table border="1"> <tr> <td>SIMPLE LINEAR</td> <td>NON LINEAR</td> <td>2 PART LINEAR</td> </tr> </table>	SIMPLE LINEAR	NON LINEAR	2 PART LINEAR	<table border="1"> <tr> <td>NONE</td> </tr> <tr> <td>A</td> </tr> <tr> <td>B</td> </tr> <tr> <td>C</td> </tr> <tr> <td>D</td> </tr> <tr> <td>E</td> </tr> <tr> <td>F</td> </tr> <tr> <td>G</td> </tr> <tr> <td>H</td> </tr> </table>	NONE	A	B	C	D	E	F	G	H
AUTO LINEAR	AUTO 2 LINEAR																										
AUTO EQUALISE	AUTO GAUSSIAN																										
MANUAL	NEGATE STRETCH																										
SAVE STRETCH	RESTORE STRETCH																										
PIXEL VALUE	TRANSECT GRAPH																										
HISTOGRAM	OVERLAY 4 HISTOGRAM																										
SIMPLE LINEAR	NON LINEAR	2 PART LINEAR																									
NONE																											
A																											
B																											
C																											
D																											
E																											
F																											
G																											
H																											
		<table border="1"> <tr> <td colspan="2">WHOLE EXTRACT</td> </tr> <tr> <td colspan="2">PART EXTRACT</td> </tr> <tr> <td colspan="2">LAST ONE</td> </tr> </table>	WHOLE EXTRACT		PART EXTRACT		LAST ONE																				
WHOLE EXTRACT																											
PART EXTRACT																											
LAST ONE																											
		<table border="1"> <tr> <td>ALL</td> <td>RED</td> </tr> <tr> <td>GREEN</td> <td>BLUE</td> </tr> </table>	ALL	RED	GREEN	BLUE																					
ALL	RED																										
GREEN	BLUE																										
		<table border="1"> <tr> <td>STATUS</td> <td>HELP</td> <td>RETURN</td> <td>IMAGE</td> </tr> </table>	STATUS	HELP	RETURN	IMAGE																					
STATUS	HELP	RETURN	IMAGE																								

INPUT FOR COMMAND OR QUIT

Fig 13 The 'Contrast Stretch, Histograms, Intensities' page

AUTO LINEAR

SUMMARY

This provides a simple automatic linear contrast stretch as in Fig 11 (B).

COMMAND IN DETAIL

The first command instructs the system which area to use to produce the histogram. 'Whole Extract' indicates that the whole scene displayed is to be used to provide the histogram information. If a 'Part Extract' is selected, a rectangle appears. The rolling ball is used to move the rectangle to the area of interest and after pressing the 'CHNG' button, the rolling ball can change the size of the rectangle. On pressing 'INPUT' the system produces the three histograms for the images connected to the red, green and blue guns of the television. GEMSTONE finds the points P and Q in the histogram diagram in Fig 9. This is so arranged that about 1% of the data lies below P, and 1% lies above Q. These small amounts of data are all compressed into level 0 and level 255 respectively. Three transfer functions are produced similar to Fig 11(B) and put into their respective LUT boxes, resulting in the stretched picture appearing on the screen.

The next time a contrast stretch is applied, when the user has to specify the area for the production of the histogram, GEMSTONE suggests 'Last One'. If 'Whole Extract' was specified previously, it will be used again, and similarly if 'Part Extract' was specified then it will be used again without having to redefine the area.

AUTO 2 LINEAR

SUMMARY

This provides a simple two step linear contrast stretch as illustrated in Fig 11 (C), where the median of the input data is set to level 127 or mid-grey on the output.

COMMAND IN DETAIL

The operation of this command is exactly like 'AUTO LINEAR', except that three transfer functions of the form shown in Fig 11 (C) are loaded into the three LUTs.

AUTO EQUALISE

SUMMARY

The histogram shown in Fig 9 had a large peak in it. This contrast stretch is such that it produces an histogram of the output data from the LUT boxes having a flat top. ie an equidensity or rectangular distribution.

COMMAND IN DETAIL

The operation is exactly like 'AUTO LINEAR', except that three transfer functions of the form shown in Fig 11 (D) are used.

AUTO GAUSSIAN

SUMMARY

As for 'AUTO EQUALISE' a complicated transfer function is produced. This time the resulting histogram has the shape of a standard 'Gaussian' distribution.

COMMAND IN DETAIL

The operation is exactly like 'AUTO LINEAR', except that three transfer functions of the form shown in Fig 11 (D) are used.

MANUAL

SUMMARY

Should the user consider that none of the automatic contrast stretches gives the enhancement required, even with the facility to choose a limited area for the statistics, then the transfer function can be adjusted manually to produce a range of simple linear and complex stretches.

COMMAND IN DETAIL

The user is first asked whether the contrast stretch is to be applied to all colours (as in the case of a black and white image), or to a single colour. If different stretches are to be applied to each colour, then they have to be applied sequentially. The next choice is the type of stretch to be used, namely, 'Simple Linear', '2 Part Linear', or 'Non Linear'.

'Simple Linear' results in a transfer function as shown in Fig 11 (B). Moving the rolling ball from side to side causes the top of the transfer function to move from side to side. A small diagram at the bottom right of the television screen shows the result of this operation, the number displayed being the x co-ordinate of the top of the transfer function. This would be 180 in the case of Fig 11 (B). If the 'CHNG' button is pressed, the rolling ball controls the lower end of the transfer function, and the diagram at the bottom of the screen shows the movement of this lower end of the graph, and the x co-ordinate of this end. This would indicate 30 in the case of Fig 11 (B). A further pressing of the 'CHNG' button causes the rolling ball to control the top of the transfer function again, and so on.

'2 Part Linear' starts off like 'Simple Linear'. Initially the rolling ball controls the top end of the transfer function, then, after pressing the 'CHNG' button, the bottom end, but after the next press of 'CHNG', it controls the mid-point of the output. This is shown in Fig 11 (C) where the user can control the point with co-ordinates of 80 on the input and 127 on the output. Moving the ball from side to side, causes the x co-ordinate to move whilst leaving the y co-ordinate fixed at 127. Subsequent pressings of the 'CHNG' button cause the control of the rolling ball to cycle round the three functions.

'Non Linear' stretching is normally applied after one of the other two manual stretches. A new unity transfer function is shown in the small diagram on the television screen, and this function should be considered as being applied in addition to the previous contrast stretch. Initially the user can control the gradient of the line through the mid-point of the function. Pressing the 'CHNG' button allows the user to move this mid-point in both the x and y directions using the rolling ball and GEMSTONE continuously produces smooth transfer functions to fit these constraints. At the side of the diagram are three numbers. The upper two give the co-ordinates of the point at which the slope is being controlled (initially the mid-point 128,128), and the lower number indicates of the slope at that point. 0 represents a slope of 45 degrees, 127 represents the steepest slope (about 90 degrees is parallel to the y-axis), and -127 represents the shallowest slope (about 0 degree is parallel to the x-axis).

Before using this function, it is worthwhile storing any stretch that has been applied, using 'SAVE STRETCH' (p31), since on completion of this command the LUTs are modified. Thus, if a further non-linear stretch were produced, it would be applied to the new LUT. Restoring the previous stretch would therefore result in the same starting point, before a new non-linear stretch was applied.

NEGATE STRETCH

SUMMARY

This command produces the negative of the current stretched image.

COMMAND IN DETAIL

It is best to consider this command as producing a separate transfer function in addition to the main stretch rather like the manual 'Non Linear'. This function will transfer data at level 255 to 0, and data at level 0 to 255, etc. To remove this negative, the 'Negative Stretch' command should be repeated, since two negatives make a positive.

SAVE STRETCH

SUMMARY

It is possible to save up to 8 contrast stretches using this command, so that at some later time, they can be re-applied to an image.

COMMAND IN DETAIL

Up to 8 stretches can be stored, and the only command is to label the stretch from A to H. A line of up to 70 text characters can be inserted at the terminal, and this is stored as a descriptor of the stretch. The contents of the 3 LUTs are transferred to the host computer, for later recall to GEMS or to a background program which allows the stored contrast stretch to be applied to a complete image.

RESTORE STRETCH

SUMMARY

This command allows the user to re-load a previously stored contrast stretch into the LUTs.

COMMAND IN DETAIL

The only secondary command is to specify the contrast stretch to be restored. If 'NONE' is specified, then the LUTs are set to the unity transfer function, so that no contrast stretch is applied to the image. It will be found that the command cycles round all the stored stretches. So, for example, having restored 'NONE', the next press of the 'INPUT' button will show the chapter page with the cursor over 'RESTORE STRETCH'. A further press will move the cursor to stretch 'A', and the next press will result in the image being displayed with stretch 'A'. Three more presses of the 'INPUT' button will result in the image being displayed with contrast stretch 'B', and so on. Thus the user can very quickly examine a number of pre-stored stretches to find the most suitable one.

If a line of description was supplied when the stretch was saved, using 'SAVE STRETCH', this text will appear on the terminal screen.

PIXEL VALUE

SUMMARY

The intensities in the three colours, and the co-ordinates of a pixel may be displayed on the screen. The numbers presented are the intensities in the stores, is not modified by the LUTs.

COMMAND IN DETAIL

The rolling ball or the buttons round the 'mode' button (see Chapter 4), can be used to move the cursor over the area of interest. Below the cursor cross is a continuously updated set of figures giving, on the top line, the co-ordinates of the centre of the cursor in x and y, relative to the top left of the original image, on the next line the co-ordinates relative to the top left of the screen, and on the bottom line, the intensities of the pixel at the centre of the cursor cross, in the three colours. By looking at the right hand buttons on the control panel, the user can determine which store is connected to which colour gun. These buttons may be pressed while this process is running, so that it is possible to display the pixel values in four stores very quickly. The figures presented, are held in overlay plane 3. If the 'CHNG' button is pressed, then the pixel at the centre of the cursor cross is copied into overlay plane 2. If the 'CHNG' button is pressed again before the cursor is moved, then the co-ordinates of the pixel relative to the top left of the screen are also stored in overlay plane 2. A third pressing of 'CHNG' stores the other co-ordinates, and the fourth pressing stores the intensities. In this way a series of points can be labelled with co-ordinates with or without intensities, and overlayed on the image. To terminate the command, the 'INPUT' button is pressed.

If the image has been geometrically transformed to a map base, then the position of the cursor in map co-ordinates will appear on the computer terminal.

TRANSECT GRAPH

SUMMARY

A transect across an image can be defined, and a plot displayed of intensity along the transect in one or three bands.

COMMAND IN DETAIL

Having selected one or all three colours, GEMSTONE draws a line on the screen, and positions the cursor at one end. The cursor may be moved with the rolling ball or the buttons round the 'mode' button, to place the cursor over the start point for the transect. To assist with the operation, the co-ordinates of the cursor are updated on the computer terminal. After pressing the 'CHNG' button, the final point of the transect can be selected. Further pressings of the 'CHNG' button will allow the user to reset the start point and the end point cyclically. When the 'INPUT' button is pressed, the graph of intensity along the transect is displayed in one or three colours depending on the choice made earlier. The y (or vertical) axis is linear in intensity from 0 to 255. The x axis gives the position along the transect and corresponds to the distances along the maximum projection of the transect line onto the x or y axis. Thus if the transect line is close to horizontal, the scale corresponds to the projection of the transect onto the x axis. A scale is also annotated on the transect line and there is a larger index at the end of the transect line corresponding to the origin of the transect graph. Whilst this command is being exercised, the current length (true length, not 'taxi-cab' distance) and angle of the line is given on the VDU.

If the command is executed again, the user will find that the end point is in the same position. Hence a series of transects radiating from a point may be investigated. For a contiguous sequence of transects, the start point should be put over the previous end point by shrinking the line to one pixel, and a new end point specified.

HISTOGRAM

SUMMARY

Using this command, histograms can be displayed on the television screen. These can be for the data before or after stretching. If the histograms are for individual colours, the transfer function is also displayed.

COMMAND IN DETAIL

It is first necessary to specify the area of the scene to be used for producing the histograms, ie 'Whole Extract', 'Part Extract', or 'Last One' as described in 'AUTO LINEAR' contrast stretching. If the user next chooses that 'All' three colours are to be examined, then GEMSTONE will produce the corresponding three histograms of the data after stretching, in overlay plane 3, and a further three histograms before stretching in overlay plane 2. Plane 3 is displayed, but the user can switch in and out planes 2 and 3 with the buttons on the extreme left as described in section 4. If, however, the choice was that only one colour (ie 'Red', 'Green', or 'Blue') be examined, then GEMSTONE will put two histograms in overlay plane 3 and the transfer function in overlay plane 2. The two histograms correspond to before and after stretching. Both histograms and the transfer function are displayed, but the user may again switch the planes in and out. All histograms have the corresponding mean, median, and standard deviation, beside them. Note that the vertical axis is scaled to make the largest column equal to one and that the scale is non-linear. This allows lower level detail to be examined. Note also that a vertical line is not drawn for the y-axis. If there appears to be such a line, it is actually the column for pixels of intensity 0. ie there are a lot of totally black pixels in the picture. More statistical data also appear on the VDU.

To speed up the process, the 'Whole Extract' is sub-sampled, by selecting one pixel in four and one line in four. Such a sample is still statistically reasonable with 16,384 items. When 'Part Extract' is used, integer sampling is again done so that there are less than 128 samples per line and less than 128 lines. It should be noted that for a computer generated, regularly patterned image, this procedure may not produce exactly the correct histogram. In these obscure cases, 'OVERLAY 4 HISTOGRAM' should be used.

OVERLAY 4 HISTOGRAM

SUMMARY

The principle is the same as 'HISTOGRAM' above, but the area for the statistics is defined by the contents of overlay plane four.

COMMAND IN DETAIL

By use of the graphics routines in 'Text and Graphics' or in 'Classifiers, Copy Bit Plane', or in 'Density Slice, Measure Areas', a mask is put into overlay plane four, such that only where the plane is at level one are the pixels in the image plane(s) included in the statistics. The operation of the command is the same as 'HISTOGRAM', but there is no sampling done by the system.



LANDSAT 4

TM BAND 4



SLICES:

FIG 14 DENSITY SLICING EXAMPLE

5.5 Density Slice, Measure Areas

Density Slicing is another photographic operation that is used for a much wider range of possibilities and problems. The original idea was that every pixel in a black and white image could be colour coded. For example, all the pixels from black to dark grey could be made yellow, all those from dark grey to mid-grey could be made green, etc. Since, as was mentioned in the pre-amble to contrast stretching (section 5.4), the number of intensity levels in a digital image is very large compared to the capabilities of the eye and film, this facility is necessary to enable the user to examine the subtle differences in tone in an image. The ease and the precision with which the density slice can be applied in a digital image is in marked contrast to the difficulty and imprecision of the photographic system.

It may be found in an image that every pixel between one level and another is associated with a particular type of ground cover. For example, in many images water appears very dark, and a simple density slice may isolate all the water areas. This is an example of a simple 'Classification', and the water would be a 'Class' of ground cover. In section 5.8 these ideas are developed further using more than one band of an image. However, this simple idea should be understood before the more complex classifiers are examined.

Fig 14 illustrates the density slice operation applied to the same Portsmouth scene as in Fig 12. The first slice in blue from level 0 to 13, colours in the water area, ie, this simple process can be seen as classifying this area as water, very successfully. The second slice in dark red is from level 14 to 21 and picks out mud/wet sand areas above the water line. However, there are many other areas within the town of Portsmouth at the bottom left, that are also picked out, and this is therefore a relatively inaccurate classification of mud/wet sand.

The process is implemented using LUTs in the same way as for contrast stretching (section 5.4), and again, the contents of the stores are not modified. To have a sliced image in the stores, the original requires to be copied to three stores, one through the red LUT, one through the green, and one through the blue, using the 'COPY STORE TO STORE' command (p21).

MANUAL SLICE	8-DIV	BLACK	RED	GREEN	YELLOW
SAVE SLICE	16-DIV	BLUE	MAGENTA	CYAN	WHITE
RESTORE SLICE	8-BIT	GREY	BROWN	GOLD	ORANGE
BLACK IMAGE	16-LEVEL	PURPLE	DARK BLUE	DARK GREEN	DARK RED
NORMAL IMAGE	NONE	USER COL 1	USER COL 2	USER COL 3	USER COL 4
PIXEL COUNT	A	1			
OVERLAY 4 PIXEL COUNT	B	2			
CHOOSE COLOUR	C	3			
	D	4			
	E	5			
	F	6			
	G	7			
	H	8			
		9			
		10			
		11			
		12			
		13			
		14			
		15			
		16			
		END			
		STATUS	HELP	RETURN	IMAGE

INPUT FOR COMMAND OR QUIT

Fig 15 The 'Density Slice, Measure Areas' page

MANUAL SLICE

SUMMARY

On a black and white image, it is possible to apply up to 16 density slices. These slices can have gaps between them. For each slice, there are 16 fixed and four user definable colours to choose from.

COMMAND IN DETAIL

As 16 slices have to be identified, it is suggested that the first slice is number 1. A colour is next suggested for this first slice, but it can be changed to any of the other 19 if desired. Having chosen the colour, the image is displayed with a scale ranging from 0 to 255, at the bottom of the screen. If the rolling ball is moved to the right, a line will appear progressing along the scale, and the darkest parts of the image will be sliced in the chosen colour. The slice operates up to and including the intensity at the end of the line.

Where there is difficulty reading the scale, the zoom facilities and the panning control (see section 4) can be used to enlarge the small region of the scale required. If a density wedge had been added to the image before slicing (see 'ADD WEDGE', p21), then the slice would be observed moving along the wedge. Not only can it be helpful in following what is happening for this one slice, but later when there are several, this wedge keeps a visual record of the relative and absolute positions of the slices and their colours.

If the 'CHNG' button is pressed, the rolling ball now controls the bottom limit of the slice, keeping the top limit fixed. Subsequent pressing of the 'CHNG' button cause control of the rolling ball to cycle round these two operations.

On pressing 'INPUT', the user is returned to the menu where it is suggested that a second density slice be applied and the same sequence of commands is followed. This process can be repeated for up to 16 slices or can be stopped by moving to 'End'. It is possible to change the colours of the slices after they have been specified, and even after 'End', since the slices themselves are not reset on the next execution of 'MANUAL SLICE', but control continues from the previous settings.

SAVE SLICE

SUMMARY

This command can be used to save up to 8 density slices, for later application to an image.

COMMAND IN DETAIL

The first command is to label the slice from A to H. A line of up to 70 text characters can be inserted at the terminal, and this is stored as a descriptor for the slice. The contents of the 3 LUTs are then copied into the host computer memory.

RESTORE SLICE

SUMMARY

This command allows the user to re-load a previously stored density slice into the LUTs.

COMMAND IN DETAIL

The only secondary command is to specify the name of the density slice to be restored. If 'NONE' is specified, then no slice is applied. As described in 'RESTORE STRETCH' (p31), the command cycles round all the stored slices. Thus if one has been restored, then three presses of the 'INPUT' button will result in the next slice being applied. At the end of all the slices, the next in the cycle is 'NONE'. If text was stored at the same time as the slice, it will appear on the terminal screen.

This command also allows the user to apply four standard slices, three being for the display of classified data. As a result of the classification process (section 5.8), a store may contain one class in the first bit plane of the store (see section 2 for details of the bit planes), a second class in the second bit plane, etc. The secondary commands '8-div', and '8-bit', permit the display of these classifications in different ways.

Suppose there are three classes in bit planes 1,2 and 3, with the other bit planes set to zero. Where there is an area which is only in class 1, bit plane 1 will be set and planes 2 and 3 will be zero. Hence the value of the pixels within the area will be 10000000 in binary or 128 in decimal. (It may be helpful to refer to the bit plane diagram, Fig 2, p6). Similarly, for an area where only class 2 exists, the value of a pixel will be 01000000 or 64. Where a pixel is in both class 1 and class 2, its value will be 11000000 or 192. This argument can be extended, and the results for three classes sliced using '8-div', are summarised in the first two columns of Table 2.

<u>Pixel value</u>	<u>Pixel class</u>	<u>Slice colour</u>
224-255	1 and 2 and 3	white
192-223	1 and 2	yellow
160-191	1 and 3	magenta
128-159	1	red
96-127	2 and 3	cyan
64-95	2	green
32-63	3	blue
0-31	none	black

Table 2 Pixel value, class and colour for the '8-div' slice

This command therefore permits the display of three classes together with all the overlap regions which are shown in different colours. If there is any doubt about a colour, (eg the colour cyan) a wedge added to the image ('ADD WEDGE', p21) will display them, in order. Where there is difficulty in understanding this process, the user can

simply restore '8-div', and identify the classes using the table.

If there are two classes there are only four possibilities for display, ie no class, class 1, class 2, both classes. In the case of three classes there are 8 possibilities as described, and these can be displayed with 8 slices. However the number of possibilities doubles each time a class is added and so for 8 classes there are 256 combinations. Obviously density slicing cannot show this information, or more to the point, the human cannot absorb it. Suppose there are 8 classes in the 8 bit planes arranged in order of importance with the most important class being in bit plane 1. If an area is in class 1, then the value of the pixels will be 128 as before. When other classes also lie within the area, the value of the pixels will be in the range 128 to 255. If class 1 is the most important, a slice from 128 to 255 will show all occurrences of class 1 irrespective of other classes. Similarly, a slice from 64 to 127 will indicate all areas of class 2 irrespective of classes 3 to 8 etc. The '8-bit' slice will display 8 classes in priority order as listed in Table 3.

<u>Pixel value</u>	<u>Pixel class</u>	<u>Slice colour</u>
128-255	1	red
64-127	2	green
32-63	3	blue
16-31	4	yellow
8-15	5	cyan
4-7	6	magenta
2-3	7	purple
1	8	dark green

Table 3 Pixel value, class and colour for the '8-bit' slice

The '16-div' slice is a general purpose one. Whereas '8-div' divided the intensity range into 8 equal steps, '16-div' divides it into the 16 equal steps listed in Table 4.

<u>Pixel value</u>	<u>Slice colour</u>	<u>Pixel value</u>	<u>Slice colour</u>
240-255	white	112-127	green
224-239	grey	96-111	dark green
208-223	red	80-95	cyan
192-207	dark red	64-79	blue
176-191	brown	48-63	dark blue
160-175	orange	32-47	magenta
144-159	gold	16-31	purple
128-143	yellow	0-15	black

Table 4 Pixel value and colour for the '16-div' slice

The '16-level' slice sets the first 16 intensity levels above 0, to the colours given in Table 8 (p107), and is used to re-display a maximum likelihood classification.

BLACK IMAGE

SUMMARY

This command can be used to make the background black when slicing an image. There are no secondary commands.

NORMAL IMAGE

SUMMARY

This command simply restores the image after using 'BLACK IMAGE'.

PIXEL COUNT

SUMMARY

This command enables the area of each slice to be written on the VDU.

COMMAND IN DETAIL

The total area is printed on the computer terminal, followed, for each slice and the unsliced area (UN), by the number of pixels, the percentage of the total area, and, if the scene has been geometrically transformed, the area in the units of the map (generally square metres). For a scene that has not been transformed, the number of pixels is printed again with a minus sign. If the area covered by one pixel on the Earth is known, then the ground area of a slice can be found simply by multiplying the number of pixels by the area of each pixel, remembering to take into account the sampling used to display the picture. The following table gives the approximate sizes and areas of pixels from some of the more common spacecraft images.

<u>Spacecraft</u>	<u>Sensor</u>	<u>Size</u>	<u>Area (hectare)</u>	<u>Area (acre)</u>
Landsat	MSS	57*79	0.45	1.11
Landsat 3	RBV	19*23	0.044	0.108
Landsat	TM	30*30	0.090	0.22
Spot	Colour	20*20	0.040	0.099
Spot	B/W	10*10	0.010	0.025
Seasat and	SAR	25*25	0.0625	0.15
resampled		50*50	0.25	0.62
images		100*100	1.00	2.47

Table 5 Sizes of pixels in some spacecraft images

OVERLAY 4 PIXEL COUNT

SUMMARY

This is the same as 'PIXEL COUNT' above, except that the area for the count is defined by the contents of overlay plane 4.

CHOOSE COLOUR

SUMMARY

With this command, the colour of the four user colours may be changed.

COMMAND IN DETAIL

Following selection of the user colour to be varied, the display reverts to the sliced scene, with three numbers in the bottom right of the picture. Moving the rolling ball to the right increases the left hand number, ie the red intensity of the slice colour and this can be seen changing on the screen. Moving the ball to the left reduces red. If the 'CHNG' button is depressed, the green intensity can be varied in exactly the same manner, and a small bar moves from the left hand number to the middle, to indicate which component of the slice colour is being varied. Further presses of the 'CHNG' button allow the user to vary the colour components again, in order.

The following table of the fixed colours and the red, green and blue intensities, gives the user an indication of the numbers required to produce a new colour.

Colour	Red	Green	Blue	Colour	Red	Green	Blue
Black	0	0	0	Grey	128	128	128
Red	255	0	0	Brown	80	48	48
Green	0	255	0	Gold	255	192	0
Yellow	255	255	0	Orange	255	128	0
Blue	0	0	255	Purple	128	0	128
Magenta	255	0	255	Dark Blue	0	0	64
Cyan	0	255	255	Dark Green	0	128	0
White	255	255	255	Dark Red	128	0	0

Default user colours:

User Col 1	128	192	0	Lime Green
User Col 2	192	64	0	Rust Red
User Col 3	64	128	128	Petrol Blue
User Col 4	192	64	128	Pinky Purple

Table 6 Primary components of the slice colours.

The description of the colours, particularly the user colours, in Table 6 are only approximate and will differ from user to user and from monitor to monitor.

5.6 Maths Operations

There are a number of operations grouped in this section several of which use 'spatial filtering' techniques. The basic idea behind filtering is that a patch with elements called 'weights' can be convolved with an input image to produce an output image.

If $I_{p,q}$ is the intensity of pixel (p,q) in the input image,

$O_{p,q}$ is the intensity of pixel (p,q) in the output image,

and $W_{i,j}$ is the value of the weight at (i,j) in a $2n+1$ by $2m+1$ patch,

then,

$$O_{p,q} = \sum_{i=1}^{2n+1} \sum_{j=1}^{2m+1} W_{i,j} * I_{p-(n+1)+i, q-(m+1)+j} \quad \text{--- (1)}$$

If the patch is $2n$ by $2m$, then there is no central weight, and the convention within GEMSTONE is that,

$$O_{p,q} = \sum_{i=1}^{2n} \sum_{j=1}^{2m} W_{i,j} * I_{p-n+i, q-m+j} \quad \text{--- (2)}$$

The consequence of this convention is that the image moves upwards by 0.5 of a pixel and to the left by 0.5 of a pixel.

It may be easier to visualise this convolution process with an example.

Suppose there is an image with a small piece like this — where A is the intensity of the pixel at that point etc,

A	B	C	D	E
F	G	H	I	J
K	L	M	N	O
P	Q	R	S	T

and there is a patch of three rows and three lines — where a is the value of the weight at that point etc.

a	b	c
d	e	f
g	h	i

The patch is put on top of the image like this, —

aA	bB	cC	D	E
dF	eG	fH	I	J
gK	hL	iM	N	O
P	Q	R	S	T

and the element with intensity G in the input image is replaced by the sum of all the products inside the patch. ie The output pixel has the value,

$$a*A + b*B + c*C + d*F + e*G + f*H + g*K + h*L + i*M.$$

The patch is then moved along one place to the right —

A	aB	bC	cD	E
F	dG	eH	fI	J
K	gL	hM	iN	O
P	Q	R	S	T

and the input pixel with value H is replaced by

$$a*B + b*C + c*D + d*G + e*H + f*I + g*L + h*M + i*N.$$

This process is repeated for every step along the line. The patch is then moved down one line and the whole process repeated for every step from left to right again, and so on to the bottom of the image. By choosing the weights a,b,c.... correctly, various smoothing, edge enhancement, and destriping filters can be produced. Within this chapter, the weights are fixed by GEMSTONE, but the user can design his own filters using the commands in 'Linear Filters'. (see section 5.7)

[illegible]

INPUT FOR COMMAND OR QUIT

Fig 16 The 'Maths Operations' page

SMOOTH

SUMMARY

This command allows the user to smooth an image along a line, down columns, or over an area. By changing the size of the filter, it is possible to remove high frequencies alone, or both high and mid range frequencies.

COMMAND IN DETAIL

For this filter, all the weights in the patch are equal. Thus in equations (1) and (2) on page 44, W has the same value for all i and j , and W equals one divided by the total number of weights in the patch. The sum of the weights is therefore one, and the overall intensity of the image is left unchanged by the operation. In the example on page 45,

$$a - b - c - d \dots - 1 = 1/9.$$

The user is first asked which store(s) contains the image to be smoothed, then where the result should be placed, and the area of the image over which the filter is to be applied. The next question is the size of the filter, which defines the type of smoothing to be done. Thus a filter size of 3,1 (ie 3 weights along the line and only one line), would remove only high frequency edges running from the top of the image to the bottom. Changing the 3 to a higher number would remove lower frequencies as well. The maximum size in this direction is 512, and such a filter would remove all the information in the image along the lines leaving only the changes from line to line. A filter of size 1,3 (ie 3 weights in one column) would remove high frequency edges running from left to right. Again, increasing the 3 would result in lower frequencies being removed, but in this case the maximum number is 16, to give reasonable speed. For filters other than a column or a line, frequencies will be removed in both directions according to the size of the filter. Having specified the size of the filter, the scene is processed. It should be noted that if a filter is specified with an even number of weights along one or both sides of the patch, then the output image will be shifted upwards by 0.5 of a pixel if there is an even number of weights in the vertical direction, and to the left by 0.5 of a pixel if there is an even number of weights in the horizontal direction.

The following simple example illustrates the operation of a smoothing filter. An idealised image of two fields and one noise pixel, is shown at the top of Fig 17. The intensity along one scan line AA is shown below, and the horizontal scale is marked at the boundaries between the pixels. If the smoothing filter $(1/3, 1/3, 1/3)$, is convolved with the image, the reader can check that the last graph is the resultant intensity along this one line.

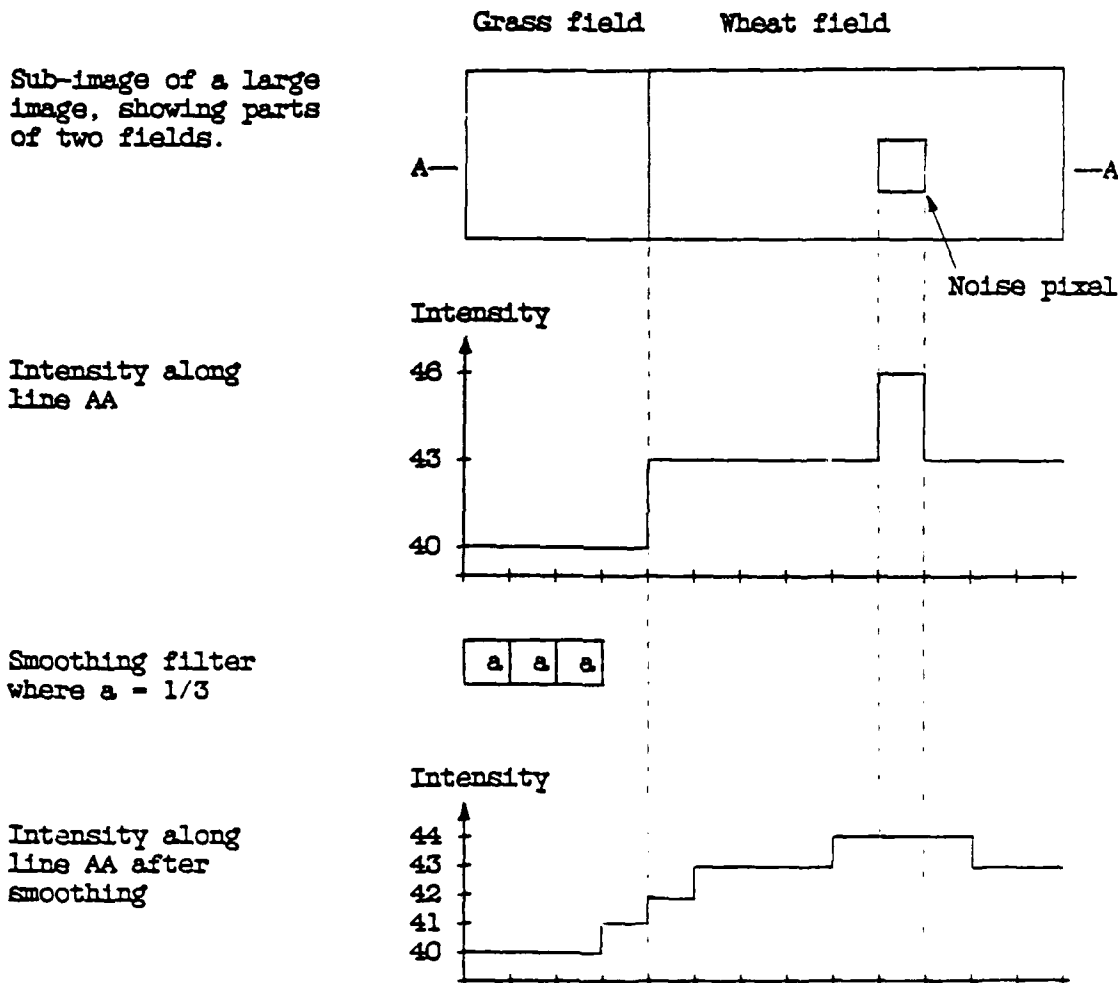


Fig 17 Example of smoothing

The main features to be noted are that the average intensities of the two fields away from the edge and the noise pixel remain unaltered, that the edge has been smoothed, and that the noise pixel has been very much reduced and spread out. Fig 18 shows, on the left hand side, examples of a 3×3 and a 7×7 filter applied to the Portsmouth scene.

ORIG.

3x3
EDGE
ENH.3x3
SMOOTH7x7
EDGE
ENH.7x7
SMOOTH3x3
EDGE
ENH.
REMOVE
LOW

FIG 18 VARIOUS FILTERS APPLIED TO THE PORTSMOUTH SCENE

EDGE ENHANCE

SUMMARY

This command can be used to enhance edges which run vertically in the image, or horizontally, or omnidirectionally, by either increasing the amplitude of the high frequencies, or decreasing the amplitude of the low frequencies. As with smoothing, the size of the filter controls the frequencies to be modified.

COMMAND IN DETAIL

As with the 'SMOOTH' command (p47), the user specifies the store containing the input image, the result store and the area to be processed. The user is next asked to 'increase high' or 'remove low'. In the former case, a patch is produced in which all the weights except the central one are equal to $-1/(T)$, where T = total number of weights in the patch. The central weight is set to $(T-1)/T$, so that the sum of the weights is zero. Thus in the example on page 45,

$$a - b - c - d - f - g - h - i = -1/9 \quad \text{and } e = +8/9.$$

Therefore if the patch is convolved with a uniform area (ie an area with no edges, or $A = B = C \dots$ in the example) the result is zero. If any of the pixels in the image are different from the neighbouring pixels, then this filter produces a non-zero output. It is therefore an edge detection filter. GEMSTONE adds the output from this filter to the original image to produce the edge enhanced result.

In the case of 'remove low', a smoothing filter with equal elements is first applied, as in the 'SMOOTH' command (p47). This smoothing filter leaves low frequencies unaltered, and reduces the high frequencies. If this resultant image is subtracted from the original, the low frequencies of the original will be exactly cancelled by the low frequencies of the smoothed image, but the high frequencies will not be removed. The net resultant image is zero plus and minus the high frequency information. Since GEMS cannot display a negative image, 128 is added to this final result.

Having chosen the type of enhancement, the size of the patch is specified as for the 'SMOOTH' command. If the patch has an even number of weights on one or both sides, then the enhanced edges are moved again, but this time by 0.5 pixel downwards or to the right or both.

The following example shows the effect of an edge enhancement filter, using 'increase high', applied to the image of Fig 17 (p48). Below the original line, is the result of convolving the edge detection filter with the line, and, as shown, there is only a non-zero output where there are edges. Adding to the original image, at the bottom, shows the enhancement of edges.

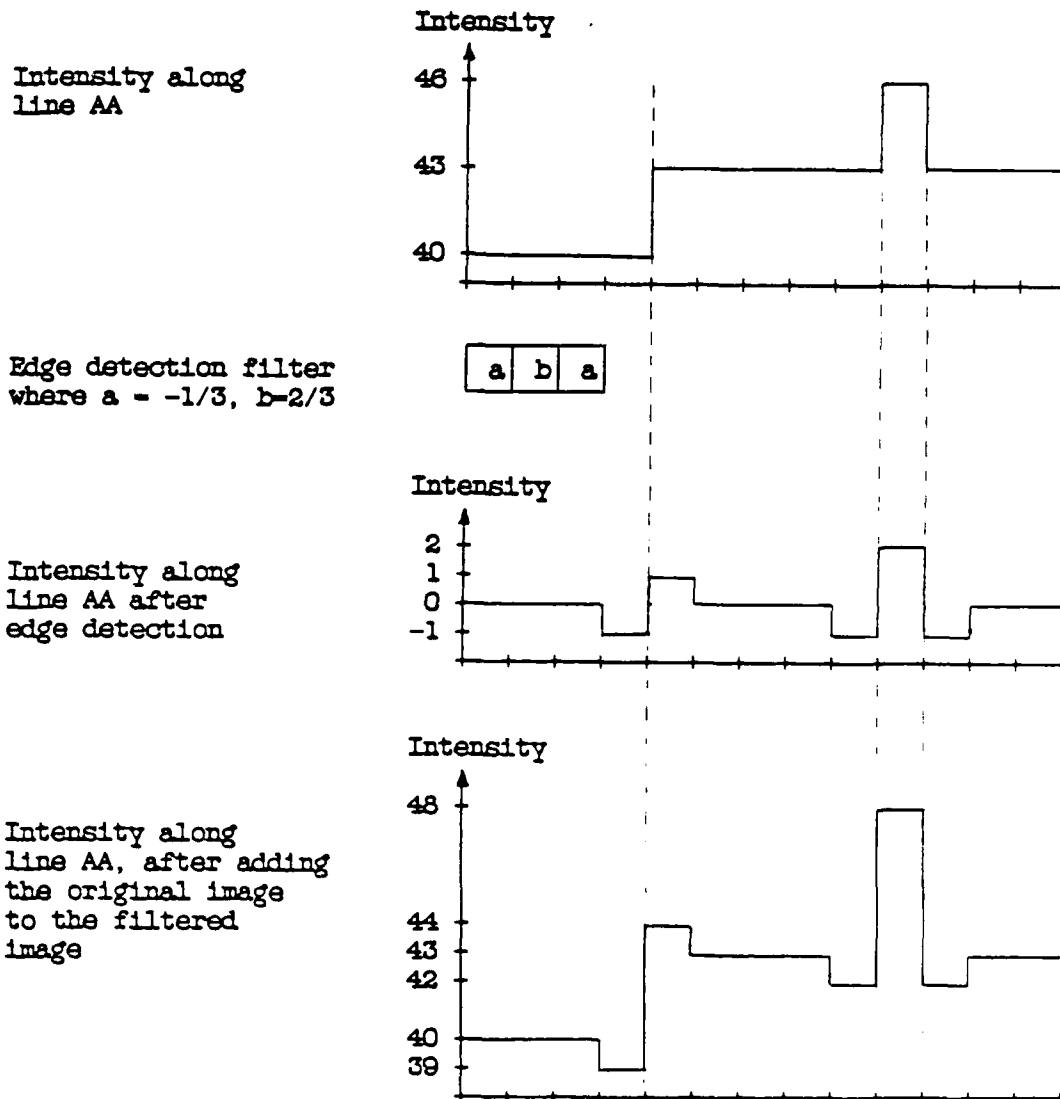


Fig 19 Example of edge enhancement using 'increase high'

In this resultant image, the average intensities of the fields away from the edge remain unaltered, there is a larger contrast change over the edge, but unfortunately, the noise pixel has been enhanced.

Fig 20 shows the effect of passing the edge enhancement filter over the image in Fig 17 (p48), using 'remove low'. Below the original line are the smoothing filter and the result of convolving this with the line. Subtracting this from the original and adding 128 to avoid negative numbers, results in the intensities shown at the bottom.

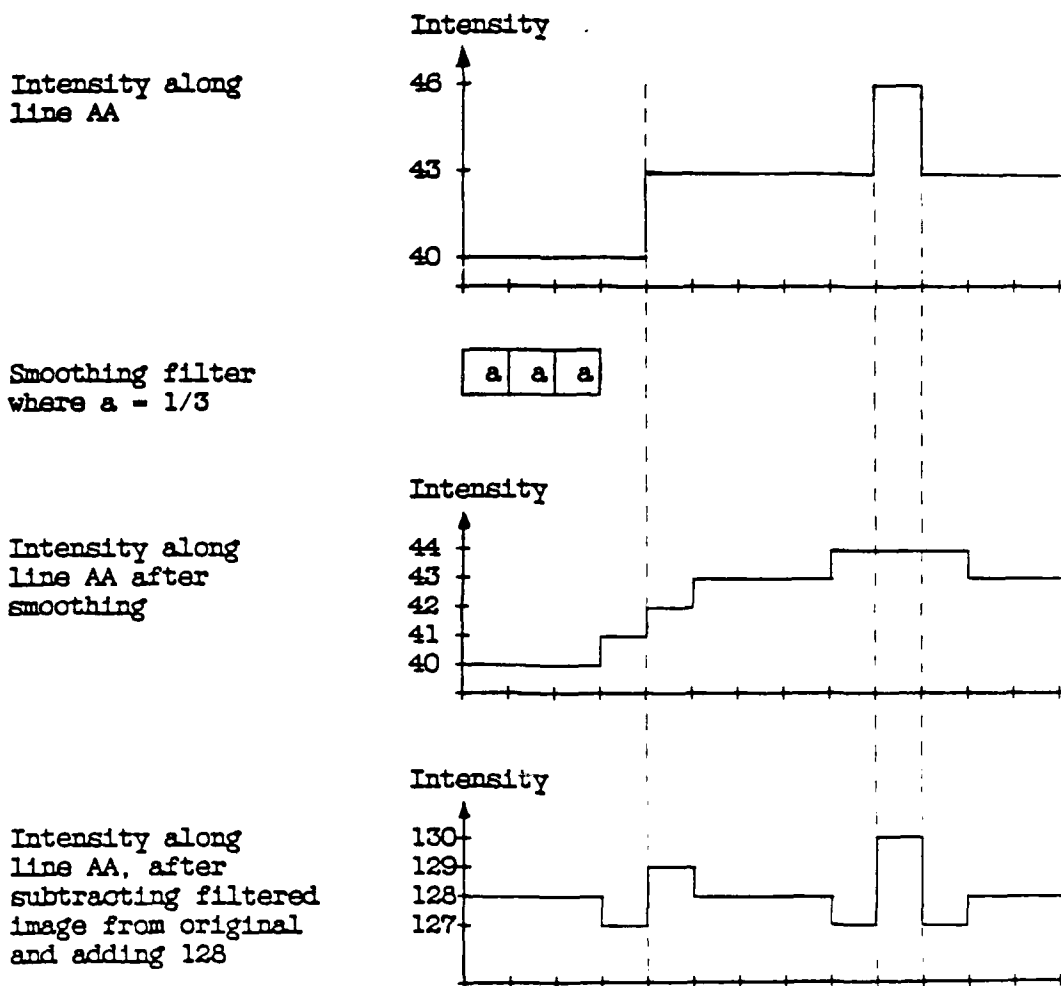


Fig 20 Example of edge enhancement using 'remove low'

In this resultant image, all the low frequency information has been removed with only the edges remaining. Thus a skeleton has been left.

In Fig 18, on the right hand side, a 3×3 and a 7×7 edge enhancement have been applied to the Portsmouth scene. The last image at the bottom right, shows the effect of edge enhancement using 'remove low'.

DESTRIPE

SUMMARY

Many images, particularly from the MSS in the Landsat series of satellites, have a horizontal striping structure. This command permits the partial or complete removal of this structure.

COMMAND IN DETAIL

The process uses the same filtering techniques already described in 'SMOOTH' and 'EDGE ENHANCEMENT' (pp47 - 53), and can be considered in three steps, although as far as the user is concerned, the command itself is only one operation. The first step is to smooth the image using a long single line filter, eg about 50 weights along one line. This results in an image in which most of the detail has been removed along the line but no change has occurred in the vertical direction and thus, if there are stripes in the image they will be more visible. Step two applies an edge enhancement using 'remove low' (see pp51 - 53). Since the low frequency information has now been removed, the image will only contain stripes. Step three subtracts this image of stripes from the original, and therefore produces a 'destriped' image.

Fig 21 illustrates the effect of this command on a Landsat MSS band 5 image of the Venice area, shown in (A). A severe contrast stretch has been used to emphasize the striping in the sea, and thus parts of the land and the lower plume at Y, are saturated. The result of applying the command using a 49*6 filter, is shown in (B) where it can be seen that the stripes have been removed from the sea, and that there is no visible change to the land data. Figs 21 (C) and (D) illustrate two of the steps in the process described above. In (C), a smoothing filter with 49 weights along the line has been applied to the original scene in (A). It will be noticed that most of the detail along the line has been removed, so that the stripes are more prominent, but large features (eg the sea) are still visible. Fig 21 (D) shows the subsequent application of edge enhancement with a 1*6 weight filter using 'remove low'. The grey level differences between large features have been removed leaving essentially the stripes alone. The last step in the process described above was to subtract the image in (D) from the original in (A) giving the result in (B).

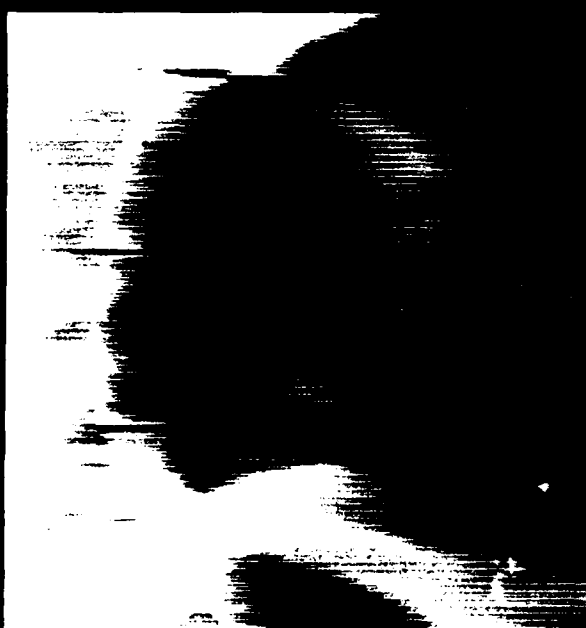
By considering the two plumes at X and Y in Fig 21 (A), and the natural horizontal feature at Z, the detailed effects of this process can be examined. The plume at Y has almost no striping in (A) and none in (B), but plume X has 'acquired' some stripes in (B). Plume Y is large compared to the horizontal smoothing filter (49*1 weights) and therefore the smoothing operation does not cause stripes to spread into the plume area. However, plume X is small and is straddled by the smoothing filter, so that when the centre of the filter is within the plume, part of the filter is outside. This causes some of the information in the striped sea area to be included in the smoothing operation and hence stripes appear in the output image.



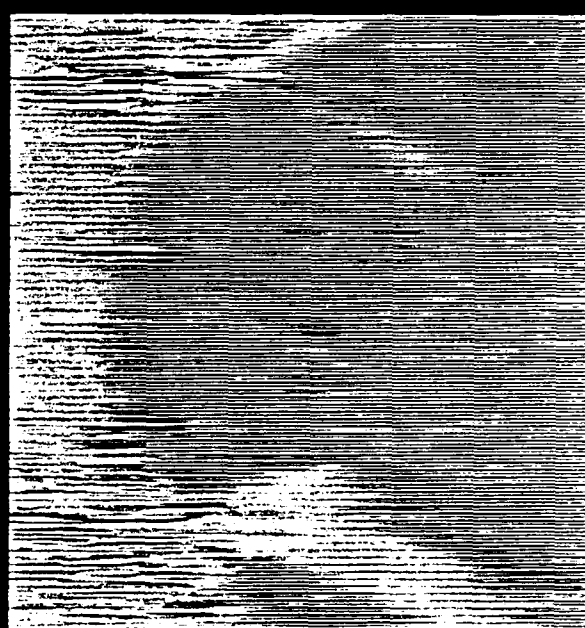
(A) ORIGINAL SCENE



(B) DESTRIPE



(C) SMOOTHED



(D) SMOOTHED AND EDGE ENHANCED

FIG 21 DESTRIPIING EXAMPLE

Reducing the size of the smoothing filter will improve the situation, but if it is made too small, then some of the real image information will be suppressed. For example, the natural horizontal feature at Z, has been slightly reduced in (B) even with 49 elements in the smoothing filter. A much shorter filter would remove the feature at Z altogether.

Similarly, the size of the filter in the vertical direction is a compromise, but this time, it should be as short as possible to minimise the modification of the real image information. It is suggested that the size be such as to encompass one cycle of the striping structure. Thus for a full resolution Landsat MSS image, with its 6 line structure, the filter should also be 6 lines in extent. Note that if the Landsat image has been sampled then the filter should be reduced in proportion. If a sampling of one line on the screen for two in the image has been used, then the filter should be 3 lines in extent (ie $6/2$).

If it is intended that a scene be stretched at any stage, it is strongly recommended that the destriping be done after the stretching, thereby minimising the quantisation noise.

If the user is not sure how the image has been modified by the destriping operation, then simply subtracting the result from the original, using the commands in 'Arithmetic' (see section 5.10), will show the differences as in Fig 21 (D).

To operate the command, the user enters the store(s) to be destriped, the store(s) for the result, whether all or part of the image is to be processed and the size of the filter in the x and y directions.

MEDIAN SMOOTH

SUMMARY

With this command, the user can smooth an image by replacing a pixel by the median of that pixel and its eight immediate neighbours.

COMMAND IN DETAIL.

This filter is different from the previous ones in that, although a 3×3 window is passed over the image, there are no weights as such. The nine pixels within the window are ranked in order of increasing intensity. The lowest four and the highest four, are discarded, leaving the middle or median intensity. If the filter is applied to the idealised image of Fig 17, the result is as shown in Fig 22.

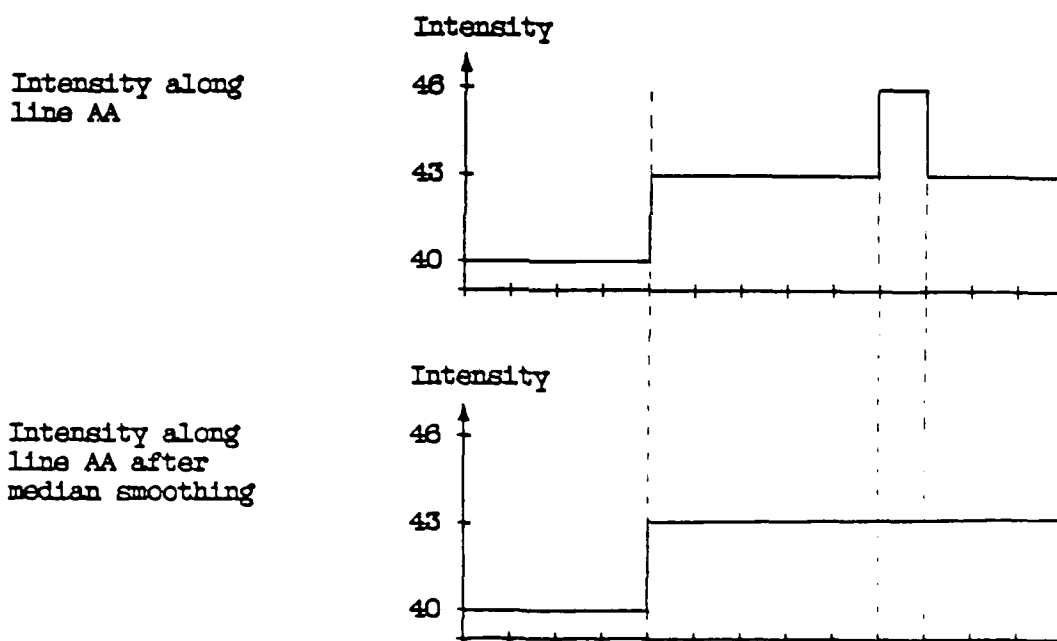


Fig 22 Example of median smoothing

The 'noise' pixel has been totally removed, and the edge has been preserved exactly. This might be considered the 'perfect' filter. However, had the 'noise' pixel been a point reflector in the field, this filtering operation would have destroyed information. The use of this non-linear process should be carefully monitored, by examining the image produced when the resultant image is subtracted from the original.

Due to the high computational load on the system, this process has been restricted to a 3×3 window, and it has been optimised to give a reasonable performance for this one case. This filter can be applied several times to remove small groups of noise pixels.



(A) ORIGINAL WITH NOISE

(B) MEDIAN SMOOTH



(C) MEDIAN SMOOTH. THREE TIMES

(D) 5*5 SMOOTH

FIG 23 MEDIAN SMOOTH EXAMPLE

Fig 23 illustrates the use of a median filter. The original image in (A) is a Landsat 4 Thematic Mapper scene of the Mississippi River, but in the top right quadrant 'noise' in the form of numerous white pixels, has been added. Applying the median filter removes nearly all of these noise pixels as illustrated in (B). A further two applications of the filter results in the removal of all this noise as shown in (C). Using a 5*5 smoothing filter produces an image in (D), which looks broadly similar to (C), but the noise has not been removed completely, and more of the real image information has been destroyed. In fact, since the noise has been smoothed as well, the whole of the top right quadrant is brighter. Slicing (section 5.5) or classifying (section 5.8) the image in (D) would give incorrect results in the top right quadrant due to the increased brightness in the smoothed image. The median filter produces neither of these defects in this case. Synthetic aperture radar images have 'speckle' which has a similar appearance to this noise, and it may be that a median filter would also produce better results than a smoothing filter.

If the image in (A) were classified (section 5.8), then an individual field would generally not come out as one homogeneous class. Careful examination of the image reveals that there is a large variation in the tone within one field, due to either missing crops or tracks through the field, etc. For a general land use classification, this level of detail would be excessive. The images in (B), (C) and (D) show a reduction in this detail, but the following points should be noted :

1. Median filtering preserves sharp edges. Smoothing does not.
2. Small inconsistent areas within an otherwise homogeneous region, are replaced by a median filter. Smoothing will subdue such areas, but will not completely eliminate them.
3. Median filtering will only produce those pixel values in the output image, which existed in the original. Smoothing can introduce a whole range of new values. This is most easily seen where there is a large intensity change over a boundary, such as at the banks of the river in Fig 23. Dark land pixels and bright water pixels contribute to a whole range of grey pixels at the boundary, when image (A) is smoothed as in (D). Thus classifiers would produce incorrect results.
4. Median filtering can be applied to a classified scene to remove fine detail. Smoothing of a classified image would produce many errors for the reasons in 3 above.

LINE FIX

SUMMARY

An image can sometimes have a defect where one or more, full or part lines, are missing and have been replaced with spurious data. This command permits the replacement of such lines with the mean of the line above and the line below.

COMMAND IN DETAIL

Every pixel in the image is compared with the mean of the pixel immediately above and below. If the difference between the pixel intensity and the mean is greater than a user supplied 'rejection number', the pixel is replaced with the mean. This process is shown applied to the grey wedge in Fig 24 (A). Two defective lines have been added, the upper one with an intensity of zero, and the lower with an intensity of 150. An intensity scale has been added at the bottom of the wedge. With a 'rejection number' of 40, 'LINE FIX' replaces most of the defective lines as shown in (B). Only in those places where the difference between the defective line and the mean is less than 40, is there no substitution. A smaller number than 40, would remove more of the defective pixels, but would also remove some of the real image data.

Part of the Portsmouth scene shown in Fig 12 has been reproduced in Fig 24 (C), with a simulated defective line at level zero. Using 'LINE FIX' with a 'rejection number' of 40 makes the line almost invisible in Fig 24 (D). Subtracting (D) from (C) illustrates in Fig 24 (E) that the process has removed most of the line with minimal modification of the real image. As in 'DESTRIFE' this command is a compromise, but when used carefully, it can remove the very distracting defective lines with little damage to the image.

Having executed the primary command, the user is asked for the store(s) to be fixed, the store(s) for the result, the area to be fixed, and the 'rejection number' as described above.

20 40 60 80 100 120 140 160 180 200 220 240

(A) GREY WEDGE WITH TWO DEFECTIVE LINES

20 40 60 80 100 120 140 160 180 200 220 240

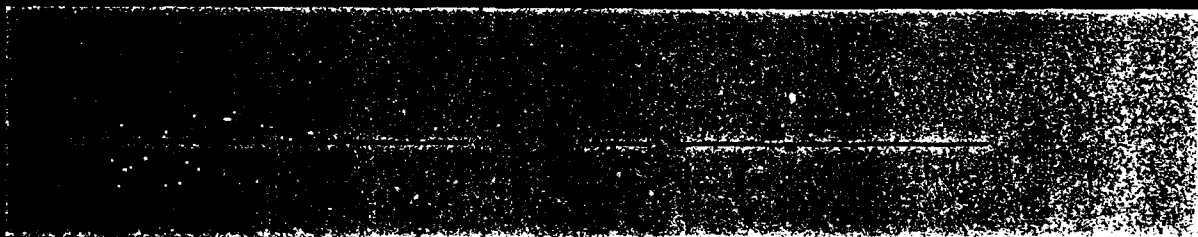
(B) LINE FIX APPLIED TO (A) WITH REJECTION - 40



(C) PART OF PORTSMOUTH SCENE WITH A BLACK DEFECT LINE



(D) LINE FIX APPLIED TO (C) WITH REJECTION - 40



(E) IMAGE (C) MINUS IMAGE (D)

FIG 24 EXAMPLE OF LINE FIX

SCATTER DIAGRAM

SUMMARY

Up to 4 two-dimensional scatter diagrams can be displayed where the number of occurrences at a point is indicated by the intensity of the displayed pixel. If this facility is used in conjunction with 'Density Slice' (section 5.5), two-dimensional histograms can be displayed in colour.

COMMAND IN DETAIL

It is sometimes instructive to examine how one spectral band varies relative to another. This is done for each pixel by plotting the intensity in one band against the intensity in another. If the two bands are highly correlated, then where a pixel is bright in one band, it will be bright in the other, etc, and so the plot will be approximately a straight line through the origin at 45 degrees to the axes. Where the data are less correlated, the plot is spread out or 'scattered'. The more scattering the lower the correlation, and the more information can be extracted from the pair of bands when processed together. It may be found that points are not simply scattered, but are in fact collected in groups or 'clusters'. These clusters often correspond to a particular type of feature within the image. e.g. water, forestry, sandy beaches, etc. If one particular cluster is found to correspond to forestry for example, then it should be possible to put a box (or more complicated envelope) round the cluster in the scatter diagram, and to highlight in the image all the pixels that lie within that box. This is the basis of all 'classifiers' including box (parallelepiped), cluster, maximum likelihood, etc.

The first secondary command specifies the stores to be displayed along the x and y axes of the scatter diagram, together with the store to be used to hold the diagram itself. Since a scatter diagram has axes from 0 to 255, it is possible to put 4 such diagrams on a screen which is 512 elements square. Therefore, the quadrant in which the diagram is to be displayed, is required. Finally, the image area to be used is given, and the diagram is then written into the specified store with the scale held in overlay plane 3. If the exact co-ordinates of a point in the diagram are required, then 'PIXEL VALUE' within 'Contrast Stretch, Histograms, Intensities' (see p32) can be used to find these, remembering that the bottom left of the diagram has co-ordinates 0,0 and the top right 255,255. The intensity of a point in the diagram is a measure of the number of pixels at that point with the largest number scaled to 255. Since the maximum number of pixels at a point may be much larger than the smallest number (other than zero), a non-linear scale is desirable. In fact a square root scale is used. If the maximum number of pixels at a point is N, this point would be at level 255, and so if another point had $N/2$ pixels, it would have a level of $255/(\text{square root of } 2)$. i.e. 180. Considering this the other way round, if a point is at level 64, then the number of pixels at this point is $(64/255)$ squared times the number at the maximum. i.e. $1/16$ of the maximum.

PRINCIPAL COMPONENTS

SUMMARY

This command applies the standard principal components analysis process to between 2 and 16 stores. This process is also known as the Karhunen-Loeve (K-L), Hotelling, or eigenvector transformation.

COMMAND IN DETAIL

If $I_{1,j}^k$ is the intensity of pixel 1 in row j in band k,

$E(I)_k$ is the expected value (or mean) of the intensity in this band,

N is the total number of pixels, and n is the total number of bands, then,

$$E(I)_k = \frac{\sum_{1,j} I_{1,j}^k}{N} = M_k, \quad E(I \cdot I)_{p,q} = \frac{\sum_{1,j} I_{1,j}^p \cdot I_{1,j}^q}{N}$$

$$\text{and } \sigma_k^2 = E(I)_k^2 - M_k^2 = \frac{\sum_{1,j} (I_{1,j}^k)^2}{N} - M_k^2.$$

The covariance matrix C of this image is,

$$C = \left\{ C_{ij} \right\} = \begin{pmatrix} \sigma_1^2 & E(I \cdot I)_{2,1} - M_2 M_1 & \dots & E(I \cdot I)_{n,1} - M_n M_1 \\ E(I \cdot I)_{1,2} - M_1 M_2 & \sigma_2^2 & \dots & E(I \cdot I)_{n,2} - M_n M_2 \\ \vdots & \vdots & \ddots & \vdots \\ E(I \cdot I)_{1,n} - M_1 M_n & E(I \cdot I)_{2,n} - M_2 M_n & \dots & \sigma_n^2 \end{pmatrix}$$

and the corresponding correlation matrix S is,

$$S = \left\{ S_{ij} \right\} = \frac{\left\{ C_{ij} \right\}}{\sigma_i \sigma_j}$$

Generally, there is some correlation between the bands of a multispectral image, in which case the matrix S is not diagonal. Principal components analysis requires a transformation of the original data so that the matrix S is diagonal. This transformation matrix U is composed of rows of the eigenvectors of the covariance matrix. First, the eigenvalues are found by solving the characteristic equation

$$\det(C - \lambda I) = 0.$$

The eigenvalues are the roots of this equation, i.e. $\lambda_1, \lambda_2, \lambda_n$.

For each eigenvalue (λ_i), there is an eigenvector (u_i) defined by,

$$C u_i = \lambda_i u_i \quad i = 1, 2, \dots, n.$$

The transformation matrix is composed of these eigenvectors arranged in rows and so if A, B, C , etc are the new spectral bands then,

$$\begin{pmatrix} A \\ I \\ ij \\ \\ B \\ I \\ ij \\ \\ \vdots \\ \\ N \\ I \\ ij \end{pmatrix} = U \begin{pmatrix} 1 \\ I \\ ij \\ \\ 2 \\ I \\ ij \\ \\ \vdots \\ \\ n \\ I \\ ij \end{pmatrix} = \begin{pmatrix} u_1 \\ \\ u_2 \\ \\ \vdots \\ \\ u_n \end{pmatrix} \cdot \begin{pmatrix} 1 \\ I \\ ij \\ \\ 2 \\ I \\ ij \\ \\ \vdots \\ \\ n \\ I \\ ij \end{pmatrix}.$$

The correlation matrix of this resulting image is diagonal and hence the bands are uncorrelated.

The following example illustrates this mathematical operation.

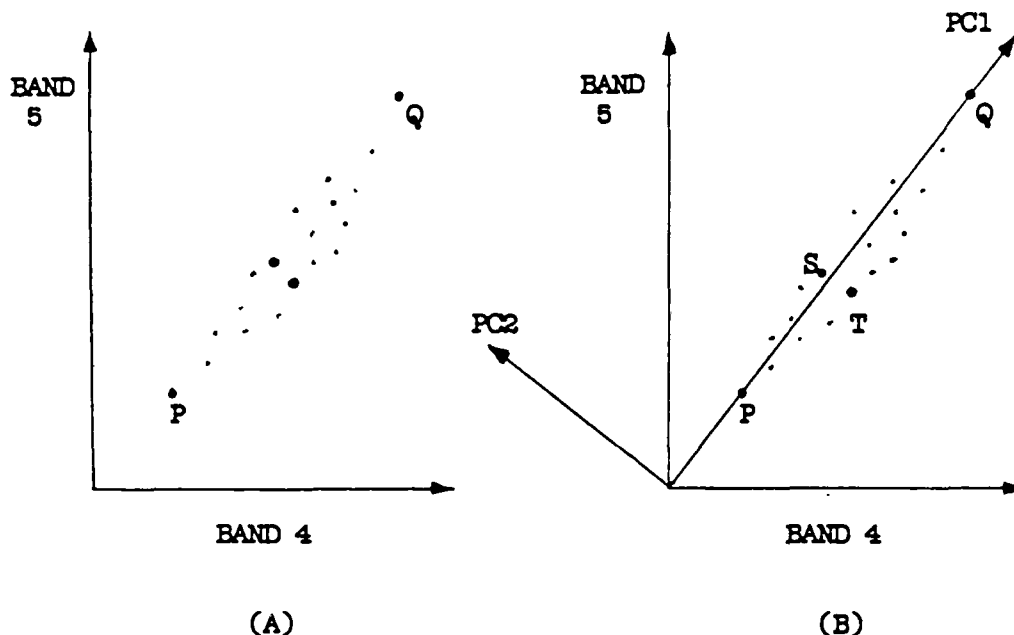
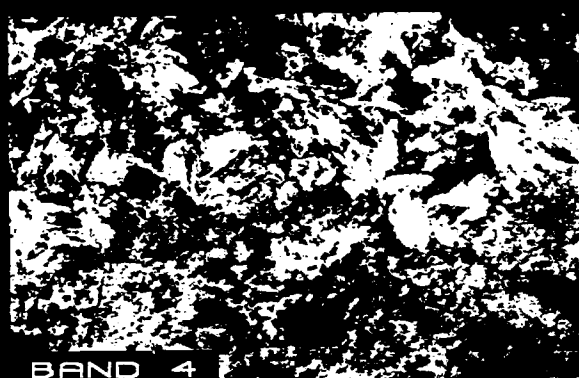


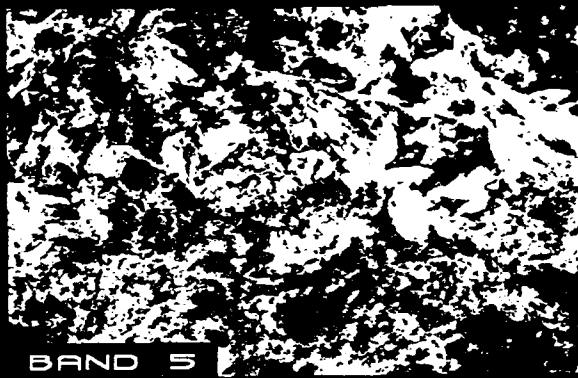
Fig 25 Typical scatter diagrams

Consider the scatter diagram in Fig 25(A). The intensity of a pixel P, in Landsat MSS band 4, has been plotted against the intensity of the same pixel in band 5. This has been repeated for pixel Q, and all other pixels within the image and it can be seen that these bands are highly correlated. Thus a pixel which is dark in band 4 is also dark in band 5 (as at P) and similarly a bright pixel in band 4 is also bright in band 5 (as at Q). Hence all the plotted points lie on or near the diagonal line at 45 degrees to the axes of the plot. As far as the interpreter is concerned, the fact that the data are highly correlated, means that, having studied band 4, he gets little further information from looking at band 5. Fig 26 which is a Landsat MSS picture of Botswana (185/77) taken on 17-Jan-73, shows this point very well, there being very little variation between the four spectral bands.

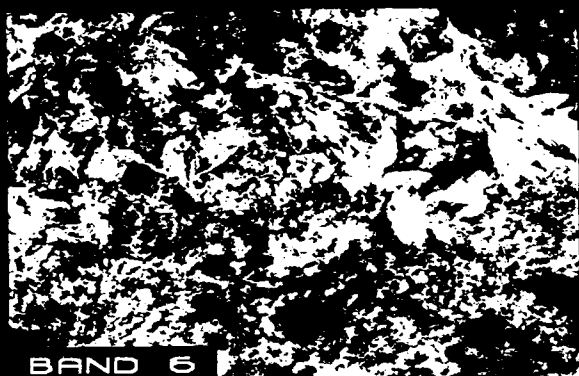
Principal components analysis is designed to remove this correlation by creating a new set of bands, each of which is made by adding different proportions of the input bands (ie linear combinations of them). This is illustrated in Fig 25(B). A new pair of axes (PC1, PC2) have been defined, created by rotating the original axes until PC1 lies along the direction of maximum variance of the data as shown. The method by which the image PC1 is produced is approximately as follows. The intensity of pixel S in PC1 is the sum of part of the intensity in band 4 and part of the intensity in band 5. If the PC1 axis has been rotated through 45 degrees, then the contributions of the two bands are equal. If the angle is less than 45 degrees, then band 4 will contribute more. Similarly, PC2 is formed by subtracting the intensity of S in band 4 from band 5. Note that it is very difficult to see the difference between points S and T in bands 4,5 and PC1, since the



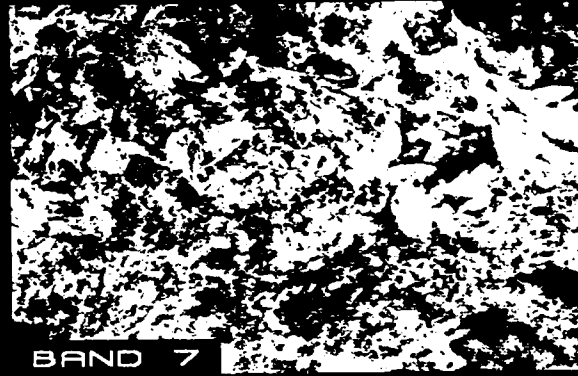
BAND 4



BAND 5

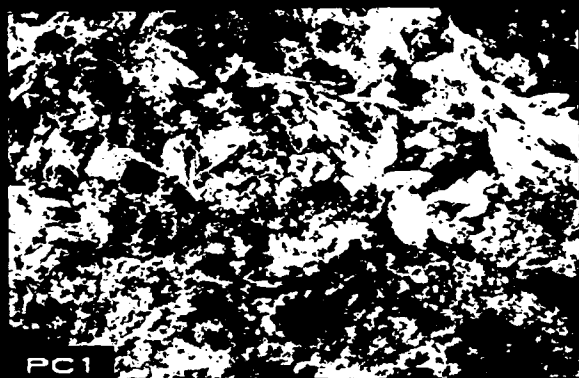


BAND 6



BAND 7

FIG 26 LANDSAT MSS OF BOTSWANA



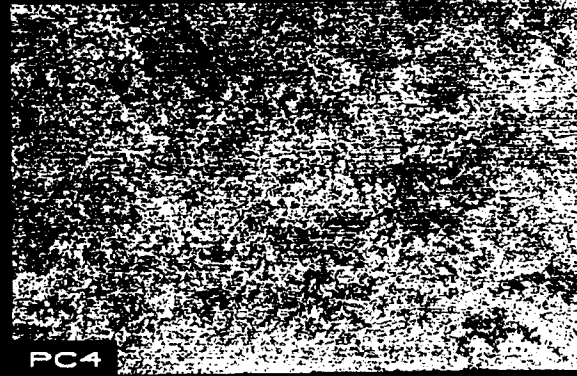
PC1



PC2



PC3



PC4

FIG 27 PCA OF FIG 26

separation of S and T is very small compared to the total intensity range of the data. This subtle difference in intensity could therefore be invisible to the eye. However, in PC2 the separation of S and T is about half of the total range of the data in this band and would therefore be easily visible. This illustrates that principal components analysis can often show subtle variations in spectral response which would not normally be visible.

In this illustration, PC1 is the average of the two input bands, and PC2 is the difference between them. This process can be applied to more than two bands, using the following rules:

PC1 is along the direction of maximum variance.

PC2 is at right angles to PC1, and is along the next direction of maximum variance.

PC3 is at right angles to both PC1 and PC2, and is along the next direction of maximum variance.

PC4 etc.

Strictly, the rotation is not about the origin, but is about the centre of gravity of the distribution, or the point with the co-ordinates which are the mean intensities in each band. Having performed this rotation, a number is added to each pixel value to put the mean at 128 in each of the transformed axes. Finally a simple linear contrast stretch is applied to each principal component to produce a distribution with a standard deviation of about 25. However, if the standard deviation of a principal component before contrast stretching is less than 2.5, the standard deviation of the output band is set to ten times the input, thereby keeping the noise in the output band to a reasonable level.

Fig 27 shows the result of principal components analysis applied to the four bands of Fig 26. The following is a copy of the information which is printed on the VDU.

ffMEAN OF INPUT BANDS

ff 57.0 70.9 71.2 32.4

ff

ffSD OF INPUT BANDS

ff 6.6 11.0 10.1 4.7

ff

ffSD OF PRINCIPAL COMPONENTS BEFORE STRETCHING

ff 16.4 3.6 2.2 1.2

ff

ffEIGENVALUES SCALED TO SUM TO 1

ff 0.93258 0.04503 0.01713 0.00527

ff

ffPRINCIPAL COMPONENTS AS COMBINATIONS OF ORIG. BANDS

ff 1 0.3542 0.6626 0.6072 0.2586

ff 2 0.8235 0.0430 -0.3325 -0.4576

ff 3 0.3104 -0.7361 0.5991 0.0542

ff 4 0.3162 -0.1317 -0.4024 0.8490

The 'MEAN OF INPUT BANDS', gives the mean intensity values in the order of the stores. In this case they are for bands 4,5,6 and 7. Similarly the 'SD' numbers are the standard deviations for those stores, followed by the standard deviations for the resulting principal components. The next four numbers are the eigenvalues of the covariance matrix, scaled to sum to one, and in this example, they show that 93.258% of the information is contained in the first principal component, 4.503% of the information is contained in the second, etc. Below 'PRINCIPAL COMPONENTS ...' is the transformation matrix. The first eigenvector is (0.3542, 0.6626, 0.6072, 0.2586) corresponding to the first eigenvalue, 0.93258, and similarly for the others. The numbers in this table imply that a pixel in the first principal component is calculated using the following rule:

0.3542 times the intensity of that pixel in Landsat MSS band 4, plus
 0.6626 times the intensity of that pixel in Landsat MSS band 5, plus
 0.6072 times the intensity of that pixel in Landsat MSS band 6, plus
 0.2586 times the intensity of that pixel in Landsat MSS band 7.

Thus, the first principal component is approximately the sum of the four bands, the second is the difference between the visible and the infrared bands, and the third and fourth are more of a mixture.

To execute this command, the user specifies the stores to be used for the operation; the stores for the results; the area to be used to produce the covariance matrix and hence the transformation; and the area in which the transformation is to be performed. Note that if no result stores are specified, the eigenvalues and eigenvectors will still be calculated, but the transformation will not be done. If fewer result stores than input stores are specified, then only the first bands up to the number of result stores, will be produced.

OVERLAY 4 SCATTER DIAGRAM

SUMMARY

The principle is the same as 'SCATTER DIAGRAM' (p65), but the area to be used is defined by the contents of overlay plane 4.

COMMAND IN DETAIL

By use of the graphics routines in 'Text and Graphics' or in 'Classifiers, Copy Bit Plane', or in 'Density Slice, Measure Areas', a mask is put into overlay plane 4, such that only where the plane is at level one are the pixels in the image planes included in the scatter diagram.

OVERLAY 4 PRINCIPAL COMPONENTS

SUMMARY

The principle is the same as 'PRINCIPAL COMPONENTS' above, but the area for the statistics is defined by the contents of overlay plane 4. The detail is the same as for 'SCATTER DIAGRAM' above.

MAGNIFY

SUMMARY

This command allows the user to magnify or reduce the size of an image by integer or non integer factors, independently in both directions. It also permits the reflection of an image about either or both axes. Resampling is performed using nearest neighbour or bilinear interpolation.

COMMAND IN DETAIL

Magnification of a digital image cannot be considered like a photographic enlargement. A pixel displayed on the TV covers a small square on the screen. Since the size of this small square is fixed, the only way to magnify the image is to increase the number of small squares per original pixel. For example, if there is an image of 256 pixels by 256 lines, it will cover only one quarter of the TV display. Magnifying digitally by two in both directions, will result in an image of 512 pixels by 512 lines which will cover the whole screen. Digital magnification therefore increases the number of pixels, and the resampling method defines the values given to these pixels, as described below.

Consider the start of the first line of an image, where A, B, and C are the intensities of the first three pixels and also mark the centres of the pixels.

A	B	C	
---	---	---	--

If the image is magnified by two in the x direction only, then for each input pixel there will be two output pixels, with intensities a,b,c.... which again mark the centres.

a	b	c	d	e	f	
---	---	---	---	---	---	--

To determine the pixel intensities at a,b,c,..., the output pixel centres are superimposed on the original. For nearest neighbour interpolation, point a, since it lies within pixel A, takes on the value of A. Similarly, b takes the value of A. Point c is within pixel B and hence takes the value of B, and so on.

a A b	c B d	e C f	
-------	-------	-------	--

More generally, for a magnification by a factor m, the centre of the first pixel, a, is at a distance $\frac{1.1}{2m}$ from the edge

of the input pixel, and the spacing is $\frac{1}{m}$ of the input pixel.

$\frac{1.1}{2m}$	$\frac{1}{m}$	
←	←	
aA	b B c	dC

For nearest neighbour interpolation, the program simply determines which old pixel contains the new point. For bilinear interpolation, the value of the output pixel is found as follows.

.	.	.
.	.	.
.	.	.
A	b B	
←	←	
p	q	

If p and q represent the distances from point b to A and B respectively, then

$$\text{Intensity at } b = (\text{Intensity at } A) \cdot \frac{q}{p+q} + (\text{Intensity at } B) \cdot \frac{p}{p+q}$$

It is therefore assumed that there is a linear change from pixel A to pixel B and b takes a value in proportion to the distances from A and B . It is not possible to apply this equation to point a , since there is no pixel to the left of it. (A is the first pixel in the line.) Pixel a , simply takes the value of A .

The following example, using the idealised image of Fig 17, illustrates the two resampling techniques.

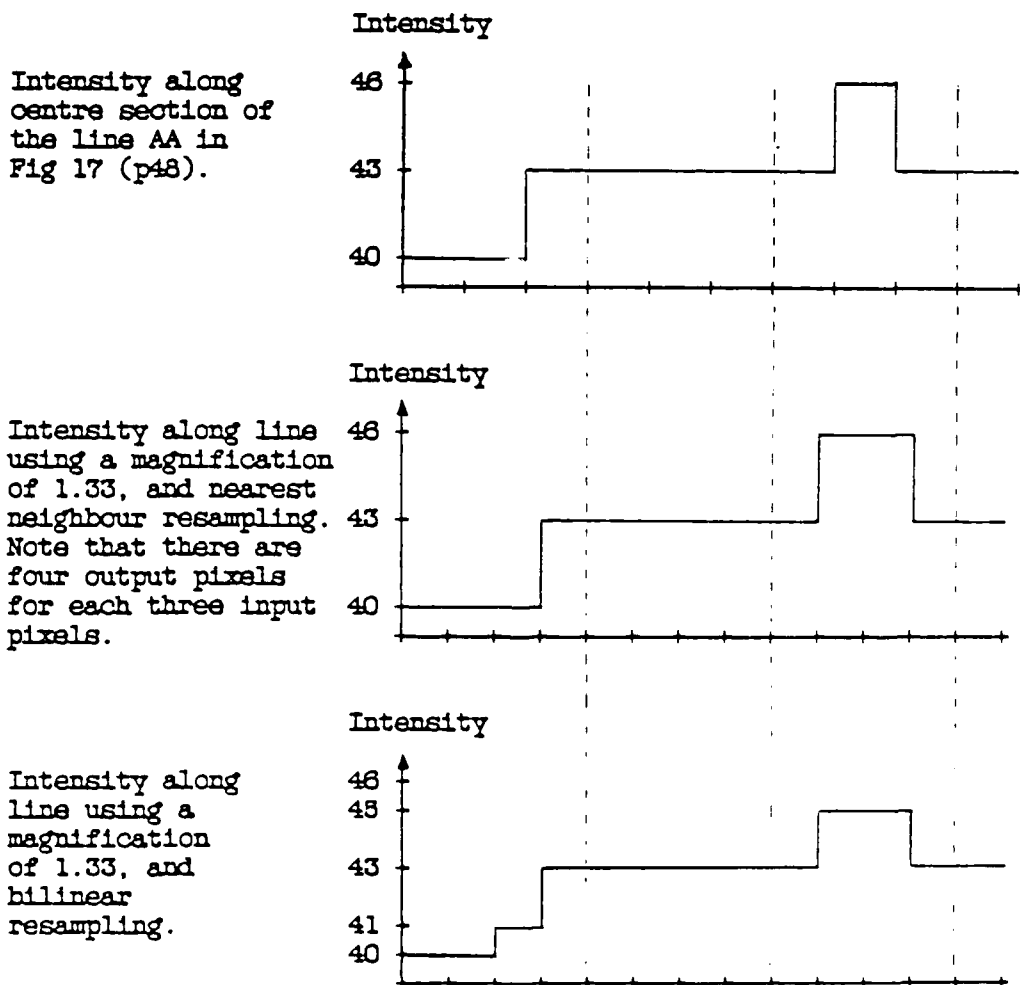


Fig 28 Examples of resampling used in magnification

Nearest neighbour resampling does not introduce smoothing or any intensities that did not previously exist, but it does distort shapes (eg the distance from the field boundary to the noise spike is now less). On the other hand, bilinear interpolation introduces new intensities that did not previously exist, but it does not distort shapes as much. (eg the field boundary is still essentially in the same position). There is also a smoothing effect (of smoothing example in Fig 17, p48), but edge enhancement will help to correct this.

The description so far has been for magnification along the line. For 2-D magnification, the resampling is performed first along the lines, and then down the columns, for both resampling methods.

After the primary command, the user gives the stores to be magnified and the result stores. The area to be magnified is next specified as the whole screen, a part of the screen (fixed using the rolling ball and the 'CHNG' button as usual) or the last specified area. The maximum area to be covered by the magnified result is next given, using the rolling ball to change the position and size of the rectangle on the screen. When 'INPUT' is pressed at the end of this operation, the magnification of the input area which just fills the output area, is displayed on the terminal. Following the choice of interpolation method, the user is asked for the magnification required in the x direction. Using the rolling ball the magnification factor at the bottom of the screen can be varied. A number greater than one will result in an enlargement; between zero and one will result in a reduction; equal to zero will result in the use of the magnification shown on the terminal (this saves having to re-enter this number); and a number less than one will result in a reversal of the image as well as the enlargement/reduction.

There is a small bar above the number on the screen. If the rolling ball is moved to the right, then the number is incremented starting at the digit below the bar. Moving the ball to the left causes the number to decrement. If 'CHNG' is pressed, the small bar can be moved to the left or right. This is useful if a number with many digits is to be entered, since the two least significant digits could be entered, the bar then moved to the left and the next two digits entered, etc. If 'CHNG' is pressed again, the decimal point is moved by the rolling ball. These three operations of changing a number, moving the bar and moving the decimal point, cycle round continuously, and so decimal numbers with many digits can be entered quickly.

Exactly the same operation is performed for the y magnification, and when 'INPUT' is pressed, the command is executed. If the magnified image area is less than the whole screen, then the rest of the image area is unmodified. Similarly, if the whole screen has been specified, but a magnification less than that required to fill the screen is used, part of the image area is again unmodified. Finally, if a magnification greater than that required to fill the area is specified, then the output image will start at the top left corner but will be truncated to the right and the bottom so that it fits the area allocated.

5.7 Linear Filters

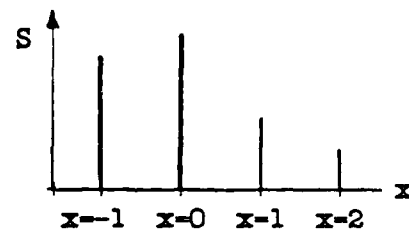
The commands within this chapter permit the creation of a wide range of filters with sizes up to 5 by 5 weights, and the subsequent application to an image. Apart from the restriction on size, there is much greater flexibility in the definition of a filter in this chapter, than in 'Maths operations'. However, the user must be quite familiar with filtering operations before using the filters in this chapter, since there is a much greater possibility of producing misleading results.

When creating a new filter, it may be helpful to analyse its effect using the following approximate method.

Consider a 1-D image where the intensity, I , varies as a function of position, x .

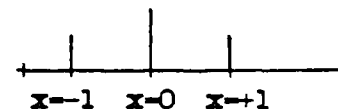
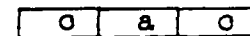


A digital image is a sampled version of this, and can be considered as the product of the image and a set of delta-functions separated by the pixel spacing, which corresponds to the Nyquist sampling frequency f .



Thus $S = I \cdot (\dots + \delta(x-1) + \delta(x) + \delta(x+1) + \dots)$

Similarly, the filter can be considered as a set of weighted delta-functions with the same spacing as the digital image. For simplicity, a three-weight symmetric filter centred on $x = 0$ is used in this example.



The filtered version of the image is thus $I = S * (\text{filter})$

On performing a Fourier Transform (FT), the convolution (*) is changed to a multiplication, so that

$$\text{FT}\{S * (\text{filter})\} = \text{FT}\{S\} \cdot \text{FT}\{\text{filter}\}$$

Hence, in order to analyse the modifying effect of the filter on the frequency response, it is simply necessary to examine the Fourier Transform of the filter. In this simple example,

$$\begin{aligned} \text{FT}(o.\delta(-1) + a.\delta(0) + o.\delta(+1)) &= o.e^{-2\pi i f} + a.e^{0} + o.e^{+2\pi i f} \\ &= a + 2.o.\cos(2\pi f). \end{aligned}$$

The easiest way to consider a filter with an even number of weights, is still to centre it on $x = 0$. Thus a two-weight filter has the form,

$$(b.\delta(-0.5) + b.\delta(+0.5)), \quad \text{the FT of which is} \quad 2.b.\cos(\pi f).$$

This filter is not strictly permissible, since it is not possible in GEMS to place filter weights at non-integer pixel positions. However, this approach avoids the need to consider imaginary sines, and the only difference is that there is a displacement. (of the filters in 'Maths operations' which shift the image by 0.5 of a pixel for filters with even numbers of weights). The general form of both frequency responses is the same.

In the following table, the complete set of 1-D symmetric filters up to five elements is given, with the corresponding Fourier Transforms.

FILTER	FT					
<table><tr><td>a</td></tr></table>	a	a				
a						
<table><tr><td>b</td><td>b</td></tr></table>	b	b	$2.b.\cos(\pi f)$			
b	b					
<table><tr><td>c</td><td>a</td><td>c</td></tr></table>	c	a	c	$a + 2.c.\cos(2\pi f)$		
c	a	c				
<table><tr><td>d</td><td>b</td><td>b</td><td>d</td></tr></table>	d	b	b	d	$2.b.\cos(\pi f) + 2.d.\cos(3\pi f)$	
d	b	b	d			
<table><tr><td>e</td><td>c</td><td>a</td><td>c</td><td>e</td></tr></table>	e	c	a	c	e	$a + 2.c.\cos(2\pi f) + 2.e.\cos(4\pi f)$
e	c	a	c	e		

Table 7 The set of 1-D symmetric filters

To produce a filter, all that is required is to choose the numbers a,b,c,d,e. Using the method outlined above, the response of the filter can then be examined. The following pages give some examples of filters, their transfer functions, and the effects on the computer generated image shown in the top strip, labelled 'NO FILTER', in Fig 29. This image is in nine sections. The leftmost section consists of a column of pixels at level 0, followed by a column of pixels at level 255, etc. This corresponds to the highest possible frequency that can be correctly recorded in the digital image. (It is assumed that there is no aliasing.) This frequency is in fact half the Nyquist sampling frequency and is thus $f/2$, with a corresponding wavelength of two pixels. The next section to the right is a sampled sine wave with a frequency of $f/3$ and thus a wavelength of three pixels. The sampled sine waves in the following sections have frequencies down to $f/10$, or a wavelength of ten pixels. These patches of pure single frequencies, cover most of the range of interest in a typical scene, when it is examined for patterns, texture, and edges.

The examples of filters have two items called scale and offset, associated with them. Each filter weight (which is restricted to integer values) is divided by 'scale' thereby permitting fractional weights. The offset number is added to the result at the end of the convolution and is generally used to shift the origin 128 levels to cater for negative weights. All the frequency response curves have the same horizontal scale which is annotated only on the first graph.

Filter in Fig 29

Frequency response

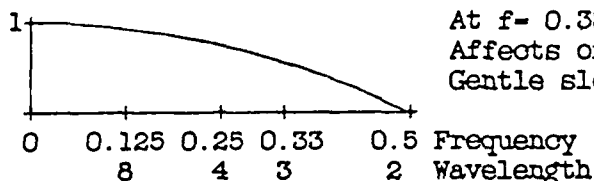
Comments

A - Low pass

1 1

Scale = 2

Offset = 0



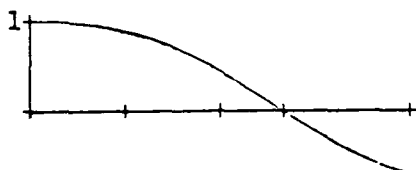
At $f = 0.33$, half power, slope = -2.72
Affects only high frequencies.
Gentle slope.

B - Low pass

1 1 1

Scale = 3

Offset = 0



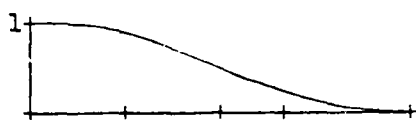
At $f = 0.21$, half power, slope = -4.06
Affects mid and high frequencies.
Steep slope, but above $f = 0.33$, the phase is inverted and gain is not equal to zero.

C - Low pass

1 2 1

Scale = 4

Offset = 0



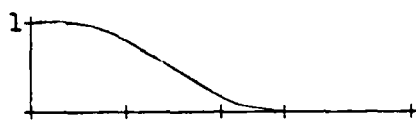
At $f = 0.25$, half power, slope = -3.14
Affects mid and high frequencies.
Moderate slope.
Note this is $A * A$.

D - Low pass

1 4 6 4 1

Scale = 16

Offset = 0



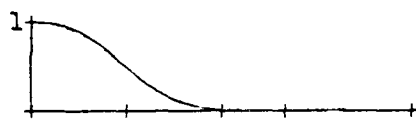
At $f = 0.18$, half power, slope = -4.04
Affects mid and high frequencies.
Steep slope.
Note this is $C * C$.

E - Low pass

Filter D

followed by

filter D



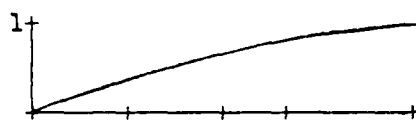
At $f = 0.13$, half power, slope = -5.47
Affects 'lowish' to high frequencies.
Very steep slope. This is the same as a single nine-weight filter.

G - High pass

1 -1

Scale = 2

Offset = 128



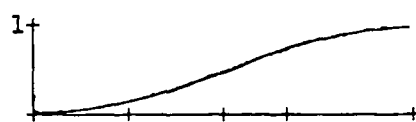
At $f = 0.17$, half power, slope = 2.72
Affects low frequencies.
Gentle slope.

H - High pass

-1 2 -1

Scale = 4

Offset = 128



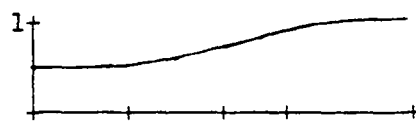
At $f = 0.25$, half power, slope = 3.14
Affects low and mid frequencies.
Moderate slope.
Note this is $G * G$.

I - High boost

-1 6 -1

Scale = 8

Offset = 64



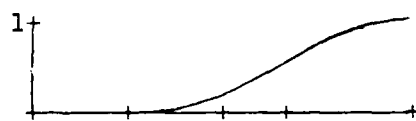
This is half of the original image added to half of the image convolved with filter H. This is edge enhancement.

J - High pass

1 -4 6 -4 1

Scale = 16

Offset = 128



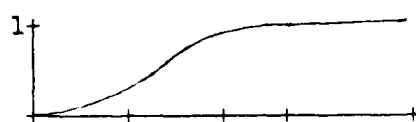
At $f = 0.32$, half power, slope = 4.04
Affects low and mid frequencies.
Steep slope.
Note this is $H * H$

K - High pass

-1 -4 10 -4 -1

Scale = 16

Offset = 128



At $f = 0.18$, half power, slope = 4.04
Affects low frequencies only. It is like filter J but the half power point is changed.

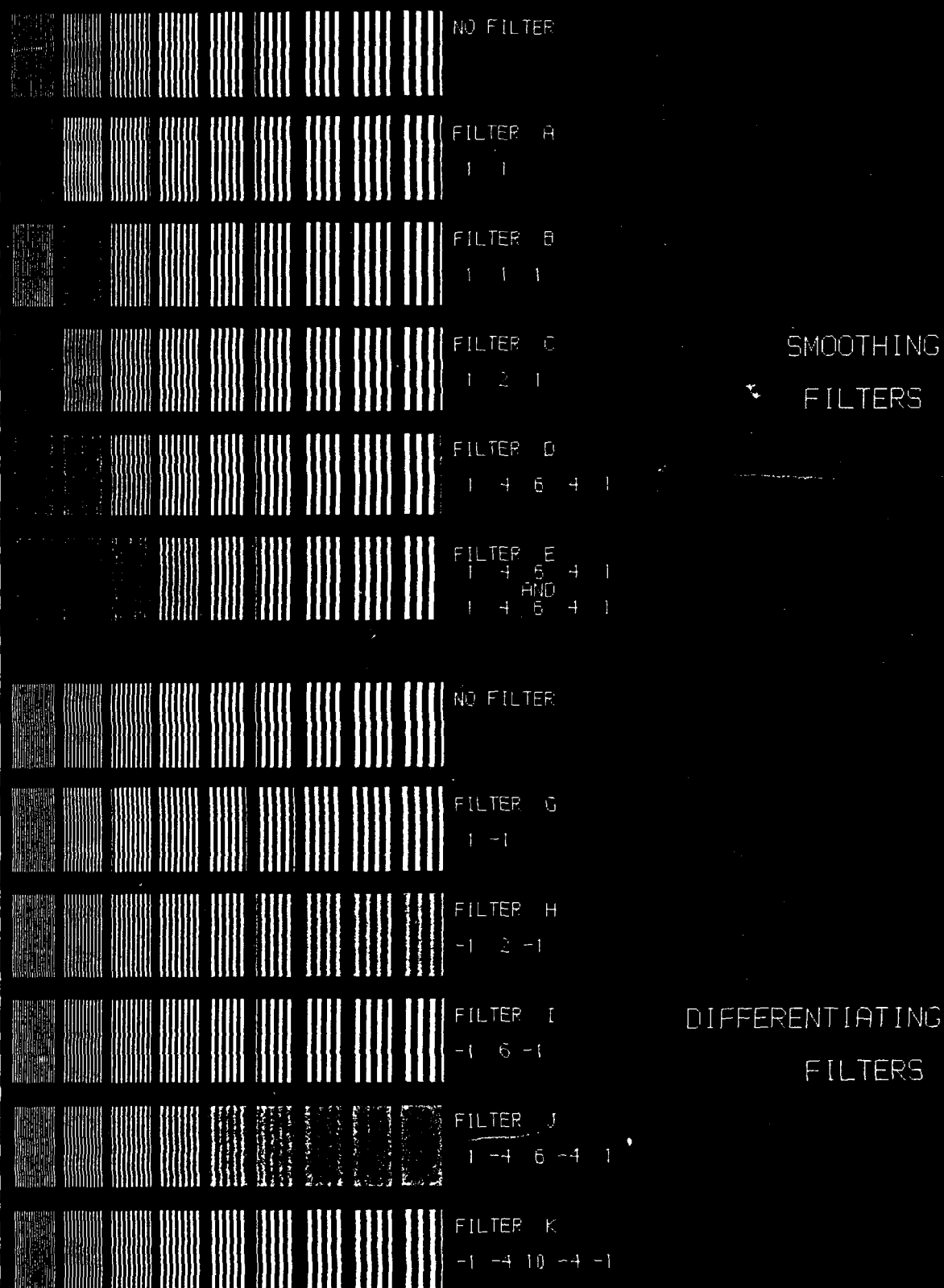
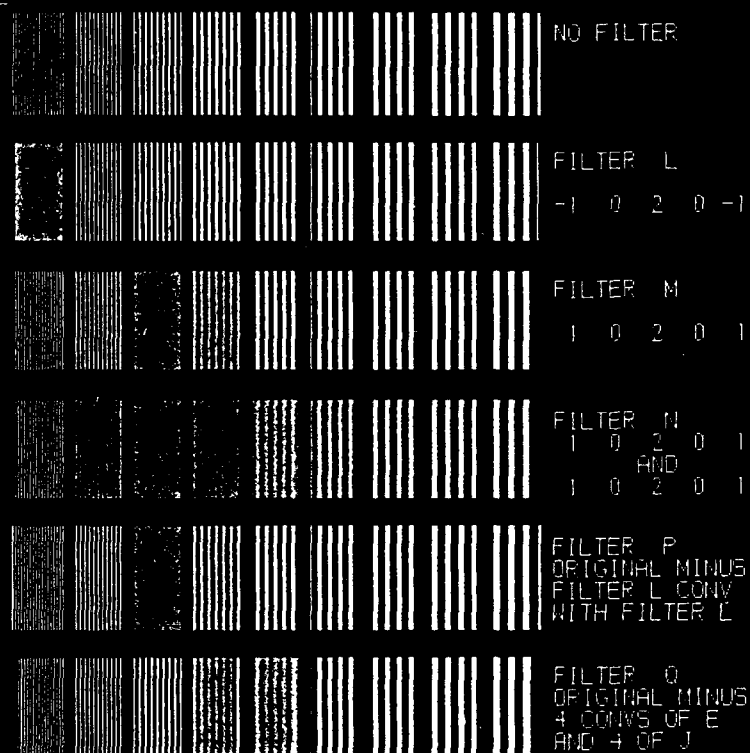
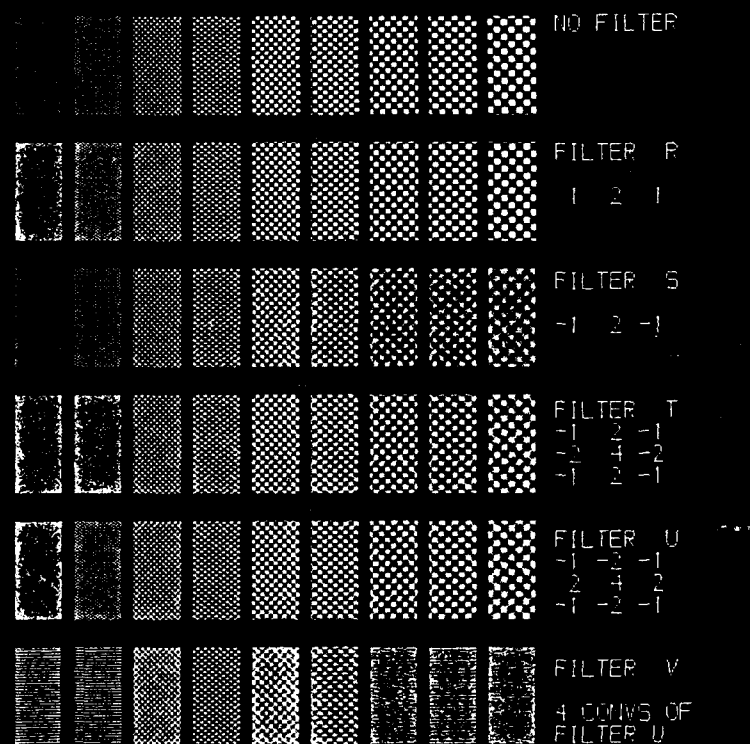


Fig 29 Filter frequency responses

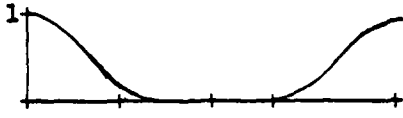
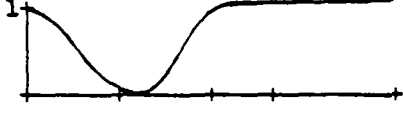


BANDPASS
AND
BANDSTOP
FILTERS



2-D
FILTERS

Fig 30 Filter frequency responses

Filter in Fig 30	Frequency response	Comments
L - Bandpass -1 0 2 0 -1 Scale = 4 Offset = 128		At $f = 0.125$ and 0.375 , half power. Affects high and low frequencies. Note this is C^*H
M - Bandstop 1 0 2 0 1 Scale = 4 Offset = 128		At $f = 0.125$ and 0.375 , half power. Affects the mid frequencies only. This is the original image less the one filtered with L.
N - Bandstop Filter M applied twice		At $f = 0.091$ and 0.409 , half power. Affects the mid frequencies only, but over a wider bandwidth than filter M.
P - Bandstop Original less filter L * L		At $f = 0.159$ and 0.341 , half power. Affects the mid frequencies only, centred on $f = 0.25$, but over a narrower bandwidth than filter M.
Q - Bandstop Original less 4 convs of E and K		At $f = 0.0683$ and 0.225 , half power. Affects the mid to low frequencies. Similar to P but centre of filter shifted to lower frequency. (0.141)

The last group of filters are concerned with filtering 2-D images. The test image is made in the same way as the previous one, but the frequencies in the patches run in both vertical and horizontal directions. Thus the first patch is made by taking an image of black/white columns, and multiplying it by an image of black/white rows, producing an image of dots. The horizontal and vertical frequencies for this patch are both $f/2$. In the subsequent patches, the horizontal frequencies are $f/3$, $f/4$, $f/5$, $f/6$, $f/7$, $f/8$, $f/9$, and $f/10$, and the vertical frequencies are $f/2$, $f/4$, $f/4$, $f/6$, $f/6$, $f/8$, $f/8$, and $f/10$.

Filter R (same as C) shows that the image is only modified in areas of high horizontal frequencies, irrespective of the vertical frequencies, and the output tends to zero at a horizontal frequency of $1/2$. This illustrates that a 1-D filter affects frequencies in both directions.

Filter S (same as H, ie high pass), suppresses all low frequencies. It is only at high horizontal frequencies, that any vertical information is passed.

Filter T is filter H convolved with filter C rotated through 90 deg. It is a horizontal high pass and a vertical low pass filter. It is similar to filter S, except that the image is suppressed in areas of high horizontal frequency.

Filter U is the same as filter T but rotated through 90 degrees, thereby passing high frequency vertical information but suppressing high frequency horizontal and all low frequency information.

Filter V illustrates that 2-D filters can be applied sequentially, and here filter U was convolved with the image four times, thereby losing all but the extreme high frequency vertical information.

With these examples, the user should be able to design many smoothing (low pass), differentiating (high pass), bandpass and bandstop filters. Filters A to K are 1-D and they may be convolved with themselves and others to produce 2-D filters (eg T, U and V). For example, filter T is made from filter H convolved with filter C rotated through 90 deg as follows:

$$(1/4) \cdot \begin{pmatrix} 0 & 0 & 0 \\ -1 & 2 & -1 \\ 0 & 0 & 0 \end{pmatrix} * (1/4) \cdot \begin{pmatrix} 0 & 1 & 0 \\ 0 & 2 & 0 \\ 0 & 1 & 0 \end{pmatrix} = (1/16) \cdot \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 \\ 0 & -2 & 4 & -2 & 0 \\ 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}.$$

using the definition of convolution on page 44. The terms (1/4) and (1/16) are the 'scales' defined on pages 78 and 83.

A command in this chapter also permits the rotation of a 1-D or a 2-D filter, to any desired orientation. Thus a filter may be defined in the easiest orientation and later rotated to enhance or smooth in the desired direction. Rather than convolving two simple filters to make a more complicated filter for application to the image, it is equally possible to convolve each simple filter with the image in turn. This is because convolution is commutative in that, given two filters, F1 and F2 and an image IM,

$$F1 * (F2 * IM) = F2 * (F1 * IM) = (F1 * F2) * IM.$$

Filters can also be added together, to produce another family of filters, since convolution is distributive, ie

$$(F1 * IM) + (F2 * IM) = (F1 + F2) * IM.$$

Filter I has the same effect as adding half of the original to half of the image convolved with the filter H on page 78.

$$\begin{aligned} & \text{ie } (H * IM)/2 + IM/2 \\ & = (H * IM)/2 + (1 * IM)/2 \end{aligned}$$

where '1' represents a filter having one in the middle and all other elements zero. This is called the 'identity' filter.

$$\begin{aligned} & = (H + 1)/2 * IM \\ & = ({\{-1, 2, -1\}}/4 + {\{0, 1, 0\}})/2 * IM \\ & = ({\{-1, 6, -1\}}/4)/2 * IM \\ & = {\{-1, 6, -1\}}/8 * IM \\ & = (\text{filter I}) * IM. \end{aligned}$$

where '4' is the 'scale'.

Similarly, filter K, which is the original less the image convolved with filter D, is

$$\begin{aligned} & (1 * IM) - (D * IM) \\ & = (1-D) * IM \\ & = ({\{0, 0, 1, 0, 0\}} - {\{1, 4, 6, 4, 1\}})/16 * IM \\ & = {\{-1, -4, 10, -4, -1\}}/16 * IM \\ & = (\text{filter K}) * IM. \end{aligned}$$

Examples A, C, D, and E illustrate that as the smoothing filter becomes longer, the half power point tends towards lower frequencies and the slope increases. Equally for the differentiating filters, G, H, and J, the half power point tends to higher frequencies and the slope increases. By subtracting a smoothed image from the original, a differentiating filter is produced, but in this case the half power point tends towards the lower frequencies as the filter size is increased. An example of this is filter K, which is the same as the original image less the one smoothed with filter D. Similarly, a high pass filtered image can be subtracted from the original, to produce a smoothed image. Thus, there is considerable control over the position of the half power point and the slope. Slopes can be increased by allowing the frequency response curve to become negative as for filter B. This negative gain, or phase reversal is often unimportant, but a further filtering operation can remove the problem. eg filter D applied before or after B would remove all the high frequencies.

A bandpass filter can be produced by successive convolution of the image with high and low pass filters. A bandstop filter can be made by subtracting the bandpassed image from the original.

The following more complex example shows how a low frequency bandpass filter can be formed. The low pass part can be made from filters A, B, C, D or E, or from the original image less G, H or J. Steep slopes are often desirable, and hence D or E would be best. By the same argument, the high pass part is best made by subtracting a low pass filtered image from the original. Thus for a pass band with a centre frequency of 0.18, the image could be filtered with D, subtracted from the original (ie high pass) and filtered again with D. These stages can be examined with the help of Table 7. Filter D has a frequency response of

$$(1/16) \cdot \{6 + 8 \cos 2\pi f + 2 \cos 4\pi f\} = \cos^4 \pi f.$$

The bandpass filter is therefore

$$\cos^4 \pi f \cdot (1 - \cos^4 \pi f),$$

which happens to be filter D followed by K. More generally, a bandpass filter can be written as

$$\cos^m \pi f \cdot (1 - \cos^n \pi f).$$

This function should have a turning point at the centre frequency, which implies that

$$n = m \cos^2 \pi f / (1 - \cos^2 \pi f).$$

For a centre frequency, f , 'm' can be chosen, and 'n' is then given by the above equation. If 'n' equals 'm', the overall filter is symmetric. Attention should be paid to the scales and offsets to ensure that the filter has a gain of one at the centre frequency. Thus for the above example, filter D is applied first with a scale of 16 and an offset of 0 as before, but the scale for filter K should be 4, with an offset of 128. The six-line banding of Landsat MSS images has a basic frequency $f = 1/6$, and can be matched reasonably well with $m = 4$ and $n = 5$ (should be 5.14), ie filters D, A, and K.

More complex filtering operations than illustrated in this example, should be performed using Fourier transforms (see chapter 3.12).

There are many filters which appear to be less amenable to this form of analysis. For example, the simple Laplacian filter

$$\begin{pmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{pmatrix} \text{ does not decompose immediately.}$$

Since the filter is symmetric, the two linear filters required to produce it, should be of the form,

$$x.(1,a,1) \text{ and } x.\begin{pmatrix} 1 \\ a \\ 1 \end{pmatrix} \text{ where 'x' is the scale.}$$

Convolving these two, results in the top left element being $x.x$ which can only equal the top left element of the Laplacian if x equals 0. However the convolution process is distributive and therefore this filter can be decomposed to,

$$\begin{pmatrix} 0 & 0 & 0 \\ -1 & 2 & -1 \\ 0 & 0 & 0 \end{pmatrix} + \begin{pmatrix} 0 & -1 & 0 \\ 0 & 2 & 0 \\ 0 & -1 & 0 \end{pmatrix}.$$

Thus the Laplacian filter produces the same result as convolving the image with the left 1-D filter above, and adding this image to the one produced by convolving the original image with the right 1-D filter. The frequency response of the Laplacian is the sum of the frequency responses of the separate filters.

Summarising, if filter A is convolved with filter B to produce C, then the frequency response of C is the product of the frequency response of A and the frequency response of B. If filter A is added to filter B to produce C, then the frequency response of C is the sum of the frequency responses of A and B. Any combination of sums and convolutions is allowed.

The frequency response of the Laplacian is therefore,

$$(2-2\cos 2\pi f(x)) + (2-2\cos 2\pi f(y)) = 2.(2-\cos 2\pi f(x)-\cos 2\pi f(y)).$$

where $f(x)$ and $f(y)$ represent the frequencies in the x and y directions. It is difficult to visualise this frequency response since it is 2-D. However, a few points can be examined.

$$\begin{array}{ll} \text{At } f(x) = f(y) = 0, & \text{the output is } 0. \\ \text{At } f(x) = 0, & \text{the output is } 2-2\cos \pi f(y). \\ \text{At } f(x) = 1/2, & \text{the output is } 6-2\cos \pi f(y). \\ \text{At } f(x) = f(y) = 1/2, & \text{the output is } 8. \end{array}$$

An image of the frequency response can be made, where the intensity, I , at a point x,y on the screen represents the filter response at $f(x)$ and $f(y)$. Thus if

$$I = 64.(2-\cos \frac{X_{pos}-1}{255}) - \cos \frac{512-Y_{pos}}{255}),$$

an image of the frequency response will result in the bottom left quadrant of the screen. (See the Arithmetic chapter for details on the functions used in this equation.) The '64' is simply to make the maximum value of I equal to 255 in the image, which is shown in Fig 31 at the top left. At the top right, the image has been sliced at eight levels, to show the loci of equal responses. At the bottom left, the 3-D display facilities in the Text and Graphics chapter, have been used to make an alternative, which may be easier to visualise.

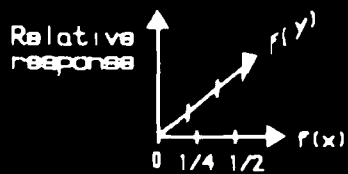
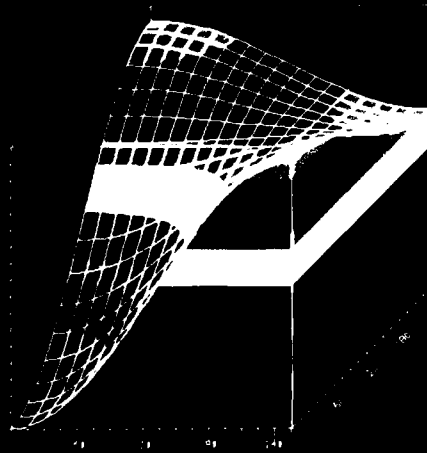
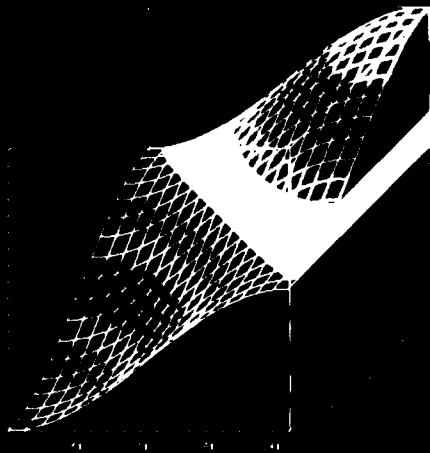
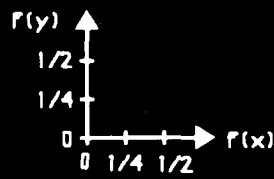
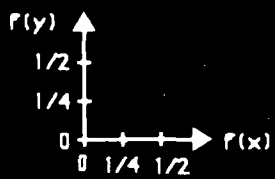


Fig 31 Filter responses

To determine the response of a filter at a particular pair of frequencies, $f(x)$ and $f(y)$, the command 'PIXEL VALUE' in the Contrast Stretch chapter (page 32) is used on the image of the response (eg Fig 31 top left).

For this example, $f(x) = 0$ at $x = 1$, and $f(x) = 1/2$ at $x = 256$
and $f(y) = 0$ at $y = 512$, and $f(y) = 1/2$ at $y = 257$.

Suppose the response is required at $f(x) = 1/4$, and $f(y) = 1/8$, the cursor is put at

$$x = 1 + (256 - 1)/2 = 129 \text{ and } y = 512 + (257 - 512)/4 = 448.$$

The intensity at this point is 82. Since the image intensities were multiplied by 32 to make most use of the intensity range, the filter response at this pair of frequencies, is $82/32 = 2.56$. Solving the equation $2(2 - \cos 2\pi f(x) - \cos 2\pi f(y))$ produces the answer 2.59, the small difference being due to quantisation errors.

Another example of an omnidirectional edge detecting filter is

$$\begin{pmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 9 & 0 \\ 0 & 0 & 0 \end{pmatrix} - \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix} = 9 - (1 \ 1 \ 1) * \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix},$$

the frequency response of which is $9 - (1 + 2\cos 2\pi f(x))(1 + 2\cos 2\pi f(y))$.

A sliced 3-D version of this is shown in Fig 31 at the bottom right, and it will be noted that the maximum response does not occur at $f(x) = f(y) = 1/2$, as in the Laplacian, but at $f(x) = 0$ and $f(y) = 1/2$, and vice versa. Both these filters can be used for omnidirectional edge detection, but the former is better for enhancing individual points whereas the latter is better for enhancing the edges themselves.

There are a great many 2-D filters with 3 by 3 weights, and some of the following examples have been taken from the books in the bibliography. Their effects on the image can be analysed using the procedures already described.

Omnidirectional smoothing -

$$1/9 \cdot \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}, \quad 1/10 \cdot \begin{pmatrix} 1 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 1 \end{pmatrix}, \quad 1/16 \cdot \begin{pmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{pmatrix}.$$

Directional smoothing -

$$1/3 \cdot \begin{pmatrix} 0 & 0 & 0 \\ 1 & 1 & 1 \\ 0 & 0 & 0 \end{pmatrix}, \quad 1/4 \cdot \begin{pmatrix} 0 & 0 & 0 \\ 1 & 2 & 1 \\ 0 & 0 & 0 \end{pmatrix}.$$

Omnidirectional edge detection -

$$\begin{pmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{pmatrix}, \quad \begin{pmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{pmatrix}, \quad \begin{pmatrix} 1 & -2 & 1 \\ -2 & 4 & -2 \\ 1 & -2 & 1 \end{pmatrix}.$$

Omnidirectional edge enhancement -

$$\begin{pmatrix} -1 \\ -1 & 5 & -1 \\ -1 \end{pmatrix}, \quad \begin{pmatrix} -1 & -1 & -1 \\ -1 & 9 & -1 \\ -1 & -1 & -1 \end{pmatrix}, \quad 1/2 \cdot \begin{pmatrix} -1 & -1 & -1 \\ -1 & 10 & -1 \\ -1 & -1 & -1 \end{pmatrix}.$$

Directional edge enhancement -

$$\begin{pmatrix} 0 & 0 & 0 \\ -1 & 3 & -1 \\ 0 & 0 & 0 \end{pmatrix}, \quad 1/3 \cdot \begin{pmatrix} -1 & 3 & -1 \\ -1 & 3 & -1 \\ -1 & 3 & -1 \end{pmatrix}.$$

Compass gradient -

$$\begin{pmatrix} 1 & 1 & -1 \\ 1 & -2 & -1 \\ 1 & 1 & -1 \end{pmatrix}, \quad \begin{pmatrix} 1 & 1 & 1 \\ 1 & -2 & -1 \\ 1 & -1 & -1 \end{pmatrix}, \quad \begin{pmatrix} 1 & 1 & 1 \\ 1 & -2 & 1 \\ -1 & -1 & -1 \end{pmatrix}.$$

(West) (N. West) (North)

Line detector -

$$\begin{pmatrix} 1 & -1 \end{pmatrix}, \quad \begin{pmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{pmatrix}, \quad \begin{pmatrix} -1 & 2 & -1 \\ -1 & 2 & -1 \\ -1 & 2 & -1 \end{pmatrix}.$$

Shift -

$$\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix}, \quad 1/2 \cdot \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 0 \end{pmatrix}, \quad \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}.$$

to the left to the left to the left
by one pixel. by 1/2 pixel. and down.

Region growing/shrinking -

$$\begin{pmatrix} 0 & 1 & 0 \\ 1 & 5 & 1 \\ 0 & 1 & 0 \end{pmatrix}, \quad \begin{pmatrix} 1 & 1 & 1 \\ 1 & 9 & 1 \\ 1 & 1 & 1 \end{pmatrix}, \quad \begin{pmatrix} 0 & 1 & 0 \\ 1 & 21 & 1 \\ 0 & 1 & 0 \end{pmatrix}.$$

The last filters are used with images having limited grey levels. For example, the result of a classification process (see chapter 5.8) may be an image of two classes, with class 0 at level 0, and class 1 at level 1. Using the first region growing filter, results in the following output:

Output = 0 to 4, class 0 centre pixel, surrounded by 0,1,2,3,4 class 1 pixels.
Output = 5 to 9, class 1 centre pixel, surrounded by 0,1,2,3,4 class 0 pixels.

If the resulting image were sliced, where,

Slices 1 & 3 (colour of red = 0) cover intensity levels 0,1,2 and 5,6,
Slices 2 & 4 (colour of red = 1) cover intensity levels 3,4, and 7,8,9,

then the effect would be the same as a median filter, and for example, spots of class 0 in an area of class 1 would be removed. Changing the slices to,

Slices 1 & 3 covering intensity levels 0,1,2,3 and 5,6,7,
Slices 2 & 4 covering intensity levels 4 and 8,9,

would cause class 0 to grow and class 1 to shrink.

The middle example of the region growing filters is again for a binary image and the last one is for a 3-level (or 3 class) image, the levels being 0,1,5. the slicing is of course much more complicated for this last filter.

DEFINE FILTER	WHOLE EXTRACT			D1	↑	↑	D1			
	PART EXTRACT							D2	↑	D2
	LAST ONE							D3	↑	D3
SYMMETRY				D4	↑	↑	D4			
FILTER IMAGE				5	↑	↑	5			
SAVE FILTER	1	2	1	6	↑	↑	6			
RESTORE FILTER	2	4	2	7	↑	↑	7			
SHIFT & ROTATE	1	2	1	8	↑	↑	8			
READ IMAGE	SIZE 3 SCALE 16 OFFSET 0			9	↑	↑	9			
WRITE IMAGE	TL PIX	TL 1/4		10	↑	↑	10			
	TOP 1/2	LEFT 1/2		11	↑	↑	11			
	TOP ROW	LEFT COL		12	↑	↑	12			
				13	↑	↑	13			
				14	↑	↑	14			
				15	↑	↑	15			
				16	↑	↑	16			
				D5	↑	↑	D5			
				D6	↑	↑	D6			
				D7	↑	↑	D7			
				D8	↑	↑	D8			
				↑END	↑	↑	↑END			

STATUS	HELP	RETURN	IMAGE
--------	------	--------	-------

INPUT FOR COMMAND OR QUIT

Fig 32 The 'Linear Filters' page

DEFINE FILTER

SUMMARY

A filter of up to 5 by 5 weights can be specified using this command. The weights can be made fractional by dividing by 'scale', and a constant can be added to the final image to permit the use of negative weights.

COMMAND IN DETAIL

The size of the square filter up to 5 by 5 is first requested. Weights can be left as zero if non-square filters are required. Having pressed 'INPUT', a cursor appears within the filter. This is moved over the weight to be specified, using the rolling ball. Alternatively, if 'CHNG' is pressed the cursor moves to the next weight in the filter. With the cursor over this weight, the 'INPUT' button is pressed, and the rolling ball is moved to set the positive or negative integer value of the weight. Pressing the 'INPUT' button again allows the user to move the cursor to specify another weight, and so on. A weight can be redefined and a weight which is not specified by the user, is assumed to be zero. When complete, the 'QUIT' button is pressed to exit from the filter patch. It is now necessary to give a 'scale' or the integer number which will be the divisor of each weight. A number is suggested, based on the weights that the user has put into the filter, but it can be changed with the rolling ball. Having pressed 'INPUT', the 'offset', is specified in the same way. This number is added to each pixel of the output image to remove any negative intensities. One more press of 'INPUT' completes this operation. Having specified a filter, 'DEFINE FILTER' can be used again to change one or more of the weights, scale or offset without affecting any other term. If the filter is symmetric, time can be saved by using the 'SYMMETRY' command below.

SYMMETRY

SUMMARY

If there is symmetry in a filter, this command will replicate weights, lines of weights etc, so that filters can be defined quickly.

COMMAND IN DETAIL

The only item required is the weight(s) to be replicated.

- TL PIX - The top left weight is copied into every location.
- TOP 1/2 - The top half of the filter is copied into the bottom half.
If there are 5 rows, the top two rows are copied into the bottom two, and similarly for 3 rows.
- TOP ROW - The top row is copied into every row.
- TL 1/4 - The top left quarter is copied into the other quarters.
If there are 5 rows, only weights from the top two rows are copied, etc.
- LEFT 1/2 - The left half of the filter is copied to the right.
If there are 5 columns, only two are copied, etc.
- LEFT COL - The leftmost column is copied into every column.

Having chosen the required option with the cursor, the 'INPUT' button is pressed and the weights in the filter are changed.

FILTER IMAGE

SUMMARY

The filter shown in the menu is applied to the image(s) with this command.

COMMAND IN DETAIL

The only secondary commands define the stores to be filtered, the stores for the results, and the area to be filtered. This area is specified as the whole area, the part area inside the rectangle, (the position of which can be changed with the rolling ball, and the size of which can be varied by pressing 'CHNG' and moving the rolling ball), or the last area specified.

SAVE FILTER

SUMMARY

This command enables a user defined filter to be stored in the host computer.

COMMAND IN DETAIL

The letter used to identify the saved filter has to be chosen. A line of up to 70 text characters can be inserted at the terminal, and this is stored as a descriptor for the filter. If no text is required, 'INPUT' is pressed. The filter weights, the scale and the offset are also stored.

RESTORE FILTER

SUMMARY

This command enables a previously stored filter to be retrieved. The header text appears on the terminal screen.

SHIFT AND ROTATE

SUMMARY

Using this command, a filter created using 'DEFINE FILTER' or a restored filter, can be shifted in the x or y direction, by integer or fractional amounts of one pixel spacing. The filter can also be rotated.

COMMAND IN DETAIL

The shift in the x direction is required first and a number appears on the screen with a small bar above it. If the rolling ball is moved to the right, then the number is incremented starting at the digit below the bar. Moving the ball to the left causes the number to decrement. If 'CHNG' is pressed, the small bar can be moved to the left or right. This is useful if a number with many digits is to then be entered, since

the two least significant digits could be entered, the bar moved to the left and the next two digits entered, etc. If 'CHNG' is pressed again, the decimal point is moved by the rolling ball. These three operations of changing a number, moving the bar and moving the decimal point, cycle round continuously, enabling decimal numbers with many digits to be entered quickly.

The shift in the y direction is next entered in the same way, and finally the rotation angle in degrees is given. A rotation of 90 deg will cause a filter weight right of centre to move to above centre, ie rotation is positive in an anticlockwise direction.

It is assumed that the filter is surrounded by weights all of which are zero. Thus if the filter is moved one pixel to the right, there will be a column of zeros at the left. The shift command is most useful for fractional pixel shifts. A sinc function is used to resample the filter; weights outside the area of the filter patch on the screen are discarded. This implies that, for example, a rotation of +12 deg followed by one of -12 deg, will not normally result in a filter having exactly the same weights as the original. Therefore, rotation should only be applied once. If, for example, an edge enhancement filter is to be applied at 15 deg, 30 deg, 45 deg, . . . , to produce a set of images with enhancements in different directions, the basic filter should be saved, and should be restored each time for each rotation, rather than rotating 15 deg, applying the filter, rotating 15 deg again, etc.

READ IMAGE

SUMMARY

This command allows the user to extract a patch from an image and write the numbers into the filter array.

COMMAND IN DETAIL

The only requirements are to specify the store to be read, and to place the box over the area of interest, using the rolling ball. The size (up to 5 by 5) of the box can be varied by pressing 'CHNG' and using the rolling ball.

WRITE IMAGE

SUMMARY

This command is the inverse of READ IMAGE in that a patch of up to 5 by 5 elements can be inserted into the store specified by the user.

5.8 Classifiers. Copy Bit Plane

In 'Density Slice, Measure Areas' (section 5.5) it was suggested that in a black and white image, a particular feature may be characterised by a certain shade of grey, (ie water may lie between black and dark grey in many images). However this simple rule may result in many errors. In this example, some areas of water may lie outside the range black to dark grey, and other features e.g. shadows, may lie within the range giving errors of both types, i.e. excluding some of the required feature and including parts of other features. This density slice could be performed on a second image (spectral band) where it may be found that different parts of other features are included and are thus erroneous. If the second slice is applied only to the areas sliced in the first band, there should be an improvement in the number of errors introduced. This process could be repeated for all the spectral bands (up to 16 in GEMSTONE) but it would be very tedious. Classifiers allow the user to do this multi-band selection with the minimum of effort.

There are three different types of classifier in GEMSTONE, presented here in increasing order of complexity, theoretical accuracy, and time of execution. They all require the user to identify one or more 'training areas'. For example if a small box is put totally inside an area which is known to be woodland, the box would enclose a training area for woodland. The three classifiers use the information differently, but ultimately produce images of areas having similar spectral characteristics to the training area. For this example, all the classifiers should produce images of the areas of woodland.

In the case of the box classifier, having been supplied with these training areas, the system will examine only pixels within these areas and will produce histograms for each spectral band. Consider the classifier operating on only one band, initially. As illustrated in Fig 9 (p22), a histogram is likely to have long 'tails', and in this one, the bulk of the data is between 50 and 150, with tails between 25 to 50 and 150 to 200. A simple density slice from 50 to 150 would include the bulk of the data. If in the same image, there is a much brighter class with a distribution centred on 200 say, then the lower tail of this distribution will overlap the upper tail of the previous one. Extending the slice limits of the previous one to 25 and 200 will therefore add a few extra correct pixels in the tails, but will add many incorrect pixels from the brighter class. Hence the tails of the distributions are sacrificed to reduce the errors, and for this classifier 1% of the distribution is neglected in each tail. A single band classification is a density slice where the limits are at the 1% and 99% points of the histogram of the training data. When the classifier has been operated, these numbers appear on the chapter page alongside the corresponding store number. For a multi-band image, this process is repeated so that a pixel is included in a class only if it passes the separate tests in all bands.

In the case of the discrete classifier, the pixels in the training area are not treated as samples of a more general distribution. Each pixel is treated as a separate training area, and the whole image is examined for pixels that have exactly the same intensities in all the bands, as one of the pixels in the training area. For example, if there is a single band image and only four training pixels with intensities, 1, 3, 4, 6, the classifier will only select a pixel in the image if the intensity of the pixel is 1, 3, 4, or 6. If there is a small number of pixels in the training area, the process is too discriminating. In such cases, a small n-dimensional box is constructed around

each pixel and the size of this box, can be changed by the user. If the box size is 2, the total intensity range (0-255) will be divided into 128 groups of 2 intensity levels, i.e. 0/1, 2/3, 4/5, 6/7, etc. Thus for the example above, the training pixel with intensity 1 lies within group 0/1, and hence any pixel in the image with a value of 0 or 1 will be classified. Similarly training pixel values of 3, 4, 6, lie within groups 2/3, 4/5, 6/7, respectively, and hence any image pixel in the range 0 to 7 will be classified. For a box size of three, there are 86 groups (i.e. 0/1/2, 3/4/5, etc).

This classifier therefore considers the training area as the complete set of individual pixel values, and will give good results where large training areas are used, and/or where the user is sure that the training area contains all the pixel values required. Unlike the other two classifiers it will operate on a multi-modal distribution. For example, in a mixed woodland with individual trees, which are spectrally quite different, it will search the image only for those particular pixel values. The other classifiers treat these samples of many separate distributions as samples of one, and the results are inferior. The difficulty with this classifier, is in ensuring that the training area actually contains pixels with all the values of interest.

For the maximum likelihood classifier, the pixels of the training area are treated as samples of a class, which is assumed to have a Gaussian distribution. From statistical theory, the probability density function of such a distribution is

$$\text{Den}(X) = \frac{1}{(2\pi)^{n/2}} \cdot \frac{1}{\det(C)} \cdot e^{-\frac{1}{2}(X-M)' \cdot C^{-1} \cdot (X-M)}$$

where C is the covariance matrix of the training area (see page 66 for the definition of the covariance matrix), M is the intensity vector of the mean of the training area, n is the number of bands or dimensions, and X is the intensity vector of a pixel in the class. The likelihood of observing one single point X, given that the pixel is in the class, is also the same function. Note that this is not the probability, since the probability of observing a pixel within a given range of values is the integral of Den(X) over that range, the probability of observing a single value is the integral of Den(X) over zero range, and is thus zero. It is convenient to use the log form of the likelihood, Lik, as in

$$\log\{\text{Lik}(X)\} = -\frac{n}{2} \log(2\pi) - \frac{1}{2} \log\{\det(C)\} - \frac{1}{2} (X-M)' \cdot C^{-1} \cdot (X-M) \quad \text{---(3).}$$

The first two terms on the right hand side are constant for one training area, and the last term is a measure of the squared distance of a pixel from the mean of the training area, normalised by the covariance matrix. For a single class and single band it is simply necessary to evaluate this last term, i.e.

$$\log\{\text{Lik}(x)\} = -\frac{1}{2} \cdot \frac{(x-m)^2}{\sigma^2} \quad \text{---(4),}$$

where x is the intensity of the pixel being tested, m and σ are the mean and standard deviation of the training area distribution, and the 1/2 in equation (3) has been neglected. If $x=m$, then from equation (4), we have the reasonable result, that the log likelihood is a maximum when the pixel intensity is the same as the mean. The log likelihood reduces as x departs from the mean, m, in a parabolic manner.

The measure of likelihood stored in the output image from this classifier in GEMSTONE is the log likelihood as defined in equation (4) for one band, or the last term of equation (3) (without the constant 1/2) for multi-band images.

This classifier is generally used to make decisions about whether a pixel belongs to class 1, or class 2, or class 3, etc. For this decision making operation, the last two terms of equation (3) are used. In the case of a single band, this reduces to,

$$\log\{\text{Lik}_1(x)\} = \log\{\sigma_1^2\} - (x - m_1)^2 / \sigma_1^2,$$

where $\text{Lik}_1(x)$ is the likelihood of the pixel with intensity x being in class 1,

and m_1 and σ_1 are the mean and standard deviation of the class 1 distribution.

Similarly, the likelihood that the pixel with intensity x belongs to class 2 is,

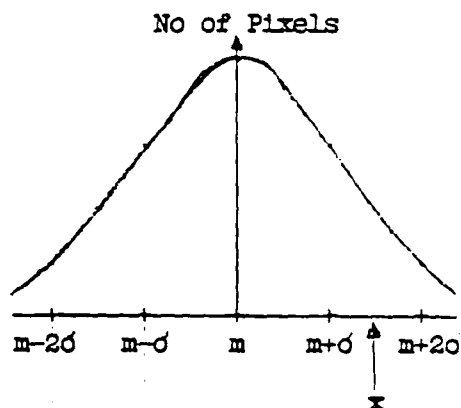
$$\log\{\text{Lik}_2(x)\} = \log\{\sigma_2^2\} - (x - m_2)^2 / \sigma_2^2.$$

These last two expressions are evaluated for every pixel in the image, and if $\log\{\text{Lik}_1(x)\}$ is greater than $\log\{\text{Lik}_2(x)\}$ for a pixel with intensity x ,

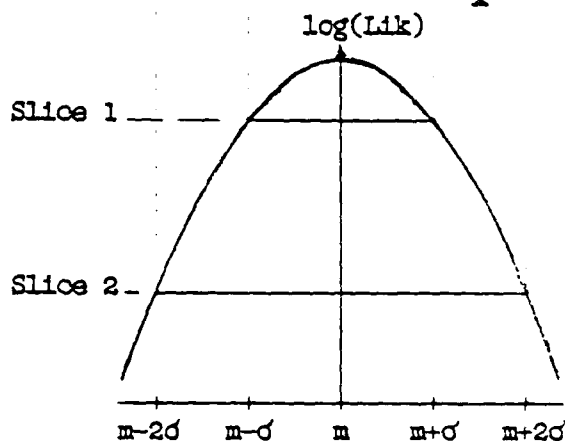
then that pixel is assigned to class 1, (or to class 2 if the opposite is true). This is the most elementary form of the decision rule used in this classifier. In the general case of multiple bands and multiple classes, the last two terms of equation (3) are evaluated for every class and every pixel. The result is that every pixel in the image will have one likelihood of belonging to class 1, and another of belonging to class 2, etc. The same decision rule is applied, so that every pixel then has associated with it the number of the class to which it is most likely to belong.

The scene can now be classified and so, for example, if there are four classes, every pixel would be assigned to its most likely class and there would be four classes displayed in four colours on the screen. There would be no pixel left as 'unclassified'. Taking this concept to the limiting case of only one class, since every pixel would have a finite likelihood of belonging to the one class, every pixel in the image would be classified as belonging to that class. This is obviously unreasonable, since a pixel with an infinitesimally small likelihood would still be classified. In GEMSTONE, the user is allowed to make one further decision to get round this difficulty. Earlier, it was stated that a Gaussian distribution is assumed for the intensities of the pixels in the class. Considering one band again, or a simple 1-D Gaussian distribution it can be shown that 68% of the points in the distribution lie between $m - \sigma$ and $m + \sigma$, where m is the mean and σ is the standard deviation of the distribution, and that 95.6% of the points lie between $m - 2\sigma$ and $m + 2\sigma$.

In the Gaussian distribution shown here, a density slice from $m-\sigma$ to $m+\sigma$, would colour in 68% of the pixels in the class, and the pixel at x would be excluded. Changing the slice to the range $m-2\sigma$ to $m+2\sigma$ would colour in 95.6% of the pixels in the class, and the point x would be included. Thus, with a density slice, the user can make a decision on the significance of point x , etc.



In GEMSTONE, it is the likelihood that is sliced, since the basic output of the classifier is the log likelihood that a pixel belongs to the class. The graph of log likelihood for one class and one band is shown here (of equation (4)). Both this graph and the Gaussian distribution have the same horizontal axis and therefore a slice of likelihood corresponds uniquely to a proportion of the total number of pixels in the class. For example, Slice 1 in the $\log(\text{Lik})$ graph corresponds to the range $m-\sigma$ to $m+\sigma$, which includes 68% of the data. Similarly Slice 2 corresponds to the range $m-2\sigma$ to $m+2\sigma$.



There are 16 slice levels allowed in GEMSTONE. They are set at the following levels, where the numbers of pixels unsliced form a geometric progression and the factor between successive numbers is the fifth root of ten. This gives a good overall range and a reasonable resolution.

Level	Percentage of pixels in class which are,	
	sliced	not sliced
15	60.2	39.8
14	74.9	25.1
13	84.2	15.8
12	90.0	10.0
11	93.69	6.31
10	96.02	3.98
9	97.49	2.51
8	98.42	1.58
7	99.00	1.00
6	99.369	0.631
5	99.602	0.398
4	99.749	0.251
3	99.842	0.158
2	99.900	0.100
1	99.9369	0.0631
0	100	0

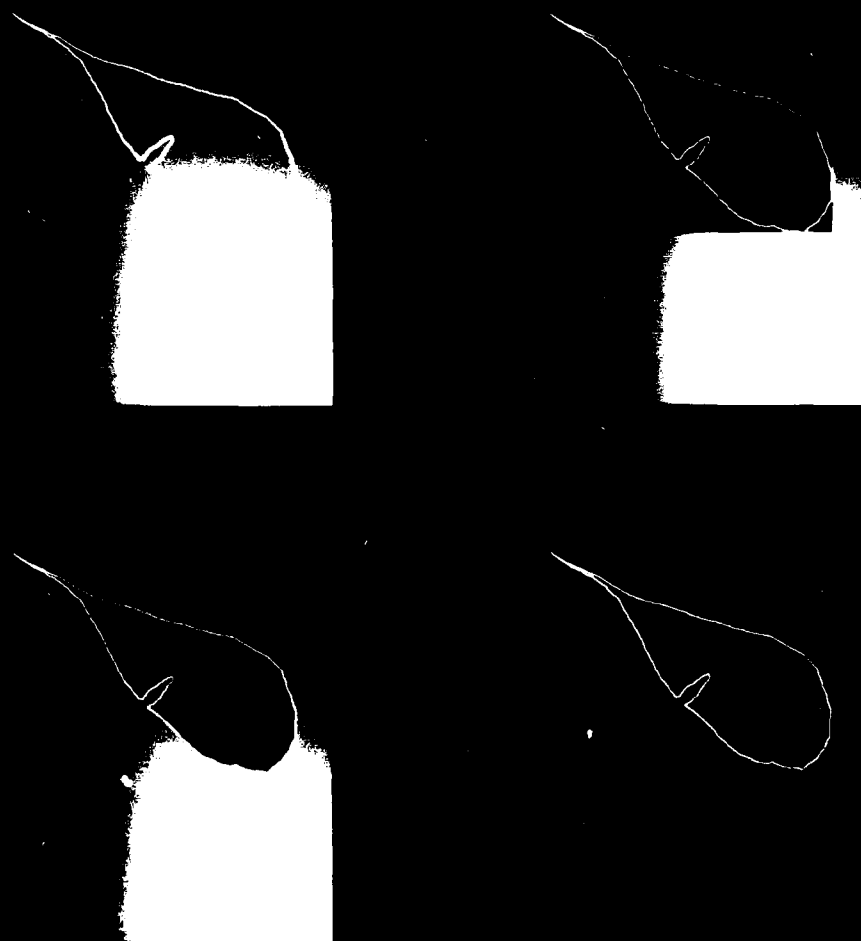
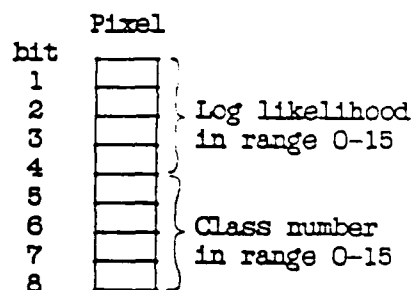


Fig 33 Classifier examples

The output image from the classifier contains two items of information for each pixel, namely, the most likely class to which the pixels belongs, and the measure of likelihood. There can be 16 classes and 16 likelihood levels, which are stored in the output pixel as shown. Because of this mixture of information, the 'Density Slice' commands cannot be used to slice the likelihood. The special command, 'DISPLAY MAX LIK' must be used.



The effect of these classifiers is illustrated in Fig 33, where the two single band images at the top are combined to form the colour picture at the middle left, by putting the top left image on the red gun of the TV, and the top right on the green. Suppose it is required to find all the areas that have a range of orange-yellow colours. The training area outline shown in the colour picture at the middle left covers most of the required colour range, although in drawing it, a mistake in the form of a kink, has been made. The performance of the three classifiers can be compared using the last three images in Fig 33, by noting how well they classify the training area. In a real application of course, the image would not have pixels whose intensities depend on spatial position, and hence other areas in the image would be classified.

The result of using the box classifier with this training area is shown at the middle right, where the classified area is in blue. Since the limits of the box are set at the 1% and 99% points of the distribution in each spectral band, a small part of the training area remains unclassified. This can be seen at the top left of the example training area. Many pixels which are not orange-yellow have been classified (eg green ones at the bottom left, and red at the top right). This illustrates the limitation of the box classifier, in that many areas quite different from the training area, can be included. However, it is fast and simple.

The result of using the discrete classifier is shown at the bottom left. Only those pixels within the image, which have exactly the same values in both spectral bands as ones in the training area, are classified. Hence in this example, the classified area is the same as the training area. Where the human error was made in drawing the training area, the image remains unclassified. This illustrates the limitation, in that the classification will be in error unless the training area contains an example of every pixel intensity required. In GEMSTONE, a small box can be constructed round the intensity point of each pixel in the training area, thereby alleviating this drawback. This classifier is extremely precise, takes longer to run than the box version, and should be used with care.

The result of using the maximum likelihood classifier is shown at the bottom right, where the intensity corresponds to the 16 possible levels of likelihood. Thus the brightest part in the image contains the pixels most likely to belong to the class. By slicing the image (using 'DISPLAY MAX LIK'), the image can be divided into two regions, in the class and outside. The slice level can be varied by the user. This classifier treats the training area as a sample of a distribution, and is thus relatively immune to the human error (cf discrete) and it results in a class which fits the training area very well (cf box). It is the most complex of the three classifiers, but should give the best results, when the process is understood.

CHOOSE CLASSIFIER	MANUAL		OK			D1	T	T	D1
DEFINE TR AREA	BOX		BOX			D2	T	T	D2
COPY BIT PLANE	DISCRETE		DRAW			D3	T	T	D3
CHOOSE TR AREA	MAX LIK		PAINT			D4	T	T	D4
CHOOSE BANDS	COLOURS	NONE	RED			5	T	T	5
	NUMBERS	GREEN	BLUE			6	T	T	6
CLASSIFY	ALL					7	T	T	7
DISPLAY BITPLANE	1					8	T	T	8
DISPLAY MAX LIK	2	OVLY 2				9	T	T	9
SAVE	3					10	T	T	10
RESTORE	4	OVLY 3				11	T	T	11
RESTORE STRETCH	5					12	T	T	12
	6	OVLY 4				13	T	T	13
	7					14	T	T	14
	8	STORE				15	T	T	15
						16	T	T	16
						05	T	T	05
						06	T	T	06
						07	T	T	07
						08	T	T	08
						+END			

NONE	BA
A	BB
B	BC
C	BD
D	BE
E	BF
F	BG
G	BH
H	MA
	MB

STATUS	HELP	RETURN	IMAGE
--------	------	--------	-------

INPUT FOR COMMAND OR QUIT

Fig 34 The 'Classifiers, Copy Bit Plane' page

CHOOSE CLASSIFIER

SUMMARY

One of the four classifiers can be selected. 'Manual' is the same as the box classifier, except that the numerical limits for the box are entered by the user. The 'box', 'discrete' and 'max lik' are as described on the previous pages.

DEFINE TR AREA

SUMMARY

With this command, one or more training areas can be stored in overlay plane 4.

COMMAND IN DETAIL

There are 5 secondary commands. 'OK' is used to inform GEMSTONE that the definition of training areas has been completed. The user is then returned to the primary commands. 'Clear' erases overlay plane 4 prior to the definition of a new training area. 'Box', 'draw' and 'paint' can all be used to define training areas, using rectangles, irregular outlines, or irregular areas, respectively.

In the case of 'box', the rolling ball is used to move the rectangle. Its size can be changed using the rolling ball after pressing the 'CHNG' button. At the bottom right of the screen, the position of the top left corner of the box is given when the box is being moved, and the size of the box is given when this is being changed. These two operations continue to cycle with further pressings of the 'CHNG' button. When the box is totally within the homogeneous area to be used for training, 'INPUT' is pressed, to return the user to the secondary commands. The number of pixels in the training area is displayed on the VDU. A further training area can be added, using any of the three methods. In this way, many small training areas can be chosen as examples of one class, and when this process is complete, 'OK' is selected. Every pixel inside the box, including those underneath the boundary of the box, is used to produce the statistics of the training area. It is therefore important to ensure that no box extends outside the homogeneous area, or erroneous classifications will be produced.

In the case of 'draw', the rolling ball moves the point at which the line is to start. After pressing 'CHNG', the rolling ball is then used to draw the line which is the boundary of the homogeneous area to be used for training. When 'CHNG' is pressed again, GEMSTONE joins the last point to the first, fills in the enclosed area, and displays on the VDU the number of pixels in this training area, together with the total number of pixels in all the training areas for this class, so far selected. The rolling ball can be moved again to describe further training areas. Pressing 'INPUT' completes the operation and the user is returned to the secondary commands.

For 'paint', the rolling ball is used to move the cursor to a point inside the required area. When 'CHNG' is pressed, and the rolling ball moved, it is as if there were a paint brush under the centre of the cursor and the training area can be described by 'painting it in'. The size of the brush can be varied by changing to mode 1 (ie by pressing the 'MODE' button so that the light under the button is extinguished). Moving the rolling ball to the right causes the 'brush' to widen, and to the left causes it to decrease to a minimum of one pixel wide. Returning to mode 0 by pressing the mode button again, results in the rolling ball controlling the painting again. If 'CHNG' is pressed once more, and the ball moved, the 'brush' now 'paints out' or 'erases' any defined training area. The size of the 'brush' can be varied as before. Note that this 'painting out' process is the way in which a training area may be removed. Pressing the 'CHNG' button again allows the user to move the starting point for another painting operation. The current function of the rolling ball (ie move, draw or erase) is displayed on the VDU. When 'INPUT' is pressed, the operation is completed, and the user is returned to the secondary commands.

COPY BIT PLANE

SUMMARY

The contents of one bit plane or overlay plane may be copied to another plane using this command.

COMMAND IN DETAIL

Since the command 'DEFINE TR AREA' always puts the training areas into overlay plane 4, 'COPY BIT PLANE' is often used to copy plane 4 to bit planes in stores. For example, an 8 class maximum likelihood classification would require 8 training area sets, one for each class. The set of training areas representing class 1 could be copied from overlay plane 4 into store 1 bit plane 1. The set representing class 2 could be copied into store 1 bit plane 2, etc. Equally, for the single class classifiers, the results are always put into overlay plane 3. For multiple executions of such classifiers for multiple classes, the results can be accumulated in bit planes of stores, by this copying process. Careful notes should be kept of where training areas and classification results are stored, otherwise a complete muddle can be produced when using this command.

The user is first asked to choose the source data for the copy and overlay planes 2,3,4 as well as any (or all) bit plane of any store can be used. If the source 'Zeroes' is selected, the number 0 will be copied. This is the method used to erase one (or many) bit plane. The bit plane(s) into which the data are to be copied is next specified. Overlay planes 2,3,4 and any (or all) bit plane of any store can be used. If all 8 bits of one store are copied to all 8 of another, the user has the further option, in that the data can be copied through a look-up table. This feature can be used for saving final results from the maximum likelihood classifier. Either the red, green, or blue LUT can be specified, or none at all in which case, the data are copied unmodified.

CHOOSE TR AREA

SUMMARY

The bit planes containing the training areas for the classifiers, are specified with this command.

COMMAND IN DETAIL

In the case of the box and discrete classifiers, at least one bit plane must be specified as the source of the training data. If the user has put two examples of training areas for one class, into two bit planes, then the classifiers can be run using one or other or both examples. This is very useful for investigating the contribution of different training areas. The user simply chooses the bit plane or planes containing the required combination of training areas. For the maximum likelihood classifier, one and only one bit plane is required for each class up to a maximum of 16. Thus, if 8 training areas are stored in the 8 bit planes of store 1, and they are all selected, a classification with 8 classes will result. However, if only 5 bit planes are chosen, only 5 classes will result.

CHOOSE BANDS

SUMMARY

This command is used to select the stores containing the bands of data to be used by any of the classifiers. Up to 16 may be selected.

CLASSIFY

SUMMARY

This command is used to execute the classification.

COMMAND IN DETAIL

Before using this command, many of the previous commands must have been executed. The type of classifier must have been selected using 'CHOOSE CLASSIFIER'. Training areas must have been defined with 'DEFINE TR AREA'. Multiple training areas must have been copied out of overlay plane 4 with 'COPY BIT PLANE'. The training areas to be used must have been specified with 'CHOOSE TR AREA', and the bands to be used by the classifier must have been selected with 'CHOOSE BANDS'. If one or more of these operations has not been performed, GEMSTONE will ask the user to supply the required information.

If the manual classifier was selected (in 'CHOOSE CLASSIFIER'), the user is asked for the limits of the box in each band. Initially both the upper and lower limits are set at 128 and displayed on the TV screen opposite the numbers of the selected stores. If the manual or box classifier was used immediately before, the previous limits are displayed on the screen. Moving the rolling ball to the right or left will increase or decrease respectively, the lower limit of the first

band, and this is displayed on the screen. When the desired value is displayed, 'INPUT' is pressed and the upper limit of the first band can be varied using the rolling ball. After pressing 'INPUT' again, the lower limit of the second band can be altered, and so on until all the limits have been set. The cursor moves each time to indicate which limit is being varied. Whether limits have been changed or not, 'INPUT' must be pressed for each upper and lower limit in each band. On completion, the classification is performed, the result displayed in overlay plane 3, and the number of pixels in the class, the % of the total number of pixels in the class, and the total number of pixels in the image all appear on the VDU. When the classifier is next run, overlay plane 3 will be overwritten. Therefore, to preserve a classification, it should be copied to another bit plane (generally in one of the stores) using the 'COPY BIT PLANE' command.

If the box classifier is selected, the classification proceeds immediately by examining the data in each band, within the training areas, and the means and standard deviations of these data are displayed on the VDU. The limits of the box are displayed on the TV. Overlay plane 3 is again used to hold the classification, and the number of pixels in the class, the % of the total, and the total all appear on the VDU. The results should be copied to a store if they are to be saved.

If the discrete classifier is selected, the user is asked for the size of the boxes in each dimension, as described on pages 95 and 96. The size in each dimension is displayed to the right of the store number to which it refers. If the cursor is moved to the size to be changed, and the 'INPUT' button pressed, then the user can change the size with the rolling ball. Pressing 'INPUT' again allows the user to select another dimension, and so on. The sequence is finished with the 'QUIT' button, whereupon the classification process starts, with the results being produced in overlay plane 3. The number of pixels in the class, the % of the total, and the total all appear on the VDU. The results should be copied to a store if they are to be saved.

If the maximum likelihood classifier is selected, the process starts immediately and the means of all the bands of each training area, together with the number of pixels in each training area are displayed on the VDU. The user is next asked for the store to be used for the results of the classification. The standard deviations of the data within each training area are then displayed on the VDU. These standard deviations relate to the principal component axes for each training area. At the conclusion of the classification, GEMSTONE passes automatically onto the command 'DISPLAY MAX LIK' described below.

DISPLAY BIT PLANE

SUMMARY

The contents of one, two or three bit planes can be examined quickly with this command.

COMMAND IN DETAIL

Suppose a number of training areas have been stored in the bit planes of store 1. If the user asks for bit one of this store to be displayed in red, then only in those places where bit one is set will the maximum red (ie level 255) be displayed, and red will be set to zero elsewhere. For this operation, all the locations in the red LUT up to and including level 127 have their contents set to zero, and all locations above 127 have their contents set to 255. By a similar method, bit plane 2 could be displayed in green. The resulting image would show all the regions in the first training area in red, and all the regions in the second training area in green. A region in both training areas (which would normally be an error), would be both red and green, ie yellow.

The secondary commands for this operation are to choose the bit plane and the colour for the display. If 'all' bit planes are chosen, this implies that all bits must be seen, and so the LUT is reset so that location 0 contains 0, location 1 contains 1, etc. If the colour 'none' is chosen, this implies a black and white image, and hence all the LUT's are made the same. This command can be used to reset all the LUT's very quickly, by choosing 'all' bit planes and setting the colour to 'none'.

DISPLAY MAX LIK

SUMMARY

This command is used to examine a maximum likelihood classification, to select the reject level, and to set the LUT's for results storage.

COMMAND IN DETAIL

A special slice is applied to the image by this command. The colours corresponding to the 16 classes are shown in the following table.

Class no.	Colour	Red	Green	Blue
0	Black	0	0	0
1	Red	255	0	0
2	Green	0	255	0
3	Blue	0	0	255
4	Yellow	255	255	0
5	Cyan	0	255	255
6	Magenta	255	0	255
7	Grey	128	128	128
8	Brown	80	48	48
9	Gold	255	192	0
10	Orange	255	128	0
11	Purple	128	0	128
12	Dark Blue	0	0	64
13	Dark Green	0	128	0
14	Dark Red	128	0	0
15	Petrol Blue	64	128	128
16	Pinky Purple	192	64	128

Table 8 Colours of displayed classes.

The special slice is set initially so that 1.6% of the data within a class are rejected. As the rolling ball is moved to the right more can be rejected, and vice-versa. This reject level is displayed on the VDU, and the whole range of possible reject levels is given on page 98. Note that the region left as black is the unclassified region and is referred to as class 0 in table 8 and elsewhere.

Having adjusted the reject level to display the best compromise between including as much of the class as possible, and yet not including erroneous pixels, the user presses 'INPUT'. The next choice is whether to leave the LUT's so that the colour picture remains on the screen, or to change the LUT's so that class 1 is displayed at level 1, class 2 at level 2, etc, with the unclassified areas at level 0. This produces a very dark picture but it is very useful for the storage of the classified image in a very simple form, using the 'COPY BIT PLANE' command. Thus, either 'colours' or 'numbers' respectively, is selected and 'INPUT' is pressed, whereupon the LUT's are changed if necessary, and a display of the numbers of pixels in each class with percentages appears on the VDU.

If 'numbers' is selected and the image is stored through the LUT's, the colour picture can be reproduced simply by slicing the image at levels 0, 1, 2, ..., 16. Within 'Density Slice' there is a standard slice called '16-level' (p41) which reproduces these slice levels and the colours in table 8.

SAVE

SUMMARY

It is possible to save classification parameters using this command.

COMMAND IN DETAIL

For the box classifier, the upper and lower limits in each band are stored. Up to eight sets of classification parameters can be store for the box classifier, and these are labelled BA to EH (for Box-A etc). Up to two sets can be stored for the maximum likelihood classifier, labelled MA and MB. These classification parameters can be recalled for application to the same scene, a different scene, or for use in a background program to process a much larger scene.

The only secondary command is to specify the name of the classification, B1, B2, ..., B8, for the box classifier, or M1, M2, for the maximum likelihood classifier. A line of up to 70 text characters can be inserted at the terminal, and this is stored as a descriptor for the classification parameters.

RESTORE

SUMMARY

This command allows the user to run a previously stored classification.

COMMAND IN DETAIL

One of the 8 box classifications, or one of the 2 maximum likelihood classifications must be selected. The header text appears on the terminal screen, including the number of bands used in the classification. The stores for these bands need to be selected again. With this command, many scenes can be classified using the same classification parameters, and so if a classification of a large image is required, it can be divided into a number of store sized images, and they can be classified separately. The resulting set of classified images can then be joined together in the host computer to produce the classification of the whole large image.

RESTORE STRETCH

SUMMARY

This command is exactly the same as 'RESTORE STRETCH' in the contrast stretch chapter, (p31). It is here simply for convenience.

COMMAND IN DETAIL

When such operations as 'DISPLAY MAX LIK' are performed, any previously applied contrast stretch is lost. Provided stretches have been saved while in the contrast stretch chapter, they can be restored here. The only secondary command is to choose the stretch.

OUTPUT PLANE	CLEAR PLANE	BOX	WHOLE BOX	OVLY 4 BOX	NONE
PIXEL VALUE	PIXEL COLOUR	CIRCLE	COPY SPRITE	OVLY 4 COPY SPRITE	A
WRITE TEXT	COPY AREA	LINES	MOVE SPRITE	OVLY 4 MOVE SPRITE	B
PAINT	DRAW LINES	LAST LINES			C
POLYGON FILL	AREA FILL	AUTO FILL			D
SEED FILL	DRAW SPRITE	AUTO INCR.	NONE	RED	E
GREY SCALE	COLOUR PALETTE	AUTO INTERP	GREEN	BLUE	F
LINEAR STRETCH	COLOUR ROTATE	1 2	AUTO	CURSOR	G
3-D FIXED	3-D GENERAL	3 4	FIXED	NUMBERS	H
SAVE STRETCH	RESTORE STRETCH	OV 4 5			
		6 7	BOTTOM	LEFT	
		8	TOP	RIGHT	
		STATUS HELP RETURN IMAGE			

INPUT FOR COMMAND OR QUIT

Fig 35 The 'Text and Graphics' page

5.9 Text and Graphics

This chapter contains a number of commands for:

- (1) Writing text in an overlay plane.
- (2) Writing text in an image within an image store.
- (3) Writing text in a contrasting background within an image.
- (4) Writing text with arrows or pointers etc.
- (5) Drawing straight or curved lines (eg for direct interpretation of images on the screen, for geological structure, roads, region outlines; production of display material or slides with flow charts, diagrams, etc).
- (6) Filling in areas for other processes (eg training areas for classifiers, areas defined in overlay plane 4 for such operations as 'OVLY 4 HISTOGRAM' (p34), masking operations, flow charts, diagrams etc).
- (7) Drawing sprites.
- (8) Erasing parts of the graphics, or images, using 'PAINT'.
- (9) Copying parts of an image to other positions in the same or another image.
- (10) Modifying the LUTs for B/W or colour presentations.
- (11) Changing the LUTs continuously to provide attractive demonstration graphics.
- (12) Making 3-D diagrams of intensity.

These operations vary widely in complexity from writing text (requiring only 'WRITE TEXT' to put text in overlay plane 4, with perhaps 'CLEAR PLANE' to remove previous text), to drawing sprites where, not only are the commands fairly complicated, but the interactions with other commands are also quite complex.

OUTPUT PLANE

SUMMARY

This command is used to specify the plane into which the graphics are to be written. Either one of the first four stores or overlay plane 4 may be used. If a graphics function eg 'WRITE TEXT' is run without selecting a plane, overlay plane 4 is assumed. The current plane number is written in the menu above 'Help'.

CLEAR PLANE

SUMMARY

Having chosen the store for the graphics using 'OUTPUT PLANE', it can be erased completely with this command.

PIXEL VALUE

SUMMARY

It is possible to write the graphics into a store at any desired grey level. If a graphics function (eg 'WRITE TEXT') is run without selecting a pixel value (or grey level), 255 is assumed.

COMMAND IN DETAIL

The only secondary command is to select the required grey level of the graphics by specifying 'number', in which case the rolling ball controls the grey level and this number is displayed on the screen. Alternatively, for 'cursor', the rolling ball can be used to move the cursor over a pixel in some existing graphics or over a pixel in a grey scale etc. The grey level of the chosen pixel is displayed on the screen. For both methods 'INPUT' is pressed to select the displayed value, and this number is copied to the menu above 'Return'. For many purposes such as writing text or masks in overlay plane 4, this command need not be exercised, since a value of 255 is then assumed. Note that overlay plane 4 is treated as a store with only one bit plane, the MSB. Thus for the overlay plane, any number over 127 can be used. For the stores, if 128 is chosen, then the MSB will be set to 1 and the others to zero. This is not the same as writing to one bit plane since the other bit planes may be altered.

A typical sequence of operations would be as follows. Choose one of the four stores for the output, erase it, choose 'PIXEL VALUE' at level 1, choose 'PIXEL COLOUR' (see below) as 255 on red, 0 on green and blue, and then draw the graphics which would appear as red and be stored at level 1. If 'numbers' were selected next time 'PIXEL VALUE' is used, GEMSTONE would increment the value by one, ie level 2 in this example, and another colour eg 0 on red and blue and 255 on green, could be selected with 'PIXEL COLOUR'. This second set of graphics would appear in green, alongside the previous graphics in red, and would be stored at level 2. Thus 256 sets of graphics could be held in one store, and be displayed in 256 colours selected from a range of $256 \times 256 \times 256$ possibilities.

PIXEL COLOUR

SUMMARY

In order to make the graphics clearer, or to display them in their final colours, it is possible to alter the numbers in the LUT location corresponding to the pixel value selected with the 'PIXEL VALUE' command.

COMMAND IN DETAIL

If 'cursor' is selected to define a colour, the cursor is moved with the rolling ball over the required colour which may be some existing graphics or a section of the colour palette (see below), etc. The intensities in red, green and blue are displayed on the screen, and the 'INPUT' button pressed to select the colour. If 'numbers' is selected, the rolling ball is first used to set the level of the red component of the colour. After pressing the 'CHNG' button the green component can be changed and so on with blue, and then red again if required etc. The 'INPUT' button is pressed when the required colour has been produced. The default colour is to set all the LUTs to the value of the grey level chosen with 'PIXEL VALUE'. If neither of the commands 'PIXEL VALUE' or 'PIXEL COLOUR' is exercised, the default pixel value is 255 and hence the default colour is 255 on red, green and blue (ie white).

WRITE TEXT

SUMMARY

Text can be written into one of the first four stores or overlay plane 4.

COMMAND IN DETAIL

Before running this command, the user can specify the store into which the text is to be written, the grey level at which it is to be written, and the settings of the LUTs so that the text is in colour. The commands required are 'OUTPUT PLANE', 'PIXEL VALUE', and 'PIXEL COLOUR' respectively. However, text can be written without specifying any of these, in which case the default values are used and the results are placed in overlay plane 4.

The text is first typed on the computer terminal. Any character can be used including upper and lower case versions. There are two pitfalls when trying to write lower case alphabetic characters. Some terminals have a key labelled 'TTY CAPS' or similar, which when it is on, has the effect of making all the alphabetic characters upper case irrespective of the 'SHIFT' key but leaves the others, eg the numeric keys, as lower case. Secondly, there is generally a key labelled 'SHIFT LOCK' which makes all keys upper case. Both of these keys should be off. Many lines of text can be typed in at once, each being terminated with the 'RETURN' key and the whole text is finished with a blank line. This is the same as pressing the 'RETURN' key twice. If the user wishes to have a blank line within several lines of text, this can be achieved by typing a space character and 'RETURN'.

Mistakes can be corrected by backspacing and overtyping. Unfortunately, terminals vary in the key used to backspace. If there is a key marked 'BACK SPACE' or 'BS' this should be used. If there is no such key on the keyboard, the next possibility is to find a set of four arrows pointing up, down, left and right. The key with the left arrow will generally backspace correctly. If a keyboard has none of these keys, there is most likely a single key with a left pointing arrow and this will generally work. Where a keyboard has both the set of arrows and a single arrow on its own, the single arrow key is likely to put a left pointing arrow into the text. Many terminals have a key marked 'RUB OUT', but this generally does nothing. A line of text can be modified with numerous backspacings, overtypings, additions, etc, but as long as the line is correct before the 'RETURN' key is pressed, the text should be written correctly on the TV screen. After the 'RETURN' key has been pressed, the text cannot be modified. If such an inaccessible error has been made, the user should continue, finish the text (ie two 'RETURN's) and then 'QUIT' on the control panel. The text can then be retyped from the beginning.

Having written the error free text on the computer terminal and pressed a second 'RETURN', the user is returned to the menu, to choose either 'Auto' or 'Fixed'. For 'Auto', a box appears on the screen, and it can be moved with the rolling ball, or its size can be varied by pressing the 'CHNG' button and using the rolling ball. The box should be placed where the text is to be written. When 'INPUT' is pressed, GEMSTONE will write the characters so that they just fill the box. This

is a very quick and convenient way of writing small amounts of text. However, it is very difficult to write many pieces of text in the same image or in different ones and still have characters of the same appearance, because GEMSTONE varies the sizes to suit the boxes that the user supplies. In such cases, 'fixed' can be selected. The user is asked to give the size, in pixels, of a single character in the 'x' direction by moving the rolling ball and finishing with 'INPUT'. In the same way, the size in the 'y' direction is next supplied, and finally the thickness of the lines in the character is given. The default sizes of 8, 12, and 1 respectively, give characters that are small, of a reasonable proportion, and quite legible. When the last 'INPUT' is pressed, a box appears on the screen and the size is such that the text as specified will just fit inside it. This box can be moved to any position on the screen, and when 'INPUT' is pressed, the text is written at that place.

Examples of text can be seen in all the photographs within this handbook. The texts in the early figures, eg figs 12, 14, 18 etc, have been written using 'Auto'. In Figs 29, 30, 31, 33, etc there are examples of text written using 'Fixed'.

COPY AREA

SUMMARY

A rectangular area in any image can be copied to any position in the output image. The area copied may be limited by the contents of overlay plane 4, and a contrast stretch can be applied during the copy. Alternatively, the area can be considered as a sprite and moved continuously.

COMMAND IN DETAIL

Six secondary commands are presented to the user. The first is 'Whole Box', which is the simplest one and is used to copy an area outlined by a box, into a new position in the same or another image. If this command is chosen, then the next requirement is to select one of the contrast stretches (ie red, green, or blue) in the current LUTs, or no stretch (ie none), to be applied during the copy. Lastly, the store containing the area to be copied, ie the source store, is chosen. The source image is then presented to the user with a box, the position of which can be changed with the rolling ball and the size of which can be varied after pressing 'CHNG', so that the area to be copied is outlined. After pressing 'INPUT', the user can move the box to the area where the copy is to be placed. Note that the size of this box cannot be changed. After pressing the last 'INPUT', the copy with or without contrast stretch is performed. If the box is moved and 'INPUT' pressed, further copies can be made into the output store, until 'QUIT' is pressed to finish.

The command 'Ovly 4 Box' is the same as 'Whole Box' except that only the area which is both inside the box for the source store, and inside the area set in overlay plane 4 is actually copied. A very useful operation for this command is outlined in the following example. Suppose it is required to contrast stretch an image which contains bright regions (eg land in a remote sensing image) and dark regions (eg

water), but to have different stretches in the two regions in order to maximise the information that can be seen. Within the Contrast Stretch chapter two stretches can be saved, one for the bright regions, and one for the dark. A mask for the bright region (and the inverse mask) could be produced in the Density Slice chapter, or by classifying, or by drawing using the other commands in this chapter. With one mask in overlay plane 4, and the corresponding stretch applied to the image, one area can be copied and stretched correctly with this command. With the other mask in the overlay plane, and the other contrast stretch restored, the other area of the image can also be copied. A colour image can be copied in a similar manner.

If the copy is from one store to another, all 8 bits are copied. For a store to overlay plane copy, only the MSB of the store is copied to the overlay plane, unless the copy is through a LUT in which case any pixel value which is changed to a value above 127 through the LUT, will be copied. In the reverse direction, the overlay plane is copied to all the bit planes of the store. This is the same as considering the output of the overlay plane as being at level 255. If the one location at 255 in a LUT is changed, then by copying through the LUT, a copy can be made at any level in the output store. An example of the use of copying through the LUT is for the operation above, where different parts of an image are stretched differently. Overlay plane 4 can be copied to itself through the LUT previously set to negate an image (see 'LINEAR STRETCH' below). In this way, a mask can be inverted quickly.

For 'Copy Sprite' (see p118 for the definition of a sprite), the area to be copied is treated as a sprite, ie it can be moved continuously. It is however limited to 4096 pixels in area, for reasons of speed. Also, the copy can only be made in the same store, ie the one specified by 'OUTPUT PLANE'. Having chosen this operation, GEMSTONE displays a box whose position can be varied with the rolling ball and whose size can be altered after pressing 'CHNG', but within the 4096 pixel limit. If the size in the x direction is increased, there will come a point when GEMSTONE will reduce the other dimension to remain within the limit. When 'INPUT' is pressed to select the area to be copied, the rolling ball controls the position of the sprite. It may be moved anywhere over the image and is not finally written into the store until 'INPUT' is pressed. Thus an object can be matched to the surroundings before being inserted, and the cursor control buttons can help with the final accurate positioning. Multiple copies can be written, and the process ends with 'QUIT'.

'Copy Ovly 4 Sprite' is the same as 'Copy Sprite' except that the area copied must lie both inside the box and inside the area set in the overlay plane.

'Move Sprite' and 'Move Ovly 4 Sprite' are the same as the 'Copy' commands above, except that when the sprite is moved, it is erased from its original position.

PAINT

SUMMARY

For this command, when the cursor is moved, it is as if there were a paint brush under the centre of the cursor. Areas can be 'painted in' (drawn), or 'painted out' (erased) with a brush of variable width.

COMMAND IN DETAIL

Initially, the rolling ball is used to move the cursor to the start point for the painting operation. When 'CHNG' is pressed, and the rolling ball moved, a 'painted' line is left behind the cursor. Since the paint brush is only one pixel wide initially, this operation can be used to draw a continuously curved line, unlike 'DRAW LINES' where curved lines are made from straight line segments. The size of the brush can be varied by changing to mode 1 (ie by pressing the 'MODE' button so that the light under the button is extinguished. See section 4). Moving the rolling ball to the right causes the brush to widen and vice-versa. Returning to mode 0 results in the rolling ball controlling the painting again. If 'CHNG' is pressed once more, and the ball moved, the brush now paints out or erases areas that were previously painted in or set by any other graphics process. Pressing 'CHNG' again allows the user to move the starting point for another painting operation. The current function of the rolling ball, ie move, draw, or erase, is displayed on the VDU. When 'INPUT' is pressed, the operation is complete and the user is returned to the menu.

DRAW LINES

SUMMARY

With this command the user can draw lines which comprise straight line segments.

COMMAND IN DETAIL

The cursor is moved to the start point with the rolling ball, and the 'INPUT' button is then pressed to mark the position. Moving the rolling ball to the next point in the line and pressing 'INPUT' causes the second point to be marked and a straight line is then drawn by GEMSTONE from the first to the second point. The rolling ball is then moved to the third point, and so on. Thus a curved line can be drawn as a series of straight line segments. When drawing straight lines or boxes etc, it should be remembered that the buttons round the 'MODE' button can also be used to move the cursor up, down, left, or right (see p11), or can lock the movement of the rolling ball in one direction (see p13), thereby making long horizontal or vertical lines easier to draw. When the line has been completed, 'QUIT' is used to finish. The length of the line is displayed on the VDU.

An example of this command appears in Fig 31 where the horizontal and vertical axes have been drawn. The tick marks and arrows were also drawn with this command making use of the cursor control buttons to form neat graphics.

POLYGON FILL

SUMMARY

This command is exactly the same as 'DRAW LINES' except that at the conclusion, the last point is connected to the first point to form a closed polygon, the interior is filled in, and the total area of the polygon is displayed on the VDU together with the length of the lines round the perimeter.

AREA FILL

SUMMARY

This command is like 'PAINT' when drawing a continuous line with a thickness of one pixel, except that on completion, the line is closed and the included area filled in.

COMMAND IN DETAIL

The cursor is moved to the start point with the rolling ball and 'INPUT' pressed to mark its position. As the cursor is moved further, a line is drawn continuously until 'INPUT' is pressed again, when the last point is then connected with a straight line to the first, the enclosed area is filled, and the total area is displayed on the VDU.

SEED FILL

SUMMARY

This command allows the user to 'plant a seed' within a closed region and then fill in the region.

COMMAND IN DETAIL

A closed region is generally produced with 'DRAW LINES' or 'PAINT' and results in a boundary line around the required region. It must be closed, ie the start point must be joined to the end point. When the cursor is moved inside the region with the rolling ball or the cursor buttons, and 'INPUT' is pressed, GEMSTONE fills in the region within the boundary line. If the region is not closed, this filling will 'leak' out of the gap and may fill the whole screen. 'DRAW LINES' followed by 'SEED FILL' can be exactly the same as 'POLYGON FILL' but in two operations. However, if 'PIXEL VALUE' is changed between the two operations, the filling can be at a different level from the boundary. This can be very useful for producing attractive display materials, (eg text could be in one colour, written on a different background, surrounded by a differently coloured border.) or allowing the user to deal with a boundary in a different manner in, for example, the classifiers.

DRAW SPRITE

SUMMARY

A sprite is an object such as a square, a circle, an arrow, a drawing of a man, a house, etc that can be moved around the screen. The sub-commands allow the user not only to draw the sprites and move them, but also manipulate them so that they are the correct size, shape, etc.

COMMAND IN DETAIL

There are several secondary commands, the first four being concerned with drawing, and the last three being optional extensions. They are:

BOX. When this is chosen, a rectangle appears on the screen and the user is able to change the box with a series of MANIPULATION SUB-COMMANDS. There are five such commands, shift, scale, shape, skew, and rotate. They use the rolling ball to shift a sprite, change the size of it, change the shape of it, skew it, and rotate it, respectively. By pressing the 'CHNG' button, the current sub-command is changed cyclically, ie shift, scale, shape, skew, rotate, shift, scale, etc, and the name of the current sub-command appears on the VDU. The following operations are possible.

SHIFT. Using the rolling ball, the sprite can be moved on the screen.

SCALE. This uses the rolling ball to change the size of the sprite. For a box, the bottom left corner remains fixed, so that if a sprite is reduced, it will eventually contract into this single point. 'Reducing' still further results in the box re-appearing on the other side of the single point. Thus, if the box were in the first quadrant initially, it would be reflected into the third. For a circle, the fixed point is on the circumference, at the middle right, and for a set of lines, it is the first point defined by the user.

SHAPE. This uses the rolling ball to change the size of the sprite in the y-direction alone, and hence the shape.

SKEW. A horizontal line through the fixed point does not change at any stage during this operation. A horizontal line above or below the fixed point is moved to the left or right by the rolling ball, and the further the line is from the fixed point, the more it moves. Thus a box, which is initially a square can be made into a parallelogram.

ROTATE. Moving the rolling ball to the right, causes the sprite to rotate in a clockwise direction about the fixed point.

All the sprites are held in overlay planes 2 and 3 at this stage. When, for example a sprite is moved, it is copied from plane 2 into 3 with the change. A further movement results in the sprite being written back into plane 2 with the new change, and so on. When a sprite of the required shape etc is in the required position, it can be copied into the store chosen with 'OUTPUT PLANE' by

by pressing the 'INPUT' button. The sprite can be further moved or changed and copied into the output plane by pressing the 'INPUT' button again. In this way numerous identical copies, or varying copies can be written into the output store. Pressing the 'QUIT' button completes the process and returns the user to the menu.

CIRCLE. A circle is displayed on the screen, and all the manipulation sub-commands described under the BOX secondary command can be used.

LINEs. A sprite or a set of sprites can be drawn with a series of straight line segments. The cursor is moved to a suitable starting point for drawing (this need not be the final resting place for the point since the sprite can be moved later) and the 'INPUT' button pressed. When the cursor is moved to another point and the 'INPUT' button pressed, a straight line is drawn between the two points. This process can be repeated for up to 64 points (ie 63 line segments). The VDU displays the number of points used, and the number of points left. If one sprite has been drawn with a few points, a second sprite can be drawn by moving the cursor to the start of the new sprite and pressing 'CHNG' to denote a new start. In this way several sprites can be drawn in a group and the manipulation sub-commands described under BOX above, operate on the whole group. The fixed point for the sub-commands is the first point in the drawing sequence. Normally this is at the start of the first line. However, if for the first point 'INPUT' were pressed, the cursor then moved and 'CHNG' pressed to denote the start of a new line, then the fixed point would be invisible and separate from the sprite itself. If for the first point 'CHNG' were pressed after 'INPUT', the cursor then moved and 'CHNG' pressed again, then the fixed point would be separate from the sprite but would be visible. Both of these examples allow the sprite to be rotated about, or magnified relative to, any point on the screen.

When the required sprite has been drawn, 'QUIT' is pressed to leave the drawing sequence. It should be noted that the sprites are only held in overlay plane 2 or 3 and have not been copied to the output store at this stage. Having pressed 'QUIT' the user can operate all the manipulation sub-commands on the sprites before using the 'INPUT' button to copy them into the store as described under BOX.

LAST LINEs. This secondary command allows the user to pick up the last lines that were drawn and use the manipulation sub-commands to change them. Note that the command only picks up the last LINEs and not the last BOX or CIRCLE which have to be recreated directly.

AUTO FILL. A sprite may be filled in immediately after it is drawn. For example, if this command were specified before using BOX, then when a box is copied into the output store by pressing 'INPUT' it is filled in at the same level as its boundary. Therefore, a number of solid boxes could be drawn on the screen, and similarly for other shapes. An open polygon, which could be the result of using LINEs, is closed by joining the last point to the first, before filling.

AUTO INCR. This command, which must be chosen before a drawing command, causes the pixel value to increment by one, each time a sprite is copied into the output store. Thus in the example of the series of solid boxes mentioned in AUTO FILL, each one would be at a different level. If the

image were then coloured using 'PIXEL VALUE' and 'PIXEL COLOUR' or 'LINEAR STRETCH' or the various methods in the 'Density slice' chapter, all the boxes could appear in different colours. Further, 'Colour Rotate' (pl23) would allow animation to be performed.

AUTO INTERP. As for **AUTO FILL** and **AUTO INCR**, this command must be chosen before one of the four drawing commands. Considering the example of the set of filled boxes described above, when the second box has been copied to the output store, **GEMSTONE** automatically interpolates along the straight line connecting the first and second sprites. If the user required the line to be divided into two pieces, there would be a sprite at the beginning, middle and end of the line. The user is asked for the number of divisions of the line and this is displayed at the bottom of the TV screen. Having selected the required number with the rolling ball, 'INPUT' is pressed. If the second sprite that the user copies to the output store is different from the first (ie by using **SCALE**, **SHAPE**, **SKEW**, or **ROTATE**) then in this example, the additional box that **GEMSTONE** inserts would be interpolated between the boxes at the two ends. A line can be divided into a maximum of 256 segments and the interpolation is performed each time a sprite is copied to the output store.

These three optional commands, which if they are required, must be specified before the drawing commands, permit the user to produce a range of sprites whose shapes change gradually from the first specified one to the next, to the next, etc, with or without filling, with or without the pixel level increasing from one to the next, and therefore, with a little thought beforehand, some very complicated graphics can be produced quite quickly.

An example of this command is shown in Fig 31 (the lower pair of axes). A sprite was drawn around the existing x-axis, and then using 'rotate' and 'skew', the y-axis was produced. Similarly, the text 'F(y)' was written in the image, a set of sprites were drawn round the characters which were rotated, skewed, and positioned at the end of the y-axis.

GREY SCALE

SUMMARY

A grey scale of any size can be positioned anywhere on the screen using this command.

COMMAND IN DETAIL

A box appears on the screen at the top left corner, with an initial size of 75 in the x direction and 90 in the y. It can be moved with the rolling ball and after pressing 'CHNG' the size can be altered. When 'INPUT' is pressed, the grey scale or wedge from level 0 to 255, is written into the image store specified by 'OUTPUT PLANE' and the user is returned to the menu. If the x dimension is equal to twice the y dimension, or less, the wedge is written in the vertical direction with black at the top. If the x dimension is greater than twice the y dimension, the wedge is written in the x direction with black at the left.

COLOUR PALETTE

SUMMARY

A set of 30 coloured boxes are displayed on the screen using this command.

COMMAND IN DETAIL

Instead of choosing colours by specifying the red, green and blue components each in the range of 0 to 255, it is sometimes convenient to select the colour from a palette. This command puts such a palette on the screen with a set of initial colours, which are given in the following table. The numbers are the red, green and blue components in order, and the colour descriptions are only given where they are likely to be consistent. Those colours without a name, can vary due to differences between monitors, and also due to the eye judging a colour differently depending on the surrounding colours. This can be demonstrated simply by changing the colour of one square to the colour of another. For example, if the second from right on the top row were picked (using 'PIXEL VALUE' followed by the 'cursor' secondary command), and the colour changed to 192,192,128 (using the 'PIXEL COLOUR' command followed by the 'numbers' secondary command), then the new patch on the top row would have exactly the same values as the patch four below it, and yet to the eye, the colours would look quite different. This example also illustrates how the initial palette may be easily changed, thereby permitting the user to examine how visible one colour is when placed next to another, or how a set of colours would match.

255,128,255 Pale Magenta	192,128,255	128,128,255 Pale Blue	128,192,255	128,255,255 Pale Cyan
255, 0,255 Full Magenta	128, 0,255	0, 0,255 Full Blue	0,128,255	0,255,255 Full Cyan
255, 0,128	128, 0,128 Dark Magenta	0, 0,128 Dark Blue	0,128,128 Dark Cyan	0,255,128
255, 0, 0 Full Red	128, 0, 0 Dark Red	0, 0, 0 Black	0,128, 0 Dark Green	0,255, 0 Full Green
255,128,128 Pale Red	255,128, 0 Orange	128,128,128 Grey	192,192,128	128,255, 0
255,255,255 Full White	255,255,128 Pale Yellow	128,128, 0 Dark Yellow	255,255, 0 Full Yellow	128,255,128 Pale Green

Table 9 Initial colours of the palette.

When the command is exercised, a box appears on the screen, and it can be moved by the rolling ball, and its size can be altered after pressing 'CHNG'. The format of the palette, is 5 patches in a row, and 6 rows, remains constant. Pressing 'INPUT' causes the palette to be written in the desired area of the output store, the LUTs to be modified, and the user to be returned to the menu. The way in which this function operates, is that the top left patch is written into the output store at level 224, the next patch to the right at 225, and so on to the bottom right corner which is at 254. The LUTs for all the grey levels between 224 and 254 are changed to give the initial colours.

LINEAR STRETCH

SUMMARY

With this command, the user is able to manipulate the contents of the LUTs.

COMMAND IN DETAIL

The principle of operation is as follows. For one location (or address) in each of the three LUTs, the user chooses red, green and blue values, which will be the new contents of that location. Three values for red, green and blue are then inserted into another location in the LUT. Between the two locations specified, GEMSTONE inserts values in the LUTs which form three linear ramps from the first location to the second. For example, if 255, 0, 0 (ie full red) were put in location 100 and 0, 255, 0 (ie full green) were put in location 103, GEMSTONE would put 170, 85, 0 in location 101, and 85, 170, 0 in location 102. The resulting image would be unchanged from level 0 to 99 and from 104 to 255. At level 100, the image would be full red with the shade changing through brown-yellow to full green at level 103. A third location in the LUTs could be chosen and there would be a linear ramp in the contents from the second to the third location, and so on. It is often helpful to put a grey wedge on the screen using the 'GREY SCALE' command to examine the effect of a particular linear stretch.

The process can be used for a wide variety of contrast stretches. A stretch to produce the negative of an image, would simply require 255 in location 0, and 0 in location 255. This command would insert all the other contents correctly. A piecewise linear contrast stretch could be produced with the same or different values in the three LUTs at each location. This would be like the manual contrast stretch on page 29, but instead of being restricted to linear or two step linear, multiple steps could be used. Considering the graphics image of a set of filled boxes on the screen, if they were drawn at different pixel values, they could be coloured in progressively. 'COLOUR ROTATE' (p123) could also be used to animate the sequence. If the linear stretch were such that there was only one non-zero entry in the LUT, then only one box in this example would be visible at any one time. 'COLOUR ROTATE' would then cause the boxes to appear in turn to give another type of animation.

The user is first asked for the colour (or the contents of the LUTs) of the first location to be changed, using either 'cursor' or 'numbers'. For the former, the cursor is put over a suitable colour in the image. eg in the colour palette. For 'numbers', each component is

varied by the rolling ball with the 'QUIT' button being used to cycle round red, green and blue. The current values of the three components appear on the screen, and there is a small bar over the component currently being altered. Once 'INPUT' has been pressed, the user is asked for the location (or address) in the LUTs for this set of three values. There is the same choice again in that 'cursor' can be used to select a pixel value in the image, or 'numbers' can be used to choose the location with the rolling ball, finishing in either case with 'INPUT'. The colour or contents of the LUTs at the second location are next required and are entered in the same way as the first. (ie If the 'cursor' was used for the first set, it will be used for all sets). Lastly, the location for the second colour is selected in the same manner as for the first, and as the location is changed, the LUT values are changed, thereby allowing the user to examine dynamically the effect of the operation. Third and subsequent contents and locations can be chosen until 'QUIT' is pressed to return the user to the menu.

COLOUR ROTATE

SUMMARY

This command is used to rotate the contents of the LUTs.

COMMAND IN DETAIL

The user is first asked for the lower location in the LUTs for this rotation operation. This is selected with the rolling ball, the value being displayed on the screen, and the number is entered with 'INPUT'. In the same way the upper limit is entered after which, the process starts. The contents of the LUTs at the lower location are transferred to the upper location whose contents move to the location one below, and so on. Thus, using the example in 'LINEAR STRETCH' above, the following shows the first steps in the sequence where 100 and 103 are the two limits.

Location	Step 0	Step 1	Step 2	Step 3
104	104, 104, 104	104, 104, 104	104, 104, 104	104, 104, 104
103	0, 255, 0	255, 0, 0	170, 85, 0	85, 170, 0
102	85, 170, 0	0, 255, 0	255, 0, 0	170, 85, 0
101	170, 85, 0	85, 170, 0	0, 255, 0	255, 0, 0
100	255, 0, 0	170, 85, 0	85, 170, 0	0, 255, 0
99	99, 99, 99	99, 99, 99	99, 99, 99	99, 99, 99

The rate at which the LUTs are circulated is controlled by the rolling ball. Moving it to the right causes the time between steps to increase, and vice versa. As mentioned in 'LINEAR STRETCH', this command is normally used for animation with colour moving through a sequence of objects, or for displaying a sequence of objects in turn. This rotation of the LUTs is terminated by pressing 'INPUT'.

3-D FIXED

SUMMARY

The result of this operation is an image which is a 3-D view of the input image with distance in the 'z-axis' being proportional to the pixel intensity.

COMMAND IN DETAIL

The first requirement is to choose the input or source image, which is selected with the rolling ball and then 'INPUT' is pressed. The output image will appear in the store selected with 'OUTPUT PLANE', but there is also an output for the 'grid' and the 'axes', either of which can be put into overlay plane 4 if desired. 'Axes' is a drawing of a box representing the axes of the 3-D image, and grid is an xy grid laid on top of the 3-D image to give another impression of the upper surface of the 3-D object. The store for the grid is selected and 'INPUT' pressed, (or 'QUIT' is pressed if no grid is required). The same process is repeated for the axes store. A box then appears on the screen, and it can be moved over the area of interest in the input image, with the rolling ball. The size is set at 256 by 256. This may be reduced if required by pressing 'CHNG' and using the rolling ball, but it cannot be increased. Having selected the area and pressed 'INPUT', a 3-D view of the area as viewed from the bottom right of the image, is written into the output store.

Fig. 31 contains two examples of this function. The image at the top left of the figure was the input image, and the result, sliced, is shown at the middle left, with the axes and grid clearly visible.

3-D GENERAL

SUMMARY

This command is similar to '3-D FIXED' except that there is much more flexibility in viewing angle and position, but there are no grids or axes.

COMMAND IN DETAIL

Having chosen the input store and pressed 'INPUT', four viewing directions, bottom, top, left and right are presented. 'Bottom' means that the observer is viewing from the bottom of the screen. When the direction has been selected and 'INPUT' pressed, the user is asked for a view angle from 0 degrees, which is a view in the plane of the image, up to 90 degrees, which is a view vertically down on the image. 45 degrees is suggested, but this can be altered with the rolling ball, and 'INPUT' is pressed at the end. A '% Scale Height' is next requested and this can best be considered for the case where the view angle is 0 degrees. If the % scale height is x%, then a pixel at level 1 will be shifted in the z-direction by 1 times x%. Thus, with the default of 25%, a pixel with an intensity of 128 would be displaced by 32 places on the screen. However, the z-direction does not exist on a 2-D TV screen but is actually up the screen in what is normally considered as the y-direction. The presentation when the view angle is 0 degrees is

correct. At other angles, the displacement is still in the vertical direction, and, although the image is now quite distorted, the illusion of a 3-D image can still be created. When the viewing angle is 90 degrees, some very odd effects can result since points in the 2-D image are displaced within the image, depending on their intensities. When the scale height has been chosen between 1% and 100%, and 'INPUT' pressed, a box appears on the screen. Both the position and the size can be varied with the rolling ball, the latter after pressing 'CHNG', and the box is placed over the area of interest in the input image. After pressing 'INPUT', another box appears, representing the area in the output image into which the 3-D picture is to be written. The position and size can be changed again, but care should be taken to ensure that the top of the image in the 'z-direction' is not truncated. When 'INPUT' is pressed again, the process starts with the 3-D image being written into the store specified with 'OUTPUT PLANE'. It is possible to produce many views from different directions or at different angles, within the one image.

SAVE STRETCH

SUMMARY

It is possible to save up to 8 stretches (ie 8 sets of the contents of the three LUTs), with this command. It is exactly the same as the 'SAVE STRETCH' command used in the Contrast Stretch chapter (p31), and uses the same locations to store the information.

COMMAND IN DETAIL

The only secondary command, is to name the stretch, from A to H. These names are exactly the same as those used in the Contrast Stretch chapter, so that if a stretch was saved as A in that chapter, it would be overwritten if a stretch were saved as A using this command. If the LUTs have been modified using 'PIXEL COLOUR', 'COLOUR PALETTE', 'LINEAR STRETCH', or 'COLOUR ROTATE', this command will hold the LUT contents which resulted from the operation. One line of comment can be typed on the terminal to describe the stretch, and this is saved with the stretch.

RESTORE STRETCH

SUMMARY

This command allows the user to restore a previously saved contrast stretch into the LUTs.

COMMAND IN DETAIL

The only secondary command is to give the name of the contrast stretch to be restored. If 'None' is specified, the LUTs are set to the unity transfer function, (ie there is no contrast stretch applied to the image). Any stretch saved within this chapter or in the Contrast Stretch chapter can be recalled using this command, and the descriptive text supplied when the stretch was saved, is reproduced on the VDU.

ADD
SUB
MULT
DIV
DEFINE PROGRAM
EXECUTE
PRINT PROGRAM
SAVE PROGRAM
RESTORE PROGRAM

LOAD	STORE	USE REG	END
PLUS	MINUS	TIMES	DIV
XPOS	YPOS	MIN	MAX
AND	OR	XOR	>0
SQRT	ABS	CHNG SIGN	RANDOM
COS	SIN	ABS2	ARG
LOG	EXP	INT	FRAC

REG1	REG2	REG3
REG4	REG5	REG6
INTEGER STORE	FLOATING STORE	CONSTANT
INTEGER PARAMETER	FLOATING PARAMETER	CONSTANT PARAMETER
PI	255	STACK!

A
B
C
D
E
F
G
H

WHOLE SCREEN	
TOP LEFT	TOP RIGHT
BOT LEFT	BOT RIGHT

D1
D2
D3
D4
5
6
7
8
9
10
11
12
13
14
15
16
D5
D6
D7
D8

D1
D2
D3
D4
5
6
7
8
9
10
11
12
13
14
15
16
D5
D6
D7
D8

STATUS	HELP	RETURN	IMAGE
--------	------	--------	-------

INPUT FOR COMMAND OR QUIT

Fig 36 The 'Arithmetic' page

5.10 Arithmetic

In this chapter, arithmetic operations are performed where an image is treated like a number. Thus, just as two numbers can be added, two images can be added, where one pixel is added to the corresponding pixel in another image. The first four commands are very simple and permit the user to add, subtract, multiply or divide two images and produce acceptable results. The rest of the chapter is concerned with more complicated functions than these four, and with the evaluation of complex equations. Since there are no values built into the general functions to ensure that the resultant images are within the range of the stores and formats (eg 8-bit, 16-bit or floating point), the user must consider the range of the output data when running all but the first four commands.

ADD

SUMMARY

One image can be added to another, pixel by pixel.

COMMAND IN DETAIL

The secondary commands give the area to be added (either the whole screen or a quadrant), the first store, the store to be added, and the store into which the result will be written. When 'INPUT' is finally pressed, the top left pixel of the first image is added to the top left pixel of the second image, the result is divided by 2, to keep the intensity in the range 0 to 255, and the resultant top left pixel put in the output store. This process is repeated for all the other pixels in the images.

SUB

SUMMARY

One image can be subtracted from another, pixel by pixel.

COMMAND IN DETAIL

Subtraction is performed in the same way as addition, except that the equation is

$$(\text{output store}) = (\text{first store}) - (\text{second store}) + 127.$$

The reason for adding 127 is to make the results positive, since the system cannot display a pixel having a negative intensity. It is still possible to have a pair of input images which produce a pixel with a negative number and if this occurs, it is replaced by zero. Equally, it is possible to have a result greater than 255 in which case it is replaced by 255. For highly uncorrelated images, the user can either accept that negative values are all set to zero, and the large values are limited to 255, or apply a contrast stretch to reduce the range of the input images, or use the full arithmetic operations described below.

MULT

SUMMARY

One image can be multiplied by another image, pixel by pixel.

COMMAND IN DETAIL

Multiplication is performed in the same way as addition, except that the result is divided by 255 to keep the output within the range 0 to 255. The largest possible output pixel is 255 (from one image) times 255 (from the other) divided by this constant 255, giving a result of 255. If both input images are dull with maximum pixel values of 128, the output image will be very dull with a maximum value of $128 \cdot 128 / 255$, ie 64. Images should therefore be stretched to cover the full range 0 to 255 before using this command.

DIV

SUMMARY

One image can be divided by another, pixel by pixel.

COMMAND IN DETAIL

Division is performed in the same way as addition except that the result is multiplied by 100. If one image is divided by another which is very similar, the answers without scaling will be around 1. The multiplier gives the resultant image a reasonable range. If a bright image (with the brightest pixels at 250) were divided by a dark image (with the brightest pixels at 50) then the result, when multiplied by 100, will far exceed the limit of 255 ($250/50$ times 100 is 500 in this case). Pixel values greater than 255 are stored as 255. Images should therefore be stretched to cover the full range 0 to 255 before using this command.

DEFINE PROGRAM

SUMMARY

Using the wide range of secondary commands described below, complex equations or programs may be constructed.

COMMAND IN DETAIL

This command functions in a similar way to an electronic calculator. There are two sets of secondary commands, one for the functions and the other for the operands, ie images, constants. Up to 64 steps can be used to define a program, but more than one operation can be included in one step. As the program is being defined, the number of steps left is displayed on the VDU.

To illustrate the similarity with a calculator, consider adding the contents of store 1 to the contents of store 2, and putting the answer in store 3. GEMSTONE suggests that the first function to select is 'load'. This is similar to entering a number into a calculator. If 'INPUT' is pressed for 'load', GEMSTONE next suggests that an 'integer store' be chosen. If this is selected, the user is asked for a store number. This is provided by moving the cursor over the store (store 1 in this example) and pressing 'INPUT'. At this stage, store 1 has been 'loaded into the calculator'. The cursor is now back in the function secondary command area, and 'plus' can be chosen, just as the plus button on a calculator. Having selected this, an operand is required, and the cursor is positioned over 'integer store'. If this is selected, store 2 can be chosen. At this stage, we have specified that store 1 is to be added to store 2, and all that remains is to choose a store for the result. After store 2 is selected with the 'INPUT' button, the cursor moves to the functions again, where 'store' can now be selected, followed by 'integer store', followed by store 3. Since no more operations are required, the last function is 'end'. At this stage, a program has been defined, to load the 'calculator' with the contents of store 1, add the contents of store 2, and put the result in store 3. To check that the program has been entered correctly, 'PRINT PROGRAM' can be executed. The calculation can be performed by selecting 'EXECUTE'. All the calculations are performed using the floating point functions of the host computer to maintain good precision over a wide range. Integer intensities are converted to floating point numbers for the calculations and converted back at the end, if required.

Having understood the basic operation of this 'calculator', the operands and functions can be examined in more detail. These are described below : -

Reg1, Reg2, Reg3, Reg4, Reg5, Reg6. Just as in many ordinary calculators, this 'calculator' has six working registers or stores. Reg1 is the first one normally used, and for many small sums, this may be the only one used. The current register number is given at the bottom left of the television screen. If an equation of the form $\text{Output} = (\dots\dots) + (\dots\dots)$, is to be evaluated, the contents of the first pair of brackets can be calculated and the result left in Reg1. Reg2 can then be used for the second pair of brackets (with 'Load' or 'Use Reg' functions), and finally Reg2 can be added to Reg1 to give the output. When 'PRINT PROGRAM' is executed, it may be noticed that 'REGO' is sometimes used. This is another register used only by GEMSTONE to do internal calculations and can be considered as a 'private' working register. Further uses of these registers as stores for intermediate results are illustrated in the examples at the end of the chapter.

Integer Store. A store number will be given after this command. If the number is 1, 2, 3, or 4, the contents are treated as 8-bit integer intensities and if the store number is in the range 5 to 20 inclusive, the contents are treated as 16-bit integer intensities ie 15 bits and a sign as described on page 7.

Floating Store. A store number in the range 5 to 20 inclusive, will be given after this command, and the contents are treated as intensities in the floating point format described on page 7.

Constant. The user can set any constant number using this command. The number is at the bottom left of the screen and can be changed with the rolling ball. If 'CHNG' is pressed, a small bar above the number can be moved to the left or right using the rolling ball. When the rolling ball is next used to change the number, only those digits to the left of the bar are changed. In this way a number comprising many digits can be input quickly. If the 'CHNG' button is pressed again, the rolling ball controls the position of the decimal point so that decimal numbers can be input. Pressing 'CHNG' again causes the rolling ball to control the numbers again, and so on. The number under 'PI' below, could be entered as 26 followed by one press of 'CHNG' to move the bar two places to the left. After two further presses of 'CHNG', the digits 59 can be entered so that 5926 appears on the screen. This process is repeated until 31415926 is on the screen. Having set the digits 31, two presses of 'CHNG' permit the user to move the decimal point to the correct position.

Integer Parameter. When a program is to be run several times with only a few changes of parameters, it is convenient to define the program with the integer store number(s) being set when the program is executed. If 'Integer Parameter' is selected instead of 'Integer Store', a space will be left in the program. Supposing the contents of stores 1, 2 and 3 are to be multiplied by one constant, a program could be defined with the source and result stores being left as parameters for insertion at execution. The one program could then be used to perform all three operations.

Floating Parameter. This is the equivalent of 'Integer Parameter' for defining floating point stores.

Constant Parameter. This is the equivalent of 'Integer Parameter' for defining the value of a constant at the time of execution of the program. Two examples are given at the end of the chapter.

PI. This is the constant π , equal to 3.1415926....

255. This is simply the number 255.

Stack! In the example given under Reg1, Reg2,...Reg6, on page 129, the two registers Reg1 and Reg2 were added at the end. Stack! simply takes the current register and the register one less than the current, performs the required operation (eg 'plus' for this example) and puts the answer in the register one less than the current. This type of operation is similar to the 'Reverse Polish' methods used in some calculators with stack memories. These six registers do not however function as a stack. In the previous example, Reg1 contained the results of the first pair of brackets and Reg2 the second. The result of the simple addition would be placed in Reg2. Selecting 'Plus' and 'Stack!' would cause Reg1 to be added to Reg2 and the result put in Reg1, but the stack would not 'move down' as in most calculator memories.

All the possible functions are as follows :-

Load. This function is used to load a register with an image store, a constant, or the contents of another register. The first time it is used, REG1 will be loaded. On subsequent occasions 'Load' will automatically increment the register number. Immediately after the function 'Use Reg', 'Load' does not increment.

Store. The contents of the register currently being used can be placed in an image store.

Use Reg. This enables any of the six registers to be selected.

End. When the program has been completely defined including where the results are to be stored, this function is used to finish.

Plus. The contents of the current register are added to the next operand, be it a constant, a store, or another register. The result is put into the current register, overwriting the previous contents.

Minus. The next operand is subtracted from the current register and the previous contents are overwritten.

Times. The current register is multiplied by the next operand, overwriting the previous contents.

Div. The current register is divided by the next operand, and overwrites the previous contents. If an attempt is made to divide by zero, zero is changed to approximately 10 to the power -12 .

Xpos. The intensity of a pixel is made equal to its x co-ordinate and the result is put into the current register, if it has not been used before (eg if this is the first function), or into the next register otherwise.

Ypos. The intensity of a pixel is made equal to its y co-ordinate and the result put into the current register, if it has not been used before, or into the next register otherwise.

Min. The minimum of the current register and the next operand is put into the current register. This is very like the Fortran language command MIN. If for example, an image is to be limited so that intensities over 100 are changed to 100, this can be achieved by taking the minimum of the image and the constant, 100.

Max. The maximum of the current register and the next operand is put into the current register. If for example a 16 bit image is to be limited to positive values only, this can be achieved by taking the maximum of the image and the constant, 0.

And. The logical AND of the current register and the next operand is produced and put into the current register. For the first four stores, the AND is performed using the 8 bits. For the other stores, 16 bits are used if the contents are integers. If the contents are floating point numbers, they are converted to 16 bit integers before the AND function is performed. A simple example of the use of this function is where the least significant 8 bits of a 16 bit integer store are to be copied into one of the top four stores. By taking the AND of the image and 255 decimal, (ie 11111111 in binary) all bits except the bottom 8 bits are masked out.

Or. The logical OR of the current register and the next operand is put into the current register in a similar fashion to 'And' above.

Xor. The logical exclusive OR of the current register and the next operand is put into the current register in a similar fashion to 'And' above. A simple example of this function is that the exclusive OR of an image with the constant 255 produces the negative of the image.

0. If the value in the current register is positive, the number 1 is put into the current register. If the value is negative or equal to zero, the number 0 is put into the current register. This command provides the basic function required by many decision making processes and is illustrated in an example at the end of the chapter.

Sqrt. The contents of the current register are replaced by their square root. If the contents of the register are negative, the output is set to zero. This is a quick and convenient way of reducing the range of a 16 bit integer image (in stores 5 to 20), into 8 bits for display in one of the first four stores.

Abs. The contents of the current register are replaced by their absolute value. ie any negative signs are removed.

Chng Sign. The sign of the number in the current register is changed.

Random. A random number in the range 0 to 1 is put into the current register. To perform this function, the computer clock is read as a 32 bit integer, and the first pixel is produced by adding this integer shifted 8 bits to the left, to this integer shifted 8 bits to the right. The second pixel is produced by adding the first pixel (in 32 bit form) shifted 8 bits to the left, to the first pixel shifted 8 bits to the right, and so on for all the other pixels. During the shifting operations, undefined bits are set to 0. This algorithm produces images without visible patterns unlike many other types of 'random' number generators, and also produces different images each time the function is executed, since the starting point, the computer clock, is changing continuously.

Sin. The contents of the current register are treated as radians, and replaced by their sine.

Cos. The contents of the current register are treated as radians, and replaced with their cosine.

Abs2. This function and the next are used to convert from rectangular to polar co-ordinates and can therefore be used to convert a complex image, where the real part is in one image store and the imaginary part is in another, into images of magnitude and phase. This function produces the magnitude by taking the square of the current register, adding it to the square of the next operand, and putting the square root in the current register.

Arg. This is the second of the pair of commands mentioned under 'Abs2', and gives the argument or phase of a pair of images. The angle in radians, whose tangent is the next operand divided by the current register, is put in the current register. For the inverse operation, the image in the x-direction (real part) is produced by multiplying the magnitude image (possibly produced by Abs2), by the 'Cos' of the argument or phase image (possibly produced by Arg). The y-direction image is made from the magnitude times the 'Sin' of the argument.

Log. The contents of the current register are replaced by their logarithm to the base e. If the current register is zero or negative, it is set to 0.000001 before taking the log.

Exp. The contents of the current register are replaced by their exponential. This function can also be viewed as the antilog of 'log' above. Raising an image to a fractional power can be done simply by taking the log, multiplying by the power number, and taking the antilog. The maximum value of this function is $\exp(80)$, and any argument greater than 80 is set to 80.

Int. The contents of the current register are replaced by their integer part. If 3.14 were in the register, this command would produce 3.

Frac. This is the complimentary function to 'Int' and replaces the current register with its fractional part. If 3.14 were in the register, this command would produce 0.14.

EXECUTE

SUMMARY

The last program to be defined (with 'DEFINE PROGRAM') or the last program restored (with 'RESTORE PROGRAM'), is executed using this command. If parameters require setting, they will be asked for in the order in which they appeared in the program.

PRINT PROGRAM

SUMMARY

The last program to be defined (with 'DEFINE PROGRAM'), or the last program to be restored (with 'RESTORE PROGRAM'), is printed using this command. Some outputs from this command are given in the examples at the end of the chapter.

SAVE PROGRAM

SUMMARY

Up to 8 programs can be saved using this command. One line of text can be typed at the terminal and saved with the program to identify it.

RESTORE PROGRAM

SUMMARY

One of up to 8 previously saved programs can be restored for subsequent execution (with 'EXECUTE'). The description of the program which the user previously typed, appears on the VDU.

The following are examples of some of the functions described above, and they also illustrate the use of the registers etc. On the left side of each example, are the sequences of commands, comments are given in the middle, and on the right side are the lines given on the VDU when 'PRINT PROGRAM' is executed.

Example 1. Suppose there are images in stores 1 and 2, and that it is required to make store 3 equal to $(\text{store 1} - \text{store 2}) / (\text{store 1} + \text{store 2})$. This essentially shows the difference between the two input images and by dividing by the average, the common intensity variations are removed. For remote sensing images, this is used to show small differences between two spectral bands, and remove variations due to differing sun angles, to at least a first approximation.

Command	Comment	'PRINT PROGRAM'
Load Integer Store Whole Screen D1	Load the whole of the integer image store 1 into the calculator. 'D1' is the first store, and 'WS' is Whole Screen.	REG1: LOAD ST1 WS
Plus Integer Store D2	Evaluate the denominator first. Add the second store. REGO is used by GEMSTONE for the calculation.	REGO: LOAD ST2 WS REG1: PLUS REGO
Load Integer Store D1	Keep the denominator in REG1 for the moment, and load the first numerator store. Note that 'Load' increments the register number.	REG2: LOAD ST1 WS
Minus Integer Image Store D2	Subtract the second store.	REGO: LOAD ST2 WS REG2: MINUS REGO

Div REG1	Divide the numerator by the denominator in REG1.	REG2: DIV BY REG1
Times Constant (200, Rolling Ball) Plus Constant (128, Rolling Ball)	The range of the data must be adjusted. If the average value of each image was 100, and the max difference was 100, then the value of the result would be about $+ \text{ or } - (100)/(100+100)$, ie $\pm 1/2$. Multiplying by 200 and adding 128 gives a good range and no negative values, respectively.	REG2: TIMES 200 REG2: PLUS 128
Store Integer Store D3 End	Put the result in store 3 and finish. Note that the current register is stored.	REG2: STORE ST3 WS END

This program is a very small one, taking only 8 steps out of the possible 64. Consider now two variations, one that the numerator is evaluated first, and the second that the maximum difference between the two images is not known and therefore the multiplier (which was 200) cannot be specified. If on executing with a multiplier, the resultant image has a poor range because there is little spread or there is considerable saturation, the program can easily be run again with another value for the multiplier. The program is essentially the same to 'Div' except that the numerator is in REG1 and the denominator is in REG2, which is still the current register, as before. From immediately before 'Div', the program would be as follows.

Command	Comment	'PRINT PROGRAM'
.	.	.
Use Reg Reg1 Div REG2	If the current register (REG2) were divided by REG1, as above, the inverse would be produced. These commands correct this.	REG1: DIV BY REG2
Times Constant Parameter Plus . . .	Since the multiplier is not known in advance, it is left as a number to be specified at execution.	REG1: TIMES PARAMETER . . .

Example 2. A wide variety of graphics images can be generated with the tools within this package. The top left image in Fig 31 was generated using the equation given on page 86, ie

$$\text{Output image} = 64. \left(2 - \cos\left(\frac{X_{\text{pos}} - 1}{255}\right) - \cos\left(\frac{512 - Y_{\text{pos}}}{255}\right) \right).$$

The following sequence of commands is one way of producing the result.

Command	Comment	'PRINT PROGRAM'
Xpos	The argument of the first cosine	REG1: XPOS
Minus	is evaluated.	
Constant		
(1, Rolling Ball)		REG1: MINUS 1
Div	Note the use of the system	
255	supplied constants, 255 and PI.	REG1: DIV BY 255
Times		
PI		REG1: TIMES 3.142
Cos	Take the cosine of the argument,	REG1: COS
Chng Sign	and take it away from 2.	REG1: CHNG SIGN
Plus		
Constant		
(2)		REG1: PLUS 2
_____	_____	_____
Ypos	Repeat for the second cosine.	REG2: YPOS
Chng Sign		REG2: CHNG SIGN
Plus		
Constant		
(512)		REG2: PLUS 512
Div		
255		REG2: DIV BY 255
Times		
PI		REG2: TIMES 3.142
Cos		REG2: COS
_____	_____	_____
Use Reg	The total is being accumulated in	
Reg1	Reg1. Change to Reg1 and subtract	
Minus	Reg2.	
Reg2		REG1: MINUS REG2
_____	_____	_____
Times	Multiply by the constant to fill	
Constant	the range, and store the result.	
(64)		REG1: TIMES 64
Store		
Integer Store		
Whole Screen		
D1		REG1: STORE ST1 WS
End		END

Example 3. Suppose it is required to generate an image of a $(\sin t)/t$ pattern in two dimensions, centred in the middle of the screen and having about four rings around the central peak. The problem can be split into two parts, the evaluation of $(\sin t)/t$, and the evaluation of t . It should be noted that when t approaches 0, the value of the function is 1. However, since $(\sin t)$ and t are evaluated separately in GEMSTONE, the function would reduce to 0/0 and thus be indeterminate. All that is required is to limit t to a number close to but not equal to zero. If

$t = \sqrt{(x-255.99)^2 + (y-256)^2} / 10$, where $**2$ means to the power 2, then $(\sin t)/t$ would be centred, and have 4 cycles of 2π . The maximum value of the function is 1.0, and the minimum is -0.21723. A program to produce this image is given below.

Command	Comment	'PRINT PROGRAM'
Xpos	Load Xpos and centre it.	REG1: XPOS
Minus		
Constant		
(255.99)		REG1: MINUS 255.990
Load	Copy Reg1 into Reg2 and multiply	
Reg1	them to produce the square.	REG2: LOAD REG1
Use Reg		
Reg1		
Times		
Reg2		REG1: TIMES REG2
Xpos	Repeat for Ypos. Note that Reg2	REG2: YPOS
Minus	is overwritten by Ypos by the	
Constant	automatic incrementing of the	
(256)	current register, which was Reg1.	REG2: MINUS 256
Load		
Reg2		REG3: LOAD REG2
Use Reg		
Reg2		
Times		
Reg3		REG2: TIMES REG3
Plus	Add the two parts together.	
Reg1		REG2: PLUS REG1
Sqrt	Take the square root.	REG2: SQRT
Div		
Constant	At this stage, 't' has been	
(10)	evaluated.	REG2: DIV BY 10

Load	Since 't' is used in two places	
Reg2	it is copied into Reg3.	REG3: LOAD REG2
Sin	Take the sine of Reg3.	REG3: SIN
Div		
Reg2	Divide (sin t) by t.	REG3: DIV REG2
Plus		
Constant	By adding 0.22, all the negative	
(0.22)	values are removed.	
Times		REG3: PLUS 0.220
Constant	By multiplying by 209, the number	
(209)	1.22 (maximum value = $1.0 + 0.22$)	
	becomes 255.	REG3: TIMES 209
Store		
Integer Store	Put the result in store 4.	
Whole Screen		
D4		REG3: STORE ST4 WS
End		END

The image at the top right of Fig 39 was produced by this method. It will be noticed that the number of the current register has been increasing as the program has developed. This is generally unimportant in a small program such as this with only 12 steps out of the 64 possible. In this example, 'x squared' was held in Reg1, and 'y squared' in Reg2. When they were added, Reg2 could have been added to Reg1 with 'Use Reg', 'Reg1', 'Plus' and 'Reg2'. By this means, Reg2 would be made available again. As it was, 'x squared' was left in Reg1 and was not used again. Alternatively, the Stack! command could be used when REG2 is the current register. Instead of 'Plus' and 'Reg1' in the example, 'Plus' and 'Stack!' would add Reg1 to Reg2 and put the result in Reg1. For a larger program, this example illustrates that it is worthwhile planning how the sum is to be evaluated, and how the registers are to be used.

Example 4. This last example shows how a simple conditional step can be made in a non-linear filter. Assume that the image to be filtered (image R) is in store 1 and that store 2 contains an average (image S). The principle of this non-linear filter is that if a pixel is within a small range of intensity of the local average, it should be kept, or if it is outside this range, it should be replaced by the local average. Such a filter could be used in a similar way to 'MEDIAN SMOOTH' in 'Maths Operations' (p58), to remove noise spikes in an image. 'SMOOTH' in the Maths Operations (described on p47) can be used to produce the average quickly with for example a 3 by 3 filter giving an average which includes the middle pixel. If only the 8 neighbours and not the centre pixel, are to be used, a filter with a zero in the centre and 8 ones round the neighbouring locations could be produced using 'Linear Filters'. There is thus great flexibility in producing the 'average'.

This problem can be split into two parts, namely finding if image R is significantly different from S, and then producing the output image of pieces of R and pieces of S.

Let $Q = >0 (\text{Abs} (R-S) - x)$

where ' >0 ' is the function described on page 132. If the absolute difference between R and S is greater than the parameter ' x ', the value of Q is 1. If the difference is less than or equal to ' x ', the value of Q is 0. Finally, if the output image is $R(1 - Q) + S.Q$, the non-linear filtering operation will be performed.

Command	Comment	'PRINT PROGRAM'
Load		
Integer Store	The 'average' image is subtracted	
Whole Screen	from the original. ie R-S.	
D1		REG1: LOAD ST1 WS
Minus		
Integer Store		
D2		REG0: LOAD ST2 WS
		REG1: MINUS REG0
Abs		
Minus	Q is calculated next.	REG1: ABS
Constant Parameter		
>0	This is ' x ' in the equation above.	REG1: MINUS PARAMETER
		REG1: >0
Load		
Reg1	A copy of 'Q' is left in Reg1.	
Chng Sign		REG2: LOAD REG1
Plus	R(1-Q) is evaluated first.	REG2: CHNG SIGN
Constant		
(1)		REG2: PLUS 1
Times		
Integer Store		
D1		REG0: LOAD ST1
		REG2: TIMES REG0
Use Reg		
Reg1	Q times S is now found.	
Times		
Integer Store		
D2		REG0: LOAD ST2
		REG1: TIMES REG0
Plus		
Reg2	The two parts are added and the	
Store	result put in store 3.	REG1: PLUS REG2
Integer Store		
D3		REG1: STORE ST3 WS
End		END

This program of only 12 steps can be run with a series of different parameters ' x ', to vary the effect of the filter.

PLANE FLICK	PLANE 1					
PLANE SEQUENCE	PLANE 2					
COPY	PLANE 3					
ROTATE	PLANE 4					
DRAW GRID	PLANE 5					
RGB TO INT HUE SAT	PLANE 6					
INT HUE SAT TO RGB	PLANE 7					
STEREO PAIR	PLANE 8					
	OVERLAY 1					
	OVERLAY 2					
	OVERLAY 3					
	OVERLAY 4					
	END					

NORMAL	TOP BOTTOM	
LEFT RIGHT	ROTATE 180 DEG	
ROTATE 90 DEG	ROTATE -90 DEG	SMALL
TRANS POSE		LARGE

WHOLE SCREEN	
TOP LEFT	TOP RIGHT
BOT LEFT	BOT RIGHT

STATUS	HELP	RETURN	IMAGE
--------	------	--------	-------

INPUT FOR COMMAND OR QUIT

Fig 37 The 'Special' page

5.11 Special

This chapter is concerned with manipulations of the display stores and overlay planes, in ways which enable the human eye to position images accurately, observe sequences of images, copy images, rotate images, overlay grids, and interpret uncorrelated sets of images.

PLANE FLICK

SUMMARY

This command allows the user to flick quickly from one image to another, to move one image relative to the other, and to measure the displacement accurately.

COMMAND IN DETAIL

The only two secondary commands specify the first and second stores to be used in the flicking operation. After pressing 'INPUT' for each store, a black and white image from the first store is displayed, followed by a B/W image from the second, then back to the first and so on. If the rolling ball is moved or one of the four cursor control buttons is pressed, the second image is moved relative to the first. When the image is zoomed, more precise movements may be made, as fine as one eighth of a pixel at a zoom of 8 times. By this means, one picture can be compared with another, and in particular, one area in the second image can be correlated by eye with a reference area in the first. When the 'INPUT' button is pressed to terminate the command, the displacement of the second image relative to the first is displayed on the VDU. If the pixel displacement is positive, the second image was positioned to the right of the first, and if the line displacement is positive, the second image was below the first.

When moving the second image relative to the first, if 'CHNG' is pressed, both images can be moved together. Thus when zoomed, an area which is outside the screen, can be moved into the display area. If the 'MODE' button is pressed so that the light is extinguished to signify mode 1, the rolling ball can be used to control the flick rate from one image to the other. Initially, one image is displayed for 1/50 second, and then the other for 1/50 second, etc. This is the fastest possible rate.

PLANE SEQUENCE

SUMMARY

A sequence of stores can be specified and displayed in turn at a variable rate. When the last store has been displayed, the sequence can either reverse direction, or start again from the beginning.

COMMAND IN DETAIL

All that is required for this command is to specify the image planes or overlay planes in the order in which they are to be displayed. Having specified the first one, the user has the option of displaying the whole image, or a quadrant. Assuming the former option, all that is necessary is to specify the other planes in the sequence, finishing with 'End'. The display sequence then commences from the first to the last, followed by the first again, etc. The rolling ball controls the rate at which the planes are displayed. If the 'CHNG' button is pressed, the sequence is from the first to the last and then from the last to the first with the rolling ball controlling the rate. Thus if the first three stores had been selected in that order, initially the sequence would be, 1-2-3-1-2-3-1-2... and after pressing 'CHNG' the sequence would be, 1-2-3-3-2-1-1-2...

This command can be used to look at a sequence of 4 full images in a GEMS system with 4 image planes. If the 'Part Screen' option is chosen, then only a quadrant of the image plane will be displayed at each step. There is an automatic zoom by two so that the quadrant fills the screen. Thus a sequence of four pictures each of 256 pixels and 256 lines, could be put into an image store, and the display sequence could cycle around the four sub-images in the one store. Using all four stores would permit the display of 16 sub-images in a sequence.

Finally, for those systems which have 1024 by 1024 pixel stores, there is a further option of 'Small' or 'Large'. For 'Small', only the top left 512 pixel square is used, whereas for 'Large', the whole store is displayed. Thus 'Part Screen' followed by 'Small' would result in quadrants of the 512 square image being displayed, and 'Part Screen' followed by 'Large' would result in quadrants of the 1024 square image being displayed.

The process is stopped by pressing 'INPUT'.

COPY

SUMMARY

This command allows the user to copy the whole of a store to itself or to another store. A quadrant of one store may be copied to another quadrant in the same store, or to another store. Inversions and rotations during the copy process are also possible.

COMMAND IN DETAIL

The user is first asked to select one of seven possible operations to be performed during the copy process. 'Normal' results in a straight copy with no processing. 'Top Bottom' causes the top line to be replaced by the bottom line, etc in the output image. This is the same as flipping the image about a horizontal line through the centre. 'Left Right' causes the rightmost column to be replaced by the leftmost column, etc. This is the same as flipping the image about a vertical line through the centre. 'Rotate 180 Deg' results in an output image rotated through 180 degrees about its mid point. 'Rotate 90 Deg' results in an output image

which is rotated through 90 degrees in an anticlockwise direction, about its mid point. 'Rotate -90 Deg' is the same but with a rotation in the clockwise direction. Finally, 'Transpose' swaps the x and y co-ordinates. This is the same as flipping the image about the leading diagonal, ie the line from the top left to the bottom right.

Having chosen the processing to be performed, the input store is selected, and the area specified (ie the whole area or one of the four quadrants). The result store is next selected (together with the quadrant, if the whole area was not selected for the input store). Finally, for systems with 1024 pixel square stores, 'Small' can be selected for 512 pixel square images, or 'Large' for the full 1024 pixel square images. The operation is performed by pressing 'INPUT'.

ROTATE

SUMMARY

This command allows the user to select an area from one image and copy a rotated version to the same or to another store.

COMMAND IN DETAIL

The first two commands specify the input and result stores. Two boxes are put on the screen, but since they overlap, they appear as one. Initially, the rolling ball controls the position of the centre of the area in the output store, into which the copy is to be made, and one of the boxes on the screen, representing this area, moves with the rolling ball. After pressing 'CHNG', the rolling ball controls the size of the area in the output store, into which the copy is to be made, and both boxes on the screen change accordingly. Pressing 'CHNG' again permits the user to move the centre of the area to be copied from the input store, and after the final 'CHNG', the box representing this area can be rotated. On the VDU, there are a few words to inform the user of the function of the rolling ball at each step, and the rotation angle in degrees. The 'CHNG' button operates in a cyclic manner allowing the parameters to be changed again, if required. When 'INPUT' is pressed, the area in the input store defined by the box, is copied and rotated into the defined area in the output store. Nearest neighbour resampling is used to produce the result.

DRAW GRID

SUMMARY

A rectangular grid can be put into overlay plane 3, to assist in the location of points or areas. There are no secondary commands, and the grid is produced on a 50 pixel/line spacing if the image has not been geometrically transformed. For a transformed image, the pixel/line spacing is chosen to give about 10 lines in each direction, and reasonable numbers along the axes.

RGB TO INT HUE SAT

SUMMARY

This command performs the transformation from RGB to a representation in intensity, hue and saturation co-ordinates.

COMMAND IN DETAIL

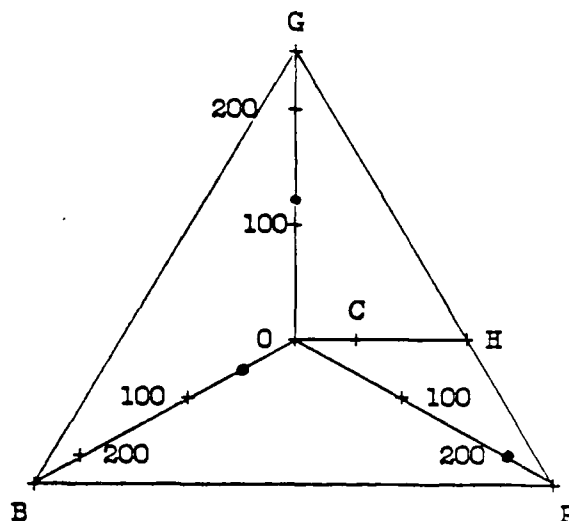
Although the receptors in the human eye are sensitive to red, green, and blue, the brain processes this information and the human would not describe a colour in terms of red, green, and blue, but in terms of intensity (or brightness), hue (or colour), and saturation (or whether it is a strong colour or a pale colour). For example, an area of colour on the TV screen with the red, green and blue components set at 200, 200 and 100 respectively, would be described as fairly bright, yellow and slightly pale, rather than in terms of the primary colours. What is more, the human can determine intensity reasonably well irrespective of colour and saturation, colour moderately well irrespective of intensity and saturation, and saturation with some difficulty. Thus if three sets of information are presented to the human in terms of intensity, hue and saturation, the human mind can separate them. In GEMS, if three stores contain uncorrelated data, (eg the first three images of a principal components analysis, or three images of one area taken in the visible, thermal infra-red, and microwave regions of the spectrum), and the first image is used to control the intensity of the final display on the TV screen, the second the colour, and the third the saturation, the human can separate the three sets of information (ie uncorrelated data remain reasonably uncorrelated to the human). Displaying the stores directly in red, green and blue, mixes the information as far as the human is concerned, and such a scene may be described as garish, confusing, or uninterpretable. The examples below show these effects. It should be noted that the visual spatial resolution is good in terms of intensity, moderate for colour, and poor for saturation. Therefore, if the three GEMS stores contain images having different spatial resolutions, or 'quality' (eg noise, striping, etc), the highest resolution, highest 'quality' image should control the intensity, the next the hue, and the poorest the saturation.

The TV screen has red, green, and blue phosphors, and so cannot display intensity, hue and saturation directly. Thus, given three stores which are to be displayed as intensity, hue and saturation, a transformation must be performed, to produce three new images of the red, green, and blue components, so that they can be displayed on the TV. This operation is performed by the next command. The current command performs the inverse transformation, and the commands are often used as a pair as shown in the examples.

In the particular algorithm in GEMSTONE which is used for the transformation, it is reasonable to make the intensity, I , the sum of the three components, R, G, B . Since each component can have a value of up to 255, ' I ' could be out of the range of the 8 bit store. Hence,

$$I = (R+G+B) / 3.$$

For hue, consider this diagram where the RGB axes are shown inside a triangle with the origin in the middle, and the maxima at the apexes of the triangle. As an example, the colour with RGB components 200,125,50 is shown at C, which is the median of the triangle formed by these components. If a line is drawn from the origin, O, through the point C, it cuts the boundary at H. The position of H is independent of I as defined above. If the boundary has a linear scale with 0 at the top of the R axis, 85 at the top of the G axis, 170 at the top of the B axis, and 255 at the top of the R axis again, a number can be given to every possible colour within the RGB triangle.



From simple geometry, it turns out that the hue (H) is,

H =	$\frac{G-B}{3(I-B)} \cdot 85$	$\frac{B-R}{3(I-R)} \cdot 85 + 85$	$\frac{R-G}{3(I-G)} \cdot 85 + 170$
for colours between	red and green	green and blue	blue and red

Despite the fact that the equation is defined in three parts, it is continuous at the boundaries. For a monochrome image, the denominators are all zero and the equations are indeterminate. Since the saturation is also 0, it does not matter which number is chosen for hue, and so it is set to zero.

Turning to saturation, consider a colour between red and green, if there is no blue component at all, then the colour cannot be made any stronger, and the saturation is set to 255. In the diagram above, as the blue component tends towards the origin, the colour, C, tends towards the RG boundary. When the blue component is zero, it cannot be reduced further, and hence the colour, C, is at maximum saturation. As the colour, C, is moved to the origin, the saturation reduces, to 0. The equation used in GEMSTONE for saturation (S) is :

S =	$\frac{I-B}{I} \cdot 255$	$\frac{I-R}{I} \cdot 255$	$\frac{I-G}{I} \cdot 255$
for colours between	red and green	green and blue	blue and red

The inverses of these operations are given by the equations,

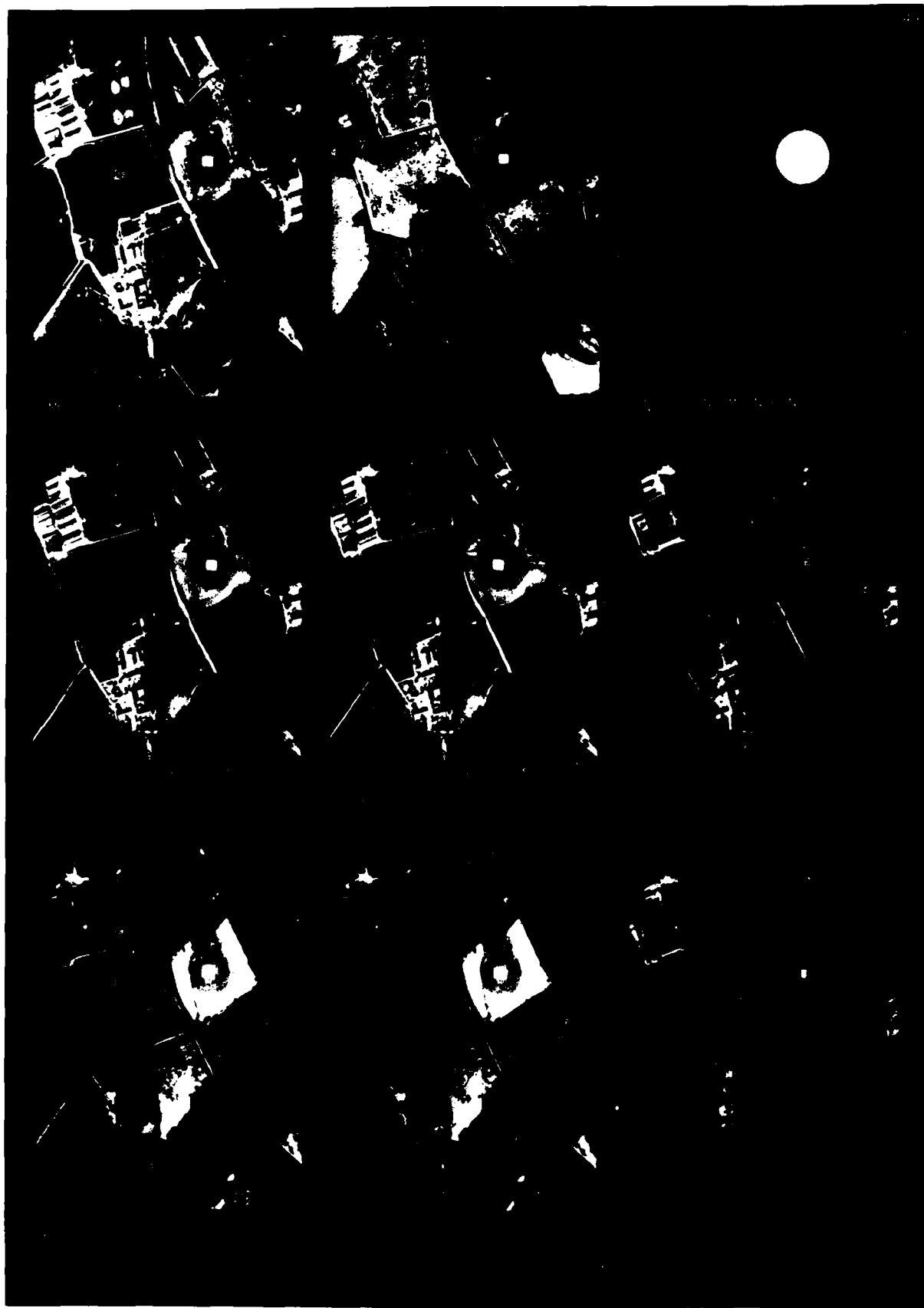
$$R = I(1 + \frac{2S}{255} - \frac{SH}{85.85}), \quad G = I(1 - \frac{S}{255} + \frac{SH}{85.85}), \quad B = I(1 - \frac{S}{255})$$

It should be noted that whereas the transformation to IHS has no scaling problems, it is possible to exceed the range of an 8 bit store doing the inverse operation. For example, the bright red colour with RGB components 255,0,0 would have IHS components of 85,0,255. If the intensity image, containing the 85, were stretched so that the 85 became 170, then the inverse transformation would result in RGB components of 510,0,0, which is outside the 8-bit range. The 510 would in fact be truncated to 255, the maximum value. In general, the user should check if the histograms of the RGB components resulting from the IHS to RGB transformation, show significant saturation, and if so, the intensity image should be contrast stretched to reduce the maximum value, and the transformation repeated.

The ideas above are illustrated in the following examples. The colour picture at the top left of Fig 38 shows a region near Bosherton in Wales, taken by an airborne scanner system. The pixel size is about 2m. To the right and below this picture are the images of intensity, hue and saturation. Whereas the intensity image looks reasonable, the hue image looks odd until it is noticed that the brightest parts correspond to areas in the original which are blue-red, and that the darkest parts are red-green in the original, etc. The saturation image is also difficult to relate to the original. Here the brightest parts are the most saturated. Thus the river is highly saturated, which is difficult to see in the original, because the intensity is so low.

As a first example, it may be considered that the false colour composite has a restricted range of colours. There are certainly no shades of green, and a histogram of the hue image showing some well separated peaks confirms this. If the hue image is stretched using 'AUTO EQUALISE' in the contrast stretch chapter (see p28), then all colours become equally likely. On performing the IHS to RGB transformation using this stretched hue image, the more colourful picture at the bottom left of Fig 38 results. If it is now considered that the colours in this picture are weak, the saturation image can be stretched (using 'AUTO LINEAR' or 'MANUAL' in the contrast stretch chapter). The resulting picture after the IHS to RGB transformation, displayed at the middle bottom of Fig 38, shows a wide range of strong colours. This example demonstrates an alternative way of stretching an image to make it more colourful or attractive, and also more matched to the eye and brain, thereby maximising the information transfer to the human. By carefully choosing the contrast stretch function, particularly for the hue image, a wide range of colour images can be produced, including ones with colour biases, ones with particular ranges of colour expanded, and as shown in the next example, ones with certain colours excluded.





The second example shows how uncorrelated images can be merged more effectively using IHS techniques. At the top right of Fig 38 is a thermal infra-red image of the region, with the bright parts corresponding to areas which are hot, or have high surface emissivities, or both. Replacing the red component of the false colour composite with the thermal image results in the picture at the middle right. Some bright red areas are obviously 'hot', but elsewhere, it is difficult to determine the 'temperatures', since the intensity of the red component in a colour cannot easily be seen. By replacing the hue image with the thermal image, and setting the saturation image to a constant value of 200 throughout, the IHS to RGB transformation yields the image at the bottom right. The information from the false colour composite is clearly seen as intensity, and the thermal information is clearly seen in different colours. Since red is normally considered hot and blue cold, a stretch was applied to the thermal image before the transformation. A linear stretch, with 255 being mapped to 0 and 0 being mapped to 170 ensured that the bright parts of the thermal image would be red in the final image, the dark parts would be blue, and the mid-grey parts would be green. This stretch also resulted in there being no values between 170 and 255, (ie between blue and red), since this would confuse the 'temperature' scale.

The last example at the top of Fig 39, shows a principal components image displayed first as RGB and secondly as IHS. Having decorrelated the three bands in the false colour composite by the principal components process, the RGB presentation mixes up the data again, as far as the human is concerned. The IHS presentation attempts to preserve the decorrelation, with intensity showing the first component, hue the second, and saturation the third, although it is very difficult to see the saturation information. There is generally considerable noise in the third band of a principal components analysis, and hence it is convenient to display this band as saturation, to which the human is least sensitive.

One further example not illustrated here concerns the combination of images with different spatial resolutions. Since the eye is most sensitive to intensity variations, the highest resolution imagery should be used to control the intensity. For example, the Spot satellite produces a single band image with 10m pixels and a three band image with 20m pixels. Starting with the three band image magnified to give 10m pixels, (note that this mathematical process does not change the inherent 'resolution') the three bands are transformed to the IHS representation. The intensity image is then replaced by the single band high resolution image and the inverse transform is performed. Whereas the original colour image had 20m resolution, this new image appears to be a colour image with 10m resolution. Other sets of images of different resolutions may be combined in a similar way to give high resolution colour pictures. The low resolution images must be processed to have the same pixel size, orientation, etc before the conversion to IHS representation.

All that is required for this command, is to specify the three source stores holding the red, green, and blue bands, whether the whole screen or a quadrant is to be used, and the result stores for the intensity, hue and saturation images.

INT HUE SAT TO RGB

SUMMARY

This command performs the transformation from intensity, hue, and saturation, to RGB co-ordinates.

COMMAND IN DETAIL

This command is the inverse of the 'RGB TO INT HUE SAT' command and uses the equations described above. All that is required, is to specify the three input stores holding the intensity, hue, and saturation images, whether the whole screen or a quadrant is to be used, and the result stores for the red, green and blue images. Care should be taken to ensure that the results are within the 8-bit range of the stores, as described above.

STEREO PAIR

SUMMARY

Using this command, one image can be converted into a stereo pair of images, the displacements in the stereo images being controlled by a separate 'height' image.

COMMAND IN DETAIL

If the user were to hold a pencil in front of one eye pointed directly at the eye, then with the other eye shut, the top of the pencil would appear to lie directly over the bottom of the pencil. This would be the plan view, or the non-stereo image. If the pencil were pointed at the nose, then, to the left eye, the top of the pencil would appear to be displaced to the right of the bottom and vice versa for the right eye. The amount of the displacement is proportional to the 'height' of the pencil. At the mid point of the pencil, the displacement would only be half that at the top. This principle can be applied to produce a stereo pair of images from a single one. If there is information about the height at all points within the image, then a 'high' pixel can be moved to the right for the left eye and vice versa for the right eye. The 'higher' the pixel, the further the displacement. In a line of 512 pixels, there will be places where a number of pixels are apparently squeezed together, and other places where they are pulled apart. This line of 512 irregularly spaced pixels is resampled using nearest neighbour interpolation to produce a line of 512 regularly spaced pixels. In GEMSTONE, height is represented on a scale of 0 to 255. A pixel at height 0 is moved in neither stereo image. A pixel at height 8 is moved one place to the right for the left eye but the image for the right eye is unaltered. It is not until the height has reached 24 that the image for the right eye is moved one place to the left. At 16 more units of height, the image for the left eye is displaced one more step to the right. This process repeats with pixels in alternate images being displaced until the maximum height is reached. There are 16 steps in this displacement process, and therefore 16 height levels in the stereo pair. The number of levels is a compromise between giving a good visual height resolution and not distorting the image too much.

A digital terrain model (DTM), is a set of point measurements of height on a regular grid. This is exactly like a digital image, the only difference being that the pixel value is a measure of the average height of the area covered by the pixel. If the regular grid of points in the DTM corresponds exactly to the pixel spacing in the image, then the DTM can be used to control the generation of the stereo pair.

At the top right of Fig 39 is a computer generated DTM of a mountain. It is in fact the image of the sine function described in example 3 in the arithmetic chapter. The brightest parts of the image represent the highest points of the mountain. It is a conical hill with rings round it. Density slicing shows the contours. 'TRANSECT GRAPH' in the contrast stretch chapter would illustrate the side view of the hill, and the 3-D routines in the text and graphics chapter would show more general views. This mountain, which is centred on the circular feature in the aircraft scanner image, has been used to control the generation of the stereo pair of images shown in the middle of Fig 39. Many users will be able to see the stereo effect in the two B/W images without the aid of a stereoscope, simply by holding the photograph at a comfortable distance, looking at a bright feature in one of the two images, and allowing the eyes to relax as though looking at an object at infinity. The two images will fuse and after some effort to refocus, the stereo effect should become apparent. A stereoscope could be used for the same effect. In the case of the colour picture at the middle right, one of the stereo pair of images is in red, and the other in green. The distortion from one image to another is obvious. If this image is viewed through coloured spectacles, with a red filter over the left eye and a green one over the right, a stereo effect should again be seen. Stereoscopy is possible directly on the TV screen using the red and green guns only, and the coloured spectacles, but for the separate images, photographic products would have to be made.

It is not necessary to use real height information to control the generation of the stereo pair. The thermal image taken by the aircraft scanner for example, could be used to produce an image where the apparent height is a measure of 'temperature'. This is shown in the pair of pictures at the bottom left of Fig 39. It also shows that stereo pictures can be made in colour. There are some fairly sharp boundaries between light and dark areas in the thermal image. This would correspond to very steep (unrealistic) slopes being observed in the stereo pair, and in fact the example shown is difficult to interpret in places. If the thermal image is smoothed quite severely, the steep slopes are removed, and a much more interpretable pair of images results. These are shown at the bottom right in red and green. This illustration with a thermal image controlling height is simply an example of using one, generally uncorrelated image to control the height information. It is an example of another way of displaying uncorrelated images in such a way that the human can still see them uncorrelated.

The inputs required for this command are the source image, whether the whole screen or a quadrant is to be used, the control image and the two stores for the results.

5.12 Fourier Transforms

Within this chapter, the user can produce Fourier transforms (FT) of images, perform a range of manipulations in the transform space, and then perform the inverse Fourier transform. It is assumed that the user is familiar with the theory and so only a summary is provided below. This summary only addresses those aspects directly applicable to GEMSTONE, and is far from rigorous.

Considering first the continuous Fourier transform in one dimension, let $f(X)$ be a continuous function of a real variable, X . The Fourier transform of $f(X)$, denoted by $F(U)$, is given by the equation,

$$F(U) = \int_{-\infty}^{+\infty} f(X) \cdot \exp(-pUX) \cdot dX$$

where $p = 2\pi i$, i being the square root of -1 . The inverse Fourier transform is

$$f(X) = \frac{1}{2\pi} \int_{-\infty}^{+\infty} F(U) \cdot \exp(pUX) \cdot dU$$

Whether $f(X)$ is a real or complex function, $F(U)$ is generally complex, and can be written in the form,

$$F(U) = R(U) + iI(U)$$

In GEMSTONE, the Fourier transform produces a real image (ie $R(U)$) in one store, and an imaginary image (ie $I(U)$) in another. An alternative way of representing $F(U)$ is in the form of magnitude and phase, where,

$$F(U) = |F(U)| \cdot \exp(i\phi(U)).$$

$|F(U)|$ is the square root of $(R(U) \cdot R(U) + I(U) \cdot I(U))$

and $\phi(U)$ is the phase angle whose tangent is $I(U)/R(U)$.

U is often called the frequency variable. $|F(U)|$ is known as the Fourier spectrum of $f(X)$ and in GEMSTONE, the image of this function is extremely useful for examining the spectral content of an image. In general, it is virtually impossible for the human to interpret the phase image.

Digital images are not continuous functions, but discrete samplings of the real world into pixels. Still considering one dimension, it is assumed that there is a finite number of samples, N , and that the sampling is performed at regular intervals with a spacing of x . Thus the sample points are at nx , where n can have values of $0, 1, 2, \dots, N-1$. The Fourier transform of the N samples is defined to be

$$F(U) = 1/N \sum_{n=0}^{N-1} f(nx) \cdot \exp(-punx)$$

The form of this is similar to the continuous case described above.

However, $F(U)$ is still a continuous function, providing problems for a computer. In the same way that $f(X)$ is sampled, the transformed function can be sampled at $U = ku$, for $k = 0, 1, 2, \dots, N-1$, with $u = 1/(Nx)$. Substituting for u in the equation above gives the discrete Fourier transform (DFT),

$$F(ku) = \frac{1}{N} \sum_{n=0}^{N-1} f(nx) \cdot \exp(-pkn/N) \quad \text{for } k = 0, 1, \dots, N-1.$$

If k is replaced by $k+N$, the value of the right side of the equation remains the same. Therefore $F(ku) = F((k+N)u)$, etc, and so the DFT is cyclic with period N . The inverse transform is also cyclic implying that $f(nx)$ is cyclic, ie it has defined values in the range $n=0$ to $n=N-1$, which are repeated in the intervals $n=N$ to $2N-1$, $n=2N$ to $3N-1$, etc, and $n=-N$ to -1 etc. This assumption of the periodicity of the input function has effects that will be explored later.

To compute the DFT requires N operations (one complex multiplication and addition) for each value of k . For the N values of k there are $N.N$ operations, and as N increases, the calculation becomes impractically large. The fast Fourier transform (FFT) overcomes this problem as follows. If $N = 2.M$, then

$$\begin{aligned} F(ku) &= \frac{1}{2M} \sum_{n=0}^{2M-1} f(nx) \cdot \exp(-pkn/2M) \\ &= \frac{1}{2M} \sum_{n=0}^{M-1} f(2nx) \cdot \exp(-pkn/M) + \frac{1}{2M} \sum_{n=0}^{M-1} f((2n+1)x) \cdot \exp(-pkn/M) \cdot \exp(-pk/2M) \end{aligned}$$

Substituting $k+M$ for k , shows that $F((k+M)u)$ is

$$= \frac{1}{2M} \sum_{n=0}^{M-1} f(2nx) \cdot \exp(-pkn/M) - \frac{1}{2M} \sum_{n=0}^{M-1} f((2n+1)x) \cdot \exp(-pkn/M) \cdot \exp(-pk/2M)$$

It should be noticed that this last equation is the same as the previous, apart from one sign change. Thus, if $F(0)$ is calculated, $F(M)$ can be found by subtracting the two summations, and similarly for $F(1)$, $F(2)$, etc. The calculation effort has been approximately halved. It should also be noticed that the form of each summation is the same as the DFT above except that each summation is over M ($= N/2$) terms. If M is not a prime number, each summation in this last equation can be further divided, thereby further decreasing the effort. Most implementations of the FFT on computers require N to be a power of 2, so that this process of subdividing can be repeated until there is only one term in the summation, ie no summation at all. For GEMSTONE, $N = 256$ or 512 in each dimension. The calculation is reduced from a large number of complex multiplications and additions to a few such operations plus a great deal of re-organisation of the original data and partial results, so that the multitude of terms are added and subtracted correctly to give $F(ku)$ for all values of k . This sequence of adding and subtracting terms is called a 'butterfly'. It can be shown that, whereas there are $N.N$ operations in the DFT, there are $N \cdot \log_2(N)$ operations in the FFT, where $\log_2$ is log to the base 2. For $N = 256$, $N.N = 65,536$ and $N \cdot \log_2(N) = 2048$, so that the FFT is 32 times faster. This does not include the extra time required to re-organise the data in the FFT, but nevertheless there are still considerable savings. For an image of 512.512 elements, the difference in time should be greater than 1000.

The DFT can be extended to 2 dimensions, (x, y) in real space, and (u, v) in transformed space, so that

$$F(ku,lv) = \frac{1}{N \cdot N} \cdot \sum_{n=0}^{N-1} \sum_{m=0}^{N-1} f(nx,my) \cdot \exp(-p(kn+lm)/N)$$

and the inverse transform is

$$f(nx,my) = \sum_{k=0}^{N-1} \sum_{l=0}^{N-1} F(ku,lv) \cdot \exp(p(kn+lm)/N)$$

A square array has been assumed in GEMSTONE. It should be noted that the summations can be separated and two 1-D transforms can be performed. Thus

$$F(ku,lv) = 1/N \sum_{n=0}^{N-1} F(nx,lv) \cdot \exp(-pkn/N)$$

$$\text{where } F(nx,lv) = 1/N \sum_{m=0}^{N-1} f(nx,my) \cdot \exp(-plm/N)$$

In summary, GEMSTONE performs a 2-D Fourier transform on arrays of 256 or 512 pixels square, by separating the problem into two 1-D transforms as above, and performing the two sequential FFTs as outlined earlier.

The following examples demonstrate the operation of the Fourier transform on some simple functions.

Example 1 : $f(nx) = A$, a constant.

$$\begin{aligned} F(ku) &= 1/N \sum_{n=0}^{N-1} A \cdot \exp(-pkn/N) \\ &= A/N \sum_{n=0}^{N-1} [\cos(2\pi \cdot kn/N) - i \cdot \sin(2\pi \cdot kn/N)] \text{ since } \exp(iZ) = \cos(Z) + i \cdot \sin(Z) \\ &= \begin{cases} 0 \text{ for } k \neq 0, \text{ because the sum of equispaced samples of a sine or cosine over a complete cycle is zero.} \\ A/N \sum_{n=0}^{N-1} \cos(0) + i \cdot \sin(0) = A \text{ for } k=0. \end{cases} \end{aligned}$$

Thus, if the input function is a constant, the FT has only one non-zero value at $F(0)$, equal to the constant. More generally, it can be shown that $F(0)$ is the average value of the input function.

Example 2 : $f(nx) = A \cos(2\pi n/N)$,
as shown in the diagram.

$$F(ku) = \frac{1}{N} \sum_{n=0}^{N-1} A \cos(2\pi n/N) \cdot \exp(-j2\pi kn/N)$$

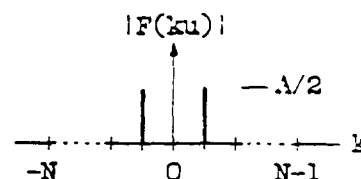
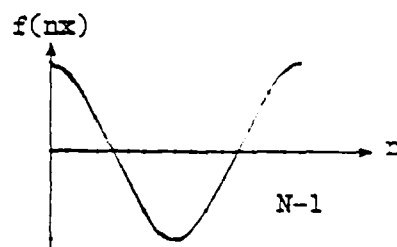
$$= \frac{A}{N} \sum_{n=0}^{N-1} \{ \cos(2\pi n/N) \cdot \cos(2\pi kn/N) - j \cos(2\pi n/N) \cdot \sin(2\pi kn/N) \}$$

$$= \frac{A}{2N} \sum_{n=0}^{N-1} \{ \cos(2\pi n(1+k)/N) + \cos(2\pi n(1-k)/N) - j \sin(2\pi n(1+k)/N) + j \sin(2\pi n(1-k)/N) \}$$

$$= \begin{cases} 0 & \text{for } k \neq 1 \text{ or } k \neq -1, \text{ by the same argument as above.} \\ A/2 & \text{for } k = 1 \text{ or } k = -1. \end{cases}$$

The magnitude, as defined earlier, is $A/2$, with the phase equal to zero. The diagram of the magnitude is shown at the right. Note that $F(0)$ is zero, confirming that the average value of $f(nx)$ is zero.

A similar calculation for $A \sin(2\pi n/N)$ would give the answer, $F(-1) = j.A/2$ and $F(1) = -j.A/2$, F being zero for all other values of k . The amplitude is still $A/2$ as for the cosine, but the phase is now $\pi/2$.



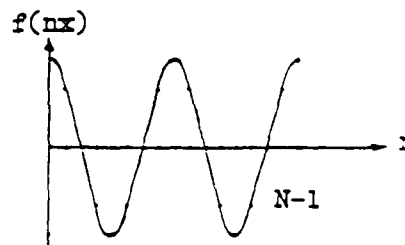
Example 3 : $f(nx) = A \cos(2\pi 2n/N)$,
as shown in the diagram.

By the same process as used in example 2, the reader can verify that

$$F(ku) = 0 \text{ for } k \neq 2 \text{ and } k \neq -2$$

$$F(2u) = A/2$$

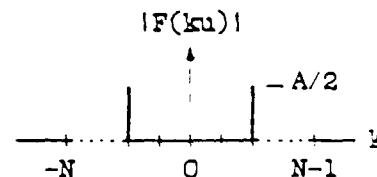
$$F(-2u) = A/2$$



The diagram of the DFT of this function is shown at the right. It is the same as in example 2 except that the spikes have moved to $k = 2$ and -2 .

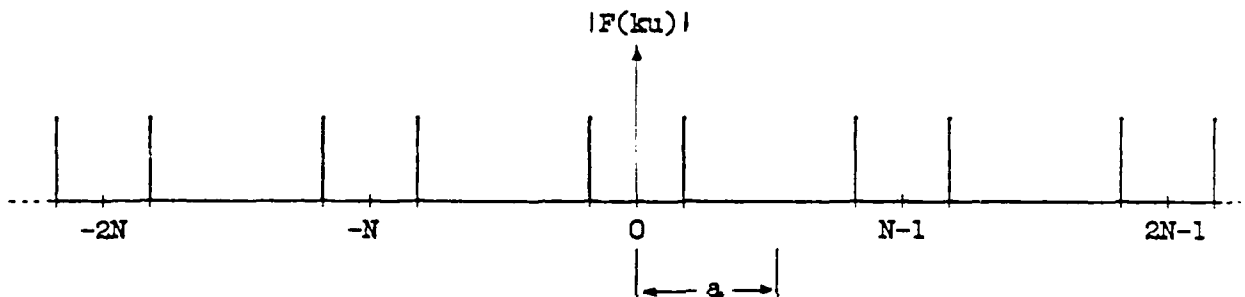
This process could be repeated for functions with more cycles within the interval 0 to $N-1$ up to the maximum frequency of $N/2$. At this maximum frequency, the function is positive

at one sampling point, negative at the next point, positive at the next, etc., and all points have the same amplitude. If a function with just over $N/2$ cycles were sampled at N points, it could not be differentiated from a function with just less than $N/2$ cycles. This phenomenon is called aliasing, and for GEMS users it should not be a problem, provided the creators of the original image data have performed the reasonable operation of limiting the bandwidth of the input function before sampling.



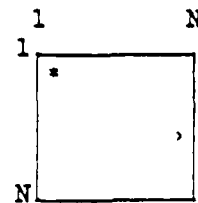
The conclusions from these examples are that a constant function is transformed to a spike at zero and a sinusoid is transformed into a pair of spikes whose distance from the origin is proportional to the frequency of the sinusoid. For a complicated input function, the FT will produce an output of many spikes at many frequencies with the amplitude of each spike being a measure of how much of each frequency is in the input function. By manipulating the transformed function, certain frequencies can be enhanced, reduced, or even removed. This will be shown later.

Before considering examples of simple images in GEMS, it is necessary to introduce one more property of the DFT. For a real function $f(kx)$, the complex conjugate of $F(-ku)$ is equal to $F(ku)$. This can be shown by taking the complex conjugate of the DFT equation for a real input function, and substituting $-k$ for k . It follows that $|F(-ku)|$ is equal to $|F(ku)|$ which is apparent in the diagrams of $|F(ku)|$ above. With this and the previous property that the FT is periodic with period N , the full transformed space for a single sinusoid is :

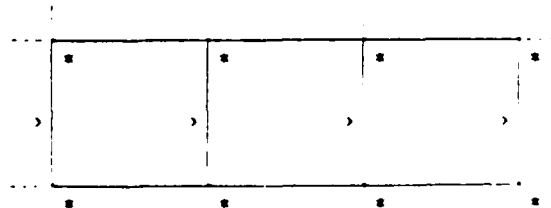


The region denoted by 'a' is all that is required to describe the whole space. This property of using $N/2$ samples in the transformed space (for real input functions only) rather than all N is used in GEMSTONE to save computing time. This last diagram also provides a graphical explanation of aliasing. In the region 'a', there is one spike which moves to the right as the frequency increases. Unfortunately, at the same time, the spike to the left of $N-1$ moves to the left. When the frequency exceeds, $N/2$, this latter spike enters the region 'a' giving erroneous answers.

Extending these considerations to two dimensions, consider an N by N image ($N = 256$ or 512 in GEMS), containing the objects * and . . A diagram of the image which would appear on the GEMS screen is shown here.



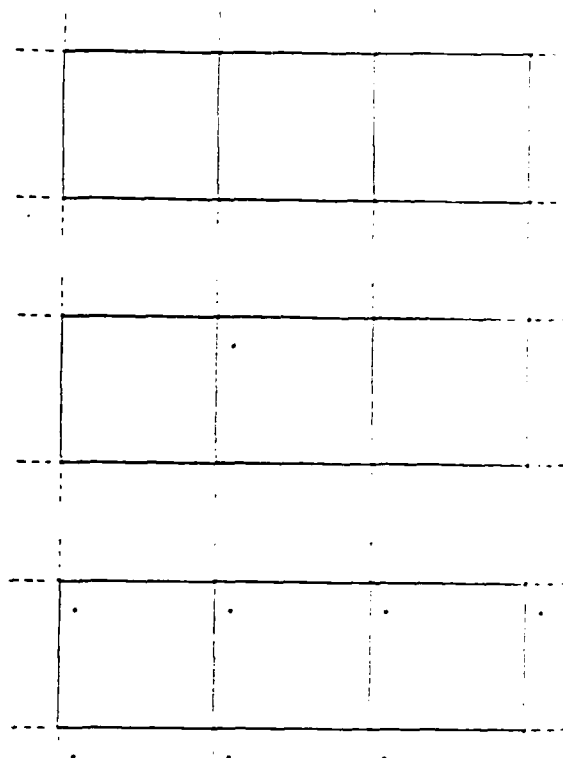
The assumption of symmetry means that this N by N image is only one of an infinity of copies, and the input image to the DFT is actually that illustrated on the right.



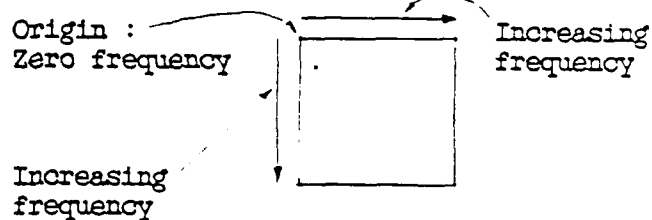
As has been shown, the output from a DFT in 1-D is an infinite line made from copies of a line segment which is Nx long. The extension to 2-D is illustrated here, where the space has been divided into an infinite number of N by N boxes.

The DFT of a sinusoid, is essentially a spike. The 2-D version of this is a spot in the plane as shown here and illustrated later in example 6. The intensity of the spot is proportional to the amplitude of the spike.

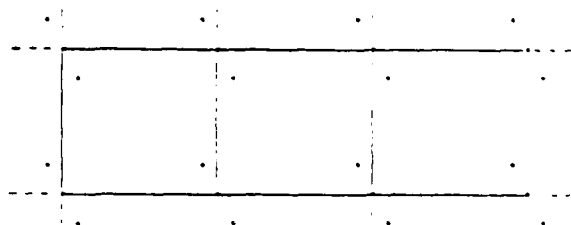
Because of the periodicity of the DFT, ie for 2-D, $F(ku,lv) = F((k+N)u,(l+N)v)$ there are an infinite number of spots in the transform plane as shown here.



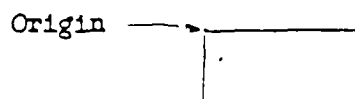
Of course, GEMS cannot display an infinite plane. A single N by N box contains all the information for all possible inputs including complex ones such as synthetic aperture radar images.



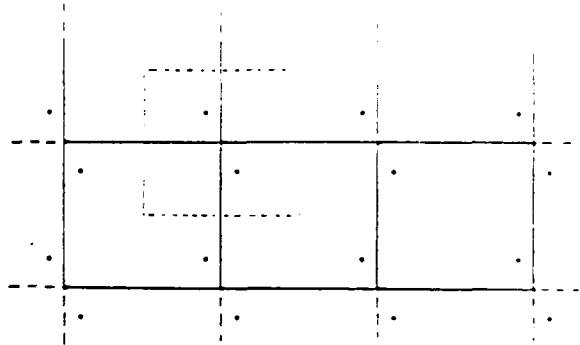
For the case of a real image, which is the type that most users are likely to handle, the extra condition that the complex conjugate of $F(-ku,-lv)$ is equal to $F(ku,lv)$, duplicates all the points in the transform plane as shown here.



There are now two redundant spots within the N by N square. Thus only half the square need be calculated and displayed. In GEMSTONE the top half has been chosen.



Many presentations of the magnitude of a DFT have the zero frequency in the middle. This can be achieved by shifting the N by N box as shown by the dotted line in this diagram. Starting from the previous display of half a square, GEMSTONE can recreate the missing half, and re-order the information to produce this display.



Example 4 : An image of a sine wave is shown in Fig 40(a). It was generated with the Arithmetic package, using the equation,

$$\text{Intensity} = 127 \cdot \{ \sin(\pi \cdot (xpos-1)/8) + 1 \}$$

There are exactly 16 cycles in the 256 pixel wide image. The DFT (Fig 40(b)), has a bright spot (enlarged for clarity) at the origin indicating a non-zero average for the input image, and one other bright spot (also enlarged) down the ku axis, indicating a single frequency in the input image in the nx direction. This is analogous to the 1-D versions given earlier, in examples 2 and 3. It may be considered that the DFT axes are the wrong way round, but they are quite arbitrary, and with this arrangement, the process runs at maximum speed.

Example 5 : This is like example 4 but the sine wave is in the other direction, as shown in Fig 40(c). The equation used was,

$$\text{Intensity} = 127 \cdot \{ \sin(\pi \cdot (ypos-1)/32) + 1 \}$$

The DFT of this image (Fig 40(d)) also has a spot at the origin and this time a spot on the lv axis, because the wave was in the my direction. It is closer to the origin than was the case in example 4, as the frequency is lower.

Example 6 : A combination of examples 4 and 5 with the equation,

$$\text{Intensity} = 127 \cdot \{ \sin(\pi \cdot ([xpos-1]/8 + [ypos-1]/32)) + 1 \}$$

results in an image of a sine wave at an angle to the axes (Fig 40(e)). The DFT (Fig 40(f)) shows one spot at the origin and another in the plane and not on an axis. Considering a spot in the transform plane as corresponding to a sine wave in a particular direction in the original image is very important in filtering operations. To further help the user, there is a command in GEMSTONE ('DISPLAY WAVELENGTH') which, when given a point in the transform plane, will draw two lines as adjacent crescents of a sine wave at the correct orientation.

Example 7 : All the examples of sine waves so far have had periods which are sub-multiples of 256. Therefore, the displayed image fits smoothly with the next copy of the image in the infinite space that has been assumed. When the period is not a sub-multiple of 256, there is a discontinuity or sharp edge. Edges produce a wide range of frequencies, and this effect is shown using the image in Fig 40(g) where the sine wave from example 5 has had '32' changed to '31'. In the transform image in Fig 40(h), many spots are visible, (again, the top line has been copied to the next two for clarity) corresponding to the many frequencies generated by these edges. It is not that these are errors, (the inverse transform will reproduce the input image correctly), but the user should be aware of them when filtering. If, for example, the high frequencies are removed by a smoothing filter then, on performing the inverse transformation, not only will the sharp edges within the body of the image be removed, but there will be odd effects at the boundaries of the image. These effects are inevitable, but will be limited to the region close to the boundary, and the user should ensure that his area of interest does not approach this boundary.

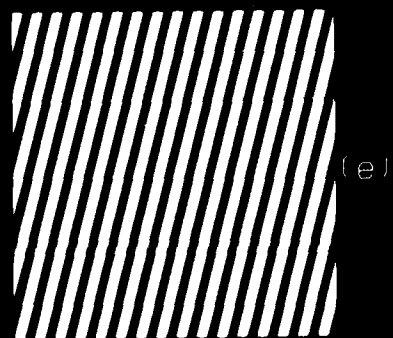
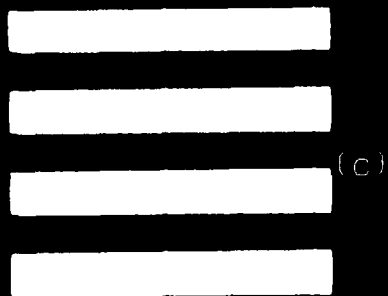
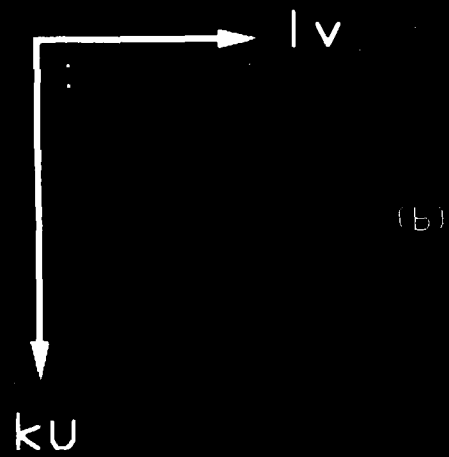
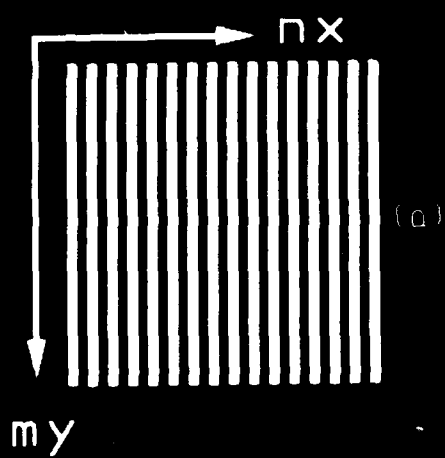


Fig 40 Examples of sine wave transforms

Having outlined the theory, and the display of the magnitude of the transforms, some uses of the FT are now demonstrated with the aid of the following examples.

Fig 41(a) is part of the Welsh scene introduced on page 146. Suppose it is required to apply a smoothing filter to it. The first operation is to perform the 'FORWARD TRANSFORM' which produces a real and an imaginary image as described at the start of this chapter. With the command 'MAGNITUDE PHASE', these two images can be converted into a magnitude image which can be stored in one of the display stores, (and a phase image, if required). The magnitude image in Fig 41(b) has the origin (zero frequency in both directions) at the top left. Immediately below this in Fig 41(d) is the re-ordered magnitude image with the origin at the centre. This was created using 'ADJUST TRANSFORM'. Features such as dense clusters in the magnitude image can be identified with objects in the original image. For example, the cluster forming the line A-A consists of many frequencies in one direction. Many frequencies suggests a sharp edge, and a cluster of reasonable intensity suggests quite a major feature in the original image. With the command 'DISPLAY WAVELENGTH' the cursor can be put over any part of the cluster, to determine the direction of a major edge. In this case, it was found that the bridge at 'A' in the original image (Fig 41(o)) was in the direction indicated by the cluster A-A. Similarly, the cluster in the line B-B was associated with the road at 'B', and the cluster C-C with the field boundaries and road at the places labelled 'C'. The cluster lines D-D and E-E are associated with defects in the image. At the bottom right corner of the original, there are some dark vertical lines which are artefacts of the imaging system. These correspond to the lines E-E. D-D is associated with a horizontal line structure generated by the imaging system and this is virtually invisible to the eye. In images where there is a false structure, which is disturbing for the human, or for a classification process, etc, this structure can be removed with little damage to the rest of the image. The method is illustrated later.

The basic idea behind filtering, using Fourier transforms, is that having transformed the image into the frequency domain, parts of these frequency images can be brightened (ie multiplied, or stretched, etc) so that on performing the inverse transform, these frequencies will be enhanced. The picture in Fig 41(f) has two sectors of a circle, at level 200, fading gently into a background at level 50. Multiplying the real and imaginary transformed images by this picture, using 'MULTIPLY TRANSFORMS' will mean that the low frequencies are multiplied by 200 and the high ones by only 50. The high frequencies are therefore reduced by a factor of 4 relative to the low ones. This picture of a filter, was produced using 'DESIGN FILTER', and the position of the sectors of the circle was set with the aid of the magnitude image (Fig 41(b)). When the modified real and imaginary images were transformed back, using 'BACKWARD TRANSFORM' (dividing by 200 in the process), the low frequencies remained unchanged. This can be seen in Fig 41(e). However, the high frequencies, multiplied by only 50 in the transformed space, and divided by 200 during the inverse transformation, have been reduced by a factor of 4. Thus the sharp edges have been partially smoothed. Note that '200' was chosen simply so that the filter image could be seen. Any number could have been chosen provided that the same number was used during the inverse transform. Since floating point format rather than fixed integer format is used for all these calculations, there are no overflow problems.

A similar operation was performed for an edge enhancement filter shown in Fig 42(b), where the background is at level 200 and close to the origin is at level 50. The number 50 was used again in the inverse transform which resulted in an image which kept the low frequencies constant, but enhanced the high ones, thereby sharpening the edges as can be seen in the image in Fig 42(a).

Both of these simple examples could have been performed using the commands within 'Linear Filters'. Where this method excels over convolutional filtering, is in the ease with which highly specific filters can be constructed. Suppose it is required to remove the road labelled 'B' in Fig 41(c). Since a very specific cluster, B-B in the transform space (Fig 41(d)), is associated with the road, a filter for this can be created, as shown in Fig 42(d), where the dark part is at level 0. On multiplying the real and imaginary images by this filter, the frequencies associated with the road are completely removed. The inverse transform results in the image in Fig 42(c), where little trace of the road remains. Notice that the bridge, which also has sharp edges in almost the same direction remains with just a little reduction in contrast. A practical convolution filter is unlikely to be so selective. This technique can also be used to remove systematic noise from an image. If there is a line structure in the image, such as 'banding' from a linescan detector system, it can be identified in the transformed images, and removed almost completely with little damage to the rest of the scene. It must be noted that if part of the image has a structure very close to that which is to be removed, it will be removed as well. However, as illustrated here with the road and the bridge, very little information need be lost, if the filter is designed carefully.

It has just been shown how selective a 'smoothing' filter can be made. Equally, edge enhancement filters can be constructed, and for example, if the dark part in Fig 42(d) were lighter than the background it would enhance edges over a very small angle. Thus the road could be enhanced without affecting the bridge. It should be noted however, that in the same way that the road was so effectively removed, for edge enhancement, 'non-existent' edges can be created. If a highly selective edge enhancement filter were applied to a random image the human eye would probably see edges in it. The more highly tuned the filter, the greater the chance of the eye being deceived into 'seeing' non-existent edges. This is particularly important in such applications as the use of remote sensing images for examining geological structure. When such filters are used, they should also be applied to random images to give the user an impression of the likelihood of false edges.

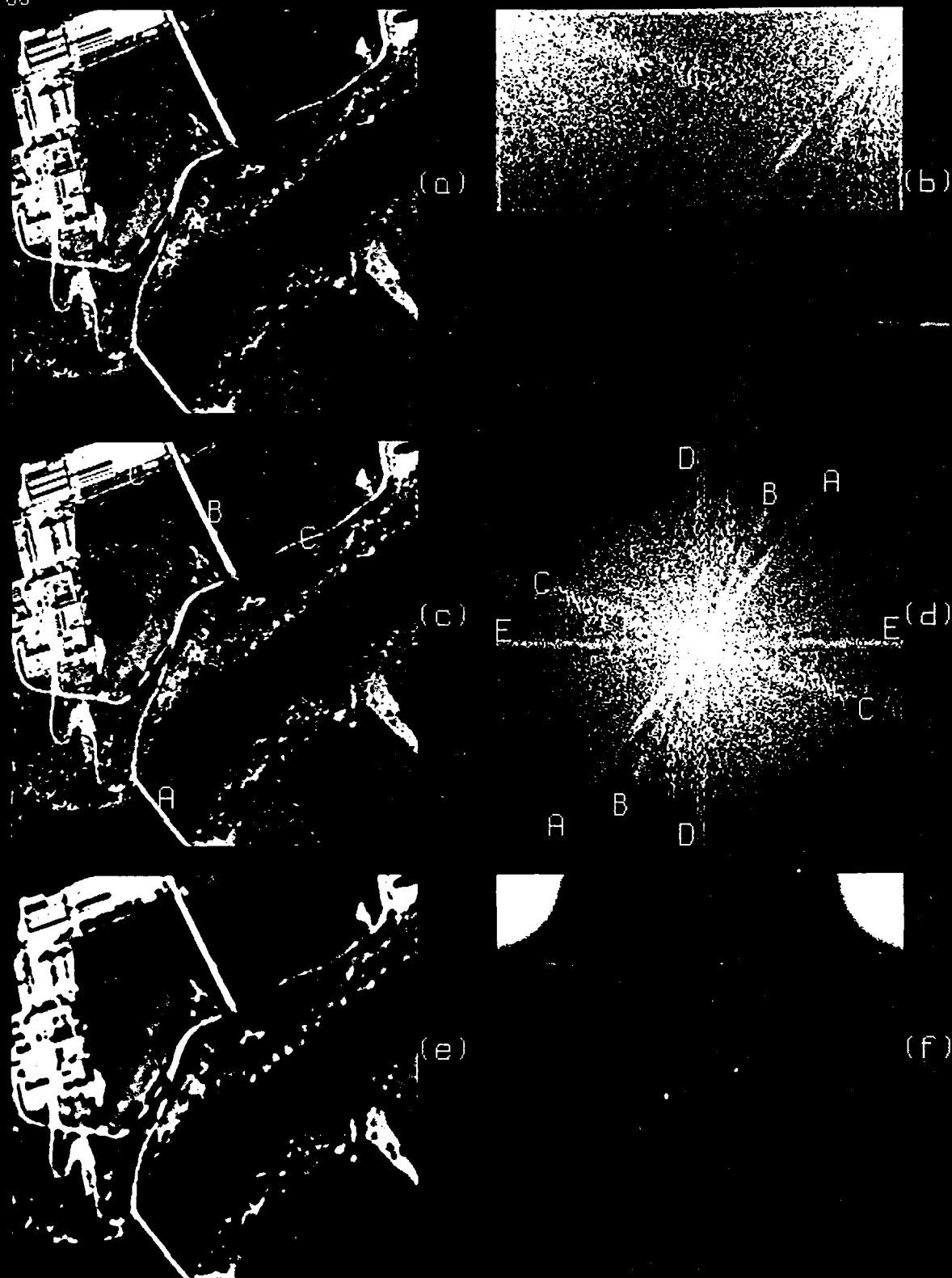


Fig 41 Identification of transform features and image smoothing

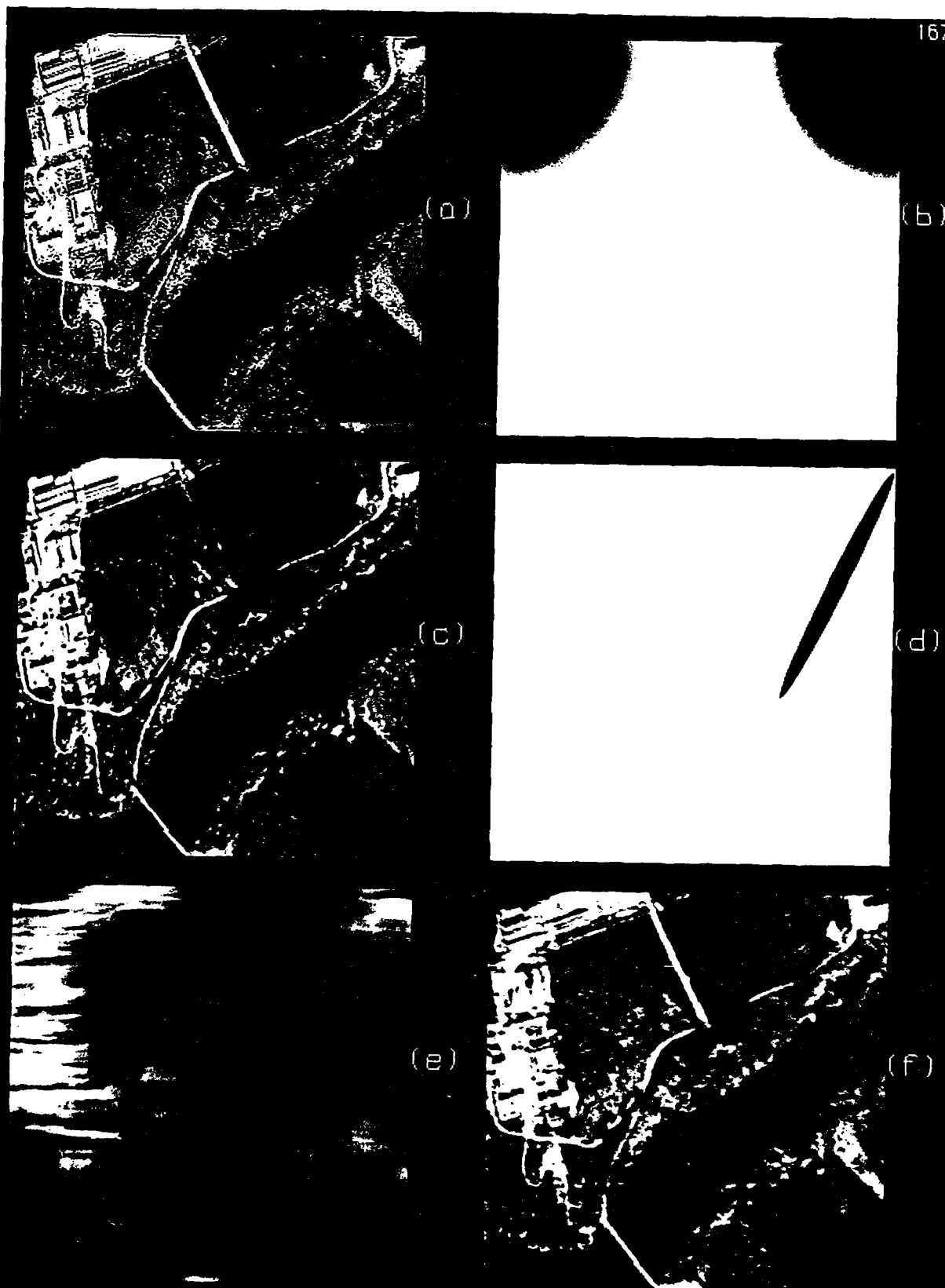


Fig 42 Edge enhancement, object removal, and inverse filtering

The last application outlined in this chapter is 'inverse filtering'. A severe smoothing filter consisting of 32 weights, of value one, in a horizontal line, was applied to the image of Fig 41(a) using the 'SMOOTH' command in the 'Maths Operations' chapter. The results are displayed in Fig 42(e). This process simulates horizontal motion blurring. Let the original image be f , with Fourier transform F , the filter (or disturbing function) be h (with FT, H) and the final image be g (with FT, G). Then,

$$g = f * h, \quad (f \text{ convolved with } h).$$

and using a further property of Fourier transforms,

$$G = F \cdot H \quad (F \text{ multiplied by } H).$$

If we wish to recover the original image, f , it could be found from the inverse transform of F , which is simply, G/H . All that is required is to determine H .

If an image, r (with FT, R) of a single point is smoothed with the same line filter, to produce an image s (with FT, S), then,

$$S = R \cdot H$$

Since S and R are known, H can be found and substituted in the equation for F ,

$$\begin{aligned} \text{ie } F &= G/H \\ &= (G \cdot R)/S \quad (\text{see page 175 for the actual equation}). \end{aligned}$$

There are, however, problems, since if the function S is ever zero, the equation breaks down. In this example, the disturbing function, h , is a rectangular window, the FT of which is a sinc function. Unless the window is only one pixel wide, this sinc function will have zero values in the transformed plane, within the box defined by u, v in the range $-N/2$ to $+N/2$. With 32 elements in the smoothing filter, H obviously has many zeroes and so will S . S can also become zero if there is noise in the system, and so for both these reasons, the equation for F is modified to,

$$F = (G \cdot R)/(S + a)$$

where a is a small constant fixed by trial and error. The image in Fig 42(e) was transformed to give G , the transform of an image of a spot gave R , and the transform of the smoothed spot gave S . The constant ' a ' was given a value of 10 to the power -9 , and the equation was evaluated using the commands 'MULTIPLY TRANSFORMS' and 'DIVIDE TRANSFORMS'. Reasonable reconstruction is evident in the output image in Fig 42(f), which shows the compromise between recovering the fine detail (by having a very small value of ' a '), and suppressing artefacts such as the ghosts of the very strong features. As the value of ' a ' is increased, the image looks more like Fig 42(e), and the artefacts are reduced.

Generally, the disturbing function, h , is not known. If there is linear motion blurring, then a reasonable guess can be made of the blurring function, and the process above undertaken to try to recover the image. It is however a trial and error operation, but by using as much other information as possible (eg the speed of the vehicle and the shutter speed of the camera, etc) and some inspiration, reasonable reconstructions are possible. Other disturbing functions which are amenable to this treatment are, blurring due to incorrect focus, and blurring due to the imaging system itself.

It is assumed in these reconstruction processes that the image is linear, ie the number representing the intensity is proportional to the incident light on the imaging system. For a photographic film, the 'gamma' of the film may not be unity, in which case the density on the film is not proportional to the intensity of the light which falls on it. Equally, the response of the digitiser may not be linear. All of the steps in the chain from the original scene to the digital image should be examined for non-linearities, and these should be removed using the commands in the 'Arithmetic' chapter. It may be possible to record the image of a grey step wedge using the same imaging system, in which case the non-linearities introduced by all parts of the system can be measured at once. Rather than using the Arithmetic package to remove the non-linearities, a LUT can be set up with the 'LINEAR STRETCH' command in 'Text and Graphics', with one entry in the LUT for each step in the step wedge. Thus each step in the recorded wedge would be replaced with the value in the original scene.

One final point to note is that in the reconstruction process, artefacts can propagate from the edges, ie the places of sharp discontinuities. Some effort spent in removing these edges is well worthwhile. In the example in Fig 42, the 256 square image was reflected about the right hand edge making a 512 pixel, 256 line image. The image was then moved 128 pixels to the right, with the rightmost 128 pixels copied to the left side. Therefore, the original image was moved to the middle of the screen, with continuous edges to the right and left. When the reconstruction was performed, the artefacts were not only decreased, but since they reduce with increasing distance from the edges of the image, and the extreme left and right 128 pixel sections were redundant, most of the artefacts were removed from the final 256 square image.

Completing this section on convolution, it has been stated that if

$$f = g * h \quad \text{then} \quad F = G . H$$

and it can also be shown that if

$$f = g . h \quad \text{then} \quad F = G * H$$

There is a similar pair of relationships for correlation, where, if 'o' represents correlation, and H' means the complex conjugate, then if

$$f = g \circ h, \quad F = G . H'$$

and if

$$f = g . h' \quad \text{then} \quad F = G \circ H$$

The command 'CORRELATE TRANSFORMS' multiplies one transform by the complex conjugate of the other, thereby giving the correlation of two images when transformed backwards.

FORWARD TRANSFORM

SUMMARY

Operation of this command produces the discrete Fourier transform using the FFT method outlined earlier in this chapter.

COMMAND IN DETAIL

The first secondary command concerns the type of transform. In most cases a real input image will be transformed into a complex one. For images such as those from a synthetic aperture radar which are in the complex form with the real part in one store, and the imaginary part in another, the complex image can be transformed into another complex image. The transform, complex to real, is unlikely to be used at this stage.

For the next secondary command, most real images will be in integer format, specified by 'Integer', although it is possible to have real images in floating point format, specified by 'Floating'. Complex images are more likely to be in floating point format. When the input format has been given, the next command specifies the output format. This could simply be integer, by choosing 'Linear', but the wide range in the data in the transformed image makes this generally unsuitable. It is normally better to store the output in 'Linear Floating' point format. If an image of the magnitude only is required, it can be produced directly using the 'Magnitude' option, but will result in an integer image having the same problem with the wide range of the data. If the logarithm of the magnitude is produced, by selecting 'Log Magnitude', it will have a much compressed range, which can be held in one of the display stores. The low level frequencies can be seen in the same image as a high level one. The precise algorithm used in this logarithmic compression is described under the command 'MAGNITUDE PHASE' (p176).

All that remains is to specify which store(s) contain the input image(s), and where the results are to be placed. For a real input image, there is only one store, and for a complex one there are two. A whole screen can be selected, or any of the four quadrants. It is worth remembering that a Fourier transform takes some time to calculate, and it may be advisable to use quadrants at least initially. There is a factor of about 6 in time between processing a quadrant and a full screen. It should be remembered that if a complex image is produced in floating point format, (as is generally the case), then the output must be held in the working stores. When working with quadrants, it will be found convenient to store the real output in a quadrant of one working store, and the imaginary output in the same quadrant in the next store. Other commands which use the output of this command, suggest that this is the arrangement for their input.

Most users who simply have an integer image in one of the top stores, will find that if they follow all the defaults, the correct result will be produced provided it is put into the working (ie non-display) stores.

FORWARD TRANSFORM	HALF ROTATE X	HALF ROTATE X AND Y	END	D1	T	T	D1
BACKWARD TRANSFORM	REFLECT TOP	INVERT REFLECT	BACKGROUND VALUE	D2	T	T	D2
ADJUST TRANSFORM	REFLECT TOP HALF ROTATE	INVERT REF HALF ROTATE	MEAN VALUE	D3	T	T	D3
DISPLAY WAVELENGTH	COMPLEX TO COMPLEX	REAL TO COMPLEX	BOX	D4	T	T	D4
MULTIPLY TRANSFORMS	COMPLEX TO REAL		ELLIPSE	5	T	T	5
CORRELATE TRANSFORMS			BOX REFLECTED	6	T	T	6
DIVIDE TRANSFORMS			ELLIPSE REFLECTED	7	T	T	7
MAGNITUDE PHASE			PHASE ONLY	8	T	T	8
STRETCH FOR PHASE	INTEGER	SPACE PIXEL	PHASE & MAG	9	T	T	9
DESIGN FILTER	FLOATING	REPL. IMAGE	NO STRETCH	10	T	T	10
MAGNIFY		REPL. PIXEL	TOP HALF	11	T	T	11
	LINEAR		ALL IMAGE	12	T	T	12
	LINEAR FLOATING			13	T	T	13
	LOG MAGNITUDE			14	T	T	14
	MAGNITUDE			15	T	T	15
				16	T	T	16
			WHOLE SCREEN	D5	T	T	D5
			TOP LEFT TOP RIGHT	D6	T	T	D6
			BOT LEFT BOT RIGHT	D7	T	T	D7
				D8	T	T	D8
			STATUS				
			HELP				
			RETURN				
			IMAGE				

INPUT FOR COMMAND OR QUIT

Fig 43 The Fourier Transform page

BACKWARD TRANSFORM

SUMMARY

This command is the inverse of the previous command.

COMMAND IN DETAIL

The same set of secondary commands are used here as in 'FORWARD TRANSFORM'. Briefly, these specify the type of transform, (generally 'Complex to Real', occasionally 'Complex to Complex' but never 'Real to Complex'), the format of the input image (generally 'Floating'), the format of the output image (generally 'Linear'), the source store(s) and the result store(s). The only additional secondary command is the option to divide the output intensity by a number, called 'Output Scaling'. In the examples on filtering described earlier, the transformed images were multiplied, one by 200 and the other by 50. If these images were transformed back directly, the intensities would be greater than 255. This command can be used to remove such multiplying factors.

If, as an exercise, an integer image were transformed forwards by the previous command, it could be transformed backwards using this command with the value '1' as the output scaling, to reproduce the original. Only the default options would be used.

ADJUST TRANSFORM

SUMMARY

For a real input image, the standard form of the output from the command 'MAGNITUDE PHASE', covers only half of the plane, with the origin at the top left. If the full plane with the origin at the centre is required, this command can be used to re-arrange the image.

COMMAND IN DETAIL

There are six secondary commands allowing a variety of possible re-arrangements, not only for images created in this 'Fourier Transform' chapter, but also for images in general. The function of the first option 'Half Rotate X' is best illustrated using the following example on one line of an image.

Original line	:	A	B	C	D	E	F	G	H
Half Rotated line	:	E	F	G	H	A	B	C	D

This command performs the rotation for half a revolution. Two applications of this operation would reproduce the original image. For the function 'Half Rotate X and Y', the rotation is performed in both the X and Y directions. This results in the top left corner of the original image being placed in the centre of the output image, which is exactly what is required to put the origin in the middle of the plane for magnitude and phase images which cover the whole plane, (ie created from complex input images). For a real input image, GEMSTONE does not calculate the bottom half of the complex transformed image.

The command 'Reflect Top' will reproduce the bottom half of the magnitude image correctly, and 'Invert Reflect' does the same reflection but also inverts the image intensity (ie subtracts it from 255) as required for the phase image, remembering that $F(u,v)$ is equal to the complex conjugate of $F(-u,-v)$. It should be noted that the reflection is not about the horizontal axis, but about the centre point. Thus the top left of the image is copied to the bottom right.

'Reflect Top Half Rotate' combines the operations of reflecting the top and rotating in both the X and Y axes. It will take the half plane of magnitude data, reproduce the bottom, and move the origin to the centre, as was illustrated in the conversion of Fig 41(b) into Fig 41(d). 'Invert Ref Half Rotate' does the same operation but inverts the intensity upon reflection, as required for the phase image.

All that is required for operation of this command is to choose one of the six manipulations, specify the input store and quadrant, and the output store and quadrant.

DISPLAY WAVELENGTH

SUMMARY

This command provides visual assistance in the interpretation of the magnitude image.

COMMAND IN DETAIL

Once the store and quadrant (or whole screen) containing the magnitude image have been specified, the cursor appears on the screen together with a pair of parallel lines. If the cursor is moved over a spot in the magnitude image, the parallel lines will indicate the direction of the sine wave in the original image, corresponding to the spot in the magnitude image. The parallel lines indicate adjacent crests for this sine wave and hence give the wavelength. Where the user wishes to identify a distracting line structure, such as banding in the image this command can be used to locate the frequencies associated with the structure, so that a filter can be designed specifically to remove it. Similarly, faint lines can be identified so that a specific edge enhancement filter can be designed.

MULTIPLY TRANSFORMS

SUMMARY

This command enables real and complex images to be multiplied together.

COMMAND IN DETAIL

Basically, one complex image can be multiplied by another. If the imaginary part of either image does not exist, then simply pressing the 'QUIT' button informs GEMSTONE that it is missing. In this way a real image can be multiplied by a real, a real by a complex, or a complex by a complex.

Up to four input stores are required for the real and imaginary parts of the first and second images. For each store, the store number must be given, the quadrant (or whole screen) and whether the contents of the store are in 'Integer' or 'Floating' point format. After the first store is chosen there is one extra option to specify 'Top Half' or 'All Image'. The former is used for the transformed versions of real input images where only half the output plane is calculated, and the latter for the transformed versions of complex images, or for general complex multiplications of images. Two output stores are required to hold the results, and as before, the store numbers, the quadrants, and the formats must be specified.

CORRELATE TRANSFORMS

SUMMARY

This command is exactly the same as 'MULTIPLY TRANSFORMS' except that one complex image is multiplied by the complex conjugate of the other. The main use of this operation is to permit the correlation of two images, as outlined in the introductory text.

DIVIDE TRANSFORMS

SUMMARY

Using this command, a real image can be divided by another real image, a real by a complex, a complex by a real, or a complex by a complex.

COMMAND IN DETAIL

If A and B are two complex images, and B' is the the complex conjugate of B, then the division equation is,

$$A \text{ divided by } B = (A \cdot B') / (B \cdot B' + 10^a)$$

The term $B \cdot B'$ appears because it is not possible to divide by a complex number directly, and the constant term at the end, ensures that the denominator is never equal to zero. The user is invited to specify the term 'a', and the number '-3' is suggested, (ie the constant is 0.001). It is most unlikely that this exponent would be positive, since generally it would dominate in the denominator. Having the constant as a power of 10 permits a wide range of numbers to be given easily. If more resolution than a power of 10 is required, fractional powers can be given as shown in the following table, where the top line is the power of 10 and the bottom line is the actual number:

Power (a)	: -3.0	-2.7	-2.5	-2.4	-2.3	-2.2	-2.1	-2.0
Number	: .001	.002	.003	.004	.005	.006	.008	.01

The exact numbers and powers can be found using a calculator.

The inputs required for this command are as for 'MULTIPLY TRANSFORMS' namely up to four input stores and two output stores, but with the addition of specifying the exponent 'a'.

MAGNITUDE PHASE

SUMMARY

The command 'FORWARD TRANSFORM', produces a complex image, and it is then often necessary to examine the magnitude image. This command produces the magnitude and phase images from a complex one.

COMMAND IN DETAIL

The store containing the real part of the complex transformed image is first given, together with the quadrant (or whole screen). As was stated earlier, if a real input image is transformed, only half of the output plane is calculated, whereas for a complex input image, the whole output plane must be determined. The next secondary command specifies whether 'Top Half' is to be used, for real input images, or 'All Image' for complex input images. Whether the contents of the store containing the real part, are in 'Integer' or 'Floating' point format is given next. Similarly, the store containing the imaginary part of the transformed image, is specified together with the format. There are two output images, one for the magnitude information, and one for the phase, and these should be put into the display stores. If the phase image is not required, (as is generally the case), it can be omitted by pressing the 'QUIT' button when asked for the store for the phase image.

Before considering the next secondary command, the range of the magnitude data needs to be examined. For an 8 bit integer input image, the maximum value of the DFT occurs at $F(0,0)$, when every pixel in the input image is at level 255. This can be verified by substituting the constant 255 for 'f' in the equation for the DFT. The minimum value (other than zero) is when there is only one occurrence of a single discrete frequency. From the equation of the DFT, it can be seen that this value is $1/N.N$, where N is 256 for a quadrant image or 512 for a full image, and so the minimum value of the DFT is $1/512.512$ ie $1/262,144$. Since the maximum is 255, the total possible dynamic range is $66,846,720 : 1$, far too much for an integer store, be it 8 or 16 bits. The range is greater for a 16 bit integer input, since the maximum positive number is 32,767, and finally for a floating point input the range is even greater. Although, by choosing 'Magnitude' a linear output of the magnitude can be produced, it is not generally advisable because of this range problem.

The other possible form of output is 'Log Magnitude', and the equation used for this is,

$$\text{Log mag} = \text{scale} \cdot \ln \left\{ (\text{mag})^2 \cdot 2^{36} + 1 \right\}^{0.5}$$

where mag stands for magnitude, and $\ln$ is the natural logarithm. The minimum value for the magnitude for an 8 bit integer image is $1/512.512$, which exactly cancels the term 2 to the power 36, giving

$$\text{Log mag} = \text{scale} \cdot \ln(1.414) = 4 \text{ (rounded)} \quad \text{if scale} = 12.$$

Equally, substituting the maximum value (255) gives $\text{Log mag} = 216$, using the same default scale factor of 12. This algorithm therefore copes well with the range, and even for the case of a 16 bit integer input image, the scale factor has only to be reduced to 11 to give a maximum Log mag of 252. Finally, the term '+1' in the equation ensures that if nothing of one frequency exists, the value of Log mag is zero. The option 'Log Magnitude' will therefore normally be chosen, in which case the scale factor needs to be fixed. The default value of 12 is generally suitable. It is possible to use the contrast stretch facilities in the 'Contrast Stretch, Histograms, Intensities' chapter to examine this magnitude image in more detail.

The phase information is stored such that a pixel with a phase angle of zero will have an intensity of zero. The intensity increases linearly with phase angle, until almost 360 degrees (actually $360.255/256$) when the intensity is 255. One other use of this command is to produce a special image for the 'STRETCH FOR PHASE' command below, for which, both the magnitude and phase must be held in the same store, when both are to be displayed. If the same store is specified for the output of the 'MAGNITUDE PHASE' command, the magnitude information will be stored in the top four bits and the phase in the bottom four.

STRETCH FOR PHASE

SUMMARY

This command is intended to aid the interpretation of either the phase image on its own or the magnitude and phase images together.

COMMAND IN DETAIL

The only secondary command is to choose a stretch for 'Phase Only', 'Phase and Mag' or 'No Stretch'. For the first choice, it is assumed that the phase is in one store and that the magnitude, which is in another, is not included in the operation. A stretch is set up in the LUTs so that if the phase angle is zero, the displayed colour is magenta. As the phase angle increases, there is a smooth change in colour through blue, cyan, green, yellow, red, and back to magenta when the phase is 360 degrees (ie level 255). It may be convenient to set up a grey wedge at the side, using 'GREY SCALE' in 'Text and Graphics' so that the range of colours can be seen and the phase angles can be identified.

For the second option of 'Phase and Mag', the images of phase and magnitude must be in the same store, as described in the command 'MAGNITUDE PHASE' above. The colours are in the same sequence as above, but the intensity of the colour is governed by the magnitude image. The last option 'No Stretch' simply resets the LUTs to the unity transfer function, ie the stretches are removed.

DESIGN FILTER

SUMMARY

This command produces a range of filters for use with transformed images.

COMMAND IN DETAIL

The simplest output from this operation is an image where most pixels are at one level, a region defined by a square or an ellipse is set to another, and there is a smooth transition between them. Examples of this are shown in Fig 41(f), Fig 42(b) and Fig 42(d). A cosine squared function is used to make the transition in intensity from one region to another across several pixels.

After choosing the store for the filter image, including whether it covers the whole screen or a quadrant, and whether only the top half is to be produced (for real input images), a range of secondary commands are presented. The use of these commands is perhaps best explained with an example. Consider the filter image in Fig 42(b), in which most of the image is at level 200. This was set using the secondary command 'Background Value', which simply takes the intensity level set by the user with the rolling ball, and writes this value over the whole filter image. 'Ellipse' was next selected so that a circle centred at the top left corner, (zero frequency point) was produced. Following the request 'Give Centre Value' which appeared on the screen, the rolling ball was used, to select the number 50 for the intensity of the pixels at the centre of the circle. Next, the 'Edge Value' was set to 200 with the rolling ball. This value is normally at the same level as the background, so that there are no sharp transitions in intensity in the filter image, which could lead to spurious high frequency components appearing in the final image at the end of the filtering process. Finally, a % radius for the centre value was given, and the default of 50 was used. This meant that when the circle of radius 80 pixels was later drawn, there was a central circle of radius 50% of 80 (ie 40 pixels) with all the pixels at level 50, surrounded by a circle of radius 80 pixels, with the peripheral pixels at level 200. In the region between the two circles, the intensity increased in a 'cosine squared' fashion from level 50 at the inside circle to level 200 at the outside.

Returning to the secondary commands, if 'Ellipse' is chosen, a circle appears on the screen and the rolling ball is used to move it in the x and y directions. After pressing the 'CHNG' button, the ball changes the size of the circle independently in the two directions. In this way, an ellipse can be produced. The text on the VDU, which was 'Shift' to denote that the rolling ball could shift the circle, changes to 'Scale' for this operation. The next press of 'CHNG' changes this text to 'Rotate', and the rolling ball can rotate the ellipse on the screen. A further pressing of 'CHNG' causes the control to revert to shifting the ellipse, etc. A circle or an ellipse can therefore be drawn at any position and orientation on the screen, and it will be noticed, that as the ellipse moves over the left edge, it re-appears at the right edge. For the example, the circle was reduced to a spot, moved to the top left corner (to centre it on the zero frequency), and subsequently expanded. The image of the magnitude was displayed under

the circle, so that the circle could be seen in relation to the frequencies in the image. The line of the circle represents the outside edge of the filter. In other words, the intensity falls off in the cosine squared fashion inside the line of the circle.

The command 'Box', is the same as ellipse except that a box rather than an ellipse is drawn. There is no option to rotate the box. The commands 'Box Reflected' and 'Ellipse Reflected' are the same as 'Box' and 'Ellipse' respectively, except that a second box or ellipse which is simply a reflection about a vertical axis in the centre, is produced at the same time. Where such a symmetry is required in the filter image, these commands save time.

The last command is 'Mean Value', which sets the value of the top left pixel. As was noted in the introductory text, for a forward transform, this particular pixel represents $F(0,0)$, or the mean value of the input image. The filter image is normally multiplied by some transformed image, and the inverse transform produced. Therefore, this top left pixel controls the mean value of the final image after the backward transform. If a particular filter image were zero everywhere except for one point, to extract only one frequency, the resulting inverse transform of the filtered image, would have positive and negative intensities. Setting the 'Mean Value' to 100 for example, would remove this problem. When the filter image has been defined, 'End' can be selected to write the detail of the filter into the background, as shown in Figs 41(f), 42(b), and 42(d).

The example in Fig 42(b) is a very simple one. The filter shown in Fig 42(d) is rather more tuned to the specific filtering required. It was drawn with the magnitude image, Fig 41(b) underneath. A filter to enhance a small range of frequencies in any direction (ie an omnidirectional bandpass) could be produced by drawing one circle inside another. More complicated filters can easily be produced with multiple applications of boxes and ellipses to a basic filter image where the background and the top left pixel have been set. One last example of the use of this command, not actually for filtering directly, is in 'grading' the edges of an input image to reduce the edge effects. If a box, 256 pixels square, were produced, with the central intensity extending to 90% of the width, a filter image with a bright centre fading gently to zero at the boundary would be produced. Multiplying the input image in Fig 41(a) by this, would result in the same image with edges fading to zero. Thus the image would be continuous over the boundary, thereby reducing the edge effect. Some of the image would however be lost.

Other commands within GEMSTONE can be used to produce filters. The facilities in 'Text and Graphics' can be used to create the basic shapes of filters. Density slicing the magnitude image of the transformed scene, will provide the basis of more accurate shaping of finely tuned filters. Following production of the basic filter, it should be smoothed before use, to avoid the generation of spurious high frequency effects.

MAGNIFY

SUMMARY

This command provides a utility magnifying operation for use in resampling images.

COMMAND IN DETAIL

There are three options for the magnification method. 'Space Pixel' results in the individual pixels of a small image being separated and zeroes being inserted in the gaps. 'Repl. Image' causes the small image to be replicated over the whole defined area, and 'Repl. Pixel' provides a nearest neighbour resampling to expand a small image to cover the defined area.

Having selected the magnification method, the input store and the area to be magnified are selected. The magnification is then fixed to cover either the whole screen or a quadrant. Finally, the output store is chosen and the operation performed.

Two possible uses of this utility are described below. By replicating a small image, a proper DFT of it can be produced. It has been assumed that the input image is cyclic with a period of 256 or 512 pixels. A small image in a 'sea of black' does not have the same cyclic property. Replication corrects this, provided that the dimensions of the small image are sub-multiples of 256 and therefore do not add to the discontinuities at the edges. Secondly, suppose an image were magnified by 'Space Pixel' and transformed. If this transformed image were multiplied by a sharp edged box, (the transform of the sinc function), and the inverse transform performed, the result would be an image of the same size as the original, but with the black spaces filled with pixels from a sinc resampling. Other sampling functions can be transformed to produce similar output images with different resampling functions.

THIS IS A DEMONSTRATION OF THE INFORMATION
SYSTEM IN GEMSTONE.

A SECTION OF TEXT IS PRINTED AND
A MENU IS DISPLAYED GIVING THE USER A
CHOICE OF FURTHER INFORMATION AS
FOR EXAMPLE IN PRESTEL.

THIS SYSTEM HAS TWO EXTRA
FEATURES IN THE MENU SELECTION
PROCEDURE. ONE IS TIMEOUT, THE OTHER IS
RANDOM DEFAULT SELECTION. THESE
MAY BE COMBINED AND WILL BE DEMONSTRATED AS PART
OF THIS PARTICULAR SYSTEM.

+++ NOW CHOOSE FROM THE FOLLOWING ITEMS
YOU HAVE 15 SECONDS +++

RAE GEMSTONE NEWS	GEMS GEMSTONE NEWS
USER INFO	END

Fig 44 The Information page

5.13 Information

This chapter provides a wide range of information to the user, in the form of text presented on the TV screen. 'Help' information of various commands, details of other GEMS installations, background processing functions, user experience in particular disciplines, and examples of image processing operations can all be provided in this package.

The way in which the system works is that the system administrator can create standard ASCII text files containing the information. By means of the simple menu system within this chapter, the user can access the files that have been created. GEMSTONE simply provides the framework for the user to access the information and does not contain any information itself. It is for the system administrator to add the information. Equally, it is for the users themselves to provide the system administrator with the material to be inserted, or to request that particular information be added.

The operation of the system is self evident in the interactive menu, and the user simply chooses the topic on which information is required. If no choice is made, the system slowly displays all the information in the files, in sequence.

Acknowledgements

I am very grateful to Alfred Odell of Space Department who has written all of the software for this project, as well as defining many of the algorithms. My thanks are also due to GEMS of Cambridge who, with the co-operation of RAE staff have designed and built all the hardware.

Bibliography.

The following are general references for digital image processing.

- 1 GONZALES, R.C. and WINTZ, P. DIGITAL IMAGE PROCESSING, 1977.
ADDISON-WESLEY, MASSACHUSETTS. 431pages.
- 2 BERNSTEIN, R. DIGITAL IMAGE PROCESSING FOR REMOTE SENSING,
1978. IEEE PRESS, NEW YORK. 473pages.
- 3 PRATT, W.K. DIGITAL IMAGE PROCESSING, 1978.
JOHN WILEY, NEW YORK. 750 pages.
- 4 ROSENFELD, A.(Editor) TOPICS IN APPLIED PHYSICS
VOL 11 - DIGITAL PICTURE ANALYSIS, 1976.
SPRINGER VERLAG, NEW YORK. 351pages.

The following are specifically for Fourier transforms.

- 5 JAVID, M. ERENTER, E. ANALYSIS, TRANSMISSION, AND FILTERING OF SIGNALS,
1963 MCGRAW-HILL, NEW YORK.
- 6 GOLD, B. RADER C. M. DIGITAL PROCESSING OF SIGNALS, 1969
MCGRAW-HILL, NEW YORK.

The next is specifically for the FFT.

- 7 COOLEY, J.W. TUKEY, J.W. An algorithm for the machine calculation of complex Fourier series. MATH COMPUT, 19, 297-301 (1965).

Index of Primary Commands

<u>Primary Command</u>	<u>GEMSTONE chapter</u>	<u>Page</u>
3-D FIXED	Text and Graphics	124
3-D GENERAL	Text and Graphics	124
ADD	Arithmetic	127
ADD WEDGE	Copy an Image	21
ADJUST TRANSFORM	Fourier Transforms	173
AREA FILL	Text and Graphics	117
AUTO EQUALISE	Contrast Stretch, Histograms, Intensities	28
AUTO GAUSSIAN	Contrast Stretch, Histograms, Intensities	29
AUTO LINEAR	Contrast Stretch, Histograms, Intensities	28
AUTO 2 LINEAR	Contrast Stretch, Histograms, Intensities	28
BACKWARD TRANSFORM	Fourier Transforms	173
BLACK IMAGE	Density Slice, Measure Areas	42
CHOOSE BANDS	Classifiers, Copy Bit Plane	105
CHOOSE CLASSIFIER	Classifiers, Copy Bit Plane	103
CHOOSE COLOUR	Density Slice, Measure Areas	43
CHOOSE TR AREA	Classifiers, Copy Bit Plane	105
CLASSIFY	Classifiers, Copy Bit Plane	103
CLEAR PLANE	Text and Graphics	111
COLOUR PALETTE	Text and Graphics	121
COLOUR ROTATE	Text and Graphics	123
COPY	Special	142
COPY AREA	Text and Graphics	114
COPY BIT PLANE	Classifiers, Copy Bit Plane	104
COPY IMAGE TO STORE	Copy an Image	19
COPY STORE TO STORE	Copy an Image	21
CORRELATE TRANSFORMS	Fourier Transforms	175
DEFINE EXTRACT	Copy an Image	20
DEFINE FILTER	Linear Filters	92
DEFINE PROGRAM	Arithmetic	128
DEFINE TR AREA	Classifiers, Copy Bit Plane	103
DESIGN FILTER	Fourier Transforms	178
DESTRIPE	Maths Operations	54
DISPLAY BIT PLANE	Classifiers, Copy Bit Plane	106
DISPLAY MAX LK	Classifiers, Copy Bit Plane	107
DISPLAY WAVELENGTH	Fourier Transforms	174
DIV	Arithmetic	128
DIVIDE TRANSFORMS	Fourier Transforms	175
DRAW GRID	Special	143
DRAW LINES	Text and Graphics	116
DRAW SPRITE	Text and Graphics	118

<u>Primary Command</u>	<u>GEMSTONE chapter</u>	<u>Page</u>
EDGE ENHANCE	Maths Operations	51
ERASE STORE	Copy an Image	21
EXECUTE	Arithmetic	133
FILTER IMAGE	Linear Filters	93
FORWARD TRANSFORM	Fourier Transforms	171
GREY SCALE	Text and Graphics	120
HISTOGRAM	Contrast Stretch, Histograms, Intensities	33
INT HUE SAT TO RGB	Special	152
LINEAR STRETCH	Text and Graphics	122
LINE FIX	Maths Operations	62
MAGNIFY	Fourier Transforms	180
MAGNIFY	Maths Operations	73
MAGNITUDE PHASE	Fourier Transforms	176
MANUAL	Contrast Stretch, Histograms, Intensities	29
MANUAL SLICE	Density Slice, Measure Areas	39
MEDIAN SMOOTH	Maths Operations	58
MULT	Arithmetic	128
MULTIPLY TRANSFORMS	Fourier Transforms	174
NEGATE STRETCH	Contrast Stretch, Histograms, Intensities	30
NORMAL IMAGE	Density Slice, Measure Areas	42
OUTPUT PLANE	Text and Graphics	111
OVERLAY 4 HISTOGRAM	Contrast Stretch, Histograms, Intensities	34
OVERLAY 4 PIXEL COUNT	Density Slice, Measure Areas	42
OVERLAY 4 PRINCIPAL COMPONENTS	Maths Operations	72
OVERLAY 4 SCATTER DIAGRAM	Maths Operations	72
PAINT	Text and Graphics	116
PIXEL COLOUR	Text and Graphics	112
PIXEL COUNT	Density Slice, Measure Areas	42
PIXEL VALUE	Contrast Stretch, Histograms, Intensities	32
PIXEL VALUE	Text and Graphics	111
PLANE FLICK	Special	141
PLANE SEQUENCE	Special	141
POLYGON FILL	Text and Graphics	117
PRINCIPAL COMPONENTS	Maths Operations	66
PRINT PROGRAM	Arithmetic	133

<u>Primary Command</u>	<u>GEMSTONE chapter</u>	<u>Page</u>
READ IMAGE	Linear Filters	94
REMOVE WEDGE	Copy an Image	21
RESTORE	Classifiers, Copy Bit Plane	109
RESTORE FILTER	Linear Filters	93
RESTORE PROGRAM	Arithmetic	134
RESTORE SLICE	Density Slice, Measure Areas	40
RESTORE STRETCH	Classifiers, Copy Bit Plane	109
RESTORE STRETCH	Contrast Stretch, Histograms, Intensities	31
RESTORE STRETCH	Text and Graphics	125
RGB TO INT HUE SAT	Special	144
ROTATE	Special	143
SAVE	Classifiers, Copy Bit Plane	108
SAVE FILTER	Linear Filters	93
SAVE PROGRAM	Arithmetic	134
SAVE SLICE	Density Slice, Measure Areas	39
SAVE STRETCH	Contrast Stretch, Histograms, Intensities	31
SAVE STRETCH	Text and Graphics	125
SCATTER DIAGRAM	Maths Operations	65
SEED FILL	Text and Graphics	117
SHIFT AND ROTATE	Linear Filters	93
SMOOTH	Maths Operations	47
STEREO PAIR	Special	152
STRETCH FOR PHASE	Fourier Transforms	177
SUB	Arithmetic	127
SYMMETRY	Linear Filters	92
TRANSECT GRAPH	Contrast Stretch, Histograms, Intensities	33
WRITE IMAGE	Linear Filters	94
WRITE TEXT	Text and Graphics	113

General Index

In this index, keywords which are likely to arise in connection with image processing operations, are given on the left hand side. Qualifiers, comments, and suggestions are given in the middle, and the relevant page numbers on the right. A chapter in this handbook and in GEMSTONE is written with quotes (eg 'Arithmetic') and a primary command is written in capitals as well (eg 'ADD'). To save space in this index, the names of the following chapters have been truncated as follows:

'Copy'	- 'Copy an Image'
'Contrast'	- 'Contrast Stretch, Histograms, Intensities'
'Density'	- 'Density Slice, Measure Areas'
'Maths'	- 'Maths Operations'
'Linear'	- 'Linear Filters'
'Class'	- 'Classifiers, Copy Bit Plane'
'Text'	- 'Text and Graphics'
'Arith'	- 'Arithmetic'
'Fourier'	- 'Fourier Transforms'

<u>Keyword</u>	<u>Pages</u>
2-Dimensional	
histogram : 'SCATTER DIAGRAM' in 'Maths'	65
3-Dimensional	
views of an image : '3-D FIXED' in 'Text'	124
example of '3-D FIXED'	87
or '3-D GENERAL' in 'Text'	124-125
Add	
two images together : 'ADD' in 'Arith'	127
or "plus" under 'DEFINE PROGRAM' in 'Arith'	131
a constant to an image : "plus" under 'DEFINE PROGRAM' in 'Arith'	131
or 'MANUAL' in 'Contrast'	29-30
or 'LINEAR STRETCH' in 'Text'	122-123
Examples of addition	134-139
Example of the operation of "plus"	129
a wedge to an image : 'ADD WEDGE' in 'Copy'	21
or 'GREY SCALE' in 'Text'	120
Aliasing	157-158
Alternate the display between two images : 'PLANE FLICK' in 'Special'	141
Analysis of spatial filter frequency responses	77-89
Angle of a line in an image	
: use 'TRANSECT GRAPH' in 'Contrast' to produce a parallel line	33
or 'DISPLAY WAVELENGTH' in 'Fourier' as an aid to filtering	174

- Animation
 : 'COLOUR ROTATE' and 'LINEAR STRETCH' in 'Text' 122-123
 or 'PLANE SEQUENCE' in 'Special' 141-142
- Annotation
 with text : 'WRITE TEXT' in 'Text' 113-114
 in general : see 'Text and Graphics' chapter 110-125
- Area
 of a slice : 'PIXEL COUNT' in 'Density' 42
 of a slice in a pre-defined area : 'OVERLAY 4 PIXEL COUNT' in 'Density' 42
 of a class : for box and discrete classifiers, area is given
 immediately after classification 106
 for maximum likelihood, 'DISPLAY MAX LIK' in 'Class' 107-108
 of a class in a previously classified image
 : density slice image and use 'PIXEL COUNT' in 'Density' 42
- Arithmetic
 Simple operations in 'Arith' : 'ADD', 'SUB', 'MULT', 'DIV' 127-128
 More general functions in 'Arith' :
 plus, minus, times, div, xpos, ypos, min, max, 131
 and, or, xor, >0, sqrt, abs, chng sign, random, sin, cos, 132
 abs2, arg, log, exp, int, frac 133
- Automatic
 contrast stretching, theory 22-25
 commands in 'Contrast' chapter 28-29
- Average : see Mean
- Bandpass filter
 by convolution, theory and examples 82-83,85
 by Fourier transforms, theory 164
 operation : 'DESIGN FILTER' in 'Fourier' 178-179
- Bandstop filter : see Bandpass filter
- Bilinear interpolation 73-74
- Binary number : see Digital number
- Bit 5-6
- Bit Plane
 description 6-7
 to store / display classifications 40-41,106-107
 copy : 'COPY BIT PLANE' in 'Class' 104
- Blow-up : see Magnify

Blurring	
Removal of blurring in an image	169-170
To blur an image : 'SMOOTH' in 'Maths'	47-49
or smooth with the commands in 'Linear'	76-94
Example of an image with motion blurring	169
Box classifier	
Theory	95
Example	100-101
Operation : using the commands in 'Class'	103-106
Brightness : see Intensity	
Button. Description of operation of buttons : see Control Panel chapter	10-13
Calculator. Use of 'Arithmetic' package as a calculator	129
Change	
store being displayed	4,12
colour of, image : by displaying stores in different colours	4,12
by the commands in 'Contrast'	22-34
by 'RGB TO INT HUE SAT' etc in 'Special'	144-152
overlay plane	13
class in overlay plane (ie overlay plane colour)	13
density slices : 'MANUAL SLICE' in 'Density'	39
'CHOOSE COLOUR' in 'Density'	43
graphics : 'PIXEL COLOUR' in 'Text'	112
'COLOUR PALETTE' in 'Text'	121-122
graphics, continuously : 'COLOUR ROTATE' in 'Text'	123
position of, cursor with the rolling ball or buttons	11,12
rectangles, objects, etc, with rolling ball only	12
one image relative to another : 'PLANE FLICK'	
in 'Special'	141
image on the screen by panning	11
image in a store by reselecting area : 'DEFINE EXTRACT'	
in 'Copy'	20
image in a store by copying from another : see Copy	
image by an integer or fractional number of pixels	
: see Shift	
sign of an image : see Negative	
pixel values, in a small area of an image : 'READ IMAGE' and 'WRITE	
IMAGE' in 'Linear'	94
'PAINT' etc in 'Text'	116
generally, by storing through the LUTs : see Copy	
Charts : see 'Text' chapter	110-125
CHNG button on the control panel	12
Choose	
image : see 'Choose an Image' chapter	16-17
colour : see Change	

Classification	
by density slicing	37
General theory of box, discrete and maximum likelihood classifiers	95-101
Comparison of classifier performances	100-101
Operation of classifier commands	103-108
(Summary of method under 'CLASSIFY' in 'Class')	105
Manual classifier under 'CLASSIFY' in 'Class'	105-106
Display of classifications : 'RESTORE SLICE' in 'Density'	40-41
'DISPLAY MAX LIK' in 'Class' (note storage method for redisplay later)	107-108
Holding a classification in a store	106
Area of a class : see Area	
Saving/restoring classification parameters : 'SAVE' and 'RESTORE' in 'Class'	108-109
Clusters	65
Colour	
Formation of a colour picture on the TV screen	4
Change colour : see Change triangle	145
of density slices	40-41, 43, 107
of classes in a maximum likelihood classification	107
of the colour palette	121
Manipulation of colour using 'RGB TO INT HUE SAT' in 'Special'	144-152
Commands. Primary and secondary commands	8-9
Compass gradient filter	90
Complex image	
storage format	7
Use in Fourier transforms	154-170
multiplication : 'MULTIPLY TRANSFORMS' in 'Fourier'	174-175
multiplication by complex conjugate : 'CORRELATE TRANSFORMS' in 'Fourier'	175
division : 'DIVIDE TRANSFORMS' in 'Fourier'	175
conversion, real/imaginary to amplitude/phase :	
'MAGNITUDE PHASE' in 'Fourier'	176-177
"abs2" and "arg" under 'DEFINE PROGRAM' in 'Arith'	133
amplitude/phase to real/imaginary :	
"sin" and "cos" under 'DEFINE PROGRAM' in 'Arith'	132
(see notes in "arg" under 'DEFINE PROGRAM')	133
Computer	4, 5
Contrast stretch	
Theory, examples and commands : in 'Contrast' chapter	22-30
: 'LINEAR STRETCH' in 'Text'	122-123
Different stretches applied to different parts of one image :	
'COPY AREA' in 'Text'	114-115
Saving/restoring stretches : see SAVE... and RESTORE... in the index of primary commands	136
Control panel	10-13

Convolution

Theory	44-45, 76-80
Operation using the commands in 'Maths' chapter	46-57
Operation using the commands in 'Linear' chapter	91-94
Theory using Fourier transforms	169-170

Co-ordinates : see Change position

of a pixel : 'PIXEL VALUE' in 'Contrast' 32
of boxes etc, are generally given at the bottom right of the TV
screen, and are sometimes repeated on the computer terminal
A co-ordinate grid can be overlaid using 'DRAW GRID' in 'Special' 143

Copy

an image from the host computer into GEMS	
: 'COPY IMAGE TO STORE' in 'Copy'	19
an extract from an image in the host computer, into GEMS	
: 'DEFINE EXTRACT' and 'COPY IMAGE TO STORE' in 'Copy'	19-20
an image from one store to another. There are several ways of doing this and the commands have optional facilities which are most useful within their respective chapters. Most of these commands are listed in the following table, where,	

- W - can copy the whole screen
- Q - can copy a quadrant specifically
- R - can copy a rectangular area (which could be a quadrant)
- I - can copy an irregular area
- M - can make multiple copies directly
- A - can magnify or reduce in the copy process
- C - can copy with a contrast stretch
- V - can invert left to right or top to bottom
- O - can rotate in fixed steps of 90 degrees
- P - can rotate generally
- S - can shift or reflect in the copy process
- L - can deal with 1024 pixel square images

'COPY STORE TO STORE'	in 'Copy'	W	Q		C		21
'COPY BIT PLANE'	in 'Class'	W			C		104
'COPY AREA'	in 'Text'	W	R	I	M	C	L 114-115
'COPY'	in 'Special'	W	Q			V	O L 142-143
'ROTATE'	in 'Special'	W	R			P	L 143
'MAGNIFY'	in 'Maths'	W	R		A	V	73-75
'MAGNIFY'	in 'Fourier'	W	R		A		180
'ADJUST TRANSFORM'	in 'Fourier'	W	Q			S	173-174
'SMOOTE'	in 'Maths'	W	R				47
'FILTER IMAGE'	in 'Linear'	W	R				93
'DEFINE PROGRAM'							
and 'EXECUTE'	in 'Arith'	W	Q		M	C	128-133

a part of an image to the same store, after moving the part image	
continuously : 'COPY AREA' in 'Text' using "Copy Sprite"	114-115
or "Move Sprite" (erases original)	114-115
from an overlay plane or a bit plane of one store to another	
bit plane : 'COPY BIT PLANE' in 'Class'	104
or 'COPY AREA' in 'Text' (limited)	114-115

Covariance	
matrix	66
matrix used in the maximum likelihood classifier	96
Correcting typing errors on a VDU	113
Correlation	67-68
of one image with another, by eye : 'PLANE FLICK' in 'Special'	141
matrix	67
by Fourier transform method	170
Preserving uncorrelated data for the human eye, using IES	144-151
using stereo	152-153
Count : see Area	
or 'HISTOGRAM' in 'Contrast'	33-34
or 'TRANSECT GRAPH' in 'Contrast'	33
Cursor	3,8,9,11,12
Freezing cursor movement in one direction	13
Defective, pixel or small group of pixels : repair using	
'PAINT' in 'Text'	116
or 'READ IMAGE' and 'WRITE IMAGE' in 'Linear'	94
Line : repair using 'LINE FIX' in 'Maths'	62
Delta function	76
Density slice : see 'Density' chapter	36-43
Destripe : 'DESTRIFE' in 'Maths'	54-57
or see 'Linear' chapter	85
or see 'Fourier' chapter	164
Diagram : see 'Text' chapter	
Differentiation of an image	
: 'EDGE ENHANCEMENT' in 'Maths' using "remove low"	51-53
or commands in 'Linear' chapter	92-94
or commands in 'Fourier' chapter	171-179
Digital image : see Image	
Digital number	
Binary number	5-6
of a pixel : 'PIXEL VALUE' in 'Contrast'	32
of pixels in a line : 'TRANSECT GRAPH' in 'Contrast'	33
of a small area of pixels : 'READ IMAGE' in 'Linear'	94
Mean, median and standard deviation of a small group of pixels	
: 'HISTOGRAM' in 'Contrast' (rectangular areas)	33-34
: 'OVERLAY 4 HISTOGRAM' in 'Contrast' (non-rectangular areas)	34
under 'CLASSIFY' in 'Class'	105-106

	193
Digital terrain model (DTM)	153
Directional filtering	
by convolution	82-83, 85, 89-90
by Fourier transform method	163-167
Discrete classifier	
Theory	95-96
Example	100-101
Operation	103-106
Discrete Fourier transform (DFT)	155-160
Displacement	
Measurement of displacement of, a point : 'PIKEL VALUE' in 'Contrast'	32
an image : 'PLANE FLICK' in 'Special'	141
Display	
a store on the TV screen	4, 12
an overlay plane on the TV screen	5, 11
a bit plane : 'DISPLAY BIT PLANE' in 'Class'	106-107
or 'RESTORE SLICE' in 'Density'	40-41
a classification : 'DISPLAY MAX LIK' in 'Class'	107-108
or 'RESTORE SLICE' in 'Density'	40-41
two images alternately : 'PLANE FLICK' in 'Special'	141
a sequence of images : 'PLANE SEQUENCE' in 'Special'	141-142
uncorrelated images : 'RGB TO INT HUE SAT' in 'Special'	151
or 'STEREO PAIR' in 'Special'	152-153
images of different resolutions : 'RGB TO INT HUE SAT' in 'Special'	151
Distance	
between two points : 'TRANSECT GRAPH' in 'Contrast'	33
or 'PIKEL VALUE' in 'Contrast' for co-ordinates	32
Divide	
one image by another : 'DIV' in 'Arith'	128
"div" under 'DEFINE PROGRAM' in 'Arith'	131
one image by a constant : "div" under 'DEFINE PROGRAM' in 'Arith'	131
one complex image by another : 'DIVIDE TRANSFORMS' in 'Fourier'	175
Draw	
straight line segments : 'DRAW LINES' in 'Text'	116
or 'DRAW SPRITE' in 'Text'	118-120
continucus line : 'PAINT' in 'Text'	116
area : 'PAINT', 'AREA FILL', 'POLYGON FILL', 'SEED FILL' in 'Text'	116-117
sprite (or moveable object) : 'DRAW SPRITE' in 'Text'	118-120
several sprites with linear interpolation : 'DRAW SPRITE' in 'Text'	118-120
a grid overlay : 'DRAW GRID' in 'Special'	143
Duplicate : see Copy	

Edge

detection : 'EDGE ENHANCEMENT' in 'Maths' using "remove low"	51-53
see 'Linear' chapter	76-94
see 'Fourier' chapter	154-180
enhancement : 'EDGE ENHANCEMENT' in 'Maths' using "increase high"	51-52
see 'Linear' chapter	76-94
see 'Fourier' chapter	154-180
Example of edge detection : filter H	78
edge enhancement : filter I	78

Eigenvalue, eigenvector, eigenvector transformation	66-67
---	-------

Enhancement : an imprecise term with many meanings.

See Contrast stretch, Density slice, Colour, Edge enhancement, Smooth, Noise, Destripe, Filter, etc

Enlarge : see Magnify

Equidensity contrast stretch : 'AUTO EQUALISE' in 'Contrast'	28-29
--	-------

Erase

a whole image (W) or a quadrant (Q) of an image	
: 'ERASE STORE' in 'Copy' (W) and (Q)	21
or 'CLEAR PLANE' in 'Text' (W)	111
or 'COPY BIT PLANE' in 'Class' using "zeroes" (W)	104
or 'DEFINE PROGRAM' in 'Arith' (W) and (Q)	128-132
or 'FILTER IMAGE' in 'Linear' with a null filter (W) and (Q)	93
a regular or irregular part of an image : 'PAINT' in 'Text'	116
a single bit plane	
: 'COPY BIT PLANE' in 'Class' using "zeroes"	104
or 'CLEAR PLANE' in 'Text' (overlay plane 4 only)	111
text in an image : as for erasing an image	
text on the VDU	113

Eye

Grey scale resolution	22
Colour sensitivity	144

Extract

from an image : 'DEFINE EXTRACT' in 'Copy'	20
--	----

False colour composite	4
------------------------	---

Fill in an area

: 'PAINT', 'AREA FILL', 'POLYGON FILL', 'SEED FILL' in 'Text'	116-117
---	---------

Graphics : see 'Text' chapter	110-125
Animated graphics : see Animation	
Grey scale : 'ADD WEDGE' in 'Copy'. Note that the image can replace the	21
wedge by using 'REMOVE WEDGE' in 'Copy'	21
or 'GREY SCALE' in 'Text' (image displaced by wedge is lost)	120
Grid : 'DRAW GRID' in 'Special'	143
Help	9
High boost filter	78-79
High pass filter	78-79
Histogram	
Description	22
for a rectangular area in an image : 'HISTOGRAM' in 'Contrast'	33
Examples	25
for a non-rectangular area : 'OVERLAY 4 HISTOGRAM' in 'Contrast'	34
2-D histogram : 'SCATTER DIAGRAM' in 'Maths'	65
Host computer	4-5
Hotelling transformation	66
Hue : see Colour	
: 'RGB TO INT HUE SAT' in 'Special'	144-151
I1, ie 8 bit integer format	5-6, 17
I2, ie 16 bit integer format	7, 17
Image	
Structure of a digital image	4
Sampling used to form a digital image	76
The GEMSTONE command IMAGE, (at the bottom of each chapter page)	9
Image restoration by inverse filtering	169-170
Image store description	4-7
Imaginary image : see Complex image	
Input button	10, 12

Intensity	
of a pixel : 'PIXEL VALUE' in 'Contrast'	32
of a small area of pixels : 'READ IMAGE' in 'Linear'	94
along a line of pixels : 'TRANSECT GRAPH' in 'Contrast'	33
of an area, ie mean, median and standard deviation	
: 'HISTOGRAM' in 'Contrast'	33-34
'OVERLAY 4 HISTOGRAM' in 'Contrast' for irregular areas	34
of a pixel set to be same as its position along the x-axis	
: 'DEFINE PROGRAM' in 'Arith' using "xpos"	131
: similarly for the y-axis, use "ypos"	131
hue and saturation : 'RGB TO INT HUE SAT' in 'Special'	144-151
Interpolation	
Nearest neighbour	73,75,143
Bilinear	73-75
Sinx/x	94,180
Inverse filtering using Fourier transforms	169-170
Invert	
an image : see Copy	
a pixel (ie intensity) : see Negative	
Karhunen-Loeve transformation	66
Laplacian filter	86-87
Least significant bit (LSB)	6
Likelihood	96
Limit the range of intensity within an image	
: 'DEFINE PROGRAM' in 'Arith' using "min" and "max" or "and"	131
Line drawing : see Draw	
Line defects : see Defective	
Linear	
stretch	22-30,122-123
filter	76-94
interpolation	73-75
Location : see Co-ordinates	
Logical functions : under 'DEFINE PROGRAM' in 'Arith'	132

Look-up table (LUT)

description	4,22
use in, contrast stretch commands	22-31
density slice commands	37-43
'COLOUR PALETTE', 'LINEAR STRETCH', in 'Text'	121-123
'PIXEL COLOUR' in 'Text'	112
'DISPLAY MAX LIK' in 'Class'	107-108
'STRETCH FOR PHASE' in 'Fourier'	177
Rotation of LUT : 'COLOUR ROTATE' in 'Text'	123
see Copy for copying an image through a LUT	
Saving the contents of a LUT : 'SAVE SLICE' in 'Density'	39
'SAVE STRETCH' in 'Contrast'	31
'SAVE STRETCH' in 'Text'	125
Restoring a previously saved LUT : 'RESTORE SLICE' in 'Density'	40
'RESTORE STRETCH' in 'Contrast'	31
'RESTORE STRETCH' in 'Class'	109
'RESTORE STRETCH' in 'Text'	125
Comparison of data before a LUT and after : 'HISTOGRAM' in 'Contrast'	33-34

Low pass filter 78-79

LSB (Least significant bit) 6

LUT : see Look-up table

Magnify

by sampling an image in the host computer at smaller intervals, down to every pixel, using 'DEFINE EXTRACT' in 'Copy'	20
by zooming (ie pixel replication on the screen)	11
by zooming with different factors in the x and y directions	13
by resampling : 'MAGNIFY' in 'Maths'	73-75
by Fourier transform method : 'MAGNIFY' in 'Fourier'	180

Manual

contrast stretch : 'MANUAL' in 'Contrast'	29-30
classifier	103, 105-106

Mask

: 'PAINT', 'DRAW LINES', 'POLYGON FILL', 'AREA FILL', 'SEED FILL', 'DRAW SPRITE', 'COPY AREA' in 'Text' can be used to draw masks	114-120
: 'DESIGN FILTER' in 'Fourier' can draw masks having edges with smooth changes in intensity	178-179
Use of masks to limit areas for processing	
: 'OVERLAY 4 HISTOGRAM' in 'Contrast'	34
'OVERLAY 4 PIXEL COUNT' in 'Density'	42
'OVERLAY 4 PRINCIPAL COMPONENTS' in 'Maths'	72
'OVERLAY 4 SCATTER DIAGRAM' in 'Maths'	72
multiply commands in 'Arith'	127-139
out bits in a binary number : "and" under 'DEFINE PROGRAM' in 'Arith'	132

Matrix	
Correlation matrix	67
Covariance matrix	66
Maximum likelihood classifier	
Theory	96-101
Example	100-101
Operation	103-108
Mean	
intensity of an area : 'HISTOGRAM' in 'Contrast' (rectangular area)	33-34
Examples	25
'OVERLAY 4 HISTOGRAM' in 'Contrast'	
(non-rectangular areas)	34
under 'CLASSIFY' in 'Class'	105-106
intensity in Fourier transform space	179
used in the maximum likelihood classifier	96-98
used in principal components analysis	66,71
Measure	
intensity : see Intensity	
area : see Area	
position: see Co-ordinates	
distance : see Distance	
Median	
Definition	24
intensity of an area : 'HISTOGRAM' in 'Contrast' (rectangular area)	33-34
Examples	25
'OVERLAY 4 HISTOGRAM' in 'Contrast'	
(non-rectangular area)	34
Manipulation of the median intensity	
: 'AUTO 2 LINEAR' in 'Contrast'	28
'MANUAL' in 'Contrast' using "2 part linear"	29-30
smoothing of an image : 'MEDIAN SMOOTH' in 'Maths'	58-61
Mem off button	12
Merge	
Images from different sources/wavelengths	
: 'RGB TO INT HUE SAT' in 'Special'	144-151
'STEREO PAIR' in 'Special'	152-153
an image into the background by grading the edges	179
Mix : see Merge	
see Colour	
Mode	11-13
Most significant bit	6
Motion blurring : see Blurring	

- Move
- cursor, by the rolling ball 8,12
 - by buttons 11
 - an image : see Copy, Shift and Sprite
 - one image continuously relative to another : 'PLANE FLICK' in 'Special' 141
 - a sprite : 'COPY AREA' in 'Text' 114-115
 - 'DRAW SPRITE' in 'Text' 118-120
- Movie sequence : see Animation
- MSB : see Most significant bit
- Multiply
- one image by another : 'MULT' in 'Arith' 128
 - "times" under 'DEFINE PROGRAM' in 'Arith' 128-133
 - one image by a constant
 - : "times" under 'DEFINE PROGRAM' in 'Arith' 128-133
 - by contrast stretch - example 22
 - one complex image by another
 - : 'MULTIPLY TRANSFORMS' in 'Fourier' 174-175
 - one complex image by the complex conjugate of another
 - : 'CORRELATE TRANSFORMS' in 'Fourier' 175
- Nearest neighbour interpolation 73,75,143
- Negative of an image
- : 'NEGATE STRETCH' in 'Contrast' 30
 - in a store : Copy through a LUT : see Copy
 - Example 'COPY AREA' in 'Text' 114-115
 - using "xor" command under 'DEFINE PROGRAM' in 'Arith' 132
- Noise removal from an image
- by linear filtering 48-49
 - by a median filter 58-61
 - by Fourier transform method 164
 - See also Destripe and Defective
- Non-linear
- contrast stretch
 - description 23-24
 - : 'AUTO GAUSSIAN', 'AUTO EQUALIZE', 'MANUAL' in 'Contrast' 28-30
 - using a piecewise linear function : 'LINEAR STRETCH' in 'Text' 122-123
 - filtering
 - : 'MEDIAN SMOOTH' in 'Maths' 58-61
 - See example 4 using arithmetic functions 138-139
- Number : see Digital number
- Nyquist sampling frequency 76-77

	201
Object : see 'DRAW SPRITE' in 'Text'	118-120
Output store or output plane. The store which will collect the output image as a result of some operation. The input for the operation is held in the input or source store.	
Overlay plane	5,7,11
Changing the colour of an overlay	13
Ovly button	11
Palette : 'COLOUR PALETTE' in 'Text'	121-122
Panning over an image	11
Parallelepiped classifier : see Box classifier	
Patch	
used for linear filtering	44-45
To patch an image : see Defective	
Pi, the number 3.14159....	130
Picture : see Image	
Piecewise linear	
contrast stretch	
: 'AUTO 2 LINEAR' in 'Contrast' (two pieces)	28
: 'MANUAL' in 'Contrast' (two pieces)	29-30
: 'LINEAR STRETCH' in 'Text' (many pieces)	122-123
Line drawing	
: 'DRAW LINES' in 'Text'	116
: 'DRAW SPRITE' in 'Text'	118-120
Pixel	
definition	4
storage	5-7
intensity or value : see Intensity	
count : see Area	
position or co-ordinates : see Co-ordinates	
colour : see Colour	
To change pixel colour, intensity : see Change colour	
Plane	
: see Bit plane	
: see Overlay plane	
Plus : see Add	
Polar co-ordinates	
: "abs2" and "arg" under 'DEFINE PROGRAM' in 'Arith'	133

Position : see Co-ordinates, Change position of one image relative to another : 'PLANE FLICK' in 'Special'	141
Power. To raise an image or number to a power : "exp" under 'DEFINE PROGRAM' in 'Arith'	133
Primary command	8-9
Principal components	
Theory	66-72
using statistics from a rectangular area 'PRINCIPAL COMPONENTS' in 'Maths'	66-72
using statistics from an irregular area: 'OVERLAY 4 PRINCIPAL COMPONENTS' in 'Maths'	72
display : 'RGB TO INT HUE SAT' in 'Special'	151
Print	
a pixel : see Digital number	
details about an area : see Area	
position or co-ordinates : see Co-ordinates	
eigenvalues or eigenvectors : see Principal components	
text : 'WRITE TEXT' in 'Text'	113-114
Probability density function	96
Quit button	12
Random number generator	132
Reconstruction of an image	169-170
Reduce	
size of an image : 'MAGNIFY' in 'Maths'	73-75
intensity of an image : see Contrast stretch	
area for processing : see all commands of the form 'OVERLAY 4' in the command index	185
: see 'COPY IMAGE TO STORE' in 'Copy'	19
noise in an image : see Noise	
Reject level : 'DISPLAY MAX LIK' in 'Class'	107-108
Rejection number : 'LINE FIX' in 'Maths'	62
Region growing/shrinking	90

	203
Remove	
cursor	12
wedge : 'REMOVE WEDGE' in 'Copy'	21
bad pixel/group of pixels/line : see Defective	
stretch : see 'RESTORE STRETCH' in command index and use the "none" secondary command	186
slice : see 'RESTORE SLICE' or 'RESTORE STRETCH' in command index and use the "none" secondary command	186
overlay from screen	11
a whole or part of an image : see Erase	
text : see Erase	
noise from an image : see Noise	
stripes from an image : see Destripe	
Replace a bad pixel/group of pixels/line : see Defective	
Resample : see Interpolation	
Resolution. Display of images with different resolutions : 'RGB TO INT HUE SAT' in 'Special'	144-151
Return	9
RGB ie red green blue. Discussion	144-151
Rotate	
LUT : 'COLOUR ROTATE' in 'Text'	123
image in steps of 90 degrees : 'COPY' in 'Special'	142-143
image generally : 'ROTATE' in 'Special'	143
Rub-out : see Erase	
Sampling : see Interpolation	
Integer sampling of an image in the host computer : 'COPY IMAGE TO STORE' and 'DEFINE EXTRACT' in 'Copy'	19-20
Saturation : 'RGB TO INT HUE SAT' in 'Special'	144-151
Scatter diagram or plot	
: 'SCATTER DIAGRAM' in 'Maths' (rectangular area)	65
: 'OVERLAY 4 SCATTER DIAGRAM' in 'Maths' (irregular areas)	72
Examples	68
Scene : see Image	
Secondary command	8-9
Select : see Choose	

Sequence

- Display a sequence of images : 'PLANE SEQUENCE' in 'Special' 141-142
- Display a sequence of two images : 'PLANE FLICK' in 'Special' 141
- Display a sequence of moving images : see Animation

Sharpening of images : see Edge enhancement

Shift

- Measure the shift required to make one image overlay another
 - : 'PLANE FLICK' in 'Special' 141
- an image by an integer number of pixels
 - : 'DEFINE EXTRACT' and 'COPY IMAGE TO STORE' in 'Copy' 19-20
 - : see Copy
- an image by a fractional number of pixels
 - : 'DEFINE FILTER' in 'Linear' to create a filter with a single non-zero weight 92
 - 'SHIFT AND ROTATE' in 'Linear' to move this single weight by a fraction of a pixel 93-94
 - 'FILTER IMAGE' in 'Linear' to move the image 93
 - : see Move and Sprite

Sinc is $\sin x/x$

- Interpolation 94, 180
- image 137-138, 153

Slice : see 'Density' chapter 36-43

Smooth

- : 'SMOOTH' in 'Maths' 47-49
- : 'MEDIAN SMOOTH' in 'Maths' 58-61
- : see 'Linear' chapter 76-94
- : see 'Fourier' chapter 154-180

Source store or input store. The store containing the image to be processed.
The results from the process are collected in the output store.

Spatial

- filter : see Filter
- frequency 47, 51, 76-89

Sprite

- : 'DRAW SPRITE' in 'Text' 118-120
- To move or copy, like a sprite, a small piece of an image
 - : 'COPY AREA' in 'Text' 115

Square root of a number or an image 132

Standard deviation

- of intensity : 'HISTOGRAM' in 'Contrast' (rectangular area) 33-34
 - Examples 25
 - : 'OVERLAY 4 HISTOGRAM' in 'Contrast' (irregular area) 34
- used in the maximum likelihood classifier 96-98
- used in principal components analysis 66, 71

Status

9, 19

Step wedge : see Grey scale	
Stereo. Production of stereo images : 'STEREO PAIR' in 'Special'	152-153
Stop	9
Store : see Image store	
To store an image : see Copy	
Stretch : see Contrast stretch	
Striping : see Destripe	
Subtract	
one image from another : 'SUB' in 'Arith'	127
or "minus" under 'DEFINE PROGRAM' in 'Arith'	131
a constant from an image : "minus" under 'DEFINE PROGRAM' in 'Arith'	131
or 'MANUAL' in 'Contrast'	29-30
or 'LINEAR STRETCH' in 'Text'	122-123
Examples of subtraction	134-139
Television	4
Template : see Mask	
Text : 'WRITE TEXT' in 'Text'	113-114
Thermal images mixed with others	
: 'RGB TO INT HUE SAT' in 'Special'	144-151
or 'STEREO PAIR' in 'Special'	152-153
Times : see Multiply	
Training area for use by the classifiers	
Drawing/defining training area : 'DEFINE TR AREA' in 'Class'	103-104
Copying training area to a bit plane : 'COPY BIT PLANE' in 'Class'	104
Choosing which training areas (already defined) are to be used in the classification process : 'CHOOSE TR AREA' in 'Class'	105
Transect : 'TRANSECT GRAPH' in 'Contrast'	33
Transfer : see Copy	
Transfer function	23
Transformation	
matrix	67
Karhunen-Loeve, Hotelling, eigenvector, or principal component transformation	66-72
Fourier transformation : see 'Fourier' chapter	154-158
'RGB TO INT HUE SAT' and 'INT HUE SAT TO RGB' transformations in 'Special'	144-152

Trigonometric functions : under 'DEFINE PROGRAM' in 'Arith'	132
Truncate intensity : see Limit	
TV	4
Typing errors. Correction of them	113
Uncorrelated images : see Correlation	
Unity transfer function	23
Value : see Intensity	
VDU	4
Virtual store	7
Wedge : see Grey scale	
Weight used in filtering	44,76
Working store	7,17
Write	
text in an image or overlay plane : 'WRITE TEXT' in 'Text'	113-114
graphics in an image or overlay plane : see 'Text' chapter	110-125
image data into an image : 'WRITE IMAGE' in 'Linear'	94
Zoom : see Magnify	

REPORT DOCUMENTATION PAGE

Overall security classification of this page

UNCLASSIFIED

As far as possible this page should contain only unclassified information. If it is necessary to enter classified information, the box above must be marked to indicate the classification, e.g. Restricted, Confidential or Secret.

1. DRIC Reference (to be added by DRIC)	2. Originator's Reference RAE TR 86062	3. Agency Reference	4. Report Security Classification/Marking UNCLASSIFIED
5. DRIC Code for Originator 7673000W	6. Originator (Corporate Author) Name and Location Royal Aircraft Establishment, Farnborough, Hants, UK		
5a. Sponsoring Agency's Code	6a. Sponsoring Agency (Contract Authority) Name and Location		
7. Title Handbook of the GEMS image processing system			
7a. (For Translations) Title in Foreign Language			
7b. (For Conference Papers) Title, Place and Date of Conference			
8. Author 1. Surname, Initials Bagot, K.H.	9a. Author 2	9b. Authors 3, 4	10. Date September 1986
11. Contract Number	12. Period	13. Project	14. Other Reference Nos. Space 664
15. Distribution statement (a) Controlled by - (b) Special limitations (if any) - If it is intended that a copy of this document shall be released overseas refer to RAE Leaflet No.3 to Supplement 6 of MOD Manual 4.			
16. Descriptors (Keywords) (Descriptors marked * are selected from TEST) Image processing. Remote sensing*.			
17. Abstract This Report describes the operation of the GEMS interactive image processing system and gives details on the background to the functions as well as details of how the operations are controlled by the user. The Report relates to the 'Diamond' version of the software.			