Colour displays and lookup tables : real time modifications of digital images

Lutz, René William; Pun, Thierry; Pellegrini, Christian

Computerized Medical Imaging and Graphics, Vol. 15, No. 2, pp. 73–84, 1991
Printed in the USA.

0895-6111/91 $3.00 + .00

COLOUR DISPLAYS AND LOOK-UP TABLES: REAL TIME MODIFICATION OF DIGITAL IMAGES

René W. Lutz, Thierry Pun* and Christian Pellegrini
Centre Universitaire Informatique, 12, rue du Lac, CH-1207 Geneve, Switzerland

(*Received* 4 *January* 1990, *Revised* 30 *August* 1990)

Abstract—Image processing in biomedical research has become customary, along with use of colour displays to run image processing packages. The performance of softwares is highly dependent on the device they run on: architecture of colour display, depth of frame buffer, existence of look-up table, etc. Knowledge of such basic features is therefore becoming very important, especially because results can differ from device to device. This introductory paper discusses hardware features and software applications. A general architecture of colour displays is exposed, comparing the features of the most commonly used devices. Basic organisation of memory, electron gun and screen are analysed for each type of display, concluding with a more detailed study of raster scan devices. Frame buffer and look-up table organisation are then analysed in relation with overhead expenses such as time and memory. Relation between image data and displayed images is discussed. By means of examples, the manipulation of colour tables is examined in detail, showing how to improve display of images without altering image data. Finally, the basic operations performed by the look-up table editor developed at University of Geneva are presented.

Key Words: Look-up table, Colour map, Frame buffer, Grey scale, Pixel Histogram, Pseudo colour

INTRODUCTION

Up to a few years ago, colour displays were hardly used. Their price was prohibitive, the colour definition unsatisfactory for an every day purpose and therefore only specific research fields requiring colour would use them. Presently, not only has quality improved but prices have reduced considerably, making colour displays very attractive. Their utilization has become common; many health care facilities routinely employ such devices. In relation with this sudden popularity of colour monitors, application of all kinds have appeared, providing medical research with a large array of useful tools. In this context, it seems important to understand more about colour displays and how colour or grey scale images are managed. The purpose of this paper is twofold. First it exposes precise knowledge about the basics of colour displaying, second it shows how to display images according to the user's own interests and technical environment.†

Look-up tables are machine dependent, but most machines follow the same general organisation. We shall therefore analyse the architecture of a typical colour device in the first section. We will also define frame buffer and look-up table, and show their respective position in the machine's internal organisation (1, 2). Different kinds of software programs and libraries offer means to display images. The way this is performed will be discussed, analysing relations between pixels, memory, and displayed colour.

In the second section several applications are presented. They show how little changes in a colour map can produce sharp, contrasted, application oriented images. Different classes of look-up tables are discussed; dynamic range adjustment, thresholding, pseudo colours, etc. (3–5).

Examples of Section 2 have been produced by a colour map editor developed at the Computer Science Centre (C.U.I) of the University of Geneva. The main features of this editor are presented in the last section of this article, including HLS colour space and basic operations.

1. ARCHITECTURE AND ORGANIZATION

Most of the modern image processing softwares run on workstations providing fast and powerful research tools. Presently, the use of colour is of great importance in many research fields, therefore an increasing amount of workstation are equipped with colour monitors. We shall therefore study the general architecture of a workstation (other systems have similar organisations). The path from internal representation to effective screen painting will be followed explicitly.

*To whom correspondence should be addressed.

†Since this paper is being published in black and white, the authors have tried to find examples as illustrative as possible to simulate the use of colours.

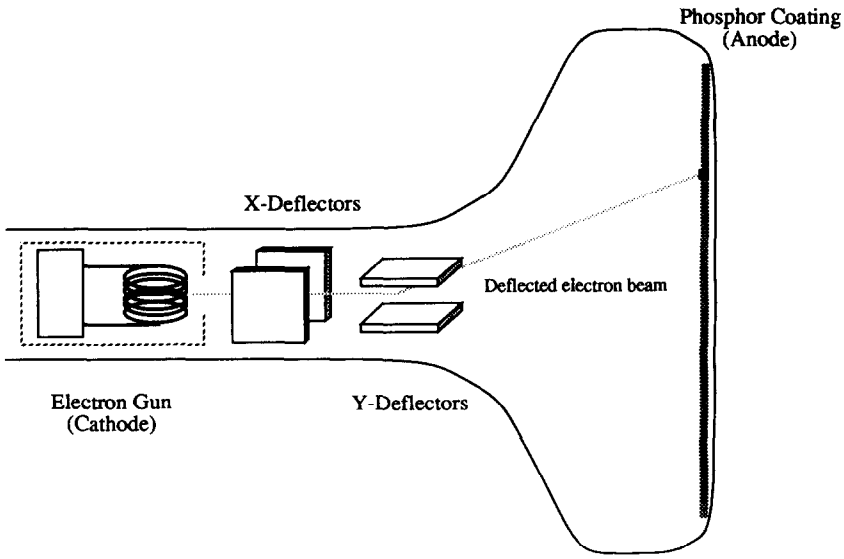

Fig. 1. Cathode ray tube architecture.

1.1 *Definitions*
Colour specifications:

R, G, B Red, Green, Blue
[r, g, b] colour specified in RGB colour space
H, L, S Hue, Lightness, Saturation

Hardware features:

CRT	Cathode Ray Tube
DAC	Digital to Analog Converter
LUT[x]	position in the Look-Up Table
pixel	**pic**ture **el**ement, dot on the raster scan screen
pixel value	integer value associated with each pixel
Mbit	Mega bit (one million bits)

1.2 The three types of displays

The three most widespread classes of displays are reviewed, namely *storage tube, random scan,* and *raster scan* displays. Main features and problems of each class will be outlined, following the technical evolution toward our all purpose colour display.

Computer monitors, like those of television sets, are CRT's (Cathode Ray Tube) (Fig. 1). These devices use a deflected beam of electrons to activate the phosphor coating of the screens. Storage tube displays and random scan displays are both line drawing devices. This means that a line can be drawn from any point on the screen to another. These classes of colour displays are gradually disappearing from the computer market. The direct-view storage tube can be seen as a CRT coated with a very persistent phosphor. A mark on the screen will remain visible for a prolonged time (as long as one hour). Erasing is performed in a 1/2 second by flooding the entire screen with a certain voltage. Individual items cannot be erased separately. Such displays are severely handicapped by their erasure mode and therefore not indicated for interactive applications.

On a random scan display (*calligraphic refresh displays*), the phosphor coating is of short persistence. Redrawing (*refreshing*) has to be performed at a rate of 30 to 50 times per second to maintain the visibility of displayed objects. Fast refreshing rates enable local erasure; the repainting process uses a display controller that scans a list containing all the objects do be repainted. An object to be erased is withdrawn from the list or flagged to be invisible and thus disappears at the following refreshing process. The main drawback of this system is the limited size of the object list. Given a refreshing rate, only a finite number of items can be read from the list and painted on the screen in time; if the list is too long, flicker will appear.

Raster scan devices work on a scheme that is rather different from the list of line segments used in line

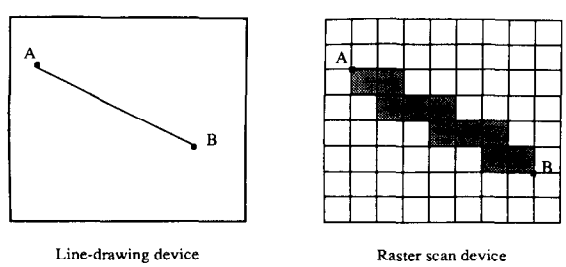

Fig. 2. Comparison of line-drawing and raster scan.

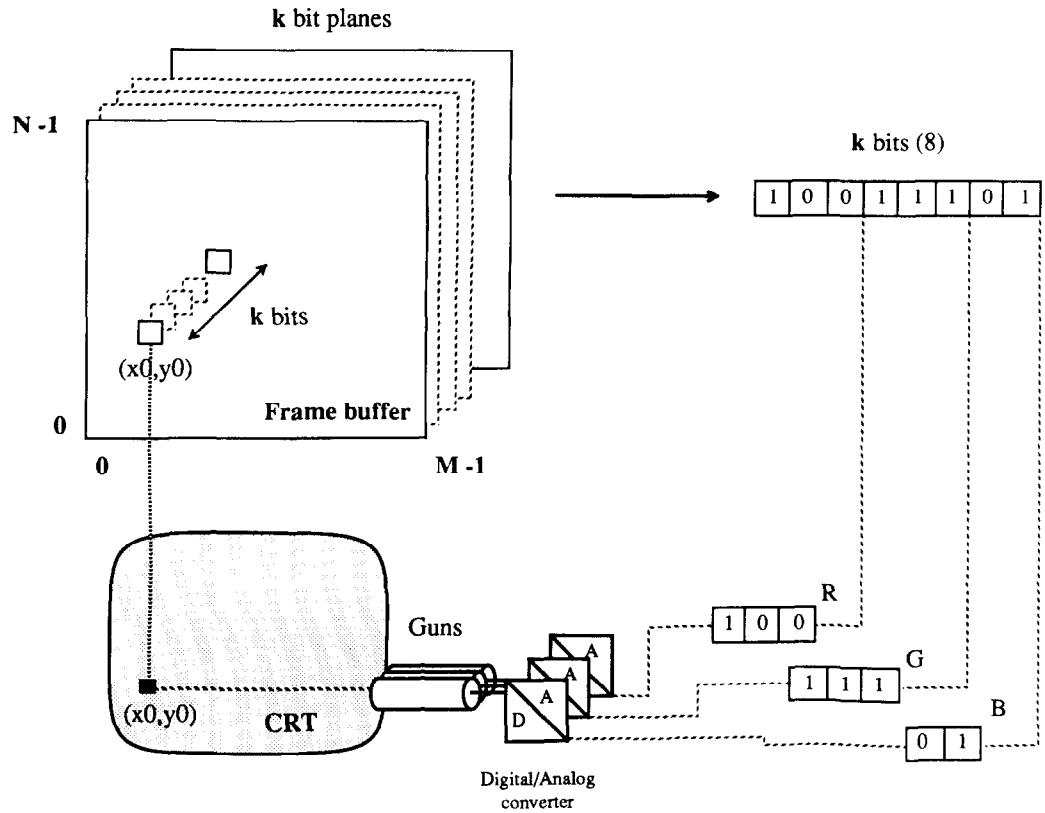

Fig. 3. General colour display architecture.

drawing devices (Fig. 2). The screen of a raster CRT can be considered as matrix of small **pict**ure **el**ements called *pixels*. Each pixel corresponds to one screen memory position and can be addressed separately, allowing fast individual changes of colours (including erasure). Refreshing rates are similar to the random scan display. As a screen is composed of a finite number of pixels, the screen's memory size is therefore constant. No matter how many objects are being displayed, constant memory access time achieves flicker-free refreshing. However, if only a few objects are to be refreshed, the great amount of memory used to address the entire screen is out of proportion. For this reason, animation is often realised on random scan displays that can perform faster refresh rates with few objects. These features have made the raster scan device by far the most popular, ranging from PC monochrome monitors to graphic workstation full colour devices.

1.3 Frame buffer

The value of all screen pixels are represented in a large, contiguous piece of random access memory, the *frame buffer*. Such a piece of memory representing each pixel on one bit is called a *bit-plane*. We define the *depth* of a frame buffer as the number of bits (number of bit-planes) required to carry the necessary information for a pixel. A monochrome display will require one bit per pixel (on/off) whereas a full colour monitor can require up to 24 or more bits per pixel. The general raster scan device is presented in Fig. 3.

On colour screens different shades are created by mixing quantities of red, green, and blue light. In Fig. 3 the colours are coded on 8 bits (3 red bits, 3 green bits, and 2 blue bits), in full colour frame buffers, all pixels are represented by 24 bits (8 red, 8 green, 8 blue). In the later case, the three basic colours being coded on 8 bits, 256 shades are available; combining the three shades will produce more than 16 million ($256 \times 256 \times 256$) colours, an amount that is far above the limit of human colour separation. This considerable amount of colours requires an equally great amount of memory. Let's consider large screens with typically 1000 lines and 1000 columns: one million pixels. One corresponding bit plane will contain 1,000,000 bits (one Megabit, *Mbit*). A full colour frame buffer requires, therefore, 24 Mbit of memory. Correct flicker-free screen refresh can only be achieved by using very fast memory circuits, special registers, and very efficient scanning algorithms.

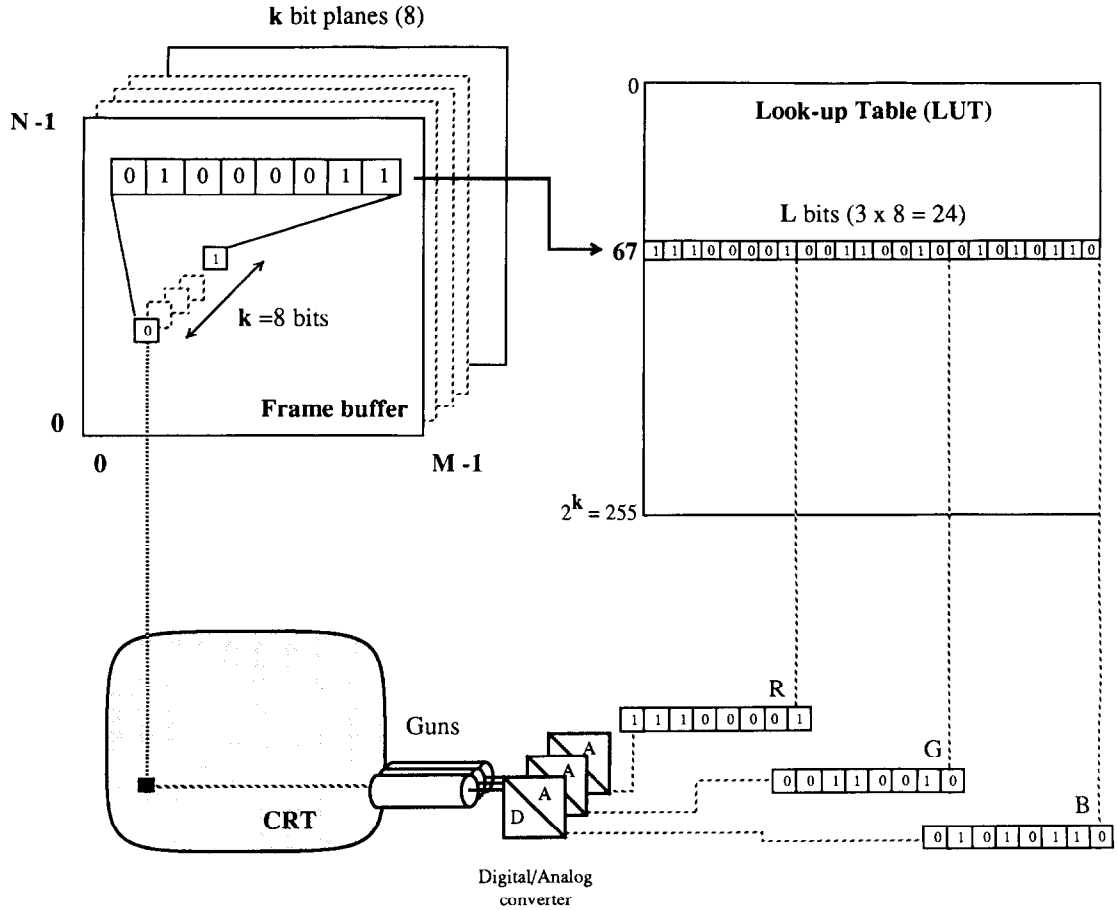

Fig. 4. Colour display architecture with look-up table.

1.4 Look-up table

The insertion of a *look-up table (LUT)* considerably reduces memory and time access expenses. Figure 4 shows a modified version of Fig. 3. Notice the position of the inserted look-up table as well as the modified addressing mechanism.

A look-up table (or *colour map*) is an intermediate table between the frame buffer and the colour display device. The frame buffer contains LUT addresses (not colours), each pixel therefore has a value corresponding to a LUT entry. A set of colours is stored in the colour table. When screen refreshing is performed, pixel values (look-up table addreses) are read from the frame buffer, thus selecting colours from the LUT; selected colours are then displayed on the screen. In Fig. 3 the frame buffer information is fed directly into the Digital/Analog Converter whereas in Fig. 4, pixel information selects a colour map position which will in turn provide the three colour components to be given to the DAC.

Most colour maps have 256 entries, so pixels take values ranging from 0 to 255. Generally, a look-up table position contains **L** = 24 bits, providing the full colour gamut as explained above. In the configuration presented in the previous example, we can predict that the use of a look-up table will have two main restrictions. Firstly, the number of simultaneous displayed colours is limited by the number of colour map entries. Secondly, the look-up table being common to all screen pixels, it is impossible to alter the colour of one single pixel without affecting all the other pixels addressing the same LUT position.

On the other hand, the use of a look-up table has some interesting advantages. The major one is to offer considerable reduction of the frame buffer size. On colour workstations equipped with the LUT mechanism, the frame buffer uses typically 8 bit planes (~ 8 Mbit). Compared with the full colour frame buffer (24 Mbit) the memory economy is of roughly 16 Mbit. Another advantage of using a look-up table is to be able to change pixel colours without altering any of the original frame buffer data. All pixel values are left unchanged, only the colours stored in the colour map are modified. In a full colour workstation where colours are defined explicitly in the frame buffer, changing a pixel colour

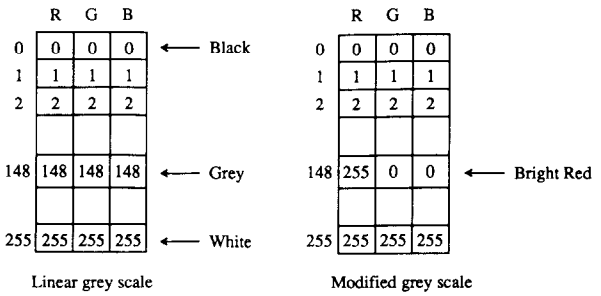

Fig. 5. Changing look-up table attributes.

requires a frame buffer modification. A look-up table can be totally rewritten without touching the frame buffer (i.e., the pixel values), as in Fig. 5. In other words, several colour maps can be loaded successively, emphasising one or the other detail in an image. Of course loading the original map will bring the original shades back immediately. Finally, the impossibility to change a single pixel is compensated by the advantage of being able to modify entire classes of identically valued pixels performing one single modification in the colour map.

A typical *grey scale* colour map ranges from black (value 0) to white (value 255), all intermediate greys taking values from 1 (very dark grey) to 254 (very pale grey). Let's define a given look-up entry by LUT[i] = [ri,gi,bi], i being the look-up table address and [ri,gi,bi] the three colour parameters. Mixing an equal amount of all three basic colours will produces shades of grey so we can now define a linear grey scale colour map as:

$$\text{LUT}[0] = [0,0,0],\ \text{LUT}[1] = [1,1,1],\ \ldots,$$
$$\text{LUT}[i] = [i,i,i],\ \ldots\ \text{LUT}[255] = [255,255,255].$$

If we now change LUT[148] and store a bright red[255,0,0] instead of the stored grey[148,148,148], all pixels with value 148 will be painted red. Of course performing the inverse operation would bring back grey[148,148,148]. These manipulations are shown in Fig. 5.

The most important fact illustrated by Fig. 5 is that we can change pixel colours without modifying values stored in the frame buffer. All pixels addressing the same look-up table position (148 in Fig. 5) will be painted the same colour. Loading a new colour map being practically instantaneous, the main advantage of using look-up tables is to provide the possibility to perform all sorts of image processing in real time (multicoloured thresholding, for example). The time in which the colours change on a look-up table driven device (one operation *per table entry*) is far smaller than the time necessary to change the colour of a pixel region (one operation *per pixel*) on a 24 Mbit frame buffer device.

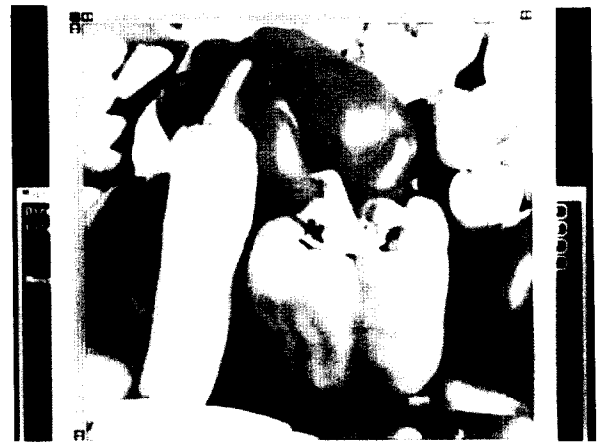

Fig. 6. Typical grey scale digital image with pixel histogram.

1.5 Displaying images

Colour map modifications are mostly used after an image has been displayed. Image processing often applies operators, filters, transforms, etc. on image data (5). On the other hand colour map modification does not affect the original signal. Operations are performed on a small table instead of on thousands of pixels. On devices such as those we are concerned with, image data is first mapped on integer values ranging from 0 to 255 and then displayed. Most of the time colour maps are initialised with a grey scale, displayed images often being monochrome.

Pixel value reveal some interesting details when a *pixel histogram* is calculated. Pixel histograms are a normalized plot showing for each table entry (i.e., 0–255) the number of pixels painted in its colour. Such an histogram can show features such as insufficient dynamic range, histogram region peaks, etc. (5). Figure 6 shows a typical grey scale image with its pixel histogram drawn on the look-up table laid out beneath. The look-up table is displayed with LUT[0] at the left end and LUT[255] at the right end. The exposed grey scale map is linear, as in Fig. 5. The histogram, for example, indicates that there are no pixels having a grey value in the range 212–255. We emphasise the fact that due to the LUT transformation, the pixel histogram is not necessarily the same as the original image histogram.

2. LOOK-UP TABLES AND MANIPULATION OF IMAGE APPEARANCE

Digital images are most of the time the result of a scanning process often carried out on photographs. In

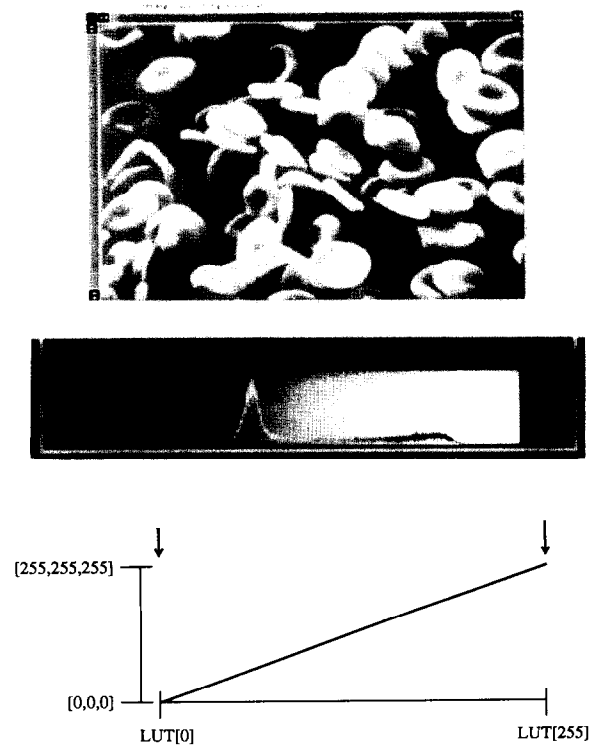

Fig. 7. Original red cell image: low contrast, poor dynamic range.

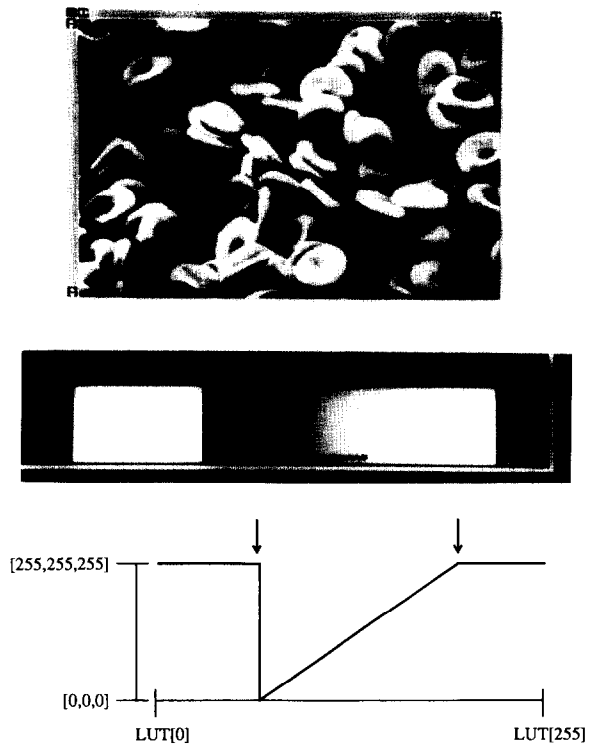

Fig. 8. Adjusted dynamic range.

the past years, the quantity of available digital pictures has never ceased to increase, providing images of all kinds and quality: large, small, dark, clear, underexposed, dull, etc. The first application of colour map editing is to correct the defects of photograph scanning such as those enumerated above.

We can now begin a more pragmatic approach, using the above explained concepts to improve the image display on our colour monitor. It is important to remember that image values are not directly displayed, but rather are indexes into the look-up table. As we have seen in the first chapter, the look-up table is rather small, changing its attributes can therefore be performed in real time. High speed is a main asset of LUT modification, giving the user the possibility to perform real time multicoloured thresholding.

2.1 Adjusting dynamic range

Lets consider a typical example of an overexposed image with poor dynamic range. Figure 7 shows the original grey scale digital image of red cells. Below the pixel histogram is painted on the colour map. We have also drawn a graphical representation of the LUT's content. In this first expample, the displayed grey scale is linear.

The pixel histogram deserves explanation. We can clearly distinguish four regions from left to right: a flat zone, followed by a large pointed peak, a rather small, slow growing bulge and, at the end, another flat region. The flat regions show look-up table addresses that are never used. The tall, pointed peak shows a great concentration of pixels painted in the same colour: they correspond to the background. Finally, starting at the bottom right end of the background peak, we find a larger peak. It contains a large interval of used addresses we can associate to the shades of gray the red cells are painted in.

Let's first consider the two flat regions. As we have said, they contain the look-up table addresses that are never used. This phenomenon is typical in the case of a bad dynamic range. If we concentrate the grey scale on the remaining two regions, cells should appear more clearly. Figure 8 shows the same image with corrected look-up table; all grey levels are used to display the pixels. The predicted result is clearly visible. Notice that both displayed histograms are the same, since the original image data has not been be modified.

2.2 Real time thresholding

Another interesting operation on look-up tables is real time multicoloured thresholding. Although all following

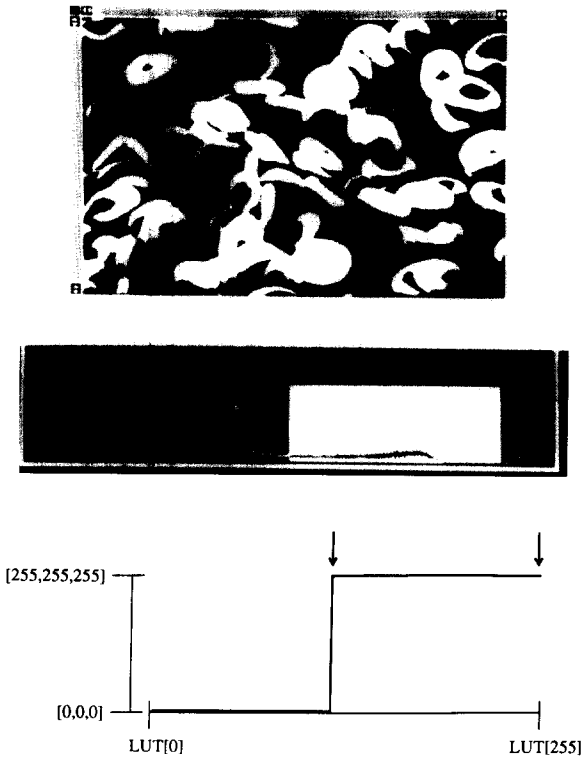

Fig. 9. Black and white single threshold.

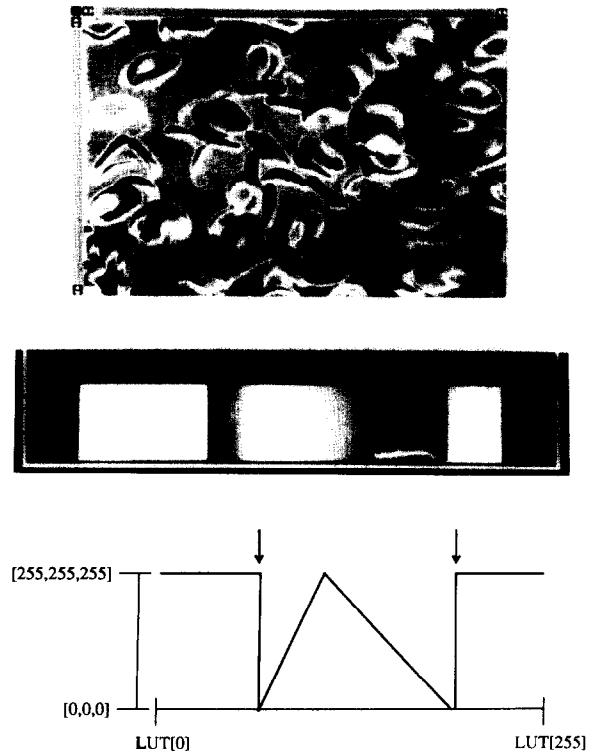

Fig. 10. Specific look-up table.

examples are black and white, colours can be used producing even sharper contrasts. Figure 9 shows a simple threshold. All look-up table entries are initially painted black. A threshold is defined: pixels above a given value are painted white, the others remaining black. The threshold is then moved into the middle of the table as in Fig. 9; all red cells are painted white, the background pixels remaining black. Moving the threshold is performed in real time; the pixels are painted instantly.

More thresholds can be defined as in Fig. 14. When comparing to the original image (Fig. 11) notice the clear edges appearing on the left rim of the skull.

2.3 Specific look-up tables

Let's assume the user decides to compare results of a given experience performed a number of times. Using several digital images taken under the same conditions, loading the same specific colour table for every image will enhance the same class of details on each image. Keeping the same images displayed and loading new colour tables will bring other details in the foreground. Figure 10 shows the same red cell image with a special map. Notice how the outlining of the cells becomes clearly visible. Notice also that pixels values at both extremities of the colour table are painted in the same colour.

Another set of examples shows the use of various look-up tables. Figure 11 shows the original image, Fig. 12 dynamic range adjustment, Fig. 13 specific region enhancement look-up table, and Fig. 14 multiple thresholding.

Although the figures presented in this document are only grey scale images, the use of colour tables has produced interesting results without performing any computations on the image data. Look-up table manipulations often provide more striking results when using colours mostly because the colour contrast reveals more details.

2.4 Discussion

On medical images, so called *"pseudo colours"* are often used to enhance interesting details. Pseudo colours are bright, contrasted colours. When a look-up table is loaded with a few pseudo colours, sets of pixels are painted in the same colour as when performing multiple thresholding. Regions with the approximate same features (i.e., painted in the same colour) will then appear. However, the use of pseudo colours can bring some mystifying results such as creating new edges. Let's imagine a picture of a ball. When displaying the image using a grey scale, smooth changes of grey shades will occur on the ball surface until its boundary. If we load a

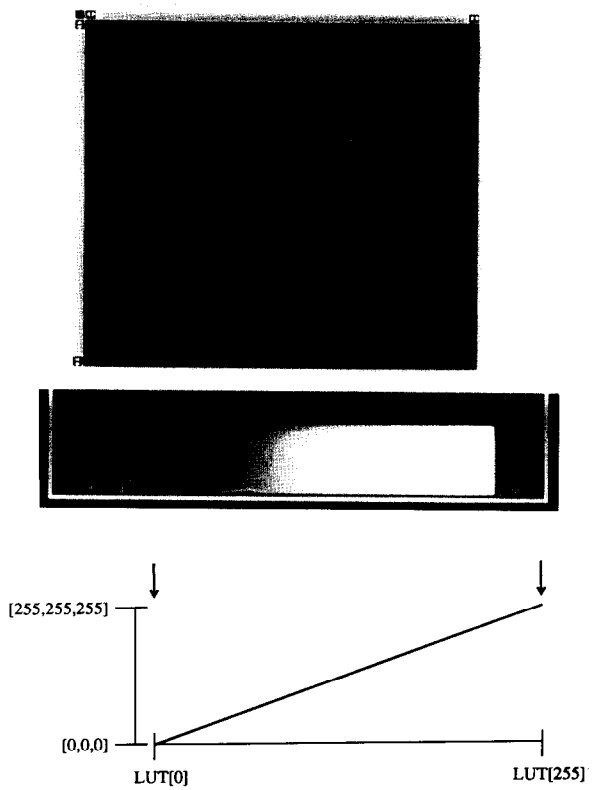

Fig. 11. Original skull image.

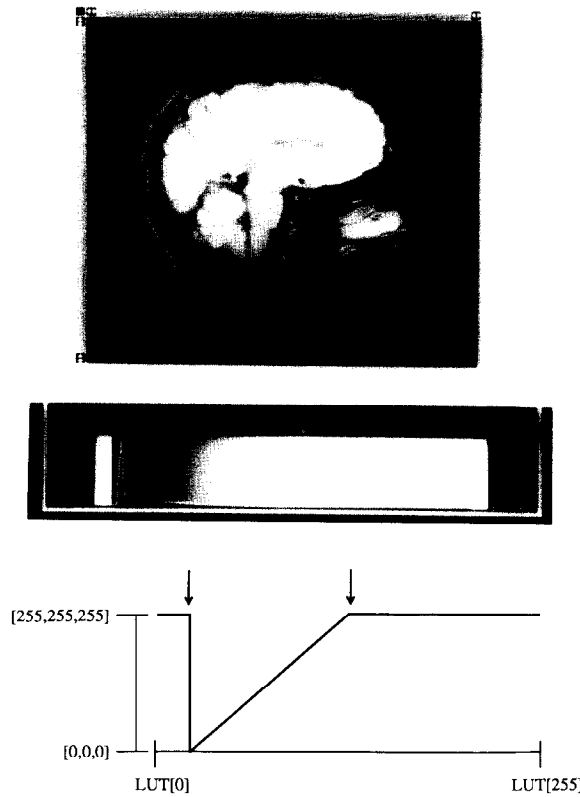

Fig. 12. Adjusted dynamic range.

look-up table containing only four different colours, changes will not be smooth but abrupt, the ball will disappear under concentric rings. This phenomenon is called false contours. False contours can be misleading because they can appear at places where nothing should be seen. Especially in the medical world where pseudo colour are often used, details can appear that have no significance. Conclusions from pseudo coloured images have to be drawn with precaution, especially because unnecessary details are often enhanced.

3. MAPEDIT

The growing importance of optimal use of colour tables encouraged us to develop a specific look-up table managing software. We have therefore written a colour map modification tool that can be included in other softwares. This tool gives the possibility to visualise, create, and edit look-up tables, working on dynamic maps that can be stored in files. From a practical point of view, MapEdit was designed to be essentially driven by mouse operations: except from file naming, all operations can be performed with the mouse. MapEdit was written in C language for SUN[tm] workstations running on SunView window manager. A new version in Xwindows should soon be issued.

The tool (displayed in Fig. 15) provides the following facilities:

- Two colour spaces: RGB and HLS
- Loading/storing colour map files from/on disk
- Linear or logarithmic first degree interpolation
- Option to freeze and extend a given colour interval for thresholding
- ....**one** to change one table entry at the time
- .... **multi** to change a set of contiguous table entries
- Vertical arrows delimiting the current interval
- Help button for interactive help.

3.1 Colour spaces

Two colour spaces are provided, one is computer oriented (RGB), the other is user oriented (HLS) (7). The RGB colour space was chosen because of its use in colour display architecture. As we have seen in previous chapters, look-up table fields contain [r,g,b] coordinates. Therefore, managing look-up tables in RGB coordinates requires no colour space transformations. RGB colour space is also useful when correcting overall features of digital images. Like colour displays, cameras

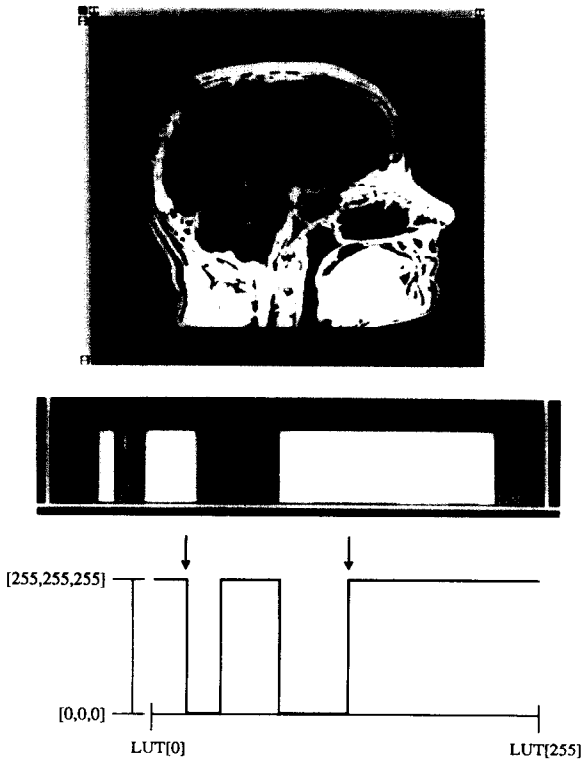

Fig. 13. Contrasted skull outlining applying multiple black and white thresholding.

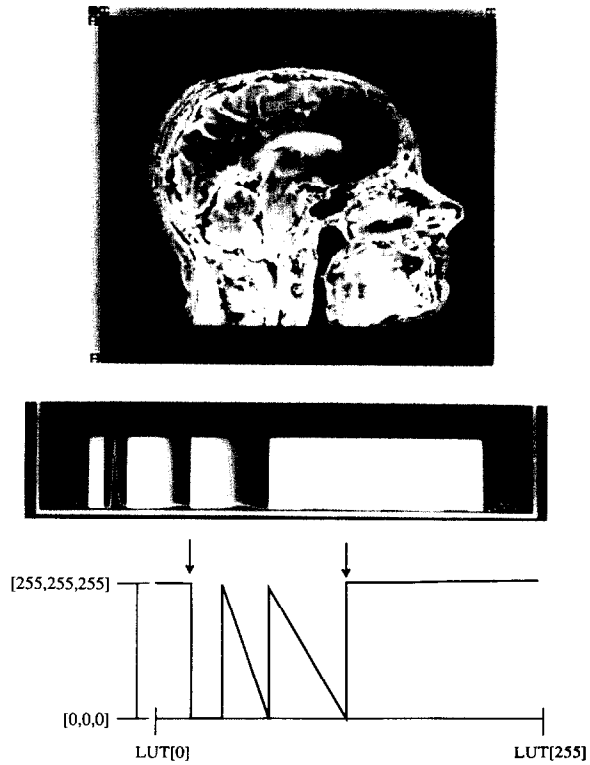

Fig. 14. Contrasted brain trough specific region enhancement.

that digitize images also handle RGB colours; colour images are passed through three filters (red, green, and blue), returning three corresponding images planes. It sometimes happens that one of the colour planes is too dominant (for reasons such as nonwhite illumination, inadequate filters, etc). For example if an image is blueish, all blue coordinates are to be reduced proportionally to obtain a balanced image.

The great disadvantage of the RGB colour space is the difficulty of hue manipulation in a intuitive way. Let us take the example of the colour orange. Orange can be defined in human terms as a yellowish red, but is becomes difficult or even impossible to express orange in RGB coordinates. It was therefore decided to provide the user with a second colour space (HLS—Hue, Lightness, Saturation) where colour manipulation is intuitive. Figure 16 shows the conic spatial representation of the HLS space.

Hue is defined by an angle. The colour range is {red(0°), yellow(60°), green(120°), cyan(180°) blue(240°), magenta(300°), red(360°)}, all intermediate colours being possible. At this stage we can already define orange quite easily as intermediate between red(0°) and yellow(60°), for example 30°. In HLS colour space, the Lightness ranges from black(0.0) to white(1.0). As lightness reduces the colour darkens. In Fig. 16, bright red becomes light-red if lightness increases, it becomes dark red if lightness reduces. Saturation defines the purity of a colour; in the colour space S is defined as the horizontal distance to the centre of the cone. When reducing saturation, bright red ($S = \max = 1.0$) becomes dull (~ brick red) and finally grey ($S = \min = 0.0$). On the vertical line in the centre of the colour space ($S = 0.0$) we find the grey scale ranging form black ($L = \min = 0.0$) to white ($L = \max = 1.0$). It is thus possible to handle colours and shades intuitively, each HLS feature corresponding to a human-like characterization of colours.

3.2 Using MapEdit

Colour space is chosen by selecting **RGB**... or **HLS**... mode; ...**multi** indicates multiple field modification, ...**one** selects single field modification.

As we can see on Fig. 15, the look-up table is displayed at the bottom of the tool. Two vertical arrows are pointing to the current limits, and thus determine an interval. The numerical position of each limit is indicated by the **Field Number** label, colour parameters are indicated by the horizontal sliders. In our example, the current interval is the whole look-up table, the colour space in HLS, and the two limits are {$L[0] = [113°, 1.0, 0.52]$; $L[255] = [251°, 0.23, 0.33]$}. All future opera-

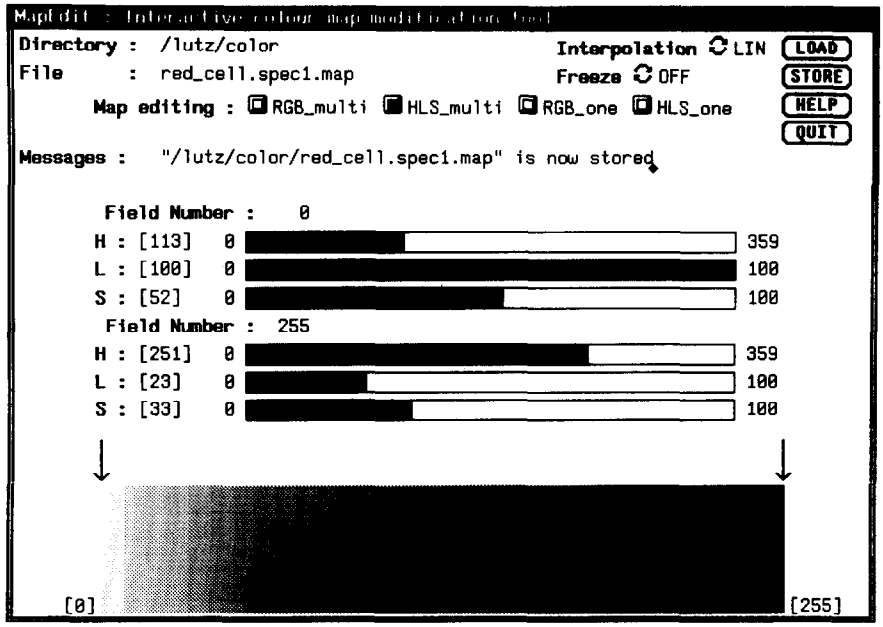

Fig. 15. MapEdit (this paper being published in black and white, the displayed LUT is gray scale; the original LUT resulting from an integration from [113,1.0,0.52] to [251,0.23,0.33] ([H,L,S]) is in colour).

tions will affect the current interval until its limits are changed. If we decide to change the intervals colour, parameters are changed by moving corresponding sliders. As soon as a new parameter has been set, all colours of the current interval are interpolated between the limits according to the selected interpolation mode (**LIN** = linear, in Fig. 15). Colour changes are instantaneous over the entire screen. Limits can be changed by moving the desired arrow into its new position or by typing the position in the corresponding **Field Number**.

Real time thresholding is performed by using the **Freeze** option. By default, **Freeze** is turned off: when vertical arrows are moved, the parameters of pointed to colours are updated and indicated by the sliders. When **Freeze** is turned on, the limit colours are "frozen": when arrows are moved, positions change but not colour features. A defined colour interval can be expanded or reduced, thus performing real time thresholding in colour.

Look-up tables can be stored in files. Colour information is coded in RGB colour space. The size of look-up table files is very small: 756 bytes (256 red + 256 blue + 256 green). The current file name must be typed, but all load/store operations are selected by the mouse.

The **HELP** button opens an information window. A set of information topics is proposed by means of buttons which can be selected.

The MapEdit version used to produce the figures was integrated in an image processing software (LaboImage,

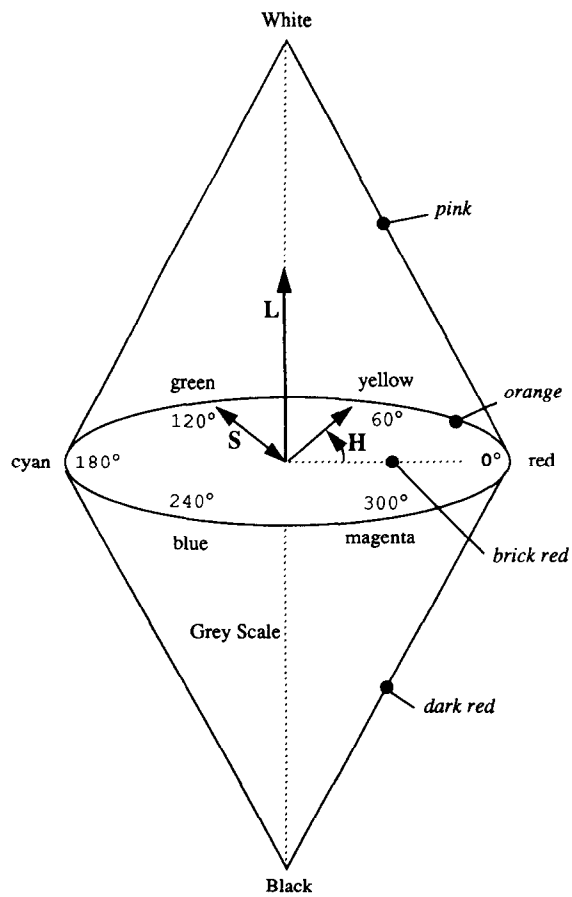

Fig. 16. HLS colour space.

birth defect (6)) also developed at University of Geneva*. Histograms (Fig. 6 to Fig. 14) were calculated and displayed on the MapEdit look-up table by LaboImage routines.

CONCLUSION

The use of look-up tables can provide powerful means to modify displayed images. The relative small memory space required by a look-up table as well as the very fast memory access time, make it possible to perform real-time operations over the entire screen. The 8-bit colour displays are less powerful than full colour displays, but adequate use of colour tables can turn some disadvantages into remarkable features. The above examples show how useful the colour map transformations can be. In a few years, technology will have produced fast, very large, and inexpensive memories dooming the use of look-up tables. But up until then, look-up table editing will provide a fast and powerful tool to manage colour or grey scale images.

SUMMARY

Presently, colour displays are spreading out very rapidly, with growing possibilities. Colour devices are more and more commonly used, users being often unaware of how colour devices work.

Notions such as CRT, raster scan displays, frame buffer, look-up table, have therefore been introduced first. Different types of displays have been discussed, explaining the basic architecture of colour display devices. Concentrating the discussion on raster scan devices, the concept of frame buffer has been defined, analysing its organisation and introducing notions such as bit-plane and frame buffer depth. Next was defined a look-up table, inserting it in the previously examined architecture. Relations between frame buffer and look-up table have been analysed in detail. Then, memory and time expenses of full colour devices opposed to look-up table equipped devices have been compared, following step by step the process of changing pixel colours by working on look-up table attributes.

After examination of the hardware of colour displaying the discussion has concentrated on the way digitized pictures are displayed on the screen. Relation between data, pixel value, and colour have been explained. The link between pixel colour and pixel value, has been examined in relation with look-up tables.

Using typical examples, displayed images have been altered by modifying look-up table attributes, but without altering image data. Illustrated by medical images, it has been shown how look-up table modifications can be of great help in many situations. Underexposed images with poor dynamic range have been corrected. Psuedo colours and multiple thresholding have been defined and explained. Specific application oriented colour maps have been loaded to enhance specific details on several images. All figures were produced by a look-up table modifier developed at the University of Geneva, whose features such as HLS colour space and basic operations have been enumerated in the last chapter.

Acknowledgements— This paper is the result of a research development by the vision group of CUI, University of Geneva. We gratefully acknowledge Mathieu Funk and Alain Jacot-Decombes for useful comments and willingness.

REFERENCES

1. Foley, J.D.; Van Dam, A. Fundamentals of interactive computer graphics. Addison-Wesley; 1982.
2. Rogers, D.F. Procedural elements for computer graphic. McGraw-Hill Book Company; 1985.
3. Rebordao, J.M. Look-up table loadings for image processing with controlled knots. Comput. Vision, Graphics, and Image Proc. 47:189–202; 1989.
4. Bates, R.T.H.; McDonnell, M.J. Image restoration and reconstruction. Oxford: Clarendon Press; 1986.
5. Gonzalez, R.C.; Wintz, P. Digital image processing, 2nd ed. Addison-Wesley; 1987.
6. Jacot-Descombes, A.; Todorov, K.; Hochstrasser, D.F.; Pellegrini, C.; Pun, T. LaboImage: A workstation environment for research in image processing and analysis. Comp. Applic. in the Biosciences; in press.
7. Murch, G. Color displays and color science. In: Durret, J.H. ed. Color and the computer. Academic Press; 1987.

About the Author—RENE W. LUTZ received his B.Sc. in 1987 and M.Sc. in 1988, both in computer science at the University of Geneva. Since then he has been working as a teaching assistant in computer graphics. He is currently working on a Ph.D thesis in computer vision.

About the Author—THIERRY PUN received his electrical engineer diploma in 1979, and his doctoral degree in image processing in 1982, both from the Swiss Federal Institute of Technology in Lausanne. From 1982 to 1985, he was a visiting fellow at the National Institutes of Health (Bethesda, MD, USA), where he developed several methods and software package for medical image analysis. From 1985 to 1986, he was at the European Laboratory for Particle Physics in Geneva (CERN), where he worked on signal and image analysis problems. Since then, he is with the Computer Science Center, University of Geneva, where he curently holds a position of associate professor of computer science. He teaches computer graphics, image processing and analysis, and computer vision. His research interests deal with medical image analysis and computer vision.

About the author—CHRISTIAN PELLEGRINI is professor of Computer Science and director of the Computer Science Department at the University of Geneva. He received his M.Sc. in physics in 1970, and his Ph.D. in computer science in 1975, both from the University of

*People interested in obtaining MapEdit or LaboImage should contact the author or A. Jacot-Descombes. E-mail: *lutz@cuisun.unige.ch (or jacot@cuisun.unige.ch)*. Theses softwares currently run on SUN 3 and SUN 4 computers, under operating systems O.S. 4.0.3.

Geneva. From 1977 to 1978, he was "IBM Post-Doctoral Fellow" at T.J. Watson Research Center where he worked on distributed system. Since 1979, he gives courses in computer science at various levels at the University of Geneva and is engaged in research in computer science. His present research interests are in the fields of artifical intelligence, computer vision, and neural networks.