38.4: Displaying True Color Images Using a Limited-Depth Frame Buffer

A. C. Barkans
Hewlett-Packard Co., Ft. Collins, CO

Abstract

For many years the only practical way to display high quality *true color* images (those with millions of colors) was on a computer with a graphics sub-system with at least 24 color planes. However due to the high cost of color graphics devices with 24 planes, many users chose 8 plane systems. Unfortunately, using these 8 plane systems requires giving up some color capabilities in order to save cost. However, a new innovation called '***Color Recovery***' provides a method for displaying millions of colors within the cost constraints of an 8 plane system. Of course pretty pictures aren't enough: therefore Color Recovery supplies the additional color capabilities without giving up interactive performance. In addition it works with all types of applications running in a windowed environment.

Introduction

Defining the term "True Color":

In this paper the term "**true color**" is used to define color reproduction such that the underlying digital quantization of the color within an image is not discernible by the human eye. In other words a continuous spectrum of color, such as in a rainbow, can be displayed such that the color appears to vary smoothly across the image.

In most computer graphics systems the true color effect is accomplished by using 24 bits of color information per pixel. With 24 bits, any single pixel can be displayed at one of 2^{24} (16.7 million) colors. Because of this the term true color is often used to describe 24 bit per pixel graphics systems. However, it is important to note that the definition used in this paper does not imply a certain number of bits per pixel. Instead the definition is based on the visual quality of the image.

Using Traditional 8 Plane Systems:

Traditional 8 plane systems can display only 2^8 (that is 256) colors. There have been two approaches that have been employed in order to get the best results with limited colors. The first is called either *pseudo color* or *indexed color*. The idea is to select a set of 256 colors and then limit the application to using only that fixed set of colors. For many applications, such as word processing and business graphics, this approach works reasonably well. This is because the resultant images are made up of very few colors. However, when an application needs more than 256 colors, such as realistically shaded MCAD (Mechanical Computer Aided Design) images or human faces in video sequences, then another approach is needed. Since more than 256 colors are required for these applications a technique to simulate more colors is used. For these applications a technique called *dithering* [1] *is* employed. The idea with dither is to approximate a single color by displaying two other colors at intermixed pixel locations. For example a grid of black and white pixels can be displayed to simulate gray. Such a grid of black and white pixels will indeed look gray when viewed from a distance. The primary problem with dithering is that most people don't work at their computer while sitting at a distance. Since people tend to work close to the display, dithered images are viewed as having a grainy or textured appearance.

The Color Recovery Process

Color Recovery is a two part process: First, true color information generated by the application is dithered and then stored in the frame buffer. The type of application generating the true color information is immaterial; for example it can be generated by a CAD application program or as part of a video sequence. The dithering may be done as either part of a device driver in software, or in the hardware of a graphics controller. The second part of the Color Recovery process is to integrate the dithered data. The idea behind the integrator is to assist the human eye in reconstructing the true color image. This is done by removing the dithering artifacts. The integrator is placed between the output of the frame buffer and the DAC's that drive the monitor.

Perhaps the most interesting thing about the Color Recovery image generation pipeline is that it looks almost like any other 8-bit graphics display pipeline. The only significant difference is in the area where most systems have a RAMDAC. In a Color Recovery system the traditional RAMDAC is supplemented by a specialized DSP (Digital Signal Processing) circuit. The entire Color Recovery process is shown at the top of the next page in Figure 1.

ISSN0097-0966X/95/2601-0840-$1.00 + .00 © 1995 SID

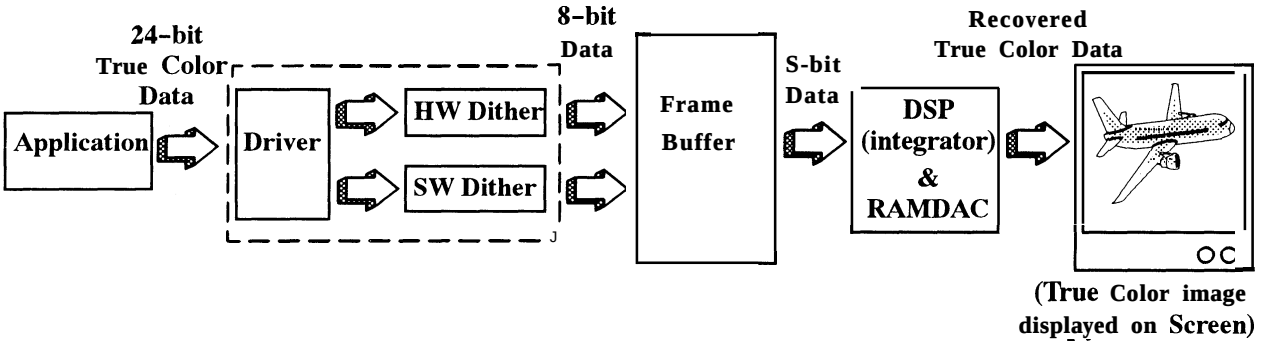

Figure 1: The Color Recovery Process

The first step shown in Figure 1 is any application that generates *true color* data. It is important to note that these applications *do not* have to be rewritten to take advantage of Color Recovery. The next two steps, shown in the dotted box, constitute the device driver. The function of the driver is to isolate the application from hardware dependencies. (Drivers are typically supplied by the hardware vendors. So in the case of HP workstations, the driver is supplied by HP.) Data output by the application is initially sent to a software layer that then determines how to dither the data for storage in the frame buffer. The driver uses the hardware dithering when possible. However, there are times when the driver must perform the dither in software. It is important to note that when compared to other dithered systems, there is no performance penalty for using the Color Recovery dither. In other words application performance will not be affected.

The fourth step of the Color Recovery process is to store the dithered image data in the frame buffer. In typical 8-bit systems the output of the dithered frame buffer is scanned to the display. In these typical systems, as the frame buffer data is being scanned it passes through a color look-up-table (LUT) in the RAMDAC. However, the LUT can not remove the patterned appearance in the image. Therefore in typical 8-bit systems the quantization of the image by dithering is readily apparent.

The fifth step shown in Figure 1 is the unique part of Color Recovery. In a Color Recovery system, as the frame buffer data is scanned it is sent through a specialized digital signal processing (DSP) circuit. The DSP is a very sophisticated circuit that removes the patterning from the dithered image stored in the frame buffer.

The last step in the Color Recovery process is to display the resultant image on the display screen.

The Color Recovery Dither Process

In Color Recovery the quality of the displayed image depends on the dither used to encode the image. During the development of Color Recovery it was found that the size of the dither region determines how well a color can be recovered. It was found that a region of 2^N pixels can recover about 'N' bits of color per component. Therefore an 8 bit frame buffer that stores data in 3-3-2 format (3 bits each of red and green with 2 bits of blue) would need a dither region of 32 pixels in order for each color component to recover 5 additional bits. Thus when using a 32 pixel dither region, an area in the image of uniform color can have the same visual quality as an 8-8-7 image.

Most dithers use a 4 X 4 dither region. Since a 4 X 4 region covers only 16 (that is 2^4) pixels, a total of 4 additional bits of color per component could be recovered. However, it was found that in order to produce the most visually pleasing images that 5 additional bits per color component were needed. Therefore with Color Recovery the dither table has 32 entries. Additionally with Color Recovery the dither table includes both positive and negative numbers. This improves the color range over which the dither is useful.

The Color Recovery dither is a little different when compared to most dithers. However, it is on the same order of complexity. In addition it is very easy to place in hardware, such as is done on all HP graphics workstations that support Color Recovery. This means that using the Color Recovery dither does not cause a decrease in performance.

The Color Recovery Intimation Process

The integration process is best shown with an example. In the example assume an 8 bit true color value of binary 01011000 is given for one of the color components. Also assume that we wish to dither this color component to 3 bits. At this point it is easiest to follow the example if we specify the true color data as a *three point five* format number. In other words place 3 bits in front of the binary

point and 5 bits behind it. Thus the original number becomes O1O.11OOO binary which we can call 2.75 in decimal. Next we will assume that the original 8 bit color component is dithered to three bits for storage in the frame buffer. For simplicity we will assume that the dither region is 2 X 2 pixels. (Note: as stated previously, the actual dither used in the implementation of Color Recovery covers 32 pixels.) If we use this 2 X 2 dither region then ideally the end result of applying the dither is that 3/4 of the pixels stored in the frame buffer are set to the value 3, and 1/4 of the pixels set to the value 2.

To make the example case more interesting we will also assume that there is an edge in the image. For the example we will assume that on one side of the edge the color component is specified as 2.75. On the other side of the edge we will assume the color component is specified as 6.25.

If the full precision of the true color data can be stored (such as in a 24 bit per pixel system) the data in the frame buffer would be as shown in Figure 2a. Note that in the figure each box represents the color component stored at a pixel location on the display screen. Also note that the figure represents only a small part of the entire display surface.

Since we our interested in the case of a limited depth frame buffer, the full precision color can not be stored. As previously stated, Color Recovery uses dithering to reduce the number of bits for storage in the frame buffer. In our example the original color specified by 8 bits will be dithered to 3 bits. Since the 3 bit dithered values are the integer part of our *three point five* format numbers, a mix of integer values should be used to approximate the true color data. In our example the color data specified as 2.75 would be stored in the dithered frame buffer with 3/4 of the pixels set to the value 3, and the remaining 1/4 of the pixels set to the value 2. The results of dithering the true color data in Figure 2a is shown in Figure 2b. Note that in a typical dithered system this is the data that is sent to the display. In these typical systems the quantization due to dither is readily apparent in the image. It is this "bumpy" frame buffer data that gives rise to the grainy or textured appearance of the image.

However, it is easy to see that if we average the 2 X 2 region of pixels inclosed in the box near the pixel shown as **Pix_1** in Figure 2b that we can perfectly recover the original true color data. In this case the average is found as follows:

$$([3 \ X3] + [2 \text{ X } 1])/4 = 2.75$$

If the operation is repeated for the pixel to the right of **Pix_1** the same answer is obtained. In general any region of constant color can be perfectly recovered.

2.75	2.75	2.75	6.25	6.25	6.25
2.75	2.75	2.75	6.25	6.25	6.25
2.75	2.75	2.75	6.25	6.25	6.25

(A)

3 Pix_1	3	3	7	6	7
2	3	2	6	6	6
3	3	3	7	6	7

(B)

Figure 2: *(a) Pixel values for original color data specified as 8 bits per color component. (b) The color data from Figure 4a after it has been dithered and stored in the frame buffer.*

However, if the averaging operation is repeated for the third pixel, shown as **Pix_3** in Figure 3, then the resultant value would be 4.50. Displaying such a value would result in the edge appearing smeared. To solve this problem a special edge detector that looks for edges in noisy data is used. The idea is to compare each pixel in the region with a value that is within plus or minus 1 of the pixel being evaluated. Since the data stored at **Pix 3** is a 3, only pixels within the evaluation region that-have a value of 2, 3 or 4 would pass the edge compare. The values that pass the edge detector are then summed. The total is now divided by the number of pixels that pass the edge compare. In the example when evaluating **Pix_3**, only **Pix_3** and the pixel below it would pass. Summing the two passing values together and dividing by 2 gives a result of 2.50. This value is slightly different than the original value of 2.75, but a better estimation to the original than the 4.50 obtained without the edge detection. The displayed values for the entire example region are shown in Figure 4.

3	3	3 Pix 3	7	6	7
2	3	2	6	6	6
3	3	3	7	6	7

Figure 3: *Evaluating the pixels in the region around* **Pix 3.**

2.75	2.75	2.50	6.25	6.25	6.25
2.75	2.75	2.50	6.25	6.25	6.25
2.75	2.75	2.50	6.25	6.25	6.25

Figure 4: *The true color data recovered from the data stored in the frame buffer of Figure 2b. This is the data data sent from the Color Recovery DSP circuit to the display.*

It is interesting to note that the human eye works better at differentiation than at integration [2]. As such small variations in color in regions that should appear as a constant color are easily detected by the human eye. However, small color errors at edges often go unnoticed. In other words, the Color Recovery process complements the human eye very well. Color Recovery works best in the regions the eye has the most trouble with. Conversely, the Color Recovery integrator starts to break down in the parts of images where the eye is strongest.

Software Considerations

Since dithered frame buffers are in common use today, it is known existing software can work with a dithered frame buffer. There are countless applications that work with dither today. These applications could all work with Color Recovery.

All HP workstations, released since January of 1994, with support for 8 bit per pixel images, have the Color Recovery logic in hardware. In these products we have chosen to have Color Recovery enabled as the default for 3D applications when run in an 8 bit visual. Thus when using Starbase, PHIGS or PEXlib, and opening an application in an 8 bit visual with true color mode, Color Recovery will normally be enabled. Of course setting an application to use a pseudo color map will disable Color Recovery and give the application the desired pseudo color capability. Because Xlib is more tied into the old color map model than the 3D libraries, Xlib applications leave Color Recovery off by default. However, a mechanism is supported that allows Color Recovery to be enabled using Xlib. (For the most part, Xlib applications must take on the Color Recovery dithering task as part of the application.)

Implementation

The implementation of the Color Recovery integration logic for the HP712 workstation performs over 9 billion ($9X10^9$) operations per second. Despite this enormous processing power the circuit is surprisingly small. The entire integrator circuit is built from approximately 35,000 transistors. When compared to the number of transistors required to increase the number of color planes this is very small. For example increasing a SVGA resolution display (1024 X 768 pixels) from 8 bits per pixel to 16 bits per pixel requires over 8,000,000 transistors (one MegaByte of additional frame buffer memory).

Acknowledgements

There are many people that have helped transform Color Recovery from an idea into a reality. The list would be too long to print here. Without the list of names I hope everyone involved knows that I appreciate their efforts. However, there are several people that I must list by name. These are Paul Martin, Larry Thayer, Brian Miller and Randy Fiscus. Additionally, a special thanks goes to Dave McAllister who took my notes and turned them into real logic. Along the way, he found many innovations that lead to a better design.

References

1. Foley, J, A. van Dam, S. Feiner and J. Hughes. *'Computer Graphics: Principles and Practice',* 2nd Ed. Addison-Wesley, 1990

2. Cornsweet, T. *'Visual Perception',* Academic Press, London 1970