PHILIPS

PHILIPS INTERACTIVE MEDIA

Technical Note #86

Improved DYUV Encoding Methods

Brent Burley

July 1, 1993

This note describes a DYUV encoding method which examines an entire scan line as opposed to the pixel-by-pixel method. This method can reduce error in the converted image and can meet the additional constraints required for DYUV "blitting." Two new tools, **rgb2dyuv** and **dyuvfit**, have been developed to take advantage of this method.

Table of Contents

Improved DYUV Encoding Methods

INTRODUCTION

DYUV, one of the available image encoding methods for CD-I, is the only available choice for producing images with 16 million colors on a base case CD-I player. This makes DYUV the best choice for natural or photographic images. DYUV has additional advantages, because it requires only one video plane (RGB555 uses two video planes), and it does not require any CLUT banks, as do CLUT4, CLUT7, and CLUT8.

DYUV encoding does, however, introduce some error into the image. The largest single pixel error may be more than 10% and can introduce noticeable "glitches" in regions of high-contrast transition.

One limitation of DYUV images is that "blitting" cannot be performed without careful preparation of the images. Blitting, or block image transferring, is a procedure where an image is copied or pasted into another, possibly larger, image. This is typically performed by a chip called a "blitter," or in the case of CD-I, by a highly optimized subroutine.

DYUV ENCODING OVERVIEW

Delta YUV, or DYUV, is based on YUV encoding. YUV encodes image data into a luminance (or brightness) component specified by Y, and two hue (color) components, specified by U and V. The YUV values are then encoded into DYUV values to reduce the storage required for the image by a factor of 3.

DYUV Codes Are Quantized Deltas

DYUV takes a YUV image and encodes the image based on a sequence of delta codes. Each delta code is a 4-bit index into a quantization table containing an 8-bit delta value. This delta value is added to the decoded value of the previous pixel (the pixel to the left) to determine the value of the current pixel.

A quantization table is required to translate the delta codes into delta values, because the delta steps are not linear. That is, there are more delta codes to choose from for small delta values and the delta steps become farther apart for larger deltas. The quantization table is reproduced on the next page.

Delta Code	Delta Value
0	0
1	1 (-255)
2	4 (-252)
3	9 (-247)
4	16 (-240)
5	27 (-229)
6	44 (-212)
7	79 (-177)
8	128 (-128)
9	177 (-79)
10	212 (-44)
11	229 (-27)
12	240 (-16)
13	247 (-9)
14	252 (-4)
15	255 (-1)

Table 1: Delta Code Quantization

Delta Codes Are Computed Modulo-256

For each delta code, a positive delta value and a negative delta value are possible. The use of a positive or negative delta is dependent on the occurrence of wraparound. Wraparound is the behavior caused by modulo-256 addition; if the value after applying the delta is greater than or equal to 256, then 256 is subtracted from the decoded value.

Consider the delta code 7, which has a delta value of either 79 or -177. If the previous pixel's decoded value is 106 and a delta code 7 is applied twice, the first application will result in a positive delta, and the next application will result in a negative delta:

```
(106 + 79) mod 256 = 185
(185 + 79) mod 256 = 8.   Or, 185 - 177 = 8.
```

DYUV Start Codes Can Be Chosen

Since DYUV is a sequence of deltas, the decoder must start with some absolute value to which it applies the first delta. The CD-I system allows you to select DYUV start codes once at the beginning of an image or at the beginning of each line. DYUV start codes specify absolute values for each of the Y, U, and V components.

It is important to note that the DYUV start codes are reapplied at the start of each line, even if there is only one set of start codes for the image. Each line is independent of every other line.

Two common start codes for Y are 16 and 128 which represent black and half-bright, respectively. U and V typically begin at 128, which is color-neutral (gray). However, any codes may be chosen as start codes.

DYUV Encoding Example

Consider an image with a sequence of Y values shown in the first column of the table and a Y start value of 128. The following table shows the desired delta values, the closest available delta values, the corresponding delta codes, and the resulting decoded values.

Desired Value	Desired Delta	Closest Delta	Delta Code	Decoded Value
70	-58	-44	10	84
65	-19	-16	12	68
77	+9	+9	3	77
99	+22	+27	5	104

Table 2: Values for DYUV Encoding Example

The above table demonstrates the following steps.

1. For each row, the desired value is given.
2. The desired delta is found by subtracting the previous row's decoded value from the desired value. For the first row, the start value of 128 is used as the previous decoded value.
3. The closest delta is found by examining the quantization table for the delta closest to the desired delta.
4. The delta code is the code corresponding to the chosen delta value.
5. The decoded value is found by adding the selected delta (closest delta) to the previous decoded value.

As can be seen in the example, the decoded value is not always the same as the desired value. *Error* can be thought of as the difference between the desired value and the decoded value.

Important Points about DYUV

1. Each pixel depends on the previous pixel (pixel to the left) and is represented by a delta code.

2. Since each delta is *quantized*, thereby reduced from an 8-bit value to a 4-bit value, the decoded value is not the same as the encoded value; some error is introduced.

3. There is more than one delta sequence that can be used to represent a particular line of an image. Since any encoding is only an approximation, the encoding method may be adjusted to achieve certain results or meet specific constraints.

   Note: *This article discusses delta codes and pixels and implies that each pixel has a delta code. Technically, DYUV is encoded in pixel pairs and there is a Y delta code for each pixel in the pair, but only one U and one V delta code for the pair. However, this distinction is not important for the purposes of this article.*

See the Green Book (CD-I Full Functional Specification), Sections V.3.4.1.1 and V.3.4.1.3 for more information on YUV and DYUV encoding.

SURVEY OF AVAILABLE ENCODING TOOLS

rgbtodyuv	A tool that converts RGB888 images to DYUV format. This tool uses the pixel-by-pixel conversion method. That is, the closest delta is chosen for each pixel without regard to the value of the following pixel. While this method may appear to produce a conversion with minimum error, it does not. This will be explained further in the article.
dyuvcut	A tool that prepares an image for DYUV blitting. This tool converts an RGB888 overlay image to DYUV with respect to a DYUV background image. First, the image is converted using the same method as rgbtodyuv. Then, 8 pixels are added to the right edge to be used as a "transition zone" to match the background pixels along the right edge of the overlay. Eight pixels are needed to guarantee that the required value may be obtained. DYUV blitting will be explained further in the article.
Photoshop Plug-in for DYUV Exporting	A plug-in utility that provides functionality equivalent to the **rgbtodyuv** tool and hence carries the same limitations.

METHODS FOR MINIMIZING ERROR

The pixel-by-pixel method for DYUV encoding (used by the **rgbtodyuv** tool) converts YUV to DYUV by comparing each pixel's ideal value with the previous pixel's encoded value and selecting the closest delta to provide the least error for that pixel. This process is very straightforward.

While it may not be obvious, it can be shown that this method does not produce a resulting image with minimum error. A quick explanation would be that this method may pick a delta code that is good for the current pixel, but inhibits the selection of a good code for the next pixel and later pixels.

Consider the following example:

Desired Y Sequence	16	116	195
DYUV Start Value	16		
Pixel-by-Pixel Encoding:			
Delta Value	+0	+79	+79
Decoded Value	16	95	174
Error	0	21	21
Total Error	42		
Improved Encoding:			
Delta Value	+9	+79	+79
Decoded Value	25	104	183
Error	9	12	12
Total Error	33		

Table 3: Example of "Looking Ahead" to Achieve Lower Error

The example illustrates that lower error can be achieved by *looking ahead*. Additional error was introduced to the first pixel to reduce error in the second and third pixels.

By looking ahead a certain number of pixels, the total error in an encoded image can be reduced. By looking at the entire line, the total error can be minimized.

> *Note: Looking at more than one line at a time does not provide any advantage, because the DYUV start codes are applied anew at each line.*

It is worth noting that *error* can be defined in more than one way. Certainly, whatever is to be defined as *error* may be minimized, but the visual effect of the result depends on the definition. It may be desirable to minimize the overall error as in the example, or it may be desirable to minimize the worst-case error, or perhaps minimize error in bright parts of an image.

The choice of encoding method should be based on the type of image. For images that contain computer generated boxes or text, a method that reduces worst-case error will likely provide the most visual improvement; this is due to the fact that large errors, or glitches, are readily apparent in computer generated images. For photographic images, a method that reduces overall error may be best, yet it is not likely to be a highly visible improvement.

Sample Images

The following images are samples from a real-world application:

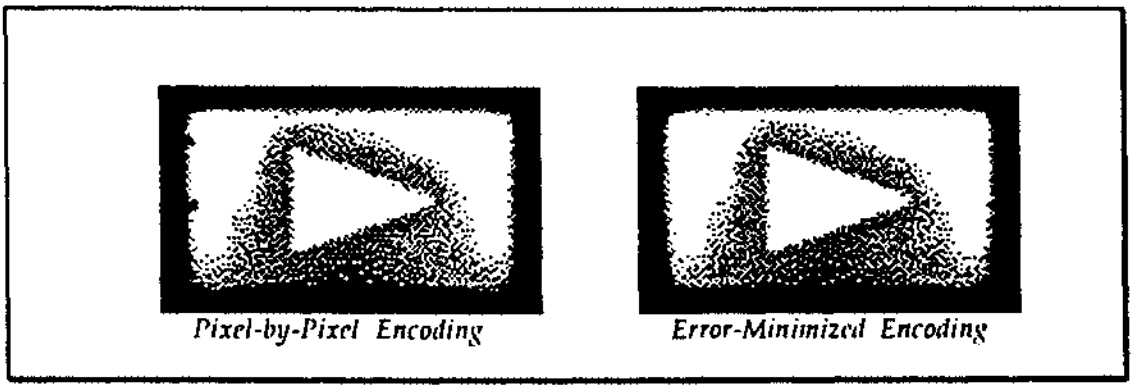

*Figure 1: Sample Images**

The jagged left edge produced by the pixel-by-pixel encoding method is not present after use of the error-minimized encoding method.

IMPROVED BLITTING

Blitting a DYUV image over a larger DYUV image requires careful image preparation. Because each DYUV pixel is encoded as a delta from the decoded value of the pixel to the left, the overlay image must conform to the decoded values of the background image on both the left and right edges. If the decoded values do not match, either the overlay image or the portion of the background image to the right of the overlay image will display incorrectly.

Normally, when encoding an image to DYUV, only the left side of the image is constrained (by the DYUV start value) and the encoding process is straightforward. However, when the DYUV image is blitted over a larger DYUV image, both the left and right sides of the image are constrained and the encoding process becomes difficult.

---

* These images were enhanced to make the differences between them visible after photocopying.

Note: Regardless of the manner of preparation, a DYUV image that is prepared for blitting is constrained to the X and Y offset that it was prepared for; that is, it cannot be moved about on the screen.

Limitations of dyuvcut Tool

The **dyuvcut** tool first encodes the DYUV image without regard to the right edge. Then, eight pixels are added to the right edge and are used as a transition zone to meet the right edge constraint. (It can be shown that a transition can be made from any value to any other in four deltas or less, and since each U and V delta applies to two pixels, eight pixels are required.) The disadvantage of this technique is that the detail for the eight background pixels to the right of the overlay is lost and the resulting appearance may be unsatisfactory.

Method for Encoding with No Transition Zone

An image may be encoded with no transition zone by picking a sequence of delta codes for each line that meets the following requirements:

1. Begin with the required left edge value.
2. End with the required right edge value.
3. Produce the minimum error.

Note: It is guaranteed that there will be such a sequence; the sequence of codes in the background meets (1) and (2) automatically and (3) is met by selecting the best sequence that meets (1) and (2).

The sequence that meets all three criteria can be found by examining all possible delta sequences.

TECHNICAL DETAILS OF IMPROVED ENCODING METHOD

Background

A need arose for a DYUV blitter that did not require the customary eight-pixel transition zone. A method was developed to search for a delta sequence that can encode a line without relying on the transition zone. The **dyuvfit** tool is an implementation of this method.

Since the search technique also minimizes the encoding error, it can be used to provide superior DYUV encoding in general. The only difference in the algorithm is that the right edge of the image is not constrained. This result is sufficiently useful that a separate tool, **rgb2dyuv**, has been implemented for general DYUV encoding.

Error Definition

For this method, error is defined as:

$$(\text{desired value} - \text{actual decoded value})^2$$

By squaring the difference, this formula places more importance on large value errors and tends to ignore small errors that are likely to be invisible. Mathematically, this formula minimizes the standard deviation of the actual values with respect to the desired values.

Search Algorithm

The important characteristic of this method is that it performs an exhaustive search to guarantee minimal error in the encoded image. If you consider that for each pixel, 16 deltas are possible, then the search problem can be viewed as searching a tree where each node has 16 children and there are as many levels in the tree as there are pixels on the scan line. An exhaustive search of this tree would examine 16^n possible paths (where n is the number of pixels per line) which would *seem* uncomputable.

What makes the exhaustive search possible is the observation that only 256 nodes are possible at each level since the range of each YUV component is 0 to 255. A search strategy that takes advantage of this fact is demonstrated in the following pseudo-code:

```
• For each pixel (p) on the line do:
    • For each of the 256 possible values (v)
      do:
        • Calculate the error (e) for the
          selected p and v
        • Find the value of the previous pixel
          (pp) that can reach the value v by one
          of the 16 available deltas and has the
          least accumulated error.
        • Add the error e to the accumulated
          error for pp and remember both the new
          accumulated error and the value of pp
          in an array.
• For General DYUV Encoding: Find the value for
  the last pixel on the line that has the least
  accumulated error.
• For DYUV Blitting Preparation: Start on the
  right edge with the value that is required by
  the background.
• Extract the least error path by following the
  pp values from right to left along the line to
  locate the previous pixel deltas.
```

Figure 2: Pseudo-Code Example of Search Strategy

NEW TOOLS

rgb2dyuv

The **rgb2dyuv** tool converts an RGB888 image to DYUV format using the previously described error minimization method. Input files must be in IFF format.

> *Note: The exhaustive search method is more than 60 times slower than the pixel-by-pixel method and can take several minutes to convert a full-screen image on a typical workstation.*

The **rgb2dyuv** tool also provides the following features:

- Optional CCIR compression
- Quick mode (using pixel-by-pixel method)

dyuvfit

The **dyuvfit** tool accepts a background image in DYUV format, an overlay image in RGB888 format, and a desired X and Y offset. It then produces a DYUV image the size of the overlay image that is suitable for blitting over the background image.

dyuvfit also provides the following features:

- Optional CCIR compression
- Optional Merged output (useful for previewing)
- Overlay width extension to preserve last pixel

> *Note: The **dyuvfit** tool cannot preserve the last pixel on each line of the overlay unless the width of the overlay is increased. Since the last pixel must match the background exactly, the value of the pixel in the overlay is ignored. In order to preserve that pixel, the width of the overlay must be increased. **dyuvfit** is capable of performing this extension automatically.*

Availability

The **rgb2dyuv** and **dyuvfit** tools are available from PIMA Developer Services in both source code and Sun executable form. The source is in C and should be portable.