NAME

Quantize - ImageMagick's color reduction algorithm.

SYNOPSIS

#include <magick.h>

DESCRIPTION

This document describes how **ImageMagick** performs color reduction on an image. To fully understand this document, you should have a knowledge of basic imaging techniques and the tree data structure and terminology.

For purposes of color allocation, an image is a set of *n* pixels, where each pixel is a point in RGB space. RGB space is a 3-dimensional vector space, and each pixel, p_i, is defined by an ordered triple of red, green, and blue coordinates, (r_i, g_i, b_i).

Each primary color component (red, green, or blue) represents an intensity which varies linearly from 0 to a maximum value, c_{max}, which corresponds to full saturation of that color. Color allocation is defined over a domain consisting of the cube in RGB space with opposite vertices at (0,0,0) and (c_{max},c_{max},c_{max}). **ImageMagick** requires $c_{max} = 255$.

The algorithm maps this domain onto a tree in which each node represents a cube within that domain. In the following discussion, these cubes are defined by the coordinate of two opposite vertices: The vertex nearest the origin in RGB space and the vertex farthest from the origin.

The tree's root node represents the the entire domain, (0,0,0) through (c_{max},c_{max},c_{max}). Each lower level in the tree is generated by subdividing one node's cube into eight smaller cubes of equal size. This corresponds to bisecting the parent cube with planes passing through the midpoints of each edge.

The basic algorithm operates in three phases: **Classification, Reduction**, and **Assignment**. **Classification** builds a color description tree for the image. **Reduction** collapses the tree until the number it represents, at most, is the number of colors desired in the output image. **Assignment** defines the output image's color map and sets each pixel's color by reclassification in the reduced tree. Our goal is to minimize the numerical discrepancies between the original colors and quantized colors. To learn more about quantization error, see MEASURING COLOR REDUCTION ERROR later in this document.

Classification begins by initializing a color description tree of sufficient depth to represent each possible input color in a leaf. However, it is impractical to generate a fully-formed color description tree in the classification phase for realistic values of c_{max}. If color components in the input image are quantized to *k*-bit precision, so that $c_{max} = 2^k-1$, the tree would need *k* levels below the root node to allow representing each possible input color in a leaf. This becomes prohibitive because the tree's total number of nodes is

$$\sum_{i=1}^{k} 8k$$

A complete tree would require 19,173,961 nodes for $k = 8$, $c_{max} = 255$. Therefore, to avoid building a fully populated tree, **ImageMagick**: (1) Initializes data structures for nodes only as they are needed; (2) Chooses a maximum depth for the tree as a function of the desired number of colors in the output image (currently $log_4(colormap\ size)+2$). A tree of this depth generally allows the best representation of the source image with the fastest computational speed and the least amount of memory. However, the default depth is inappropriate for some images. Therefore, the caller can request a specific tree depth.

For each pixel in the input image, classification scans downward from the root of the color description tree. At each level of the tree, it identifies the single node which represents a cube in RGB space containing the pixel's color. It updates the following data for each such node:

n_1: Number of pixels whose color is contained in the RGB cube which this node represents;

n_2: Number of pixels whose color is not represented in a node at lower depth in the tree; initially, $n_2 = 0$ for all nodes except leaves of the tree.

S_r, S_g, S_b: Sums of the red, green, and blue component values for all pixels not classified at a lower depth. The combination of these sums and n_2 will ultimately characterize the mean color of a set of pixels represented by this node.

E: The distance squared in RGB space between each pixel contained within a node and the nodes' center. This represents the quantization error for a node.

Reduction repeatedly prunes the tree until the number of nodes with $n_2 > 0$ is less than or equal to the maximum number of colors allowed in the output image. On any given iteration over the tree, it selects those nodes whose *E* value is minimal for pruning and merges their color statistics upward. It uses a pruning threshold, E_p, to govern node selection as follows:

```
E  = 0
 p
while number of nodes with (n  > 0) > required maximum number of colors
                            2
   prune all nodes such that E <= E
                              p
   Set E  to minimum E in remaining nodes
       p
```

This has the effect of minimizing any quantization error when merging two nodes together.

When a node to be pruned has offspring, the pruning procedure invokes itself recursively in order to prune the tree from the leaves upward. The values of n_2, S_r, S_g, and S_b in a node being pruned are always added to the corresponding data in that node's parent. This retains the pruned node's color characteristics for later averaging.

For each node, n_2 pixels exist for which that node represents the smallest volume in RGB space containing those pixel's colors. When $n_2 > 0$ the node will uniquely define a color in the output image. At the beginning of reduction, $n_2 = 0$ for all nodes except the leaves of the tree which represent colors present in the input image.

The other pixel count, n_1, indicates the total number of colors within the cubic volume which the node represents. This includes $n_1 - n_2$ pixels whose colors should be defined by nodes at a lower level in the tree.

Assignment generates the output image from the pruned tree. The output image consists of two parts: (1) A color map, which is an array of color descriptions (RGB triples) for each color present in the output image; (2) A pixel array, which represents each pixel as an index into the color map array.

First, the assignment phase makes one pass over the pruned color description tree to establish the image's color map. For each node with $n_2 > 0$, it divides S_r, S_g, and S_b by n_2. This produces the mean color of all pixels that classify no lower than this node. Each of these colors becomes an entry in the color map.

Finally, the assignment phase reclassifies each pixel in the pruned tree to identify the deepest node containing the pixel's color. The pixel's value in the pixel array becomes the index of this node's mean color in the color map.

Empirical evidence suggests that distances in color spaces such as YUV, or YIQ correspond to perceptual color differences more closely than do distances in RGB space. These color spaces may give better results when color reducing an image. Here the algorithm is as described except each pixel is a point in the alternate color space. For convenience, the color components are normalized to the range 0 to a maximum value, c_{max}. The color reduction can then proceed as described.

MEASURING COLOR REDUCTION ERROR

Depending on the image, the color reduction error may be obvious or invisible. Images with high spatial frequencies (such as hair or grass) will show error much less than pictures with large smoothly shaded areas (such as faces). This is because the high-frequency contour edges introduced by the color reduction process are masked by the high frequencies in the image.

To measure the difference between the original and color reduced images (the total color reduction error), **ImageMagick** sums over all pixels in an image the distance squared in RGB space between each original pixel value and its color reduced value. **ImageMagick** prints several error measurements including the mean error per pixel, the normalized mean error, and the normalized maximum error.

The normalized error measurement can be used to compare images. In general, the closer the mean error is to zero the more the quantized image resembles the source image. Ideally, the error should be perceptually-based, since the human eye is the final judge of quantization quality.

These errors are measured and printed when **-verbose** and **-colors** *are specified on the command line:*

mean error per pixel:
is the mean error for any single pixel in the image.

normalized mean square error:
is the normalized mean square quantization error for any single pixel in the image.

This distance measure is normalized to a range between 0 and 1. It is independent of the range of red, green, and blue values in the image.

normalized maximum square error:
is the largest normalized square quantization error for any single pixel in the image.

This distance measure is normalized to a range between 0 and 1. It is independent of the range of red, green, and blue values in the image.

SEE ALSO

display(1), animate(1), mogrify(1), import(1), miff(5)

COPYRIGHT

ACKNOWLEDGEMENTS

Paul Raveling, USC Information Sciences Institute, for the original idea of using space subdivision for the color reduction algorithm. With Paul’s permission, this document is an adaptation from a document he wrote.

AUTHORS

John Cristy, E.I. du Pont de Nemours and Company Incorporated