CAP5015: Data Compression
Fall 2005
Programming Assignment # 2
Due: Nov 2,2005

Apply DCT transform on the image, and quantize the transformed coefficients using the LBG vector quantization algorithm.

Steps: Subdivide the image into 1x8 blocks, apply 8-point DCT transform. After the transform, we have 8 coefficients for each block. Combine the 8 coefficients into a vector, and apply LBG to quantize these vectors. Also try 4x4 blocks (expand it as a 1x16 vecotr), and 8x1 blocks. Try different code book of size 4, 8, 16, 32, 64, 128, 256, 512. Try the program on two provided images.

Requirement:

1. Implement the LBG algorithm by yourself. Downloading the LBG code from Internet is not permitted.
2. Programming using C/C++ is strongly suggested. Java/Matlab is also acceptable.
3. Use the given images (can be downloaded in the course website). Two PPM images are provided. The format of the PPM file is described in the appendix. You may download fineview (freeware, http://www.fine-view.com/) to view the image. You may search the web to download source code to read/write PPM file.
4. Deliverable: short report, problems, results, program code, etc. Compare the size of the compressed images with code book of different size. Include the compressed images (decoded using IDCT) in the report. Briefly discuss the advantage/ disadvantage of the LBG algorithm. Compare the encoding algorithm used in this project with the standard JPEG. Provide the exe file (windows, or eola).
5. Email the source code, output image, exe file and the report to wsun@cs.ucf.edu.

Input image:

Appendix
(from http://astronomy.swin.edu.au/~pbourke/dataformats/ppm/).

PPM / PGM / PBM image files

Written by Paul Bourke July 1997

This note describes the format of PPM (Portable PixMap), PGM (Portable GreyMap), PBM (Portable BitMap) files. These formats are a convenient (simple) method of saving image data, they are equally easy to read in ones own applications. Unfortunately the standards aren't always implemented as well as they could.

These formats were popularised by the pbmplus image toolkit otherwise known as the "enhanced portable bitmap toolkit". The description of the toolkit from the man page is given below

DESCRIPTION

The pbmplus toolkit allows conversions between image files of different format. By means of using common intermediate formats, only 2*N conversion filters are required to support N distinct formats, instead of the N^2 which would be required to convert directly between any one format and any other. The package also includes simple tools for manipulating portable bitmaps.

The package consists of four upwardly compatible sections:

pbm Supports monochrome bitmaps (1 bit per pixel).

pgm Supports greyscale images. Reads either pbm or pgm formats and writes pgm format.

ppm Supports full-color images. Reads either pbm, pgm, or ppm formats, writes ppm format.

pnm Supports content-independent manipulations on any of the three formats listed above, as well as external formats having multiple types. Reads either pbm, pgm, or ppm formats, and generally writes the same type as it read (whenever a pnm tool makes an exception and "promotes" a file to a higher format, it informs the user).

PPM

A PPM file consists of two parts, a header and the image data. The header consists of at least three parts normally delinineated by carriage returns and/or linefeeds but the PPM specification only requires white space. The first "line" is a magic PPM identifier, it can be "P3" or "P6" (not including the double quotes!). The next line consists of the width and height of the image as ascii numbers. The last part of the header gives the maximum value of the colour components for the pixels, this allows the format to describe more than single byte (0..255) colour values. In addition to the above required lines, a comment can be placed anywhere with a "#" character, the comment extends to the end of the line.

The following are all valid PPM headers.

Header example 1

```
P6 1024 788 255
```

Header example 2

```
P6
1024 788
# A comment
255
```

Header example 3

```
P3
1024 # the image width
788 # the image height
# A comment
1023
```

The format of the image data itself depends on the magic PPM identifier. If it is "P3" then the image is given as ascii text, the numerical value of each pixel ranges from 0 to the maximum value given in the header. The lines should not be longer than 70 characters.

PPM example 4

```
P3
# example from the man page
4 4
15
 0  0  0    0  0  0    0  0  0   15  0 15
 0  0  0    0 15  7    0  0  0    0  0  0
 0  0  0    0  0  0    0 15  7    0  0  0
15  0 15    0  0  0    0  0  0    0  0  0
```

If the PPM magic identifier is "P6" then the image data is stored in byte format,

one byte per colour component (r, g, b). Comments can only occur before the last field of the header and only one byte may appear after the last header field, normally a carriage return or line feed. "P6" image files are obviously smaller than "P3" and much faster to read. Note that "P6" PPM files can only be used for single byte colours.

While not required by the format specification it is a standard convention to store the image in top to bottom, left to right order. Each pixel is stored as a byte, value 0 == black, value 255 == white. The components are stored in the "usual" order, red - green - blue.

PGM

This format is identical to the above except it stores greyscale information, that is, one value per pixel instead of 3 (r, g, b). The only difference in the header section is the magic identifiers which are "P2" and "P5", these correspond to the ascii and binary form of the data respectively.

PGM example

An example of a PGM file of type "P2" is given below

```
P2
24 7
15
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  3  3  3  3  0  0  7  7  7  7  0  0 11 11 11 11  0  0 15 15 15 15  0
0  3  0  0  0  0  0  7  0  0  0  0  0 11  0  0  0  0  0 15  0  0 15  0
0  3  3  3  0  0  0  7  7  7  0  0  0 11 11 11  0  0  0 15 15 15 15  0
0  3  0  0  0  0  0  7  0  0  0  0  0 11  0  0  0  0  0 15  0  0  0  0
0  3  0  0  0  0  0  7  7  7  7  0  0 11 11 11 11  0  0 15  0  0  0  0
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
```