图像处理实验报告

郭一隆 (2013011189)

August 24, 2015

Contents

1 基础知识 **3**

2 图像压缩编码 **5**

List of Figures

1.1 在大礼堂中心绘制红圆 4
1.2 国际象棋蒙版 5

List of Tables

1.1 图像处理工具箱函数概览 (部分) 3
2.1 WinRar 压缩文本文件 6
2.2 JpegCoeff.mat 中所含数据 6
2.3 亮度直流分量预测误差的 Category 及其 Huffman 编码 7
2.4 亮度 AC 分量的 Run/Size 及其 Huffman 编码 (部分) 8

List of Source Codes

1.1 draw_circle.m 4
1.2 chess_mask.m 5
2.1 验证变换域预处理 10
2.2 mse.m 10
2.3 mydct2.m 11

1 基础知识

在 MATLAB 中，像素值用 uint8 类型表示，参与浮点数运算前需要转成 double 型。Section 1 中“测试图像”指的是hall.mat中的**彩色图像**。

1. MATLAB 提供了图像处理工具箱，在命令窗口输入 help images 可查看该工具箱内的所有函数。请阅读并大致了解这些函数的基本功能。

Table 1.1: 图像处理工具箱函数概览 (部分)

函数名	**功能**
imshow	在 figure 中显示图像
rgb2gray	将彩色图像转换为灰度值图像
imwrite	将图像矩阵写入文件

2. 利用 MATLAB 提供的 Image file I/O 函数分别完成以下处理：

   (a) 以测试图像的中心为圆心，图像的长和宽中较小值的一半为半径画一个红颜色的圆；

   思路：利用 meshgrid 函数生成行列索引矩阵 I，J，将圆内部的像素点标为**逻辑 1**，再利用逻辑索引将测试图像圆内的部分替换为红色像素点。

```m
%% Load images
load('resource/hall.mat');
imwrite(hall_color,'images/hall_color.png');
hall_color = double(hall_color);

%% Draw red circle
[height,width,~] = size(hall_color);
center = [(1+height)/2,(1+width)/2];
radius = min(height,width)/2;
[J,I] = meshgrid(1:width,1:height);
% <height-by-width matrix> I: I(x,y) equals x
% <height-by-width matrix> J: J(x,y) equals y
area = ((I-center(1)).^2 + (J-center(2)).^2 <= radius^2);
% area equals 1 @ point inside circle

cell = mat2cell(hall_color,ones(1,height),ones(1,width),3);
cell(area) = {reshape([255,0,0],1,1,3)};
hall_color_red_circle = cell2mat(cell);

%% Write image
hall_color_red_circle = uint8(hall_color_red_circle);
imwrite(hall_color_red_circle,'images/hall_color_red_circle.png');
```

Listing 1.1: `draw_circle.m`

(a) 处理前　　(b) 处理后

Figure 1.1: 在大礼堂中心绘制红圆

(b) 将测试图像涂成国际象棋状的“黑白格”的样子，其中“黑”即黑色，“白”则意味着**保留原图**。

思路：`chess_mask` 函数提供棋盘行列数接口，计算出每块的大小，同样利用 `meshgrid` 函数确定出 `black_mask` 的位置，将图像对应位置赋为黑色。

```m
function masked_image = chess_mask(image,Nrow,Ncol)

image = double(image);

[height,width,~] = size(image);
grid_size = [ceil(height/Nrow),ceil(width/Ncol)];
[J,I] = meshgrid(1:width,1:height);
black_mask = (xor(mod(ceil(I/grid_size(1)),2),...
                  mod(ceil(J/grid_size(2)),2))==0);

cell = mat2cell(image,ones(1,height),ones(1,width),3);
cell(black_mask) = {reshape([0,0,0],1,1,3)};
masked_image = cell2mat(cell);

masked_image = uint8(masked_image);

end
```

Listing 1.2: `chess_mask.m`

按如下代码生成 64 格和 32 格棋盘蒙版

```
>> imwrite(chess_mask(hall_color,8,8),'images/hall_color_masked_8_8.png')
>> imwrite(chess_mask(hall_color,4,8),'images/hall_color_masked_4_8.png')
```

(a) 8×8 蒙版　　(b) 4×8 蒙版

Figure 1.2: 国际象棋蒙版

用看图软件浏览上述生成图片，预览效果如图1.1和图1.2，达到预期效果。

2 图像压缩编码

熵编码压缩比：

根据经验，对于文本文件，重复性越高，则压缩比越高，测试如下表

Table 2.1: WinRar 压缩文本文件

文件名	文本内容 (matlab 写入文件)	压缩前大小	压缩后大小	压缩比
high.txt	repmat(['1'],1,10000)	10000bytes	101bytes	99.01
normal.txt	int2str(2^1000)	302bytes	106bytes	2.85
low.txt	'ghjkl;'	6bytes	77bytes	0.08

本章练习题所用数据均可由“JpegCoeff.mat”导入，其内容如表2.2所示。本章练习题中“测试图像”指的是hall.mat中的**灰度图像**。

Table 2.2: JpegCoeff.mat 中所含数据

变量名	含义	说明
QTAB	DCT 系数的量化步长矩阵，式 (2.1)	
DCTAB	DC 系数预测误差的 Category 码本，表2.3	每行对应一个 Category，第一列对应 Huffman 编码的长度 L，随后 L 列对应该码字，再后全零为填充物
ACTAB	AC 系数的 (Run/Size) 的码本，完整的表2.4	每行对应一个 (Run/Size)，第一列表示 Run，第二列表示 Size，第三列表示该 (Run/Size) 对应的 Huffman 编码的长度 L，随后 L 列对应该码字，再后全零为填充物

$$Q = \begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix} \tag{2.1}$$

Table 2.3: 亮度直流分量预测误差的 Category 及其 Huffman 编码

预测误差	**Category**	**Huffman 编码**
0	0	00
-1, 1	1	010
-3, -2, 2, 3	2	011
-7, ⋯, -4, 4, ⋯, 7	3	100
-15, ⋯, -8, 8, ⋯, 15	4	101
-31, ⋯, -16, 16, ⋯, 31	5	110
-63, ⋯, -32, 32, ⋯, 63	6	1110
-127, ⋯, -64, 64, ⋯, 127	7	11110
-255, ⋯, -128, 128, ⋯, 255	8	111110
-511, ⋯, -256, 256, ⋯, 511	9	1111110
-1023, ⋯, -512, 512, ⋯, 1023	10	11111110
-2047, ⋯, -1024, 1024, ⋯, 2047	11	111111110

Table 2.4: 亮度 AC 分量的 Run/Size 及其 Huffman 编码 (部分)

Run/Size	码长	码字	Run/Size	码长	码字
0/0(EOB)	4	1010			
0/1	2	00	4/1	6	111011
0/2	2	01	4/2	10	1111111000
0/3	3	100	4/3	16	1111111110010111
0/4	4	1011	4/4	16	1111111110011000
0/5	5	11010	4/5	16	1111111110011001
0/6	6	111000	4/6	16	1111111110011010
0/7	7	1111000	4/7	16	1111111110011011
0/8	10	1111110110	4/8	16	1111111110011100
0/9	16	1111111110000010	4/9	16	1111111110011101
0/A	16	1111111110000011	4/A	16	1111111110011110
1/1	4	1100	8/1	8	11111010
1/2	6	111001	8/2	15	111111111000000
1/3	7	1111001	8/3	16	1111111110110111
1/4	9	111110110	8/4	16	1111111110111000
1/5	11	11111110110	8/5	16	1111111110111001
1/6	16	1111111110000100	8/6	16	1111111110111010
1/7	16	1111111110000101	8/7	16	1111111110111011
1/8	16	1111111110000110	8/8	16	1111111110111100
1/9	16	1111111110000111	8/9	16	1111111110111101
1/A	16	1111111110001000	8/A	16	1111111110111110
			F/0(ZRL)	11	11111111001
2/1	5	11011	F/1	16	1111111111110101
2/2	8	11111000	F/2	16	1111111111110110
2/3	10	1111110111	F/3	16	1111111111110111
2/4	16	1111111110001001	F/4	16	1111111111111000
2/5	16	1111111110001010	F/5	16	1111111111111001
2/6	16	1111111110001011	F/6	16	1111111111111010
2/7	16	1111111110001100	F/7	16	1111111111111011
2/8	16	1111111110001101	F/8	16	1111111111111100
2/9	16	1111111110001110	F/9	16	1111111111111101
2/A	16	1111111110001111	F/A	16	1111111111111110

1. 图像的预处理是将每个像素灰度值减去 128，这个步骤是否可以在变换域进行？

   变换域与时域 (或空域) 的对应关系为

$$\mathbf{C} = \sum_{x=0}^{N-1}\sum_{y=0}^{N-1} P_{x,y}\mathbf{D}^{(2)}(x,y) \tag{2.2}$$

   其中 $\mathbf{D}^{(2)}(x,y)$ 表示二维 DCT 的第 (x,y) 个基矩阵，其第 (i,j) 个分量为

$$\mathbf{D}_{i,j}^{(2)}(x,y) = \alpha_i\alpha_j \cos\frac{i(2x+1)\pi}{2N}\cos\frac{j(2y+1)\pi}{2N} \tag{2.3}$$

   对原图像进行预处理：

$$\bar{P}_{x,y} = P_{x,y} - 128$$

   则有

$$\begin{aligned}\bar{\mathbf{C}} &= \sum_{x=0}^{N-1}\sum_{y=0}^{N-1}\bar{P}_{x,y}\mathbf{D}^{(2)}(x,y)\\ &= \mathbf{C} - 128\sum_{x=0}^{N-1}\sum_{y=0}^{N-1}\mathbf{D}^{(2)}(x,y)\end{aligned}$$

   而

$$\begin{aligned}\sum_{x=0}^{N-1}\sum_{y=0}^{N-1}\mathbf{D}_{i,j}^{(2)}(x,y) &= \alpha_i\alpha_j\sum_{x=0}^{N-1}\cos\frac{i(2x+1)\pi}{2N}\sum_{y=0}^{N-1}\cos\frac{j(2y+1)\pi}{2N}\\ &= \begin{cases}\alpha_i\alpha_j N^2 = N, & i=j=0\\ 0, & elsewhere\end{cases}\end{aligned}$$

   若要在**变换域**进行预处理，则对应关系为

$$\begin{aligned}\bar{\mathbf{C}}_{i,j} &= \mathbf{C}_{i,j} - 128N, i=j=0\\ \bar{\mathbf{C}}_{i,j} &= \mathbf{C}_{i,j}, \qquad elsewhere\end{aligned}$$

   因此，这个步骤可以在变换域进行，只需把变换域直流分量减去 $128N$ 即可，取 hall_gray 左上角 8×8 验证如下：

```
>> A = hall_gray(1:8,1:8);
>> Y1 = dct2(A-128);    % preprocess on Space Domain
>> Y2 = dct2(A);
>> Y2(1,1) = Y2(1,1) - 128*8;   % preprocess on Transform Domain
```

```
>> mse(Y1,Y2)

ans =

   5.5804e-27
```

Listing 2.1: 验证变换域预处理

其中 `mse` 为自定义函数，用于计算两个矩阵的均方误差：

```
function [result] = mse(A,B)
% Mean Square Error

result = sum(sum((A-B).^2))/size(A,1)/size(A,2);

end
```

Listing 2.2: `mse.m`

由运行代码2.1中得到的 `mse` 结果可知，在变换域减去一定直流分量与预处理原图像是**等价**的。

2. 根据下式2.4自行编程实现二维 `DCT`：

$$\mathbf{C} = \mathbf{DPD}^T \tag{2.4}$$

其中

$$\mathbf{D} = \sqrt{\frac{2}{N}} \begin{bmatrix} \sqrt{\frac{1}{2}} & \sqrt{\frac{1}{2}} & \cdots & \sqrt{\frac{1}{2}} \\ \cos\frac{\pi}{2N} & \cos\frac{3\pi}{2N} & \cdots & \cos\frac{(2N-1)\pi}{2N} \\ \vdots & \vdots & \ddots & \vdots \\ \cos\frac{(N-1)\pi}{2N} & \cos\frac{(N-1)3\pi}{2N} & \cdots & \cos\frac{(N-1)(2N-1)\pi}{2N} \end{bmatrix}$$

```
function C = mydct2(P)
% 2-dimension DCT

P = double(P);
N = size(P,1);  % Assuming that P is a square matrix

[J,I] = meshgrid(1:2:2*N-1, 0:N-1);

D = sqrt(2/N) * cos(I.*J*pi/2/N);
D(1,:) = ones(1,N)/sqrt(N);

C = D*P*D.';

end
```

Listing 2.3: `mydct2.m`

仍取左上角 8×8 色块进行测试：

```
>> A = hall_gray(1:8,1:8);
>> mse(mydct2(A),dct2(A))

ans =

   2.6817e-26
```

说明 `mydct2` 与 `dct2` 计算结果相同。