

MATLAB 图像处理大作业

实验报告

姓名：马洋








学号：2015011181

班级：无 58

日期：2017 年 9 月 12 日














✧ 代码清单

1. 脚本

 image_1a	2017/8/15 14:12	M 文件	1 KB
 image_1b	2017/8/15 15:24	M 文件	1 KB
 image_2_1	2017/8/21 15:29	M 文件	1 KB
 image_2_2	2017/8/22 10:49	M 文件	1 KB
 image_2_3	2017/8/22 11:19	M 文件	1 KB
 image_2_4	2017/8/22 11:19	M 文件	1 KB
 image_2_5	2017/9/4 10:29	M 文件	1 KB
 image_2_7	2017/9/4 13:54	M 文件	1 KB
 image_2_8	2017/9/5 19:11	M 文件	1 KB
 image_2_9_coder	2017/9/5 19:11	M 文件	2 KB
 image_2_11_decoder	2017/9/5 19:11	M 文件	2 KB
 image_3_1	2017/9/8 9:56	M 文件	1 KB
 image_3_2a	2017/9/8 14:44	M 文件	4 KB
 image_3_2b	2017/9/11 10:07	M 文件	5 KB
 image_3_2c	2017/9/11 10:23	M 文件	4 KB
 image_4_1a	2017/9/11 11:23	M 文件	1 KB
 image_4_1b	2017/9/11 14:31	M 文件	1 KB
 image_4_2	2017/9/11 15:03	M 文件	2 KB

格式说明：例如 image_3_2a.m 就是第三大题的第二小题中的 a 问；

2. 函数

 AC_coding	2017/9/5 19:11	M 文件	2 KB
 AC_decoding	2017/9/5 19:11	M 文件	2 KB
 boxfilter	2017/9/11 17:44	M 文件	1 KB
 complement	2017/9/4 16:22	M 文件	1 KB
 DC_coding	2017/9/5 19:11	M 文件	1 KB
 DC_decoding	2017/9/5 14:19	M 文件	2 KB
 distance	2017/9/11 16:03	M 文件	1 KB
 eigenvector	2017/9/11 15:47	M 文件	1 KB
 i_zigzag	2017/9/5 15:15	M 文件	1 KB
 JPEG	2017/9/8 13:45	M 文件	3 KB
 matrix2str	2017/9/5 11:14	M 文件	1 KB
 str2matrix	2017/9/8 18:05	M 文件	1 KB
 zigzag	2017/9/4 14:23	M 文件	1 KB

2.1

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？请在测试图像中截取一块验证你的结论。

	1	2	3	4	5	6	7	8
1	-1.1369e-13	5.6843e-14	-2.8422e-14	-2.8422e-14	1.9984e-14	4.2633e-14	1.1369e-13	0
2	1.0658e-14	-3.3751e-14	2.0428e-14	-1.2879e-14	-9.7700e-15	1.1102e-14	2.3093e-14	3.7303e-14
3	0	8.8818e-15	1.0658e-14	1.2434e-14	-7.3275e-15	-8.2157e-15	-1.3212e-14	-1.1102e-15
4	-1.4433e-14	7.1054e-15	3.6637e-15	-1.9817e-14	-4.8850e-15	-2.5757e-14	9.1038e-15	1.2490e-14
5	-1.2434e-14	-8.8818e-16	2.3537e-14	3.5527e-15	2.4425e-15	-7.7716e-16	-1.4433e-15	-2.8866e-15
6	1.4932e-14	9.1038e-15	-1.5543e-14	-5.5511e-15	-2.0206e-14	9.9920e-15	2.7978e-14	-1.5155e-14
7	4.8850e-15	-8.2157e-15	-9.4369e-16	4.5963e-14	4.8850e-15	-7.1054e-15	-1.5543e-14	9.8255e-15
8	1.5099e-14	-1.1435e-14	-9.3259e-15	4.4409e-15	4.8850e-15	-3.9968e-15	3.7748e-15	1.3683e-14

理论上，由于 DCT 变换具有线性性，所以 $DCT(P-128)=DCT(P)-DCT(128)$ ；

由实验结果可知， $diff=DCT(P-128)-(DCT(P)-DCT(128))$ 约等于 0，与理论相符。

2.2

请编程实现二维 DCT，并和 MATLAB 自带的库函数 `dct2` 比较是否一致。

	1	2	3	4	5	6	7	8
1	4.5475e-13	6.3949e-14	-2.3803e-13	1.5099e-13	-8.5265e-14	5.3957e-13	-5.4756e-13	-1.5876e-13
2	1.4211e-13	2.7534e-14	-3.9524e-14	-2.6645e-15	-3.1086e-15	-7.8826e-15	-1.5432e-14	-4.6629e-14
3	-1.9007e-13	-3.5527e-15	-1.8874e-14	-5.3291e-15	7.3275e-15	1.0880e-14	4.7740e-15	-2.8866e-15
4	1.3967e-13	-9.7700e-15	-9.8810e-15	1.9096e-14	1.9984e-15	1.9540e-14	-2.0539e-15	-7.8271e-15
5	8.7041e-14	4.4409e-15	-1.8652e-14	1.3323e-15	-1.9984e-15	7.8271e-15	3.1641e-15	-2.1094e-15
6	5.9242e-13	-1.3323e-15	7.5495e-15	6.4393e-15	1.0658e-14	-8.8818e-15	-2.7311e-14	6.6058e-15
7	-3.7903e-13	-1.9984e-15	-1.4155e-14	-4.2133e-14	-1.0936e-14	7.1054e-15	1.4544e-14	4.5519e-15
8	-1.8141e-13	1.0658e-14	1.3323e-15	2.7756e-15	-8.2157e-15	3.3307e-15	-9.4369e-16	-1.0936e-14

基本一致。

2.3

如果将 DCT 系数矩阵中右侧四列的系数全部置零，逆变换后的图像会发生什么变化？选取一块图验证你的结论。如果左侧的四列置零呢？

➤ 理论推导：

DCT 变换后的系数矩阵由左上至右下蕴含着由低频到高频分量的信息，所以若将最右四列置零，会削弱原图中的高频分量，若将左侧四列置零，将削弱原图中的低频分量。

➤ 实验结果：

右侧四列置零：



左侧四列置零：



可以看出，右侧四列置零图像变化不大，而左侧四列置零后的图像变得不清晰，原因在于人眼是一个低通滤波器，对于信息的摄取主要集中在低频分量，所以可知，实验结果与理论相符。

2.4

若对 DCT 系数分别做转置、旋转 90 度和旋转 180 度操作 (rot90)，逆变换后恢复的图像有何变化？选取一块图验证你的结论。

2.4.1 转置

由 $C' = (D * P * D')' = D * P' * D'$ ，可知对 C 做转置等价于对原图做转置；

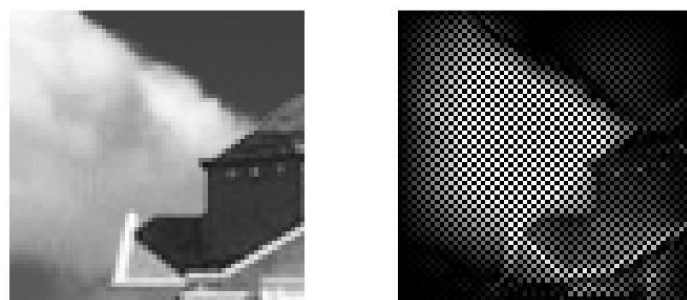
实验结果：与理论相符；



2.4.2 旋转 90°



2.4.3 旋转 180°



2.5

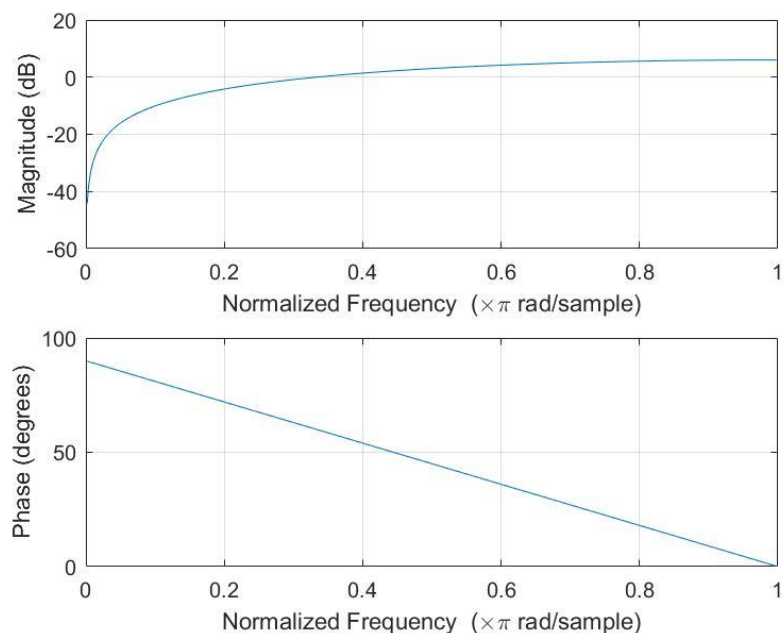
如果认为差分编码是一个系统，请绘出这个系统的频率响应，说明它是一个_____（低通、高通、带通、带阻）滤波器。DC 系数先进行差分编码再进行熵编码，说明 DC 系数的_____频率分量更多。21

系统差分方程： $s(n) = e(n) - e(n-1)$ ；

系统传递函数： $H(z) = 1 - z^{-1}$ ；

系统分子向量： $b = [1 \ -1]$ ；

系统分母向量： $a = 1$ ；



可见，差分编码系统是一个高通滤波器，说明 DC 系数低频分量更多，滤除一部分低频分量可降低熵编码所需位数。

2.6

DC 预测误差的取值和 Category 值有何关系？如何利用预测误差计算出其 Category？

若 DC 预测误差为零，则 Category 值为 0；

若 DC 预测误差大于零，则 Category 值为其二进制 bit 数；

若 DC 预测误差小于零，则 Category 值为其 1 补码的 bit 数；

2.7

你知道哪些实现 Zig-Zag 扫描的方法？请利用 MATLAB 的强大功能设计一种最佳方法。

实现 zigzag 的主要思路是实现一个 zigzag 的轨迹表；

我的实现方法：手动打表

```

function b = zigzag(A)
b = zeros(1, 64);
zz = [1 2 6 7 15 16 28 29;
      3 5 8 14 17 27 30 43;
      4 9 13 18 26 31 42 44;
      10 12 19 25 32 41 45 54;
      11 20 24 33 40 46 53 55;
      21 23 34 39 47 52 56 61;
      22 35 38 48 51 57 60 62;
      36 37 49 51 58 59 63 64;];
for i = 1:8
    for j = 1:8
        b(zz(i, j)) = A(i, j);
    end
end
end

```

网上找到的比较好的方法：计算生成

```

#include <stdio.h>
int main()
{
    int N;
    int s, i, j;
    int squa;
    scanf("%d", &N);
    /* 分配空间 */
    int **a = malloc(N * sizeof(int *));
    if(a == NULL)
        return 0;
    for(i = 0; i < N; i++) {
        if((a[i] = malloc(N * sizeof(int))) == NULL) {
            while(--i >= 0)
                free(a[i]);

            free(a);
            return 0;
        }
    }
    /* 数组赋值 */
    squa = N*N;
    for(i = 0; i < N; i++)
        for(j = 0; j < N; j++) {
            s = i + j;
            if(s < N)
                a[i][j] = s*(s+1)/2 + (((i+j)%2 == 0)? i : j);
            else {
                s = (N-1-i) + (N-1-j);
                a[i][j] = squa - s*(s+1)/2 - (N - (((i+j)%2 == 0)? i : j));
            }
        }
    /* 打印输出 */
    for(i = 0; i < N; i++) {
        for(j = 0; j < N; j++)

            printf("%-6d", a[i][j]);
        printf("\n");
    }
    return 0;
}

```

2.8

对测试图像分块、DCT 和量化，将量化后的系数写成矩阵的形式，其中每一列为一个块的 DCT 系数 Zig-Zag 扫描后形成的列矢量，第一行为各个块的 DC 系数。

图像分块我使用了 cell 这个数据结构，方便处理分块矩阵，具体代码如下：

```
1 - clear, clc;
2 - load hall.mat
3 - load JpegCoeff.mat
4 - image = double(hall_gray);
5 - hall_block = cell(15, 21);
6 - hall_block_dct = cell(15, 21);
7 - hall_block_q = cell(15, 21);
8 - for x = 1:15
9 -     for y = 1:21
10 -         hall_block{x, y}(1:8, 1:8) = image(1+(x-1)*8:x*8, 1+(y-1)*8:y*8)-128;
11 -         hall_block_dct{x, y}(1:8, 1:8) = dct2(hall_block{x, y});
12 -         hall_block_q{x, y}(1:8, 1:8) = round(hall_block_dct{x, y}./QTAB);
13 -     end
14 - end
15 - hall_q = zeros(64, 315);
16 - for i = 1:315
17 -     hall_q(:, i) = zigzag(hall_block_q{ceil(i/21), i-(ceil(i/21)-1)*21});
18 - end
```

2.9

请实现本章介绍的 JPEG 编码（不包括写 JFIF 文件），输出为 DC 系数的码流、AC 系数的码流、图像高度和图像宽度，将这四个变量写入 jpegcodes.mat 文件。

- ◆ 本题涉及到四个代码文件：脚本 image_2_9_coder.m、函数 zigzag.m、函数 DC_coding.m、函数 AC_coding.m；
- ◆ 所使用的外部文件为：hall.mat（包含原始图像）、JpegCoeff.mat（包括量化步长矩阵 QTAB、DC 系数码本、AC 系数码本）；
- ◆ 输出文件为 jpegcodes.mat：DC 码流 DC_stream、AC 码流 AC_stream、图像高 h、宽 w；
- ◆ 代码思路：
此代码实现了 JPEG 编码，步骤为：图像分块、预处理、DCT、量化、熵编码（DC/AC coding）；
其中 DC/AC coding 部分调用了自己编写的函数；
主要代码如下：

```

1      % jpeg coder
2      clear, clc;
3      load hall.mat
4      load JpegCoeff.mat
5      [h,w] = size(hall_gray);
6      image = double(hall_gray);
7      hall_block = cell(15,21);
8      hall_block_dct = cell(15,21);
9      hall_block_q = cell(15,21);
10     for x = 1:15
11         for y = 1:21
12             % partitioning & preprocessing
13             hall_block{x,y}(1:8,1:8) = image(1+(x-1)*8:x*8, 1+(y-1)*8:y*8)-128;
14             % DCT
15             hall_block_dct{x,y}(1:8,1:8) = dct2(hall_block{x,y}(1:8,1:8));
16             % quantization
17             hall_block_q{x,y}(1:8,1:8) = round(hall_block_dct{x,y}(1:8,1:8)./QTAB);
18         end
19     end

20     % entropy coding
21     hall_q = zeros(64,315);
22     for i = 1:315
23         hall_q(:,i) = zigzag(hall_block_q{ceil(i/21),i-(ceil(i/21)-1)*21});
24     end
25     % DC coding
26     DC_forecast_err = zeros(1,315);
27     DC_forecast_err(1) = hall_q(1,1);
28     for j = 2:315
29         DC_forecast_err(j) = hall_q(1,j-1)-hall_q(1,j);
30     end
31     DC_stream = DC_coding(DC_forecast_err, DCTAB);
32     %AC coding
33     AC_stream = AC_coding(hall_q, ACTAB);
34
35     save('jpegcodes.mat','h','w','DC_stream','AC_stream');

```

2.10

计算压缩比（输入文件长度/输出码流长度），注意转换为相同进制。

输入文件长度：120*168*uint8 = 161280bits；

输出码流长度：(DC_stream) 2054bits + (AC_stream) 23072bits = 25126bits；

压缩比：161280/25126 = 6.42；

2.11

请实现本章介绍的 JPEG 解码，输入是你生成的 jpegcodes.mat 文件。分别用客观 (PSNR) 和主观方式评价编解码效果如何。

客观评价：PSNR = 31.1771；

主观评价：



原图 hall_gray



编解码后图

处理后图片与原图看起来几乎无差别，编码效果较好；

2.12

将量化步长减小为原来的一半，重做编解码。同标准量化步长的情况比较压缩比和图像质量。



原始 Q 值



Q/2

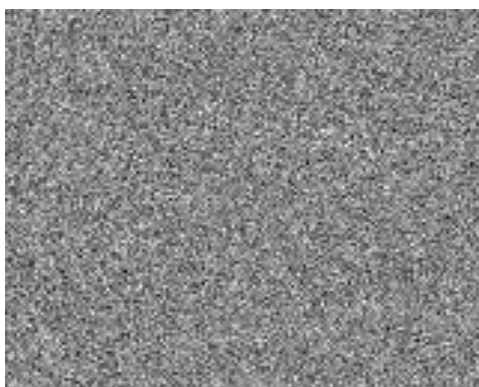
压缩比： $161280 / (2423 + 34153) = 4.4$ ；

PSNR = 34.185；

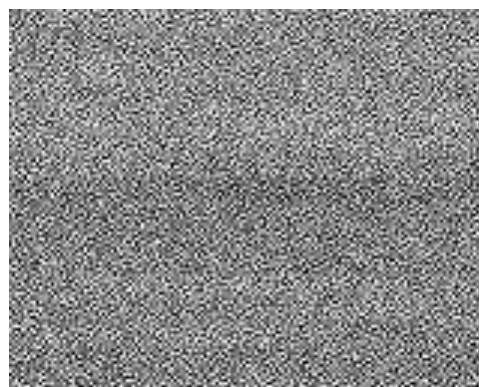
将量化步长减小后，压缩比降低，图像失真程度降低；

2.13

看电视时偶尔能看到美丽的雪花图像（见 snow.mat），请对其编解码。和测试图像的压缩比和图像质量进行比较，并解释比较结果。



原图



处理后

压缩比 = 2.50 ;

PSNR = 9.482 ;

发现压缩比较低且图像失真程度较高，分析原因可能是由于原图高频分量较多，DCT 系数矩阵右下方数据多且大，而 QTAB 量化步长矩阵在右下方的值较大，相除后产生的误差较大，导致图像处理后高频信息损失较大，从而导致 PSNR 值变小；而压缩比减小的原因同样在于原图高频分量较多，DCT 系数矩阵右下方数值较大，量化后非零值较多，所需编码数值较多，导致码流变长。

3.1

实现本章介绍的空域隐藏方法和提取方法。验证其抗 JPEG 编码能力。

JPEG 编码使用量化步长全部为 1 ;

隐藏信息为全 1 序列，120*168bits ;

提取后信息被破坏，出现许多 0 ;

原始图片（加隐藏信息）：



解码后图片：



代码：

```
1 % 空域信息隐藏&提取
2 - load hall.mat
3 - image_me = double(hall_gray);
4 - message_2 = ones(1,120*168);
5 - t = 1;
6 - for i = 1:120
7 -     for j = 1:168
8 -         s = dec2bin(image_me(i,j));
9 -         s(end) = num2str(message_2(t));
10 -        image_me(i,j) = bin2dec(s);
11 -        t = t + 1;
12 -     end
13 - end
14 - imwrite(uint8(image_me),'image_me.jpg');
```

```
15 - image_jpeg = JPEG(image_me);
16 - [h,w] = size(image_jpeg);
17 - imshow(uint8(image_jpeg));
18 - message_2 = zeros(1,120*168);
19 - t = 1;
20 - for i = 1:h
21 -     for j = 1:w
22 -         r = dec2bin(uint8(image_jpeg(i,j)));
23 -         z = r(end);
24 -         message_2(t) = str2num(z);
25 -         t = t + 1;
26 -     end
27 - end
```

3.2

依次实现本章介绍的三种变换域信息隐藏方法和提取方法，分析嵌密方法的隐蔽性以及嵌密后 JPEG 图像的质量变化和压缩比变化。

a. 方法一

◆ 加密信息：'I love you!'

因为方法一需要替换掉所有量化后 dct 系数的最低位，所以不足位补零；

```

11 % hidden information
12 message = zeros(1,120*168);
13 words = dec2bin(' I love you!');
14 [r,c] = size(words);
15 f = 1;
16 for i = 1:r
17     for j = 1:c
18         message(f) = str2num(words(i,j));
19         f = f + 1;
20     end
21 end

```

◆ 关键代码：

```

37 % information hiding
38 t = 1;
39 for x = 1:h/8
40     for y = 1:w/8
41         for i = 1:8
42             for j = 1:8
43                 if image_block_q{x,y}(i,j) < 0
44                     s = dec2bin(-image_block_q{x,y}(i,j)); % 不用补码
45                     s(end) = num2str(message(t));
46                     image_block_q{x,y}(i,j) = -bin2dec(s);
47                     t = t + 1;
48                 else
49                     s = dec2bin(image_block_q{x,y}(i,j));
50                     s(end) = num2str(message(t));
51                     image_block_q{x,y}(i,j) = bin2dec(s);
52                     t = t + 1;
53                 end
54             end
55         end
56     end
57 end

87 % information extrating
88 get_message = zeros(1,168*120);
89 result_block_a = cell(h/8,w/8);
90 t = 1;
91 for x = 1:h/8
92     for y = 1:w/8
93         % partitioning
94         result_block_a{x,y}(1:8,1:8) = coeff_q(1+(x-1)*8:x*8,1+(y-1)*8:y*8);
95     end
96 end
97 for x = 1:h/8
98     for y = 1:w/8
99         for i = 1:8
100             for j = 1:8
101                 if result_block_a{x,y}(i,j) < 0
102                     s = dec2bin(-result_block_a{x,y}(i,j));
103                     get_message(t) = str2num(s(end));
104                     t = t + 1;
105                 else
106                     s = dec2bin(result_block_a{x,y}(i,j));
107                     get_message(t) = str2num(s(end));
108                     t = t + 1;
109                 end
110             end
111         end
112     end
113 end

```

```

114 -     diff = get_message - message;
115 -     words_decode = zeros(r,c);
116 -     f = 1;
117 -     for i = 1:r
118 -         for j = 1:c
119 -             words_decode(i,j) = message(i);
120 -             f = f + 1;
121 -         end
122 -     end
123 -     mes_decode = char(bin2dec(char(words_decode+48)));

```

◆ 结果评估：



原图



加密后

```

mes_decode =

I
l
o
v
e
y
o
u
!

```

解密信息

可以看出，加密后图片左上角两个块有明显处理过的痕迹，其余部分交流分量减少，不过总体保持了原图的样式；
PSNR = 27.3354；

压缩比 = 8.104；

图片质量有所下降但压缩比升高，原因是原图 dct 系数中很多交流分量的 1 被替换成了 0；

b. 方法二

同理方法一，只需去掉隐藏信息后面补的 0 即可；



原图



加密后

PSNR = 30.51352；

压缩比 = 6.384545；

可以看出图片质量相较方法一有所提升，这也验证了图片质量下降是由交流分量减少所导致；

c. 方法三

◆ 加密信息：不足位置补 1；

```

11 % hidden information
12 - message = ones(1,h*w/64);
13 - message(1:8) = [1,-1,1,1,1,-1,-1,1];

```

◆ 关键代码：

```

36 % information hiding
37 for i = 1:h*w/64
38     for j = 1:64
39         if hall_q(j,i) ~= 0
40             if j == 64
41                 last = 64;
42             else
43                 last = j + 1;
44             end
45         end
46     end
47     hall_q(last,i) = message(i);
48 end

```

```

87 % information extrating
88 get_message = zeros(1,h*w/64);
89 for i = 1:h*w/64
90     for j = 1:64
91         if hall_q_decode(j,i) ~= 0
92             last = j;
93         end
94     end
95     get_message(i) = hall_q_decode(last,i);
96 end
97 diff = get_message-message;

```

◆ 结果评估：



原图



加密后

PSNR = 28.93531 ;

压缩比 = 6.186184 ;

可以看出，方法三相较前两种方法的优点在于加密后图片整体更均匀，不会在某些位置出现明显的加密痕迹，即使每个块都含有加密信息，图片质量也保持较高；

4.1

所给资料 Faces 目录下包含从网图中截取的 28 张人脸，试以其作为样本训练人脸标准 \mathbf{v} 。

(a) 样本人脸大小不一，是否需要首先将图像调整为相同大小？

(b) 假设 L 分别取 3,4,5，所得三个 \mathbf{v} 之间有何关系？

(a) 不需要。因为每张图训练出的人脸标准 \mathbf{u} 都是经过归一化之后的。

(b) L 的取值决定了颜色分类的精度，因为待训练的图片一般都是 8bits 表示一个像素一个通道的颜色值(0~255)，所以我先令 $L=8$ ，计算出高精度人脸标准 \mathbf{v} 后， $L=3,4,5$ 时可由 \mathbf{v} 的相应区间累加得到，具体代码如下：

◆ $L = 8$:

```
1 - clear,clc;
2 - L = 8;
3 - N = 2 ^ (3 * L);
4 - v = zeros(N, 1);
5 - for i = 1:33
6 -     image = double(imread(sprintf('Faces/%d.bmp',i)));
7 -     [rows, cols, channels] = size(image);
8 -     u = zeros(N, 1);
9 -     c_n = zeros(1, 3*L) + 48;
10 -    for x = 1:rows
11 -        for y = 1:cols
12 -            c_n(L-length(dec2bin(image(x,y,1)))+1:L) = dec2bin(image(x,y,1));
13 -            c_n(2*L-length(dec2bin(image(x,y,2)))+1:2*L) = dec2bin(image(x,y,2));
14 -            c_n(3*L-length(dec2bin(image(x,y,3)))+1:3*L) = dec2bin(image(x,y,3));
15 -            t = bin2dec(char(c_n));
16 -            u(t+1) = u(t+1) + 1;
17 -        end
18 -    end
19 -    u = u / (rows*cols);
20 -    v = v + u;
21 - end
22 - v = v / 33;
23 - save('v.mat','v');
```

◆ $L = 3, 4, 5$:

```
1 - clear,clc;
2 - load v.mat
3 - L = 3; % 3 or 4 or 5
4 - N = 2 ^ (3 * L);
5 - pack = 2 ^ (3 * (8 - L));
6 - v2 = zeros(N, 1);
7 - for i = 1:N
8 -     for k = 1:pack
9 -         v2(i) = v2(i) + v((i-1) * pack + k);
10 -    end
11 - end
```


4.2

设计一种从任意大小的图片中检测任意多张人脸的算法并编程实现（输出图像在判定为人脸的位置加上红色的方框）。随意选取一张多人照片（比如支部活动或者足球比赛），对程序进行测试。尝试 L 分别取不同的值，评价检测结果有何区别。

◆ 算法描述：

1. 分块：计算每一个 4×4 小块的特征向量，与标准向量比较，在阈值 $thre$ 内的标记为 1；

```
8      % G*G blocks
9      G = 4;
10     image = zeros(ceil(rows/G)*G, ceil(cols/G)*G, 3);
11     image(1:rows, 1:cols, :) = image_ori;
12     [h, w, cha] = size(image);
13     figure
14     imshow(uint8(image));
15     rect = zeros(h/G, w/G);
16     for x = 1:h/G
17         for y = 1:w/G
18             block = image(1+(x-1)*G:x*G, 1+(y-1)*G:y*G, :);
19             % compute the eigenvector of each block
20             u = eigenvector(block, G, L);
21             sum = 0;
22             for i = 1:length(u)
23                 sum = sum + u(i);
24             end
25             % compute the distance
26             t = distance(u, v_test);
27             if distance(u, v_test) < thre
28                 hold on, rectangle('Position', [(y-1)*G, (x-1)*G, G, G], 'edgecolor', 'r');
29                 rect(x, y) = 1;
30             end
31             % fprintf('x:%d, y:%d, sum of u:%d, dist:%d\n', x, y, sum, t);
32         end
33     end
```

2. 用 `bwlabel` 标记每个连通域，并用 `regionprops` 计算每个区域的属性，若该区域所含元素数大于某阈值，则将该区域的 `boundingBox` 画出；

```
54     rect_f2 = rect;
55     [con_region, num] = bwlabel(rect_f2);
56     region = regionprops(con_region, 'Area', 'BoundingBox');
57     figure
58     imshow(uint8(image))
59     for i = 1:num
60         if region(i).Area > 5
61             box = region(i).BoundingBox;
62             hold on, rectangle('Position', [(box(1)-1)*G, (box(2)-1)*G, box(3)*G, box(4)*G], 'edgecolor', 'r');
63         end
```

3. 另外，在标记连通区域之前，我一开始还实现了一个 `boxFilter`，想将 `rect` 矩阵低通滤波一下，再分层，补上 `rect` 中的一些洞，去掉多余的点，但是后来效果不是很理想，就没有使用，代码如下：

```

40 -     rect_f1 = boxfilter(rect);
41 -     rect_f2 = boxfilter(rect_f1);
42 -     for x = 1:h/G
43 -         for y = 1:w/G
44 -             if rect_f2(x,y) > 0.4
45 -                 rect_f2(x,y) = 1;
46 -             else
47 -                 rect_f2(x,y) = 0;
48 -             end
49 -         end
50 -     end
51 -     figure
52 -     imshow(uint8(image));
53 -     for x = 1:h/G
54 -         for y = 1:w/G
55 -             if rect_f2(x,y) == 1
56 -                 hold on, rectangle('Position', [(y-1)*G, (x-1)*G, G, G], 'edgecolor', 'r');
57 -             end
58 -         end
59 -     end

```

```

1 - function rect_f = boxfilter(rect)
2 -     [row, col] = size(rect);
3 -     rect_f = zeros(row, col);
4 -     for x = 1:row
5 -         for y = 1:col
6 -             if (x-1>0) && (x+1<row) && (y-1>0) && (y+1<col)
7 -                 rect_f(x,y) = (rect(x,y) + rect(x-1,y) + rect(x+1,y) + rect(x,y-1) + rect(x,y+1)) / 5;
8 -             else
9 -                 rect_f(x,y) = rect(x,y);
10 -            end
11 -        end
12 -    end
13 - end

```

◆ 测试结果：

参数：L = 3；thre = 0.67；Area > 9；



中间结果 (4*4 小块)



最终结果

参数：L = 4；thre = 0.84；Area > 6；



使用滤波函数：



参数：L = 5 ; thre = 0.94 ; Area > 5 ;



分析：

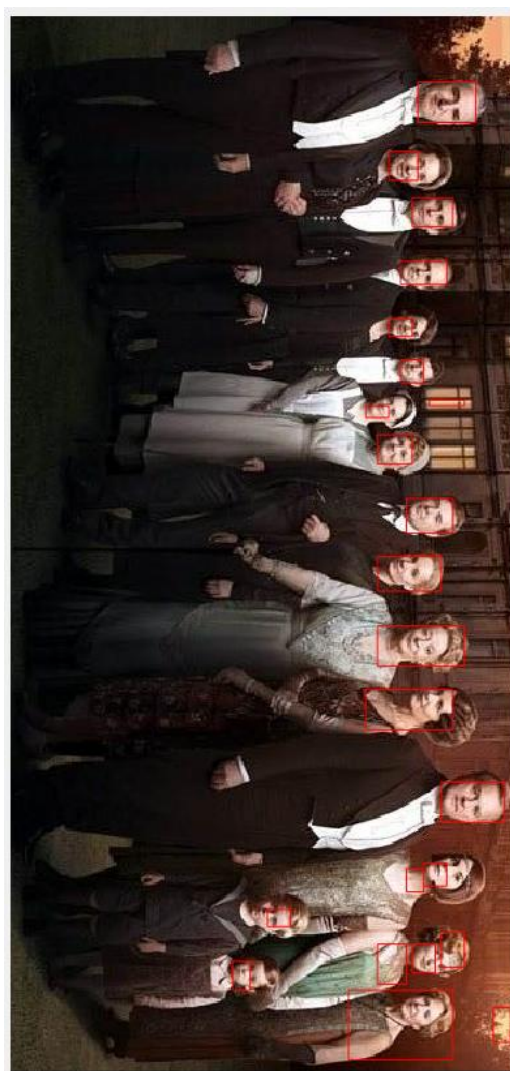
L 越大，识别准确度越高，L=3 时有些衣服上白色的部分也会被圈进，L=4、5 时基本不会；但是 L 越大程序运行时间越长；而且不同的 L 两个参数也需要相应的调节。

4.3

对上述图像分别进行如下处理后

- (a) 顺时针旋转 90° (imrotate);
- (b) 保持高度不变, 宽度拉伸为原来的 2 倍 (imresize);
- (c) 适当改变颜色 (imadjust);

再试试你的算法检测结果如何? 并分析所得结果。





分析：

参数设置：L = 4；thre = 0.84；Area > 5；

1. 顺时针旋转 90 度后结果与旋转前无明显差别；
2. 横向拉伸后，增加了一些红框，原因是原来边长不足 4 的小块被拉伸后满足阈值要求；
3. 颜色调整后原先的一些人脸部分被舍弃掉了，因为颜色发生了改变，原来的参数不再适用；

4.4

如果可以重新选择人脸样本训练标准，你觉得应该如何选取？

我认为如果想要依靠颜色提升人脸识别的准确度，应该针对不同种族不同肤色的人分别设立标准特征向量，检测时分别判断被检测区域是否与其中的一个标准特征向量相差在阈值范围内；而且一套训练样本中人脸的方向应该保持大概一致，否则也会对训练出的标准样本产生干扰。