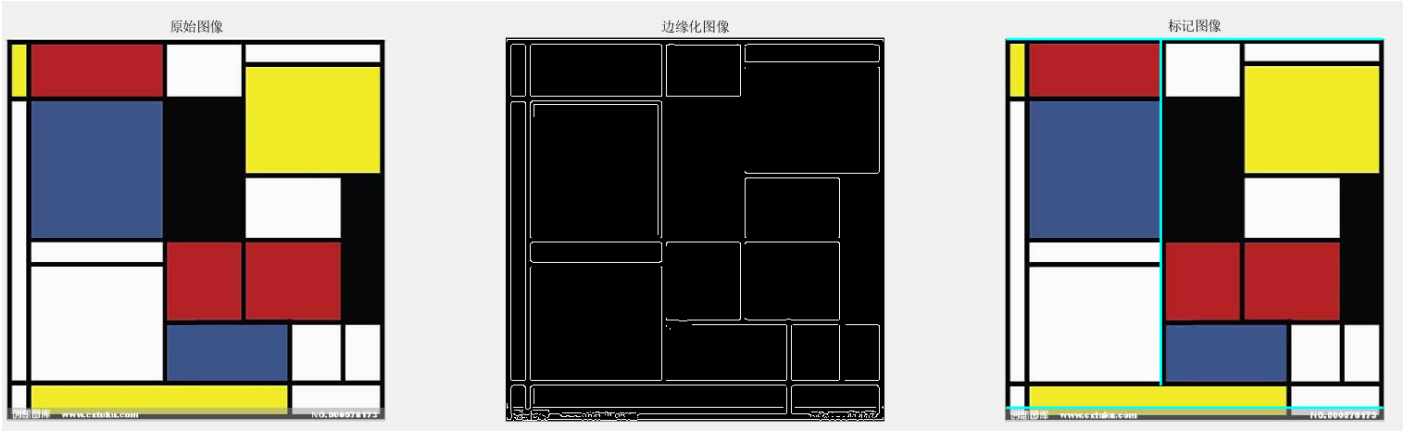
课程实验

任意给出一幅彩色图像，编写代码实现：

（1）利用Hough变换检测图像中最长的3条直线并且标记出来；

（2）利用matlab自带的边界跟踪函数标记出面积最大的3个区域。



边界跟踪


```
clear all
I = imread('rice.png');              %导入图像
figure(1),
subplot(1,3,1);
imshow(I),title('原始图像')
BW = im2bw(I, graythresh(I));        %生成二值图像
subplot(1,3,2);
imshow(BW),title('二值图像')
 [B,L] = bwboundaries(BW,'noholes'); %提取边界，并返回边界元胞数组B 和区域标志数组L
subplot(1,3,3);
imshow(label2rgb(L, @jet, [.5 .5 .5]))  %以不同的颜色标志不同的区域
title('彩色标记图像')
hold on
for k = 1:length(B)
   boundary = B{k};
   plot(boundary(:,2), boundary(:,1), 'w', 'LineWidth', 2)  %在图像上叠画边界
end
```

L中存放区域编号

L　409 x 412

L(:)　168508 x 1 即 HW x 1

area_ids 76 x 1

find(L(:)==area_ids(3))：　919 x 1

```
 9 -     [B,L] = bwboundaries(BW,'noholes');      %提取边界，并返回边界元胞数组B 和区域标志数组L
10 -     area_ids=unique(L);
11 -     area = zeros(length(area_ids),1);
12 -     for i=1:length(area_ids)
13 -         area(i)=length(find(L(:)==area_ids(i)));
14 -     end
15       %对区域面积进行排序
16       % ...
```

课程大作业二

- 输入一张灰度/真彩色图像，编程完成如下功能：
  - （1）利用Haar小波进行编码，得到中间数据文件，存储；
  - （2）针对编码后的中间存储文件，利用matlab内嵌的huffman编码函数进行二进制编码，并存为压缩文件；
  - （3）读取压缩文件，解码得到原始图像进行显示并对比压缩效率。

$$A=\begin{pmatrix}64&2&3&61&60&6&7&57\\9&55&54&12&13&51&50&16\\17&47&46&20&21&43&42&24\\40&26&27&37&36&30&31&33\\32&34&35&29&28&38&39&25\\41&23&22&44&45&19&18&48\\49&15&14&52&53&11&10&56\\8&58&59&5&4&62&63&1\end{pmatrix}$$

$$A_R=\begin{pmatrix}32.5&0&0.5&0.5&31&-29&27&-25\\32.5&0&-0.5&-0.5&-23&21&-19&17\\32.5&0&-0.5&-0.5&-15&13&-11&9\\32.5&0&0.5&0.5&7&-5&3&-1\\32.5&0&0.5&0.5&-1&3&-5&7\\32.5&0&-0.5&-0.5&9&-11&13&-15\\32.5&0&-0.5&-0.5&17&-19&21&-23\\32.5&0&0.5&0.5&-25&27&-29&31\end{pmatrix}-$$

$$A_{RC}=\begin{pmatrix}32.5&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0\\0&0&0&0&4&-4&4&-4\\0&0&0&0&4&-4&4&-4\\0&0&0.5&0.5&27&-25&23&-21\\0&0&-0.5&-0.5&-11&9&-7&5\\0&0&0.5&0.5&-5&7&-9&11\\0&0&-0.5&-0.5&21&-23&25&-27\end{pmatrix}$$

阈值裁剪

$$\begin{bmatrix}32.5&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0\\0&0&0&0&27&-25&23&-21\\0&0&0&0&-11&9&-7&0\\0&0&0&0&0&7&-9&11\\0&0&0&0&21&-23&25&-27\end{bmatrix}$$

利用Matlab自带函数对字符串进行Huffman编码

```
3  str = 'You have to believe in yourself. That is the secret of success.';
4
5  %根据字符串str得到符号集symbols，并计算各集合元素的出现概率数组p
6  len = length(str);
7  unique_str = unique(str);
8  unique_len = length(unique_str);
9
10 symbols = cell(1, unique_len);
11 p = zeros(1, unique_len);
12
13 for i = 1:unique_len
14     symbols{1,i} = unique_str(i);
15     p(i) = numel(find(str==unique_str(i))) / len;
16 end
17
18 %根据符号集symbols和概率数组p计算Huffman编码词典
19 [dict, avglen] = huffmandict(symbols, p);
```

Matlab实现基于Huffman编码的图像压缩

ENCO = huffmanenco(SIG, DICT) : 哈夫曼编码函数，SIG为输入编码信号，DICT为编码字典，由函数huffmandict（）生成；

DECO = huffmandeco(COMP, DICT) ： 哈夫曼解码函数，COMP为哈夫曼编码向量，即上面的ENCO；

DICT = huffmandict(SYM, PROB) ： 哈夫曼字典生成函数，SYM为信源符号向量，包含信息中所有符号，PROB为相应符号出现的概率；

```
clear;
clear all;
I = imread('F:\Myfile\Matlab\Test_picture\1_1.jpg');

[M,N] = size(I);
 I1 = I(:);
 P = zeros(1,256);
 %获取各符号的概率;
 for i = 0:255
     P(i+1) = length(find(I1 == i))/(M*N);
 end

 k = 0:255;
 dict = huffmandict(k,P); %生成字典
 enco = huffmanenco(I1,dict); %编码
 deco = huffmandeco(enco,dict); %解码
 Ide = col2im(deco,[M,N],[M,N],'distinct'); %把向量重新转换成图像块;

 subplot(1,2,1);imshow(I);title('original image');
 subplot(1,2,2);imshow(uint8(Ide));title('deco image');
```

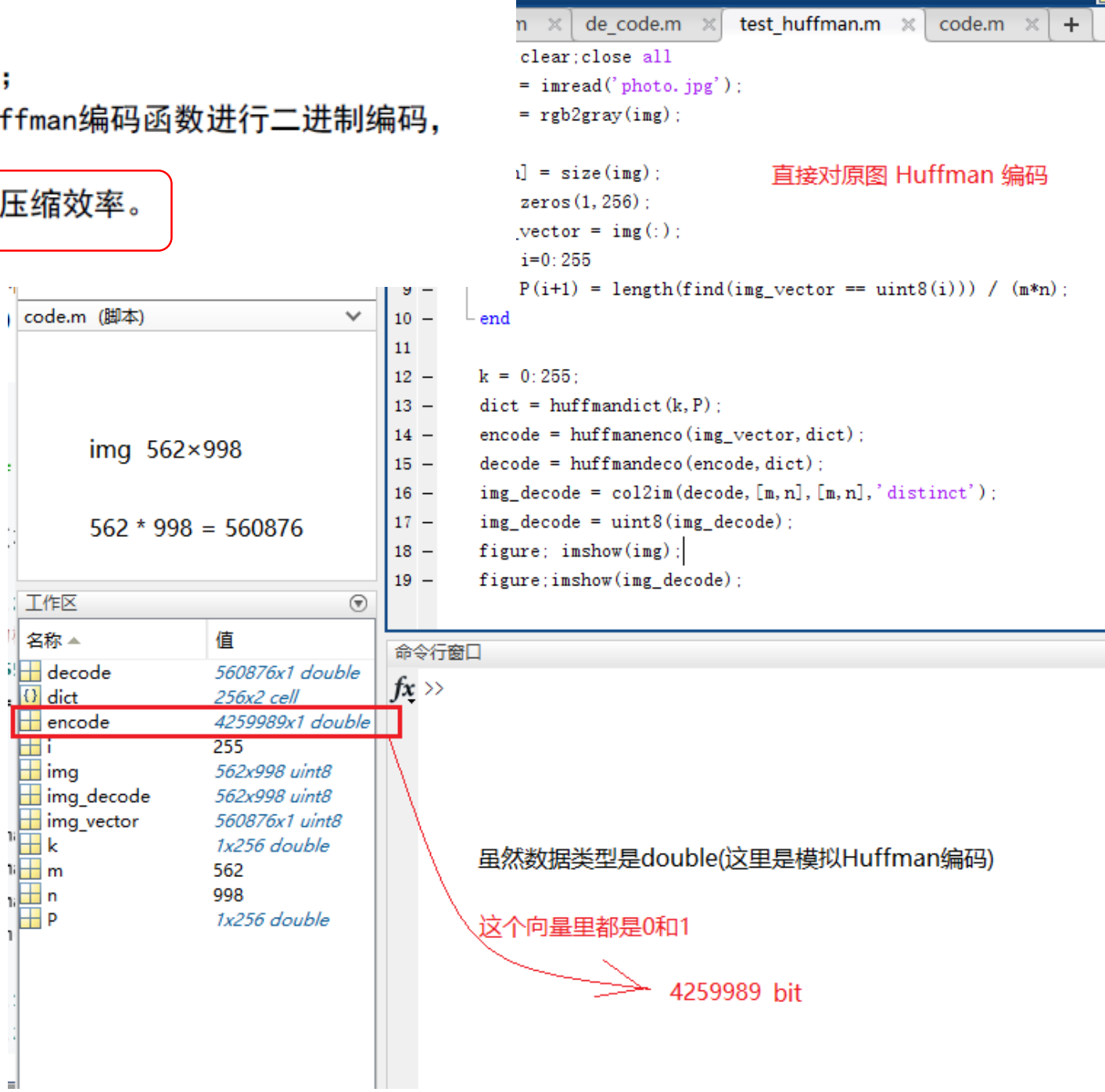
```
I1 = I(:);
k = unique(I1);

for i=1:length(k)
        P(i) = length(find(I1 == k(i))) / (N*M);
end
```

- 输入一张灰度/真彩色图像，编程完成如下功能：
  - （1）利用Haar小波进行编码，得到中间数据文件，存储；
  - （2）针对编码后的中间存储文件，利用matlab内嵌的huffman编码函数进行二进制编码，并存为压缩文件；
  - （3）读取压缩文件，解码得到原始图像进行显示并对比压缩效率。

jpg或者png格式自带压缩

压缩前：
560,876 Byte

压缩后
4,259,989bit = 532,498 Byte



img_comp = Huffman_compress(img)

未经Haar小波编码：img_comp v.s. img

img_enco = Haar_transform(img);

经过Haar小波编码并压缩：img_enco_comp v.s. img

img_enco_comp = Huffman_compress(img_enco)

NOTE： 用unique的时候转
为带符号整型，比如int16

length(unique(img)) = 256
length(unique(img_enco)) = 5128

Haar 变换之后，符号种类可能增加，但是压缩效果更好

原因：
（1）虽然符号种类数量增加，但是分布改变了
比如0.5 这个符号，占比80%， 那冗余性就很大
（2）如果0.5 占大多数，可以用阈值裁剪