

课程实验4

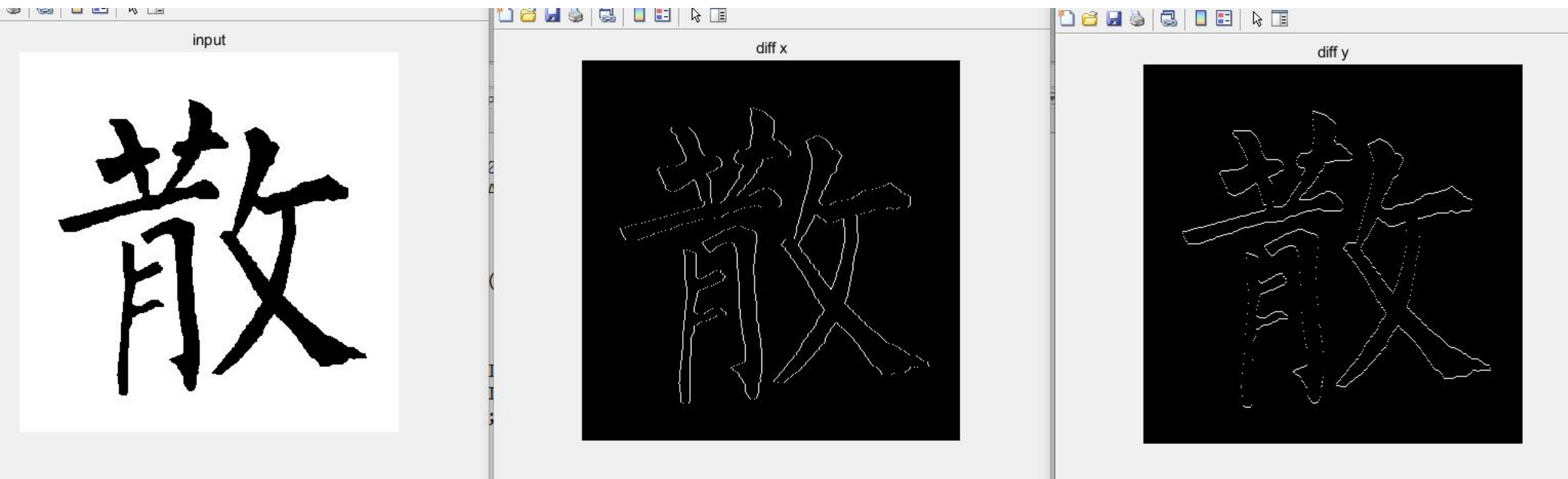
- 编写代码实现如下功能：
 - 输入灰度/真彩色图像，利用差分编码思想对其进行重编码并存储为数据文件；
 - 装载编码后的数据文件进行解码得到原始图像文件并显示。

图像差分意义:

1. 提取图像边界

例如: 汉字的轮廓

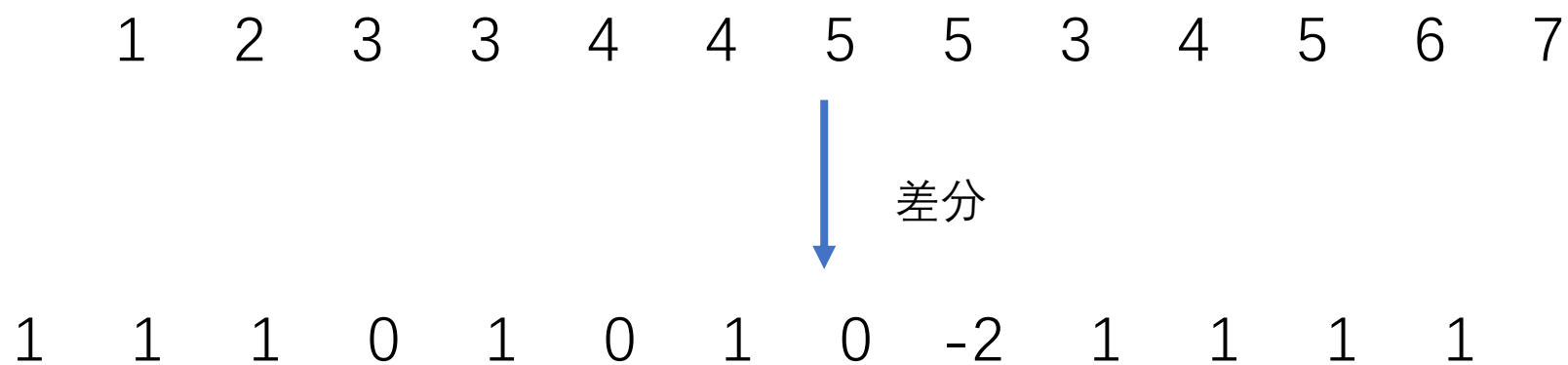
```
figure;imshow(abs(diff_I_x),[]);title("diff x");  
figure;imshow(abs(diff_I_y),[]);title("diff y");
```



图像差分的意义:

2.减小冗余性。

(图像差分本身并不是在做压缩)



图像差分的意义:

2.减小冗余性。

(图像差分本身并不是在做压缩)

原图 → Huffman 编码

vs.

原图 → 差分编码 → Huffman 编码

注意数据范围: uint8 的差分在 -255 ~ +255 之间

uint8: 0~255

int8: -128 ~ +127

int16: -32768 ~ + 32767

```
>> a = uint8(-255)
```

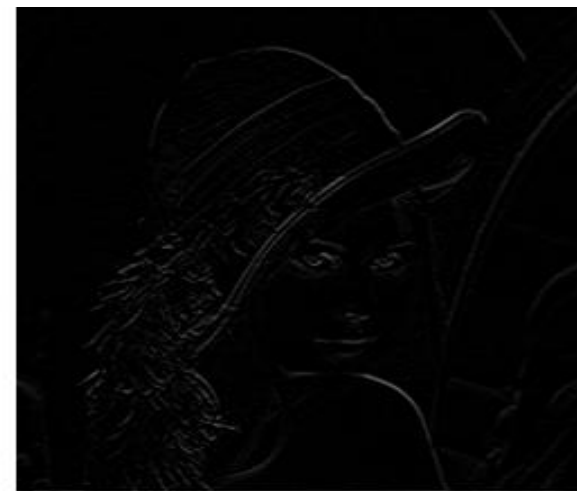
```
a =
```

```
uint8
```

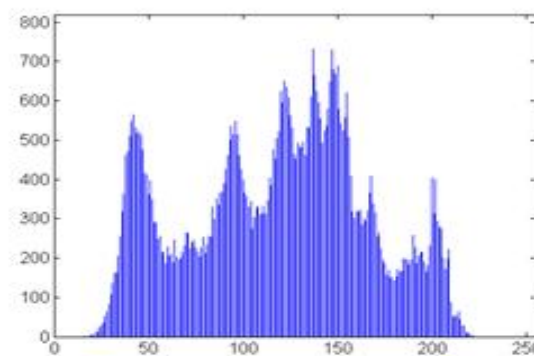
```
0
```



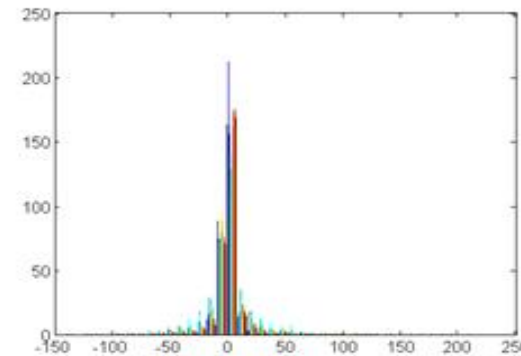
(a) 原图



(b) 差分编码结果



(c) 原图进行huffman



(d) 差分结果进行huffman

1	2	3	3	4	4	5	5	3	4	5	6	7
1	1	1	0	1	0	1	0	-2	1	1	1	1

↓ 差分

原图 → Huffman 编码

原图大小:

$560876 \times 8 = 4,487,008 \text{ bit}$

after Huffman encoding:

4,259,989 bit

$4259989 / 4487008 = 94.9\%$

The MATLAB interface displays the following components:

- Editor Window:** Contains the script `code.m` with the following code:

```
1 - clc;clear;close all
2 - img = imread('photo.jpg');
3 - img = rgb2gray(img);
4 -
5 - [m,n] = size(img);
6 - P = zeros(1,256);
7 - img_vector = img(:);
8 - for i=0:255
9 -     P(i+1) = length(find(img_vector == uint8(i))) / (m*n);
10 - end
11 -
12 - k = 0:255;
13 - dict = huffmandict(k,P);
14 - encode = huffmanenco(img_vector,dict);
15 - decode = huffmandeco(encode,dict);
16 - img_decode = col2im(decode,[m,n],[m,n],'distinct');
17 - img_decode = uint8(img_decode);
18 - figure; imshow(img);
19 - figure; imshow(img_decode);
```

A red annotation "直接对原图 Huffman 编码" (Direct Huffman encoding on the original image) points to line 13.
- Command Window:** Shows the execution of the code, with the prompt `fx >>`.
- Workspace Window:** Lists the variables created during the process:

名称	值
decode	560876x1 double
dict	256x2 cell
encode	4259989x1 double
i	255
img	562x998 uint8
img_decode	562x998 uint8
img_vector	560876x1 uint8
k	1x256 double
m	562
n	998
P	1x256 double

The `encode` variable is highlighted with a red box. A red arrow points from this box to the text "4259989 bit".

Additional annotations:

- "虽然数据类型是double(这里是模拟Huffman编码)" (Although the data type is double (here it is a simulation of Huffman encoding))
- "这个向量里都是0和1" (This vector contains only 0s and 1s)
- "'huffmandict' 需要 Communications Toolbox." ('huffmandict' requires the Communications Toolbox.)

原图 → 差分编码 → Huffman 编码

原图大小:

$560876 \times 8 = 4,487,008 \text{ bit}$

after Huffman encoding:

2,236,728 bit

$2236728 / 4487008 = 49.8\%$

The image shows a MATLAB script in the editor window, a variable list in the workspace, and a command window. The script performs the following steps:

- Reads the image 'photo.jpg' and converts it to grayscale.
- Applies differential encoding: `img_diff_enco = diff_encode(img);` and `img_diff_enco = int16(img_diff_enco);`. This step is annotated with "对原图差分编码".
- Calculates the size of the encoded image: `[m,n] = size(img_diff_enco);`.
- Creates a probability vector `P` of size 1x512, where each element represents the frequency of a specific difference value. This step is annotated with "对差分编码的结果进行Huffman编码".
- Builds a Huffman dictionary `dict` using `huffmandict(k,P);`.
- Encodes the difference image using `huffmanenco(img_vector, dict);`.
- Decodes the Huffman-encoded data using `huffmandeco`.
- Applies differential decoding: `img_decode = diff_decode(diff_img_decode);` and `img_decode = uint8(img_decode);`. This step is annotated with "Huffman解码".
- Displays the original image and the reconstructed image using `figure; imshow`.

The workspace shows the following variables:

名称	值
ans	560876
decode	560876x1 double
dict	512x2 cell
diff_img_decode	562x998 double
encode	2236728x1 double
i	512
img	562x998 uint8
img_decode	562x998 uint8
img_diff_enco	562x998 int16
img_vector	560876x1 int16
k	1x512 double
m	562
n	998
P	1x512 double

The command window shows the calculation of the total number of bits in the encoded image:

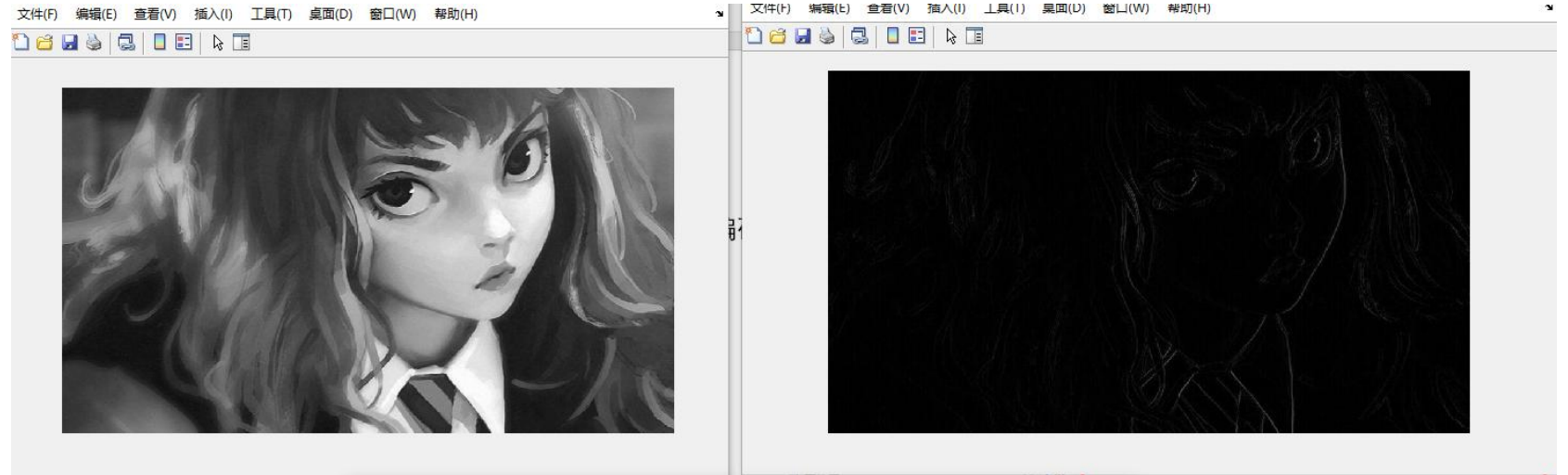
```
>> 562*998
ans =
    560876
```

An annotation points to the result 560876, stating "这个向量里都是0和1" (This vector contains only 0s and 1s) and "2236728 bit".

Below the command window, the final result is shown: 223 6728.

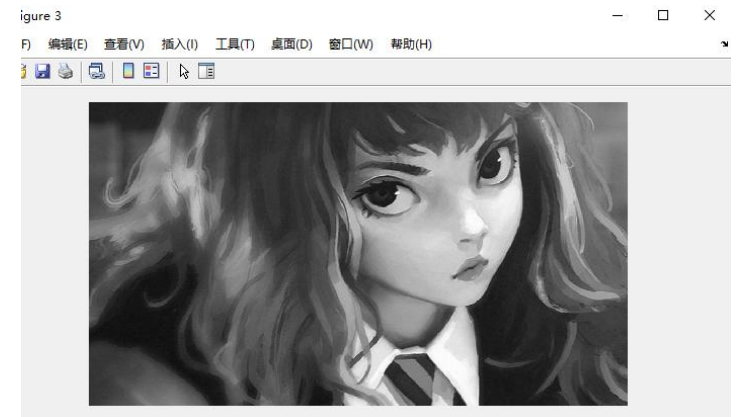
input

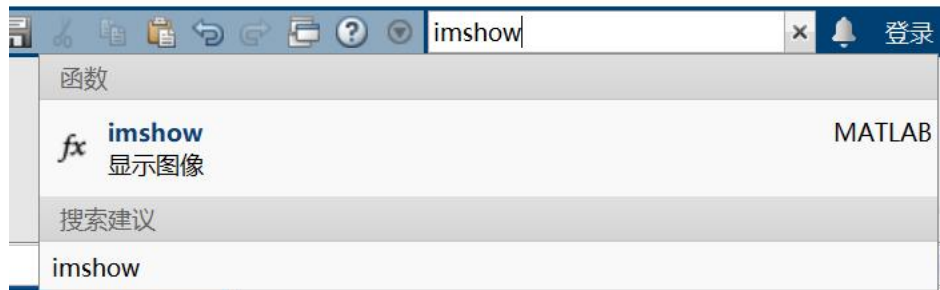
`imshow(uint8(abs(img_diff)),[])`
% 相当于画出梯度的绝对值



```
1 clear;
2
3 img = imread('1.jpg');
4 img = rgb2gray(img); %
5 img = int16(img); %
6
7 diff_img = myDiff(img); % possible range: -255 ~ 255
8 % diff_img = uint8(diff_img) % <-- this is WRONG!
9 save diff_img.mat diff_img
10
11 imshow(abs(diff_img),[]);
12
```

decode





```
>> help imshow
```

imshow - 显示图像

此 MATLAB 函数 在图窗中显示灰度图像 I。imshow 使用图像数据类型的默认显示范围，并优化图窗、坐标区和图像对象属性以便显示图像。

```
imshow(I)
```

```
imshow(I, [low high])
```

```
imshow(I, [])
```

```
imshow(RGB)
```

```
imshow(BW)
```

```
imshow(X, map)
```

```
imshow(filename)
```

```
imshow(___, Name, Value)
```

```
himage = imshow(___)
```

See also [imread](#), [image](#), [imagesc](#), [imwrite](#), [imfinfo](#), [iptsetpref](#)

[imshow 的文档](#)

语法

```
imshow(I)
imshow(I,[low high])
imshow(I,[])
imshow(RGB)
imshow(BW)
imshow(X,map)
imshow(filename)
imshow(___,Name,Value)
```

```
himage = imshow( __ )
```

说明

`imshow(I)` 在图窗中显示灰度图像 `I`。`imshow` 使用图像数据类型的默认显示范围，并优化图窗、坐标区和图像对象属性以便显示图像。

`imshow(I,[low high])` 显示灰度图像 `I`，以二元素向量 `[low high]` 形式指定显示范围。有关详细信息，请参阅 [DisplayRange](#) 参数。

`imshow(I,[])` 显示灰度图像 `I`，根据 `I` 中的像素值范围对显示进行转换。`imshow` 使用 `[min(I(:)) max(I(:))]` 作为显示范围。`imshow` 将 `I` 中的最小值显示为黑色，将最大值显示为白色。有关详细信息，请参阅 [DisplayRange](#) 参数。

`imshow(RGB)` 在图窗中显示真彩色图像 `RGB`。

`imshow(BW)` 在图窗中显示二值图像 `BW`。对于二值图像，`imshow` 将值为 0（零）的像素显示为黑色，将值为 1 的像素显示为白色。

`imshow(X,map)` 显示带有颜色图 `map` 的索引图像 `X`。颜色图矩阵可以具有任意行数，但它必须恰好包含 3 列。每行被解释为一种颜色，其中第一个元素指定红色的强度，第二个元素指定绿色的强度，第二个元素指定蓝色的强度。颜色强度可以在 `[0, 1]` 区间中指定。

`imshow(filename)` 显示存储在由 `filename` 指定的图形文件中的图像。

`imshow(___,Name,Value)` 使用名称-值对组控制运算的各个方面来显示图像。

`himage = imshow(__)` 返回 `imshow` 创建的图像对象。

Note

差分编码

uint8: 0~255,

差分范围: -255 ~ 255, 不能用uint8储存差分结果, 否则decode的时候会出现问题