《图象传感与图象处理》实验指导书

光学与电子科技学院

李子印

中国计量学院光学与电子科技学院

二零零九 年 九 月

实验（一）图像的基本操作上机实验

一.实验目的与要求：

用 MATLAB 或者 C++语言对图像进行基本操作。掌握图像读取、显示和保存的方法；并能对图像进行灰度调整、反相等基本操作；进行图像的采样和量化；理解彩色图像、灰度图像和二值图像的区别。

二.实验仪器及软件：

计算机、MATLAB 或者 Visual C++软件、Photoshop 软件。

三.实验地点：

赛博北楼综合应用技术实验室（B405）和微电子技术实验室（B404）。

四.实验原理-MATLAB 简介：

4.1 主要用途及特点

主要用途：算法研究，项目前期验证

主要特点：语句功能强大；不能生成可执行文件。

4.2 MATLAB 工作环境

4.2.1 Matlab 集成环境

如图 1.1 所示，桌面包括 5 个子窗口：命令窗口、工作空间窗口、当前目录窗口、历史命令窗口、一个或多个图形窗口（仅在用户显示图形式出现）。

命令窗口是用户在提示符（>>）处键入 MATLAB 命令和表达式的地方，也是显示那些命令输出的地方。

工作空间窗口显示对话中创建的变量和它们的某些信息。

当前目录窗口显示当前目录的内容（即路径）。

历史命令窗口包含用户已在命令窗口中输入的命令的纪录。

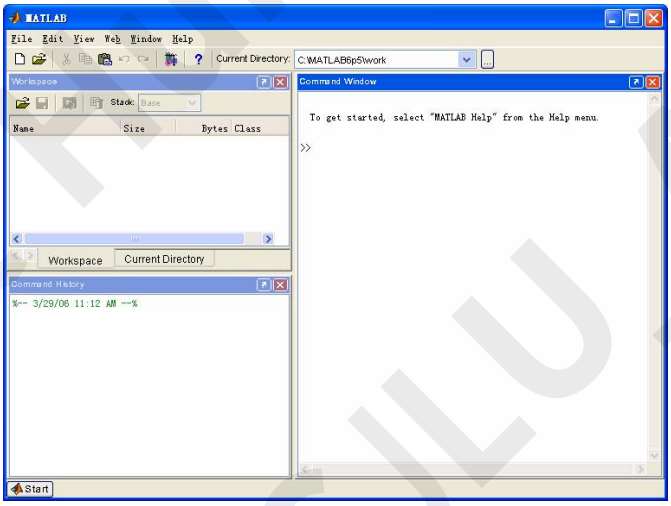


图 1.1 MATLAB 集成环境

4.2.2　使用 MATLAB 编辑器创建 M 文件

4.2.3　获得帮助

（1）help <函数名>

（2）lookfor <关键词>

4.3 数字图像的读取、显示和保存

4.3.1 图像的读取

语法：imread (‘filename’)

说明：读取图像

表 1.1 图像文件类型及其后缀名

格式名称	描述	可识别扩展符
TIFF	加标记的图像文件格式	.tif, .tiff
JPEG	联合图像专家组	.jpg, .jpeg
GIF	图形交换格式	.gif
BMP	Windows 位图	.bmp
PNG	可移植网络图形	.png
XWD	X Window 转储	.xwd

语法：[M, N]=size (‘filename’)

说明：给出一幅图像的行数和列数

4.3.2 图像的显示

语法：imshow (f, G)

imshow (f, [low high])

imshow (f, [ ])

说明：G 是显示该图像的灰度级数；小于或等于 low 的值都显示为黑色，大于或等于 high 的值都显示为白色；[ ]自动将变量 low 设置为 f 的最小值，将 high 设置为 f 的最大值。

4.3.3 图像的保存

语法：imwrite (f, ‘filename’)

说明：保存图像

4.4 数据类型和图像类型、数据类型间的转换、图像类型间的转换

4.4.1 数据类型

表 1.2 MATLAB 数据类型

名称	描述
double	双精度浮点数，范围为-10exp(308)~ 10exp(308)，8 字节
uint8	无符号 8 比特整数，1 字节
uint16	无符号 16 比特整数，2 字节
uint32	无符号 32 比特整数，4 字节
int8	有符号 8 比特整数，1 字节
int16	有符号 16 比特整数，2 字节
int32	有符号 32 比特整数，4 字节
single	单精度浮点数，范围为-10exp(38)~ 10exp(38) ，4 字节
char	字符
logical	值为 0 或 1

四种常用类型：double，uint8，char，logical。

4.4.2 图像类型

表 1.3 图像类型

名称	描述
灰度图像	uint8 类范围为[0 255]、double 类归一化为[0 1]
二值图像	图像取值只有 0 和 1 的逻辑数组
索引图像	索引图像
RGB 图像	彩色图像

4.4.3 数据类间的转换

语法：B = data_class_name (A)

举例：若 A 是 8 位图像，则 B = double (A)转换为双精度图像。

4.4.4 图像类型间的转换

表 1.4 图像类型间的转换

函数名称	将输入转换为	有效的输入图像数据类
im2uint8	uint8	Logical, uint8, uint16 和 double

im2uint16	uint16	Logical, uint8, uint16 和 double
mat2gray	double	double
im2double	double	Logical, uint8, uint16 和 double
im2bw	logical	uint8, uint16 和 double

五.实验内容：

5.1 实验前的准备工作：

a. 了解 MATLAB 或者 Visual C++ 的编程环境；

b. 了解图像的种类和格式；

c. 了解 MATLAB 图像操作的相关函数；

d. 了解 Photoshop 软件的相关功能。

5.1 实验操作要求：

a. 用 MATLAB 或者 C++语言编程，读取并显示一副 tif 格式的图像，然后将该图像在水平和垂直方向分别进行下采样，将得到的新图像存储成 bmp 格式并显示出来；

b. 用 MATLAB 或者 C++语言编程，读取并显示一副 RGB 彩色图像，然后将其转换为灰度图像并显示；

c. 用 MATLAB 或者 C++语言编程，读取并显示一副二值图像，然后对其进行取反操作，显示得到的图像；

d. 用 MATLAB 或者 C++语言编程，读取两副图像，在同一个窗口显示它们；

e. 对灰度图像分别进行 4 和 16 倍下采样，查看其下采样效果（提示：quartimg = zeros(wid/2+1,hei/2+1)。

f. 将 256 级灰度图像，转换成 128 级灰度图像，64 级灰度图像，32 级灰度图像，8 级灰度图像和 2 级灰度图像。

提示：
```
for i = 1:wid
    for j = 1:hei
        img64(i,j) = floor(b(i,j)/4);
    end
end
```

g. 用 Photoshop 软件打开不同格式的图像，进行灰度调整、反相等操作，并观察操作结果。

六.实验报告内容：

6.1 实验数据记录部分：记录实验中所用图像的分辨率、读取后所占矩阵大小；对关键代码进行注释。

6.2 分析不同类型图像的特点。

6.3 分析采样和量化对数字图像质量的影响。

七.思考题：

7.1 彩色图像和灰度图像中包含的信息内容有什么区别？

八.参考代码：

8.1

```
clear;
close all;
I=imread('原图');
imshow(I);
I2=imresize(I,0.5);
figure,imshow(I2);
imwrite(I2,'新图像');
imfinfo('新图像')
```

8.2

```
clear;
close all;
I=imread('彩色图像');
I2=rgb2gray(I);
imshow(I);
figure,imshow(I2);
```

8.3

```
clear;
```

```
close all;
I=imread('二值图像');
imshow(I);
figure,imshow(~I);
```

8.4

```
clear;
close all;
I=imread('图像 1');
I2= imread('图像 2');
imshow(I);
figure,imshow(I2);
figure,subplot(1,2,1),imshow(I);
subplot(1,2,2),imshow(I2);
```

实验（二）图像变换上机实验

一.实验目的与要求：

掌握用 MATLAB 或者 C 语言编程进行图像的变换。掌握图像变换的原理与方法；能对图像进行傅里叶变换、DCT 变换等基本操作；理解频域滤波的基本原理，能实现不同类型的频域滤波。

二.实验仪器及软件：

计算机、MATLAB 或者 Visual C++软件，Photoshop 软件。

三.实验地点：

赛博北楼综合应用技术实验室（B405）和微电子技术实验室（B404）。

四.实验原理：

4.1 应用傅立叶变换进行图像处理

傅里叶变换是线性系统分析的一个有力工具，它能够定量地分析诸如数字化系统、采样点、电子放大器、卷积滤波器、噪音和显示点等的作用。通过实验培养这项技能，将有助于解决大多数图像处理问题。对任何想在工作中有效应用数字图像处理技术的人来说，把时间用在学习和掌握博里叶变换上是很有必要的。

4.2 傅立叶（Fourier）变换的定义

对于二维信号，二维 Fourier 变换定义为：

$$F(u,v)=\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)e^{-j2\pi(ux+vy)}dxdy$$

$$f(x,y)=\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} F(u,v)e^{j2\pi(ux+vy)}dudv \qquad e^{j\theta}=\cos\theta+j\sin\theta$$

二维离散傅立叶变换为：

$$F(u,v)=\frac{1}{MN}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x,y)e^{-j2\pi(ux/M+vy/N)} \quad \text{for } u=0,1,2,...,M-1, v=0,1,2,...,N-1$$

$$f(x,y)=\sum_{u=0}^{M-1}\sum_{v=0}^{N-1} F(u,v)e^{j2\pi(ux/M+vy/N)} \quad \text{for } x=0,1,2,...,M-1, y=0,1,2,...,N-1$$

图像的傅立叶变换与一维信号的傅立叶变换变换一样，有快速算法，有关傅立叶变换的快速算法的程序不难找到。实际上，现在有实现傅立叶变换的芯片，可以实时实现傅立叶变换。

4.3 傅立叶变换后得到的频域图像的特点

对于一般的空间域图像，经过傅立叶变换可得到其频率域图像。频率域图像的能量一般主要集中在低频部分，表现为低频系数较大；而高频部分的能量比较弱，表现为大部分高频系数较小。

五.实验内容：

5.1 实验前的准备工作：

a. 了解傅立叶变换的基本原理；

b. 了解 DCT 变换的基本原理；

c. 了解傅立叶变换的实现方法；

d. 了解 DCT 变换的实现方法；

e. 了解实验使用的相关函数。

5.2 实验操作要求：

a. 使用 Photoshop 软件生成一幅大小适中的灰度图像，用于实验；

b. 用 MATLAB 或者 C 语言编程将实验用图进行傅立叶变换，显示其频谱；然后生成滤波函数，对其频谱进行滤波半径分别为 5，10，25，50 的滤波操作，并显示其结果；最后将滤波后的频谱经过反傅立叶变换还原成图像，显示结果。

六.实验报告内容：

6.1 分析经过傅立叶变换得到的频谱图像的特点，并分析其原因；

6.2 比较滤波半径不同时的滤波结果，并分析原因；

6.3 对关键代码进行注释。

七.思考题：

7.1 如何建立傅立叶变换得到的频谱分布与原始图 像之间的对应关系？

八.参考代码：

```
Image=imread('原图像');
subplot(2,2,1)
imshow(Image);
title('原图');
Spectrum=fft2(Image);
subplot(2,2,2)
imshow(Spectrum);
title('FFT 变换结果');
subplot(2,2,3)
Spectrum=fftshift(Spectrum);
imshow(Spectrum);
title('零点平移');
subplot(2,2,4)
imshow(log(abs(Spectrum)),[]);
title('系数分布图');

%低通滤波
figure;                %建立一张空白图纸
subplot(2,2,1)
imshow(log(abs(Spectrum)),[]);
title('系数分布图');
Filter=zeros(180,240);      %滤波数组赋初值,全零
r=50;                  %滤波窗口半径,从中心到半径窗口内滤波数组赋值 1
for i=(180/2-r+1):(180/2+r);
   for j=(240/2-r+1):(240/2+r);
```

```
        Filter(i,j)=1;
    end;
end;
subplot(2,2,2)
imshow(Filter,[]);
title('滤波窗口');
SpectrumN=Filter.*Spectrum;    %频谱与滤波模板卷积
subplot(2,2,3)
imshow(log(abs(SpectrumN)),[]);
title('滤波后频谱');
SpectrumN=ifftshift(SpectrumN);
I2=ifft2(SpectrumN);
subplot(2,2,4)
imshow(abs(I2),[]);
title('反变换图像');
```

实验（三）图像增强上机实验一

一.实验目的：

掌握用 MATLAB 或者 C 语言编程进行图像的灰度变换和直方图均衡化。

二.实验仪器：

计算机、MATLAB 或者 Visual C++软件、Photoshop 软件。

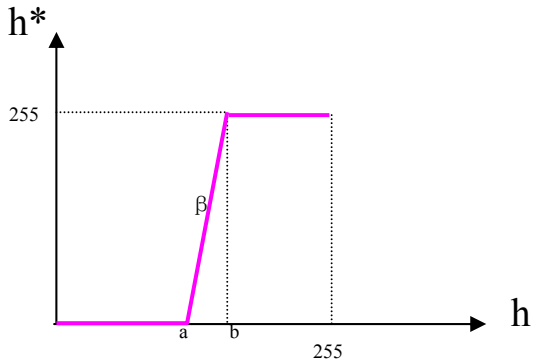
三.实验地点：

赛博北楼多媒体实验室（310）。

四.实验原理：

4.1 直接灰度变换

$$h^*(x,y)=\begin{cases}0 & h(x,y)<=a\\ \dfrac{255}{b-a}h(x,y)-\dfrac{255a}{(b-a)} & h(x,y)\in(a,b)\\ 255 & h(x,y)>=b\end{cases}$$



4.2 直方图均衡化

实现直方图均衡化的实现步骤：

1. 列出原始图像的灰度级 $f_j, j=0,1,\cdots,L-1$, 其中 L 是灰度级的个数。

2. 统计各灰度级的像素数目 $n_j, j = 0,1,\cdots,L-1$。

3. 计算原始图像直方图各灰度级的频数 $P_f(f_j) = n_j / n, j = 0,1,\cdots,L-1$，其中 n 为原始图像总的像素数目。

4. 计算累积分布函数 $C(f) = \sum_{j=0}^{k} P_f(f_j), j = 0,1,\cdots,k,\cdots L-1$。

5. 应用以下公式计算映射后的输出图像的灰度级 $g_i, i = 0,1,\cdots,k,\cdots,P-1$，P 为输出图像灰度级的个数：

$$g_i = INT[(g_{\max} - g_{\min})C(f) + g_{\min} + 0.5]$$

其中，INT 为取整符号

6．统计映射后各灰度级的像素数目 $n_i, i = 0,1,\cdots,k,\cdots,P-1$。

7．计算输出直方图 $P_g(g_i) = n_i / n, i = 0,1,\cdots,P-1$。

8．用 f_j 和 g_i 的映射关系修改原始图像的灰度级，从而获得直方图近似为均匀分布的输出图像。

五.实验内容：

5.1 实验前的准备工作：

a．了解图像增强的分类；

b．了解图像空域变换增强的主要方法；

c．了解图像空域滤波增强的主要方法；

d．了解实验使用的相关函数；

e．了解 Photoshop 软件的相关功能。

5.2 实验操作：

a．用 MATLAB 或者 C 语言编程，对一副图像分别进行直接灰度变换，显示变换结果；

b．用 MATLAB 或者 C 语言编程，对原图像进行直方图均衡化处理，同屏显示处理前后图像及其直方图，比较异同；

c．用 Photoshop 软件对图像进行灰度变换等处理，观察处理结果。

六.实验报告内容：

6.1 分析灰度变换的参数设置中 a 和 b 怎样取，增强后的图像有什么特点；

6.2 分析直方图均衡化后得到的图像的特点，描述其算法步骤；

6.3 对关键代码进行注释。

七.思考题：

7.1 灰度映射中，三种映射规则各适用于什么情况的应用中？

八.参考代码：

a.

```
I=imread('原图像');
imshow(I);
figure,imhist(I);
J=imadjust(I,[0.15 0.9], [0 1]);
figure,imshow(J);
figure,imhist(J);
```

b.

```
I=imread('pout.tif');  % 读取 MATLAB 自带的 potu.tif 图像
imshow(I);
figure,imhist(I);
[J,T]=histeq(I,64);     % 图像灰度扩展到 0~255，但是只有 64 个灰度级
figure,imshow(J);
figure,imhist(J);
figure,plot((0:255)/255,T); % 转移函数的变换曲线
J=histeq(I,32);
figure,imshow(J);  % 图像灰度扩展到 0~255，但是只有 32 个灰度级
figure,imhist(J);
```

或者自己编写直方图均衡化算法

```
clc;
f=imread('pout.tif');
subplot(2,1,1);
imshow(f,[0,255]);
q=zeros(1,256);
for x=1:291
   for y=1:240
      q(f(x,y)+1)=q(f(x,y)+1)+1;
   end
end
s=q./(291*240);
X=0:255;
subplot(2,1,2);
bar(X,s');

figure;
t=zeros(1,256);
t(1)=s(1);
for i=2:256
   t(i)=t(i-1)+s(i);
end
subplot(2,1,1);
bar(X,t');
t0=floor(255*t+0.5);
subplot(2,1,2);
bar(X,t0');

figure;
t1=zeros(1,256);
for i=1:256
   t1(t0(i)+1)=s(i)+t1(t0(i)+1);
end
subplot(2,1,1);
bar(X,t1');
```

```
f1=zeros(291,240)
for x=1:291
   for y=1:240
      f1(x,y)=t0(f(x,y)+1);
   end
end
subplot(2,1,2);
imshow(f1,[0,255]);
```

实验（四）图像增强上机实验二

一.实验目的：

掌握用 MATLAB 或者 C 语言编程进行图像的空间域平滑滤波、频域低通滤波和频域高通滤波。

二.实验仪器：

计算机、MATLAB 或者 Visual C++软件、Photoshop 软件。

三.实验地点：

赛博北楼多媒体实验室（310）。

四.实验原理：

4.1 中值滤波和均值滤波的基本原理；

4.2 频域低通和高通滤波的基本原理。

五.实验内容：

5.1 实验前的准备工作：

a. 了解图像增强的分类；

b. 了解图像空域和频域增强的主要方法；

c. 了解实验使用的相关函数；

d. 了解 Photoshop 软件的相关功能。

5.2 实验操作：

a. 对原图像加入椒盐噪声，分别采用 3x3、5x5 和 7x7 的模板进行中值滤波，同屏显示处理前后图像，比较异同；

b．对原图像加入高斯噪声，用 4-邻域平均法和 8-邻域平均法平滑加噪声图像（图像四周边界不处理，下同），同屏显示原图像、加噪声图像和处理后的图像；并请设计新的模板，对图像进行去噪操作。

c．用 MATLAB 或者 C 语言编程，利用巴特沃斯（Butterworth）低通滤波器对受噪声干扰的图像进行平滑处理，显示处理结果；

d．用 MATLAB 或者 C 语言编程，利用巴特沃斯（Butterworth）高通滤波器对图像进行锐化处理，显示处理结果；

e．用 Photoshop 软件对图像进行平滑和锐化等处理，观察处理结果。

六.实验报告内容：

6.1 比较在中值滤波中不同大小的模板的滤波效果，并分析原因；

6.2 比较均值滤波中 4-邻域平均法和 8-邻域平均法的滤波效果，并分析原因；

6.3 分析巴特沃斯低通和高通滤波中，阶数和截断频率的调整对滤波效果的影响；

6.4 对关键代码进行注释。

七.思考题：

7.1 图像增强方法（点处理和邻域处理）对图像的能量、灰度级数目、空间频谱、直方图和熵值分别有什么影响？

7.2 对于椒盐噪声和高斯噪声，分别采用哪种滤波器（均值滤波器和中值滤波器）效果更好？

八.参考代码：

a.

```
I=imread('eight.tif');
imshow(I) ;
J2=imnoise(I,'salt & pepper',0.04); % 叠加密度为 0.04 的椒盐噪声。
figure,imshow(J2);

I_Filter1=medfilt2(J2,[3 3]);  %窗口大小为 3×3
figure,imshow(I_Filter1);
I_Filter2=medfilt2(J2,[5 5]);  %窗口大小为 5×5
figure,imshow(I_Filter2);
I_Filter3=medfilt2(J2,[7 7]);  %窗口大小为 7×7
figure,imshow(I_Filter3);
```

b.

```
I=imread('eight.tif');
figure,imshow(I);title('original')
J1=imnoise(I,'gaussian',0,0.02); % 受高斯噪声干扰
figure,imshow(J1);
M4=[0 1 0; 1 0 1; 0 1 0];
M4=M4/4;              % 4 邻域平均滤波
I_filter1=filter2(M4,J1);
figure,imshow(I_filter1);

M8=[1 1 1; 1 0 1; 1 1 1];     % 8 邻域平均滤波
M8=M8/8;
I_filter2=filter2(M8,J1);
 figure,imshow(I_filter2);
```

c.

```
I=imread('原图像');
imshow(I);
```

```
J1=imnoise(I,'salt & pepper');  % 叠加椒盐噪声
figure,imshow(J1);
f=double(J1);    % 数据类型转换，MATLAB 不支持图像的无符号整型计算
g=fft2(f);       % 傅立叶变换
g=fftshift(g);   % 转换数据矩阵
[M,N]=size(g);
nn=2;          % 二阶巴特沃斯(Butterworth)低通滤波器
d0=50;
m=fix(M/2); n=fix(N/2);
for i=1:M
    for j=1:N
        d=sqrt((i-m)^2+(j-n)^2);
        h=1/(1+0.414*(d/d0)^(2*nn)); % 计算低通滤波器传递函数
        result(i,j)=h*g(i,j);
    end
end
result=ifftshift(result);
J2=ifft2(result);
J3=uint8(real(J2));
figure,imshow(J3);                  % 显示滤波处理后的图像
```

d.

```
I=imread('原图像');
imshow(I);
f=double(I);    % 数据类型转换，MATLAB 不支持图像的无符号整型计算
g=fft2(f);       % 傅立叶变换
g=fftshift(g);   % 转换数据矩阵
[M,N]=size(g);
nn=2;          % 二阶巴特沃斯(Butterworth)高通滤波器
d0=5;
m=fix(M/2);
n=fix(N/2);
```

```
for i=1:M
    for j=1:N
      d=sqrt((i-m)^2+(j-n)^2);
      if (d==0)
        h=0;
      else
        h=1/(1+0.414*(d0/d)^(2*nn));% 计算传递函数
      end
result(i,j)=h*g(i,j);
end
end
result=ifftshift(result);
J2=ifft2(result);
J3=uint8(real(J2));
figure,imshow(J3); % 滤波后图像显示
```

实验（五）图像编码上机实验

一.实验目的：

掌握用 MATLAB 或者 C 语言编程进行图像的编码。

二.实验仪器：

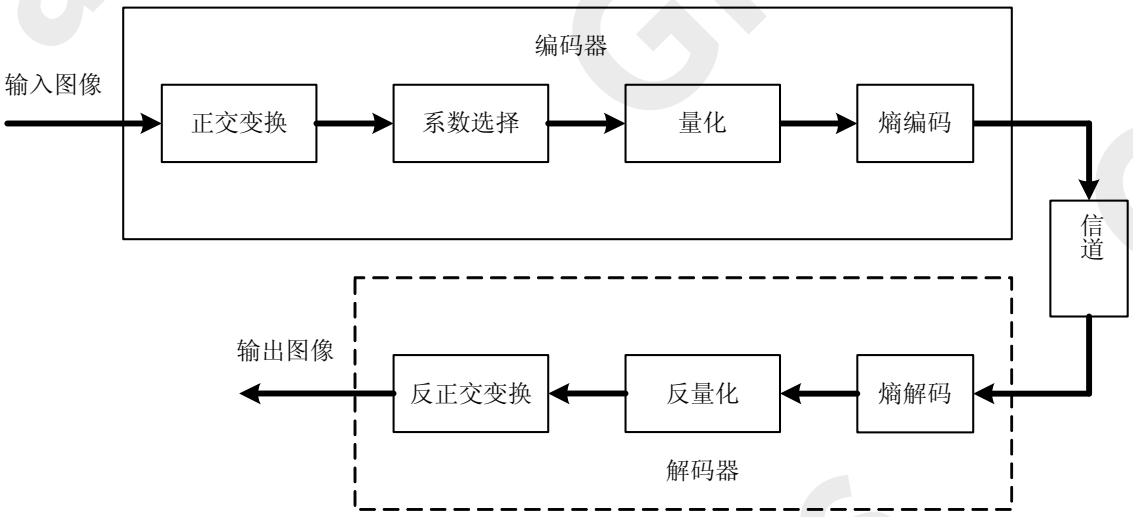
计算机、MATLAB 或者 Visual C++软件、Photoshop 软件。

三.实验地点：

赛博北楼多媒体实验室（310）。

四.实验原理：

4.1 变换编解码的基本流程



4.2 量化与反量化

量化：

$$C(u,v) = Interger\left(Round\left(\frac{F(u,v)}{Q(u,v)} \right) \right),$$

$C(u,v)$：量化器输出

$F(u,v)$：量化器输入

$Q(u,v)$：量化器步长

反量化：

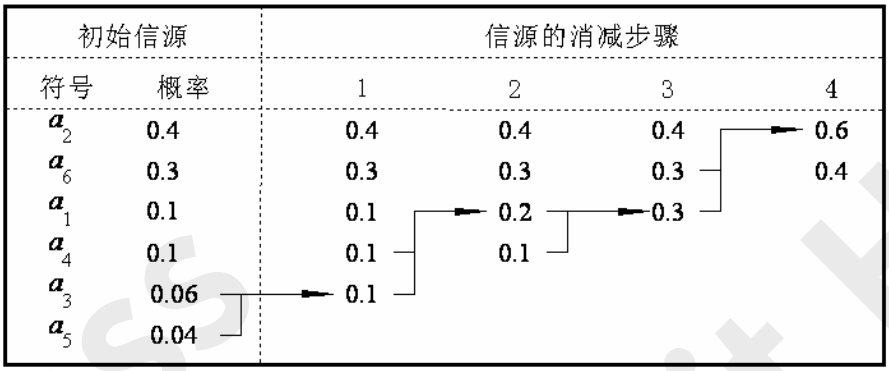
$$F'(u,v) = C(u,v) \times Q(u,v)$$

$C(u,v)$：逆量化器输入

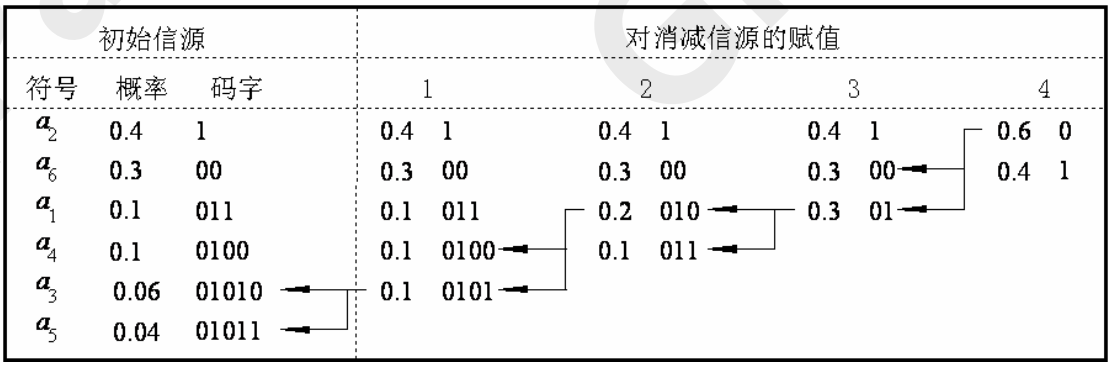
$Q(u,v)$：量化器步长

4.3 哈夫曼编码主要原理

(1) 缩减信源符号数量



(2) 对每个信源符号赋值



五.实验内容：

5.1 实验前的准备工作：

了解图像编码的原理与分类；

a. 了解图像编码的主要方法；

b. 了解主要的图像编码标准；

c．了解实验使用的相关函数。

5.2 实验操作：

a. 构造子图像，并进行二维离散余弦变换，然后利用模板选择保留的系数，对保留的系数进行反变换并进行图像拼接以得到重建图像，比较重建图像与原图像的质量；

b. 更改 a 中的模板，保留更多的低频系数，重建图像并比较其结果；

c. 在 a 的基础上添加量化环节，重建图像并比较其结果；

d. 用 Matlab 实现 Huffman 编码算法程序；要求程序输出显示所有的码字以及编码效率。

e. 用 Photoshop 软件打开一副图像，将其分别保存成 bmp 和 jpg 格式的图像，比较两副图像的质量和文件大小。

六.实验报告内容：

5.1 分析本次实验实现的变换编解码中各个环节的作用；

5.2 经过 DCT 后得到的系数矩阵有什么特点；量化矩阵应采用什么样的设计原则；

5.3 实验 a,b,c 的解压图像的对比，并分析原因；

5.4 对关键代码进行注释。

七.思考题：

7.1 MPEG、JPEG 等常用编码标准的编码过程？

7.2 计算 300*400 像素，8bit 色彩深度索引图像，使用 BMP 格式存储时的数据量？

八.参考代码：

8.1 变换编码：

```
I=imread('原图像');
imshow(I);
clear;close all
I=imread('原图像');
imshow(I);
I=im2double(I);
T=dctmtx(8);
B=blkproc(I,[8 8], 'P1*x*P2',T,T');
Mask=[1 1 1 1 0 0 0 0
      1 1 1 0 0 0 0 0
      1 1 0 0 0 0 0 0
      1 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 0 0 0 0];
B2=blkproc(B,[8 8],'P1.*x',Mask);   % 此处为点乘(.*)
I2=blkproc(B2,[8 8], 'P1*x*P2',T',T);
figure,imshow(I2);                  % 重建后的图像
```

8.2 哈夫曼编码：

```
p=input('please input a number:')  %提示输入界面
n=length(p);
for i=1:n
  if  p(i)<0
fprintf('\n The probabilities in huffman can not less than 0!\n');
 p=input('please input a number:')  %如果输入的概率数组中有小于 0 的值，
                                    %则重新输入概率数组
    end
end
if  abs(sum(p)-1)>0
```

```
    fprintf('\n The sum of the probabilities in huffman can more than 1!\n');
    p=input('please input a number:')  %如果输入的概率数组总和大于 1，则
                                       %重新输入概率数组
end
q=p;
a=zeros(n-1,n);                  %生成一个 n-1 行 n 列的数组
for i=1:n-1
  [q,l]=sort(q)          %对概率数组 q 进行从小至大的排序，并且用 l 数组
                  %返回一个数组，该数组表示概率数组 q 排序前的顺序编号
  a(i,:)=[l(1:n-i+1),zeros(1,i-1)]    %由数组 l 构建一个矩阵，该矩阵表明概
                                      %率合并时的顺序，用于后面的编码
  q=[q(1)+q(2),q(3:n),1];         %将排序后的概率数组 q 的前两项，即概率
                                  %最小的两个数加和，得到新的一组概率序列
end
for i=1:n-1
  c(i,1:n*n)=blanks(n*n);   %生成一个 n-1 行 n 列，并且每个元素的的长度
                            %为 n 的空白数组，c 矩阵用于进行 huffman 编
                            %码，并且在编码中与 a 矩阵有一定的对应关系
end
c(n-1,n)='0';          %由于 a 矩阵的第 n-1 行的前两个元素为进行 huffman 编
                       %码加和运算时所得的最
c(n-1,2*n)='1';        %后两个概率，因此其值为 0 或 1，在编码时设第 n-1
                       %行的第一个空白字符为 0，第二个空白字符 1。
for i=2:n-1
  c(n-i,1:n-1)=c(n-i+1,n*(find(a(n-i+1,:)==1))-(n-2):n*(find(a(n-i+1,:)==1)))
%矩阵 c 的第 n-i 的第一个元素的 n-1 的字符赋值为对应于 a 矩阵中第 n-i+1 行中
%值为 1 的位置在 c 矩阵中的编码值

  c(n-i,n)='0'             %根据之前的规则，在分支的第一个元素最后补 0
  c(n-i,n+1:2*n-1)=c(n-i,1:n-1)  %矩阵 c 的第 n-i 的第二个元素的 n-1 的字
                                 %符与第 n-i 行的第一个元素的前 n-1 个符
```

```
                                %号相同，因为其根节点相同
        c(n-i,2*n)='1'          %根据之前的规则，在分支的第一个元素最后补 1
        for j=1:i-1
         c(n-i,(j+1)*n+1:(j+2)*n)=c(n-i+1,n*(find(a(n-i+1,:)==j+1)-1)+1:n*find(a(n-
i+1,:)==j+1)) %矩阵 c 中第 n-i 行第 j+1 列的值等于对应于 a 矩阵中第 n-i+1 行中
                %值为 j+1 的前面一个元素的位置在 c 矩阵中的编码值
        end
       end                      %完成 huffman 码字的分配
      for i=1:n
       h(i,1:n)=c(1,n*(find(a(1,:)==i)-1)+1:find(a(1,:)==i)*n)  %用 h 表示最后的
%huffman 编码，矩阵 h 的第 i 行的元素对应于矩阵 c 的第一行的第 i 个元素
        ll(i)=length(find(abs(h(i,:))~=32))       %计算每一个 huffman 编码的长度
      end
      l=sum(p.*ll);        %计算平均码长
      fprintf('\n huffman code:\n');
      h
      hh=sum(p.*(-log2(p)));       %计算信源熵
       fprintf('\n the huffman effciency:\n');
       t=hh/l                      %计算编码效率
```

实验（六）图像分割上机实验

一.实验目的：

掌握用 MATLAB 或者 C 语言编程进行图像的分割。

二.实验仪器：

计算机、MATLAB 或者 Visual C++软件、Photoshop 软件。

三.实验地点：

赛博北楼多媒体实验室（310）。

四.实验原理：

4.1 边缘检测的基本原理；

4.2 单阈值分割的基本原理；

4.3 空间聚类算法的基本原理；

原理：令$x=(x_1,x_2)$代表一个特征空间的坐标，$g(x)$ 代表在该位置的特征值，*K*-均值法是要最小化如下指标：

$$E=\sum_{j=1}^{K}\sum_{x\in Q_j^{(i)}}\left\|g(x)-\mu_j^{(i+1)}\right\|^2$$

$Q_j^{(i)}$代表在第i次迭代后赋给类 j 的特征点集合，μ_j 表示第 j 类的均值。

算法流程：

(1)任意选 K 个初始类均值 $\mu_1^{(1)},\mu_2^{(1)},\cdots,\mu_K^{(1)}$；

(2)特征点赋类：$\mathbf{x}\in Q_l^{(i)}$ 如果 $\left\|g(\mathbf{x})-\mu_l^{(i)}\right\|<\left\|g(\mathbf{x})-\mu_j^{(i)}\right\|$；

(3)更新类均值：$\mu_j^{(i+1)}=\dfrac{1}{N_j}\sum_{\mathbf{x}\in Q_j^{(i)}}g(\mathbf{x})$；

(4)判断算法收敛

五.实验内容：

5.1 实验前的准备工作：

a. 了解图像分割算法的分类；
b. 了解图像分割的主要方法；
c. 了解实验使用的相关函数；
d. 了解 Photoshop 中的相关功能。

5.2 实验操作：

a. 用 MATLAB 或者 C 语言编程，对一副图像分别用 Prewitt 算子、LOG 算子和 Canny 算子检测图像的边缘，并给出结果；
b. 用 MATLAB 或者 C 语言编程，对一副图像进行阈值分割，显示处理结果；
c. 用 MATLAB 或者 C 语言编程，利用空间聚类方法进行图像分割，显示处理结果。
d. 用 Photoshop 软件对图像进行锐化或边缘检测，观察边缘检测效果；
e. 用 Photoshop 软件中的三种套索工具分别将图像中的一部分分割出来。

六.实验报告内容：

6.1 分析 Prewitt 算子、LOG 算子和 Canny 算子的边缘检测效果；

6.2 分析单阈值分割中阈值的选取办法，如果阈值选取过大或过小会有什么结果；

6.3 分析空间聚类算法中初始值的选择办法；

6.4 对关键代码进行注释。

七.思考题：

1．评价最常用的几种边缘增强算子的性能？

2．不同种类分割方法的特点分析与性能比较？

八.参考代码：

a.

```
I = imread('原图像');
BW1 = edge(I,'prewitt',0.04);          %  0.04 为梯度阈值
figure(1);
imshow(I);
figure(2);
imshow(BW1);

I = imread('原图像');
BW1 = edge(I,'log',0.003);  % σ=2
imshow(BW1);title('σ=2')
BW1 = edge(I,'log',0.003,3); % σ=3
figure, imshow(BW1);title('σ=3')

I = imread('原图像');
imshow(I);
BW1 = edge(I,'canny',0.2);
figure,imshow(BW1);
```

b.

```
I=imread('原图像');
imhist(I);        % 观察灰度直方图， 确定阈值
I1=im2bw(I,阈值/255); % im2bw 函数需要将灰度值转换到[0,1]范围内
figure,imshow(I1);
```

c.

```
I=imread('原图像');
I1=I(:,:,1);
I2=I(:,:,2);
I3=I(:,:,3);
[y,x,z]=size(I);
d1=zeros(y,x);
d2=d1;
myI=double(I);
I0=zeros(y,x);
for i=1:x
   for j=1:y
%根据欧式距离聚类
d1(j,i)=sqrt((myI(j,i,1)-180)^2+(myI(j,i,2)-180)^2+(myI(j,i,3)-180)^2);
d2(j,i)=sqrt((myI(j,i,1)-200)^2+(myI(j,i,2)-200)^2+(myI(j,i,3)-200)^2);
            if (d1(j,i)>=d2(j,i))
          I0(j,i)=1;
      end
    end
end
figure(1);
imshow(I);
% 显示 RGB 空间的灰度直方图，确定两个聚类中心(180,180,180)和
(200,200,200)
figure(2);
subplot(1,3,1);
imhist(I1);
subplot(1,3,2);
imhist(I2);
subplot(1,3,3);
imhist(I3);
figure(3);
imshow(I0);
```