JNDX_M

**《图像处理设计》**

**课程设计**

**题目： 简单图像处理系统**

数字媒体 学院 数字媒体技术 专业

学 号 1030515424

班 级 数字媒体技术1504

学生姓名 王林超

指导教师 狄 岚

二〇一八年六月

目录

[**第1章 目的与要求** 5](#_Toc517729487)

[**1.1 本课程设计目的** 5](#_Toc517729488)

[**1.2 要求** 5](#_Toc517729489)

[**第2章 设计的内容** 7](#_Toc517729490)

[**2.1主要功能** 7](#_Toc517729491)

[2.1.1图像变换 7](#_Toc517729492)

[2.1.2图像增强 7](#_Toc517729493)

[2.1.3图像分割 7](#_Toc517729494)

[2.1.4图像处理 7](#_Toc517729495)

[2.1.5图像表达 7](#_Toc517729496)

[**2.2 原理方法** 7](#_Toc517729497)

[2.2.1图像变换 7](#_Toc517729498)

[2.2.2图像增强 8](#_Toc517729499)

[2.2.3图像分割 8](#_Toc517729500)

[2.2.4图像处理 9](#_Toc517729501)

[2.2.5图像表达 9](#_Toc517729502)

[**第3章 系统的总体设计方案** 10](#_Toc517729503)

[**3.1系统框架图** 10](#_Toc517729504)

[**第4章 系统各模块实现程序** 11](#_Toc517729505)

[**4.1.图片的打开与保存** 11](#_Toc517729506)

[**4.1.1功能简介:** 11](#_Toc517729507)

[**4.1.2核心代码:** 11](#_Toc517729508)

[**4.1.3效果截图:** 11](#_Toc517729509)

[**4.2.图像的几何变化1** 12](#_Toc517729510)

[**4.2.1功能简介:** 12](#_Toc517729511)

[**4.2.2核心代码:** 12](#_Toc517729512)

[**4.2.3效果截图:** 12](#_Toc517729513)

[**4.3.图像的几何变化2** 13](#_Toc517729514)

[**4.3.1功能简介:** 13](#_Toc517729515)

[**4.3.2核心代码:** 13](#_Toc517729516)

[**4.3.3效果截图:** 14](#_Toc517729517)

[**4.4.图像特殊几何变化1** 15](#_Toc517729518)

[**4.4.1功能简介:** 15](#_Toc517729519)

[**4.4.2核心代码:** 15](#_Toc517729520)

[**4.4.3效果截图:** 15](#_Toc517729521)

[**4.5.图像特殊几何变化2** 16](#_Toc517729522)

[**4.5.1功能简介:** 16](#_Toc517729523)

[**4.5.2核心代码:** 16](#_Toc517729524)

[**4.5.3效果截图:** 16](#_Toc517729525)

[**4.6.图像的傅里叶变化** 17](#_Toc517729526)

[**4.6.1功能简介:** 17](#_Toc517729527)

[**4.6.2核心代码:** 17](#_Toc517729528)

[**4.6.3效果截图:** 17](#_Toc517729529)

[**4.7.图像的点运算** 18](#_Toc517729530)

[**4.7.1功能简介:** 18](#_Toc517729531)

[**4.7.2核心代码:** 18](#_Toc517729532)

[**4.7.3效果截图:** 19](#_Toc517729533)

[**4.8.图像的灰度拉伸** 19](#_Toc517729534)

[**4.8.1功能简介:** 19](#_Toc517729535)

[**4.8.2核心代码:** 19](#_Toc517729536)

[**4.8.3效果截图:** 20](#_Toc517729537)

[**4.9.图像的直方图均衡化** 20](#_Toc517729538)

[**4.9.1功能简介:** 20](#_Toc517729539)

[**4.9.2核心代码:** 20](#_Toc517729540)

[**4.9.3效果截图:** 21](#_Toc517729541)

[**4.10.图像的平滑处理算法** 22](#_Toc517729542)

[**4.10.1功能简介:** 22](#_Toc517729543)

[**4.10.2核心代码:** 22](#_Toc517729544)

[**4.10.3效果截图:** 22](#_Toc517729545)

[**4.11.图像的彩色增强** 22](#_Toc517729546)

[**4.11.1功能简介:** 22](#_Toc517729547)

[**4.11.2核心代码:** 23](#_Toc517729548)

[**4.11.3效果截图:** 23](#_Toc517729549)

[**4.12.图像的边缘检测** 24](#_Toc517729550)

[**4.12.1功能简介:** 24](#_Toc517729551)

[**4.12.2核心代码:** 24](#_Toc517729552)

[**4.12.3效果截图:** 25](#_Toc517729553)

[**4.13.图像的阈值分割** 26](#_Toc517729554)

[**4.13.1功能简介:** 26](#_Toc517729555)

[**4.13.2核心代码:** 26](#_Toc517729556)

[**4.13.3效果截图:** 27](#_Toc517729557)

[**4.14.二值图像处理** 28](#_Toc517729558)

[**4.14.1功能简介:** 28](#_Toc517729559)

[**4.14.2核心代码:** 28](#_Toc517729560)

[**4.14.3效果截图:** 28](#_Toc517729561)

[**4.15.图像的表达** 29](#_Toc517729562)

[**4.15.1功能简介:** 29](#_Toc517729563)

[**4.15.2核心代码:** 29](#_Toc517729564)

[**4.15.3效果截图:** 30](#_Toc517729565)

[**第5章课 程设计总结与体会** 31](#_Toc517729566)

[**参考文献** 32](#_Toc517729567)

**第1章** **目的与要求**

**1.1 本课程设计目的**

数字图像处理又称为计算机图像处理,是指将图像信号转换成数字信号并利用计算机对其进行处理的过错。图像处理技术始于20世纪20年代，真正的发展出现于20世纪50年代，进入20世纪80年代后,数字图像处理研究方兴未艾，形成热门的研究领域。

近十几年来,由于大规模集成电路技术和计算机技术的迅猛发展、离散数学理论的创立与完善，数字图像技术正逐渐成为其他科学技术领域中不可缺少的一项重要工具。数字图像处理技术也在从空间探索到微观研究、从军事领域到农业生产、从科学教育到娱乐游戏等越来越多的领域得到广泛应用。

数字图像处理主要研究内容包括图像变化、图像增强、图像复原、图像压缩、图像分割等，希望通过本次课程设计，让我们能够深刻理解和掌握图像的理论和处理方法。

**1.2 要求**

利用MATLAB或其他语言，设计一个简易图像处理系统:

该系统需要能够实现下列需求:

1、 对图像文件（bmp、jpg、等）进行打开、保存、退出等功能操作；

2、 几何变换

（1）图像的平移、缩放、旋转、镜像、转置等；

（2）特殊的几何变换，如：百叶窗效果、分块显示效果等。

3、图像增强

（1）空域中的各种增强方法

a) 空域中的点运算：对比度的增加、减少，亮度的增加、减少，图像 反色

b) 图像的灰度拉伸、直方图的均衡化

c) 空间域平滑算法（如均值滤波、中值滤波等）

（2）频域的各种增强方法：低通滤波、高通滤波等。

（3）色彩增强：

伪彩色增强:将一幅灰度图像转换成一幅真彩色图像；

假彩色增强: 将三幅灰度图像合成一幅真彩色图像；

真彩色增强：

a) 将一副真彩色图像进行RGB分离；

b) 将某一分离后的单色图像增加、减少；

c) 产生新的真彩色图像。

4、图像分割：

（1）边缘检测

分别用Roberts算子、Sobel算子、Laplacian算子等作边缘检测；

（2）阈值分割

a) 交互式阈值分割；

b) 迭代阈值分割。

5、二值图像处理：膨胀、腐蚀、开运算与比运算。

**第2章 设计的内容**

**2.1主要功能**

本系统主要包含4个部分的功能，各部分功能如下

2.1.1图像变换

包含两部分变化:一是通过几何运算实现图像的平移、缩放、旋转、镜像、转置以及百叶窗效果、分块显示效果。二是进行傅立叶变换展示图像的频域。

2.1.2图像增强

包含三种类型的增强: 1.空域中的各增强方法:改变对比度,亮度,反色,灰度拉伸,直方图均衡化,平滑2. 频域的各种增强方法:低通滤波、高通滤波3. 色彩增强:伪彩色增强、假彩色增强、真彩色增强。

2.1.3图像分割

包含两种功能:1.边缘检测:分别用三种算子进行边缘检测2.阈值分割:交互式和迭代阈值分割

2.1.4图像处理

主要为对二值图像的四种处理:种膨胀、腐蚀、开运算与闭运算。

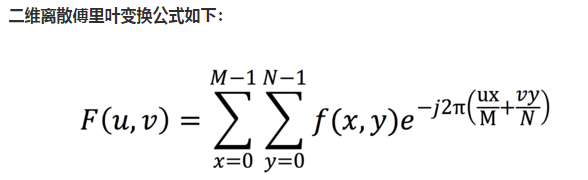
2.1.5图像表达

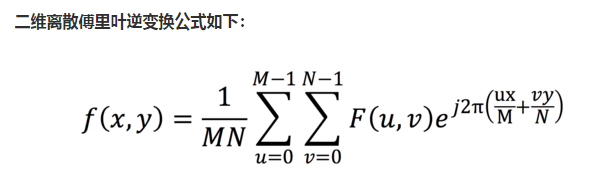
图像的3D立体表达和图像和图像的四叉树表达

**2.2 原理方法**

2.2.1图像变换

图像变化是主要是通过对图像矩阵的排列进行一定的转置而实现的。图像的傅立叶变换是通过离散傅里叶变换DFT将图像从空间变换到频率域。公式如下:





2.2.2图像增强

空域中的图像增强方法主要是通过图像的线性点运算实现:线性点运算是指输入图像的灰度级与目标图像的灰度级呈线性关系。线性点运算的灰度变换函数形式可以采用线性方程描述,即s=ar+b,r为输入点的灰度值，s为相应输出点的灰度值。通过线性改变a,b的值实现对比度,亮度,反色,灰度拉伸等效果。直方图均衡化是通过让图像的直方图均匀分布从而让图像的色调给人以协调的感觉。

空间域的平滑采用模板处理方法对图像进行滤波，以除去图像噪声或增强图像的细节，模板的中心从一个像素向另一个像素移动，通过模板运算得到该点的输出。

频域的图像增强方法的原理是通过傅里叶变换把空间域混叠的成分在频率域中分离出来,从而提取或滤去相应的图像的成分。核心为傅里叶变换。

图像的色彩增强分为伪彩色增强、假彩色增强、真彩色增强。伪彩色增强是将灰度图转化物彩色图像，原理将灰度图像的各个灰度匹配到彩色空间中的一点，从而使单色图像映射成彩色图像。假彩色增强是将可见光谱以外的图像合成一幅真彩色图像；真彩色增强的原理是将一副真彩色图像进行RGB分离后将某一分离后的单色图像增加、减少；从而产生新的真彩色图像。

2.2.3图像分割

图像的边缘检测原理与图像在空间域平滑的原理一致，也是采用模板处理方法对图像进行滤波，以增强图像的细节，模板的中心从一个像素向另一个像素移动，通过模板运算得到该点的输出。在这个系统中采用了三种模板即三种算子即Roberts算子、Sobel算子、Laplacian算子。Roberts算子利用局部差分算子寻找边缘，边缘定位精度较高，但容易丢失一部分边缘，同时由于图像没有经过平滑处理，因此不具备抑制噪声的能力。该算子对具有陡峭边缘且含噪声少的图像效果较好。Sobel算子是对图像先做加权平均处理，然后再做微分运算，所不同的是平滑部分的权值有些差异，因此它们对噪声具有一定的抑制能力。Sobel算子的边缘定位效果不错，但检测出的边缘容易出现多像素宽度。Laplacian算子是无方向的二阶微分算子，对图像中的阶跃型边缘定位准确，该算子对噪声非常敏感，它使噪声成分得到加强。这两个特性使得该算子容易丢失一部分边缘的方向信息，造成一些不连续的检测边缘。

图像的阈值分割是通过设置阈值，把像素点按灰度级分成若干类，从而实现图像分割。交互式阈值分割为人工给定一个值，迭代式阈值分割则是在开始时选择一个与之作为初始估计值，然后按某种策略不断得改进这一估计值，直到满足给定的准则为止。

2.2.4图像处理

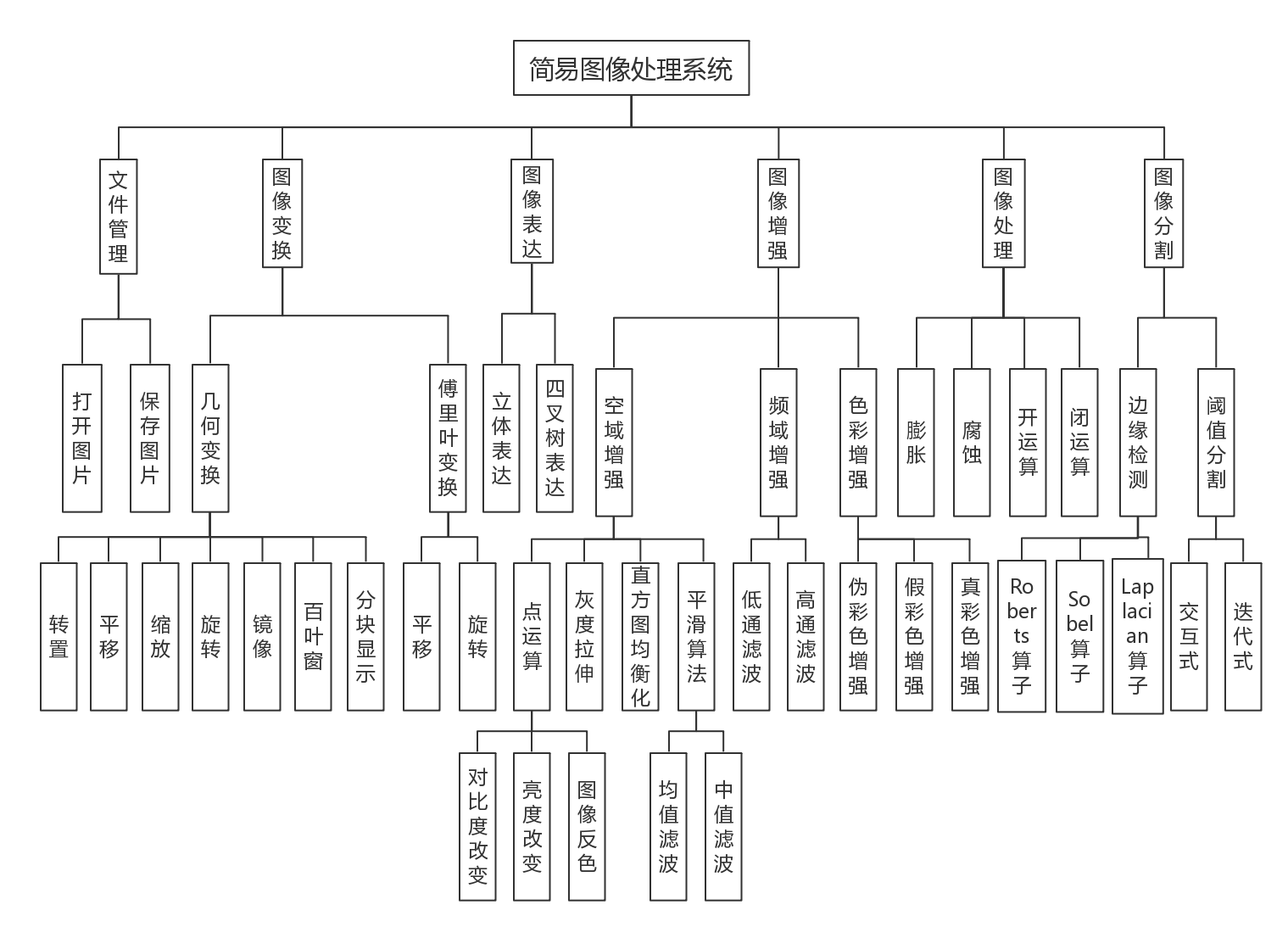
腐蚀是消除物体的所有边界点的一种过程,其结果使剩下的物体沿其周边比原物体小一个像素的面积。膨胀为将与物体接触的所有背景点合并到该物体中的过程，使物体的面积增大了相应数量的点。开运算为先腐蚀再膨胀，可使对象的轮廓变得光滑，闭运算为先膨胀再腐蚀，可消弭狭窄的间断和长细的鸿沟。

2.2.5图像表达

四叉树的的每个节点下至多可以有四个子节点，通常把一部分二维空间细分为四个象限或区域并把该区域里的相关信息存入到四叉树节点中。方法是采用递归的方法对该矩形进行划分分区块，分完后再往里分，直到每一个子矩形区域里只包含一个对象为止。

**第3章 系统的总体设计方案**

**3.1系统框架图**

**3.2 系统运行环境**

win10操作系统,matlab 实现

**第4章 系统各模块实现程序**

**4.1.图片的打开与保存**

**4.1.1功能简介:**

打开图片:通过matbla自带ui获取文件路径,路径非空显示图片

关闭图片:通过matbla自带ui获取用户指定路径,路径非空保存图片

**4.1.2核心代码:**

打开图片

[filename,pathname]=uigetfile( ...

{'\*.bmp;\*.jpg;\*.png;\*.jpeg','Image Files(\*.bmp,\*.jpg,\*.png,\*.jpeg)';

'\*,\*','All Files(\*,\*)'},...

'Pick an image');

if isequal(filename,0)||isequal(pathname,0)

return;

end

fpath=[pathname filename];

保存图片

function m\_file\_save\_Callback(hObject, eventdata, handles)

[filename, pathname]=uiputfile({'\*.bmp','BMP files';'\*.jpg','JPG files'},'Save an Image');

if isequal(filename,0)||isequal(pathname,0)

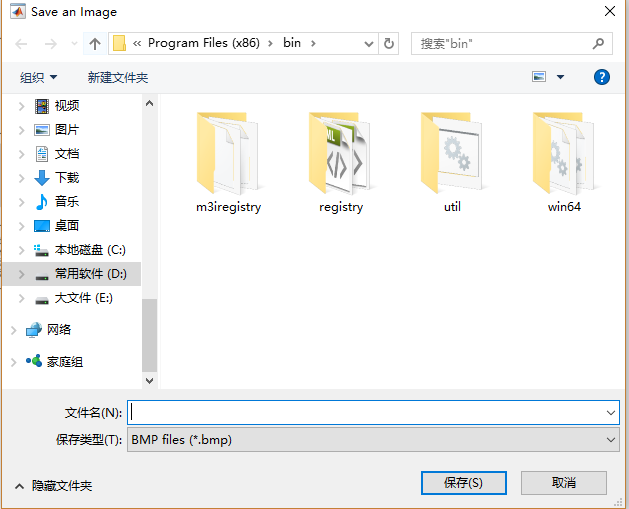
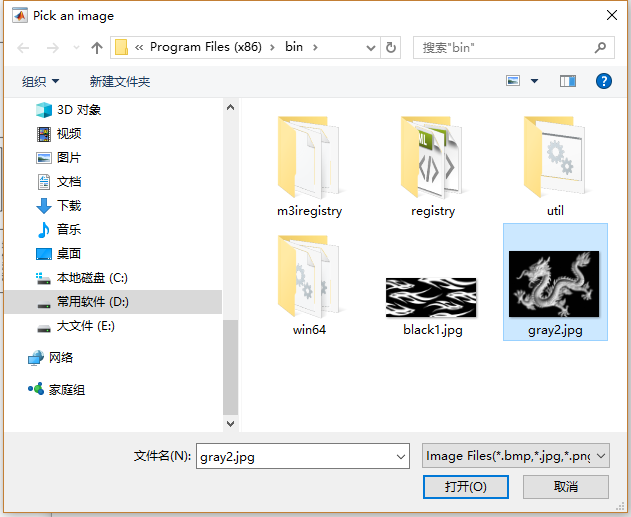
return;

else

fpath=fullfile(pathname,filename);

end

**4.1.3效果截图:**



**4.2.图像的几何变化1**

**4.2.1功能简介:**

实现图片的平移、缩放、旋转

**4.2.2核心代码:**

平移(ab代表平移量, img\_orig代表原图)

A=img\_orig;

se=translate(strel(1),[a b]);

B=imdilate(A,se);

缩放(m代表缩放倍数)

A=img\_orig;

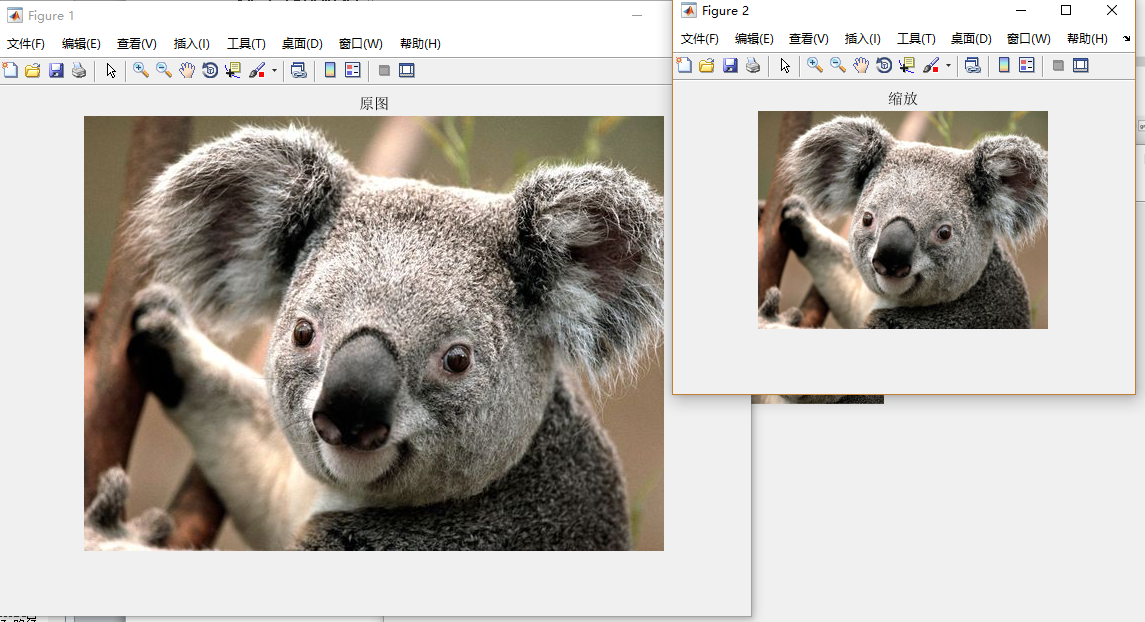
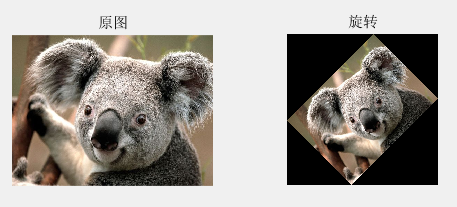
D=imresize(A,m);

旋转(b代表旋转角度)

A=img\_orig;

C=imrotate(A,b);

**4.2.3效果截图:**



**4.3.图像的几何变化2**

**4.3.1功能简介:**

通过对图片矩阵的重新排列

实现对图片的镜像、转置

**4.3.2核心代码:**

上下镜像，左右镜像

A=img\_orig;

E=size(A);

F=size(A);

for i=1:3

a0=[];

L=A(:,:,i);

for j=1:E(1)

H=[L(j,:);a0]; a0=H;

end

M(:,:,i)=H;

end

for i=1:3

a0=[];

L=A(:,:,i);

for j=1:F(2)

H=[L(:,j),a0];

a0=H;

end

N(:,:,i)=H;

end

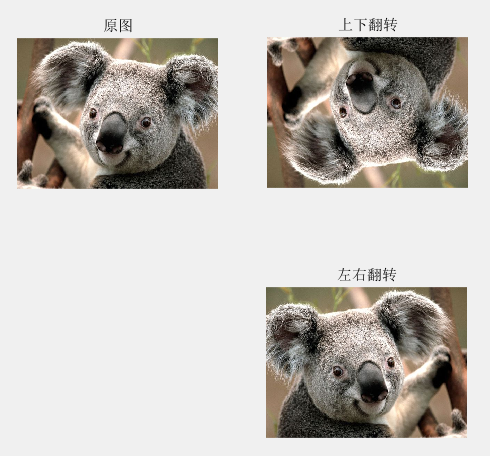
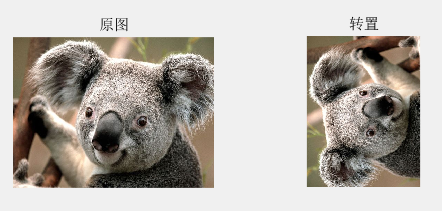
转置

I=img\_orig;

T=affine2d([0 1 0;1 0 0;0 0 1]);

dst=imwarp(I,T);

**4.3.3效果截图:**

**4.4.图像特殊几何变化1**

**4.4.1功能简介:**

百叶窗:通过图像乘以一个遮罩实现;

**4.4.2核心代码:**

百叶窗

I=img\_orig;

size\_info=size(I);

height=size\_info(1);

width=size\_info(2);

mask=zeros(height,width);

num=10;

for j=1:floor(height/num)

for i=1:num

white\_begin=floor(1+(i-1)\*(height/num));

white\_end=ceil(j+(i-1)\*(height/num));

mask(white\_begin:white\_end,:)=1;

end

J=cat(3,mask,mask,mask);

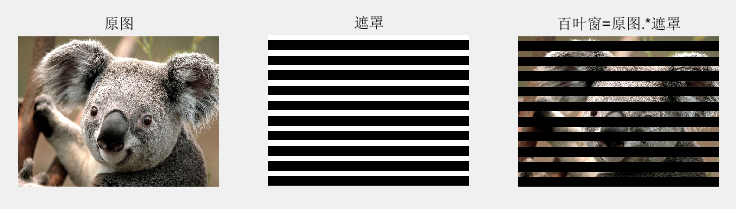
J8=uint8(J);

I8=uint8(I);

L=immultiply(I8,J8);

drawnow

**4.4.3效果截图:**



**4.5.图像特殊几何变化2**

**4.5.1功能简介:**

分块显示:将图片矩阵分成数个小矩阵实现

**4.5.2核心代码:**

分块显示

I=img\_orig;

I = imresize(I,[64\*4 64\*5]);

rs = size(I, 1); cs = size(I, 2);

sz = 64;

numr = rs/sz;

numc = cs/sz;

ch = sz; cw = sz;

t1 = (0:numr-1)\*ch + 1; t2 = (1:numr)\*ch;

t3 = (0:numc-1)\*cw + 1; t4 = (1:numc)\*cw;

figure;

k = 0;

for i = 1 : numr

for j = 1 : numc

temp = I(t1(i):t2(i), t3(j):t4(j), :);

k = k + 1;

subplot(numr, numc, k);

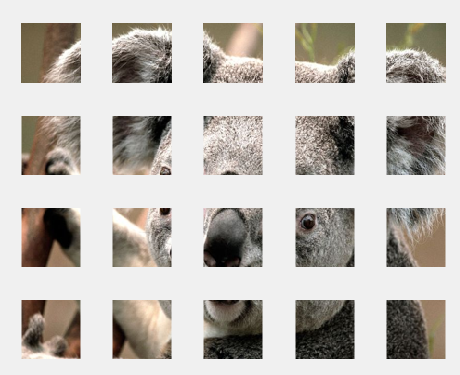
imshow(temp);

pause(0.1);

end

end

**4.5.3效果截图:**



**4.6.图像的傅里叶变化**

**4.6.1功能简介:**

傅里叶变换:将图片从时域转到频域来显示,并验证了其平移不变性、旋转一致性”；

**4.6.2核心代码:**

傅里叶变换

A=im2double(img\_orig);

B=fft2(A);

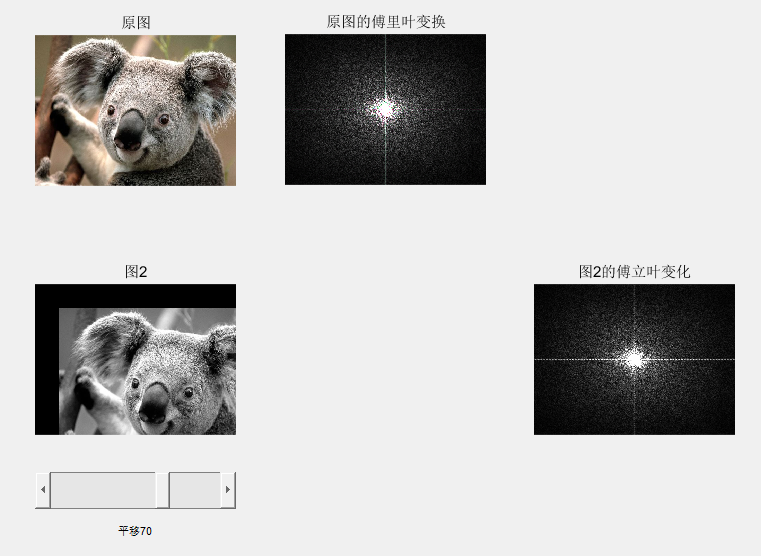
C=abs(B);

D=fftshift(C);

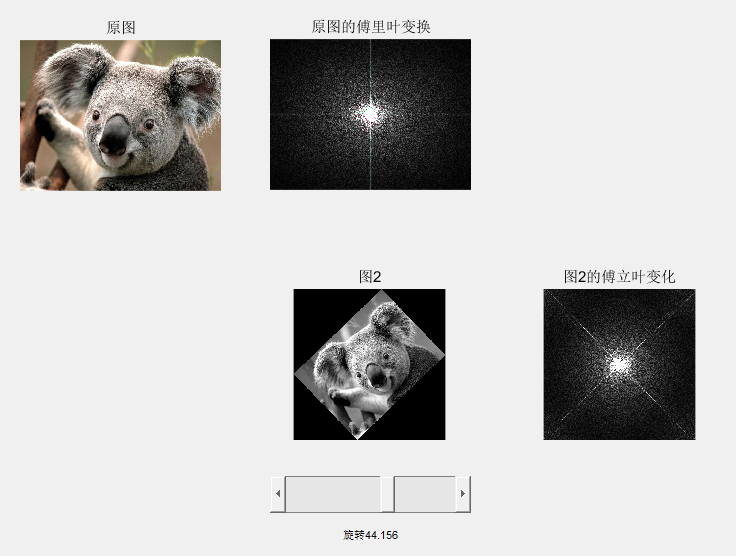
D=uint8(D);

**4.6.3效果截图:**

平移不变性



旋转一致性



**4.7.图像的点运算**

**4.7.1功能简介:**

图像对比度的改变

图像来亮度的改变

图像反色

**4.7.2核心代码:**

增加对比度

a=2;b=0;

O=a.\*I+b/255;

减小对比度

a=0.5;b=0;

O=a.\*I+b/255;

增加亮度

a=1;b=50;

O=a.\*I+b/255;

减小亮度

a=1;b=-50;

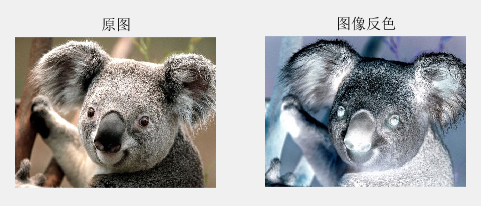
O=a.\*I+b/255;

图像反色

a=-1;b=255;

O=a.\*I+b/255;

**4.7.3效果截图:**



**4.8.图像的灰度拉伸**

**4.8.1功能简介:**

灰度拉伸:扩展图像的直方图，使其充满整个灰度级范围内。

**4.8.2核心代码:**

for i=1:nDims % ¶ÔÃ¿¸ö²¨¶ÎÒÀ´Î½øÐÐ»Ò¶ÈÀ­Éì

Sp=Image(:,:,i);

MaxDN=max(max(Sp));

MinDN=min(min(Sp));

Sp=(Sp-MinDN)/(MaxDN-MinDN); % »Ò¶ÈÀ­Éì¹«Ê½

ImageStretch(:,:,i)=Sp;

end

if nDims==3||nDims==1

axes(handles.axes1);imshow(Image);title('A');

axes(handles.axes2);imshow(ImageStretch);title('»Ò¶ÈÀ­Éì');

else

axes(handles.axes1);imshow(Image(:,:,[3,2,1]));title('Ô­Í¼');

axes(handles.axes2);imshow(ImageStretch(:,:,[3,2,1]));title('»Ò¶ÈÀ­Éì');

end

**4.8.3效果截图:**



**4.9.图像的直方图均衡化**

**4.9.1功能简介:**

直方图均衡化:通过矩阵变换实现

**4.9.2核心代码:**

[row,col] = size(I);

PMF = zeros(1,256);

for i = 1:row

for j = 1:col

PMF(I(i,j) + 1) = PMF(I(i,j) + 1) + 1;

end

end

%Compute the CDF of all pixels of the image

CDF = zeros(1,256);

CDF(1) = PMF(1);

for i = 2:256

CDF(i) = CDF(i - 1) + PMF(i);

end

%Mapping

for i = 1:256

Map(i) = round((CDF(i)-1)\*255/(row\*col));

end

for i = 1:row

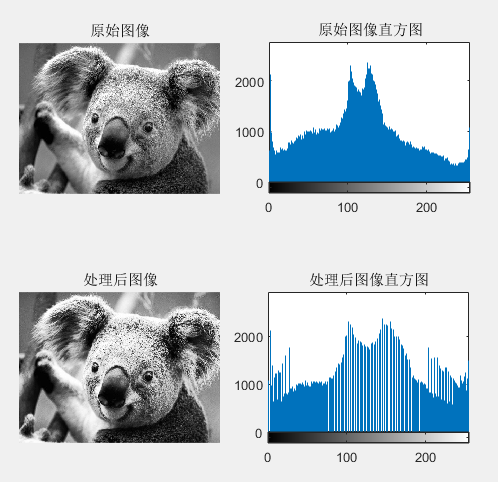
for j = 1:col

I(i,j) = Map(I(i,j) + 1);

end

end

**4.9.3效果截图:**



**4.10.图像的平滑处理算法**

**4.10.1功能简介:**

通过中值滤波和均值滤波对图片进行平滑处理

**4.10.2核心代码:**

均值滤波

B=imfilter(A,fspecial('average',5));

中值滤波

C=medfilt2(A,[3 3]);

**4.10.3效果截图:**

**4.11.图像的彩色增强**

**4.11.1功能简介:**

**伪彩色增强:将一幅灰度图像转换成一幅真彩色图像；**

**假彩色增强: 将三幅灰度图像合成一幅真彩色图像；**

**真彩色增强：**

1. **将一副真彩色图像进行RGB分离；**
2. **将某一分离后的单色图像增加、减少；**
3. **产生新的真彩色图像。**

**4.11.2核心代码:**

伪彩色增强

X=grayslice(I,16);

figure();

imshow(X,hot(16));

假彩色增强

RGBnew(:,:,1)=RGB(:,:,3);

RGBnew(:,:,2)=RGB(:,:,1);

RGBnew(:,:,3)=RGB(:,:,2);

**真彩色增强（红色减少）**

RGBnew(:,:,1)=RGB(:,:,1)/2;

RGBnew(:,:,2)=RGB(:,:,2);

RGBnew(:,:,3)=RGB(:,:,3);

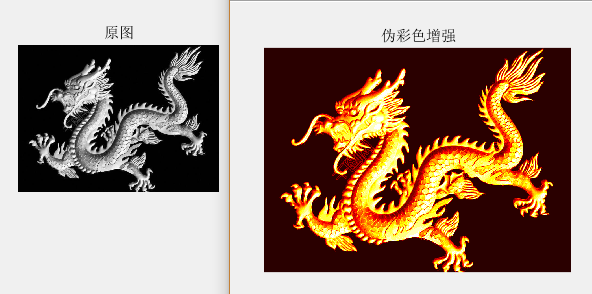
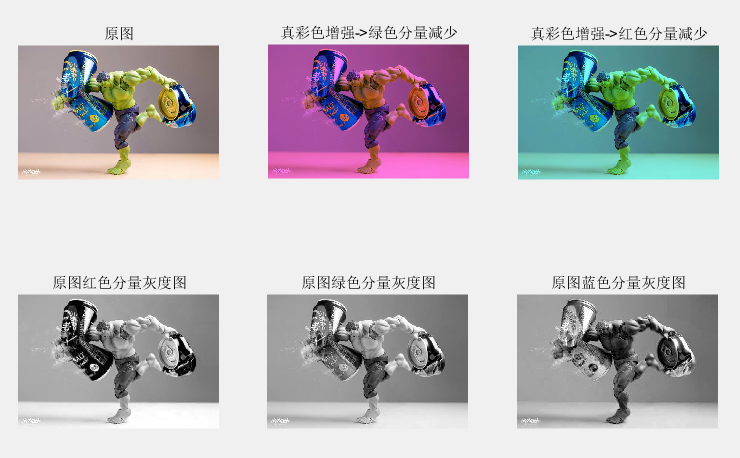
真彩色增强（红色减少）

RGBnew(:,:,1)=RGB(:,:,1);

RGBnew(:,:,2)=RGB(:,:,2)/2;

RGBnew(:,:,3)=RGB(:,:,3);

**4.11.3效果截图:**

**4.12.图像的边缘检测**

**4.12.1功能简介:**

分别用Roberts算子、Sobel算子、Laplacian算子等作边缘检测

**4.12.2核心代码:**

Roberts算子

A=edge(P,'roberts');

Sobel算子

B=edge(P,'sobel');

Laplacian算子

grayPic=P;

[m,n]=size(grayPic);

newGrayPic=grayPic;

LaplacianNum=0;

LaplacianThreshold=0.2;

for j=2:m-1

for k=2:n-1

LaplacianNum=abs(4\*grayPic(j,k)-grayPic(j-1,k)-grayPic(j+1,k)-grayPic(j,k+1)-grayPic(j,k-1));

if(LaplacianNum > LaplacianThreshold)

newGrayPic(j,k)=255;

else

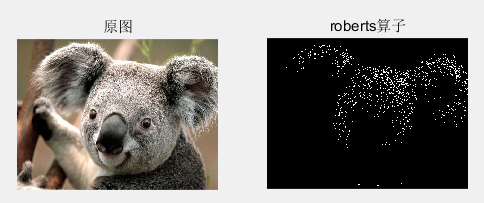
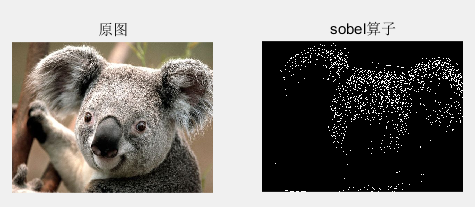
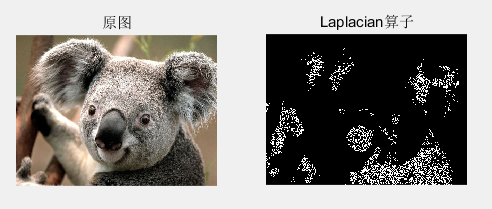
newGrayPic(j,k)=0;

end

end

end

**4.12.3效果截图:**

**4.13.图像的阈值分割**

**4.13.1功能简介:**

交互式:通过滑块自己选择阈值

迭代式：按迭代算法不断逼近获得阈值

**4.13.2核心代码:**

交互式阈值分割（T1值为用户拖动滑块得到的）

for i=1:width

for j=1:height

if(P(i,j)<T1)

P1(i,j)=0;

else

P1(i,j)=1;

end

end

end

迭代式阈值分割

T=(min(P2(:))+max(P2(:)))/2;

done=false;

i=0;

while ~done

r1=find(P2<=T);

r2=find(P2>T);

Tnew=(mean(P2(r1))+mean(P2(r2)))/2;

done=abs(Tnew-T)<1;

T=Tnew;

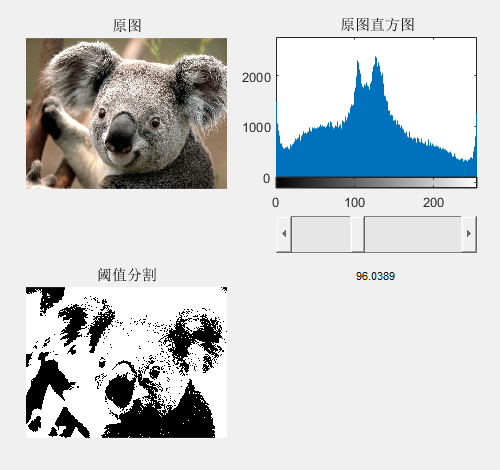
i=i+1;

end

P2(r1)=0;

P2(r2)=1;

**4.13.3效果截图:**

**4.14.二值图像处理**

**4.14.1功能简介:**

实现二值图像的膨胀、腐蚀、开运算与闭运算

**4.14.2核心代码:**

膨胀

sel=strel('square',3);

P1=imdilate(P,sel);

腐蚀

sel=strel('square',3);

P2=imerode(P,sel);

开运算

sel=strel('square',3);

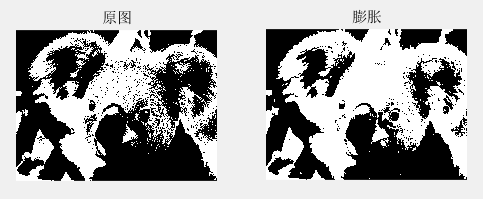
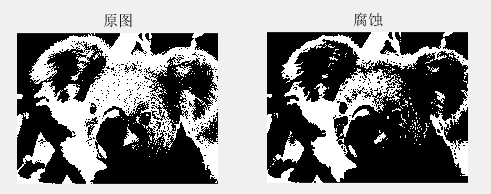
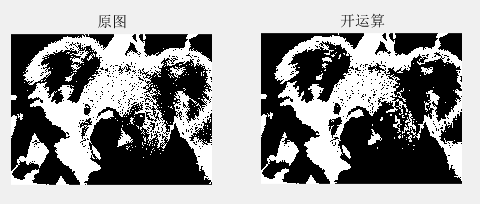
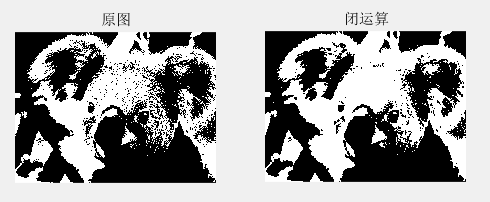
P3=imopen(P,sel);

闭运算

sel=strel('square',3);

P4=imclose(P,sel);

**4.14.3效果截图:**

**4.15.图像的表达**

**4.15.1功能简介:**

根据灰度值实现图像的3D立体表达

根据邻域像素的灰度差实现图像的四叉树表达

**4.15.2核心代码:**

3D立体表达

view(2);

mesh(P);

四叉树表达

I(1024,1024)=0;

S = qtdecomp(I,.27);

for dim = [1024 512 256 128 64 32 16 8 4 2 1];

numblocks = length(find(S==dim));

if (numblocks > 0)

[values, ~, ~] = qtgetblk(I, S, dim);

values(1:end,1,:)=0;

values(1,1:end,:)=0;

I= qtsetblk(I,S,dim,values);

end

end

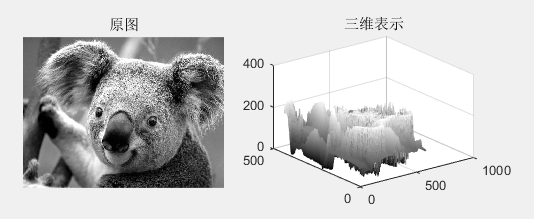
I(end,1:end) = 0;

I(1:end,end) = 0;

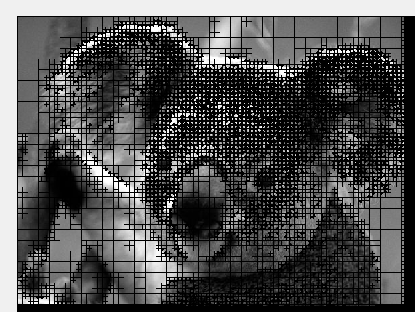
figure, imshow(I,[])

**4.15.3效果截图:**

3D表示



四叉树表示



**第5章 课程设计总结与体会**

在本次的图像处理系统设计期间，我将原来学过的图像处理知识系统总结为一个系统，并对原有的代码进行精简和修改，在这途中也遇到很多的问题，例如图像格式的不匹配，交互模式受限制等等，但通过网上查找和翻阅书籍都获得了合适的解决办法。这次的系统设计让我代码水平特别是对于GUI界面的设计方面有了一定的提高，让我获益匪浅。

**参考文献**

[1] Joshua Noble著．毛顺兵，张婷婷译《交互式程序设计》[M]北京：机械工业出版社，2014．

[2] Mandelbrot《The Fractal Geometry of Nature》[M] W. H. Freeman and Company，1983

[3] 杨杰《数字图像处理及MATLAB实现》[M]．北京：电子工业出版社，2010．

[4] 余胜威，吴婷，罗建桥《MATLAB+GUI设计入门与实战》[M]北京：清华大学出版社，2016．

[5] 罗华飞 《完整版MATLAB GUI设计学习手记第二版》[M]．北京：北京航空航天大学出版社，2010．