《数字图像处理》

实验报告

（计算机类本科生试用）

计算机科学与工程学院

华南理工大学

2020年09月

# 目 录

[目 录 2](#_Toc354477488)

[实验一 数字图像读取及色彩、亮度、对比度变化 3](#_Toc354477489)

[1．实验目的 3](#_Toc354477490)

[2．实验要求 3](#_Toc354477491)

[3．实验步骤 3](#_Toc354477492)

[4．实验图像 4](#_Toc354477493)

[实验二 数字图像变换与伽马校正 5](#_Toc354477494)

[1. 实验目的 5](#_Toc354477495)

[2. 实验要求 5](#_Toc354477496)

[3. 实验步骤 5](#_Toc354477497)

[4. 实验图像 5](#_Toc354477498)

[实验三 数字图像的噪声去除 6](#_Toc354477499)

[1．实验目的 6](#_Toc354477500)

[2．实验要求 6](#_Toc354477501)

[3．实验步骤 6](#_Toc354477502)

[4．实验图像 7](#_Toc354477503)

[实验四 图像的空间域锐化（拉普拉斯算子） 8](#_Toc354477504)

[1. 实验目的 8](#_Toc354477505)

[2. 实验要求 8](#_Toc354477506)

[3. 实验步骤 8](#_Toc354477507)

[4. 实验图像 8](#_Toc354477508)

[实验五 频率域低通和高通滤波 9](#_Toc354477509)

[1. 实验目的 9](#_Toc354477510)

[2. 实验要求 9](#_Toc354477511)

[3. 实验步骤 9](#_Toc354477512)

[4. 实验图像 9](#_Toc354477513)

[实验六 数字图像复原 10](#_Toc354477514)

[1. 实验目的 10](#_Toc354477515)

[2. 实验要求 10](#_Toc354477516)

[3. 实验步骤 10](#_Toc354477517)

[4. 实验图像 11](#_Toc354477518)

[实验七 人脸皮肤颜色分层 12](#_Toc354477519)

[1. 实验目的 12](#_Toc354477520)

[2. 实验要求 12](#_Toc354477521)

[3. 实验步骤 12](#_Toc354477522)

[4. 实验图像 13](#_Toc354477523)

[大作业 14](#_Toc354477524)

[1. 大作业目的 14](#_Toc354477525)

[2. 大作业要求 14](#_Toc354477526)

[3. 大作业内容 14](#_Toc354477527)

《数字图像处理》课程实验报告

实验题目： 实验一 数字图像读取及色彩、亮度、对比度变化

姓名： 向天翼 学号： 201836020389

班级： 18计算机联合班 组别：

合作者：

指导教师： 张星明

|  |
| --- |
| **实验概述** |
| **【实验目的及要求】**  **实验目的：**   1. 了解数字图像的存储格式； 2. 学会对图像的某些视觉特征作简单处理。   **实验要求：**   1. 从最常用的“.BMP”图像格式中读取图像数据； 2. 对数字图像的表示方式（如RGB、YUV）及各种表示方式之间的转换有初步了解； 3. 根据输入参数改变数字图像的色彩、亮度、对比度。   【实验环境】  Windows 10 |
| **实验内容** |
| 【实验过程】  一、实验步骤：   1. 根据BMP格式，将图像内容读入内存数组； 2. 通过访问数字图像RGB三个通道的对应矩阵，改变数字图像的色彩； 3. 将数字图像的RGB表示转换为YUV表示；   Y=0.30R+0.59G+0.11B  U=0.70R-0.59G-0.11B  V=-0.30R-0.59G+0.89B   1. 通过访问Y（亮度）通道，改变数字图像的亮度； 2. 通过Y（亮度）通道作灰度的线性变换，改变数字图像的对比度。   二、实验数据：    三、实验主要过程：   1. 根据BMP格式，将图像内容读入内存数组。   通过以下代码可以读取BMP文件并显示。   |  | | --- | | img = imread('lena.bmp');  imshow(img); |      1. 通过访问数字图像RGB三个通道的对应矩阵，改变数字图像的色彩。   通过定义函数get\_channel来获取数字图像的三个通道矩阵。函数的参数为h\*w\*3的原始图像和获取通道的索引号。该函数根据索引号拷贝相应的通道矩阵到result，其它通道置0。由此达到提取对应通道的效果。   |  | | --- | | %get\_channel.m  function [result] = get\_channel(img,channel\_id)  [height,width,~] = size(img);  if(channel\_id<0||channel\_id>3)  result = zeros(height,width);  else  for i = 1:3  if(channel\_id == i)  result(:,:,i)=img(:,:,i);  else  result(:,:,i)=zeros(height,width);  end  end  result = uint8(result);  end  end  %lab1.m  lena = imread('lena.bmp');  red = get\_channel(lena,1);  green = get\_channel(lena,2);  blue = get\_channel(lena,3);  subplot(2,3,1);  imshow(red);  title('Red Channel');  subplot(2,3,2);  imshow(green);  title('Green Channel');  subplot(2,3,3);  imshow(blue);  title('Blue Channel'); |      1. 将数字图像的RGB表示转换为YUV表示。   定义函数get\_YUV和函数RGB2YUV，分别负责将RGB图像分离出Y,U,V三种成分和将这三种成分组合在一起得到图像的YUV表示。   |  | | --- | | %get\_YUV.m  function [Y,U,V] = get\_YUV(img)  R=img(:,:,1);  G=img(:,:,2);  B=img(:,:,3);  Y=0.299\*R+0.587\*G+0.114\*B;  U=0.564\*(B-Y);  V=0.713\*(R-Y);  end  %RGB2YUV.m  function [YUV] = RGB2YUV(img)  [Y,U,V] = get\_YUV(img);  YUV=cat(3,Y,U,V);  end  %lab1.m  YUV = RGB2YUV(lena);  subplot(2,3,4);  imshow(YUV);  title('YUV image'); |      1. 通过访问Y（亮度）通道，改变数字图像的亮度；   定义函数set\_brightness，它先获取输入图像的Y,U,V三种成分，再将Y成分乘以输入的比例，最后将YUV表示变换回RGB表示，达到调整图像亮度的目的。   |  | | --- | | % set\_brightness.m  function [RGB] = set\_brightness(img,rate)  [Y,U,V] = get\_YUV(img);  Y=rate\*Y;  R=Y+1.402\*V;  G=Y-0.344\*U-0.714\*V;  B=Y+1.772\*U;  RGB=uint8(cat(3,R,G,B));  end  %lab1.m  half\_brightness = set\_brightness(lena,0.5);  subplot(2,3,5);  imshow(half\_brightness);  title('0.5 brightness image'); |      1. 通过Y（亮度）通道作灰度的线性变换，改变数字图像的对比度。   定义函数set\_contrast，先获取图像的Y,U,V成分，再根据变换公式对Y通道进行灰度的线性变换，最后将新的Y连同U,V转换为RGB成分组合后输出。   |  | | --- | | % set\_contrast.m  function [RGB] = set\_contrast(img)  a=100;  b=160;  c=50;  d=200;  [Y,U,V] = get\_YUV(img);  [height,width,~] = size(img);  Y3=Y;  for i=1:height  for j=1:width  if (Y(i,j)>b)  Y3(i,j)=d;  elseif (Y(i,j)<a)  Y3(i,j)=c;  else  Y3(i,j)=((double(d-c))/(double(b-a)))\*(Y(i,j)-a)+c;  end  end  end  R=Y3+1.402\*V;  G=Y3-0.344\*U-0.714\*V;  B=Y3+1.772\*U;  RGB = uint8(cat(3,R,G,B));  end  %lab1.m  contrast\_img = set\_contrast(lena);  subplot(2,3,6);  imshow(contrast\_img);  title('contrast image'); | |
| **小结** |
| 本次实验通过五次实验对数字图像的表示方式（如RGB、YUV）及各种表示方式之间的转换有初步了解，学会了从最常用的“.BMP”图像格式中读取图像数据和根据输入参数改变数字图像的色彩、亮度、对比度。加深了我对数字图像读取及色彩、亮度、对比度变化的印象。 |
| **成绩**  **指导教师：** |

《数字图像处理》课程实验报告

实验题目： 实验二 数字图像变换与伽马校正

姓名： 向天翼 学号： 201836020389

班级： 18计算机联合班 组别：

合作者：

指导教师： 张星明

|  |
| --- |
| **实验概述** |
| 【实验目的及要求】  实验目的：   1. 了解数字图像的灰度反变换； 2. 了解数字图像的g（0.4，0.6，0.8）校正。   实验要求：   1. 对图像进行灰度变换； 2. 对图像进行γ校正。   【实验环境】  Windows 10 |
| **实验内容** |
| 【实验过程】  一、实验步骤：   1. 将BMP图像内容读入内存数组。 2. 调整图像的灰度，对图像进行灰度变换（反变换）。 3. 对图像进行γ较正,分别取值为0.4,0.6,0.8。 4. 显示一幅图像的灰度直方图。   二、实验数据：    灰度变换γ较正  三、实验主要过程：   1. 将BMP图像内容读入内存数组，调整图像的灰度，对图像进行灰度变换（反变换）。   通过定义三个函数实现对图像做相应的变换，代码如下：   |  | | --- | | % logarithmic.m  function [out] = logarithmic(img)  C = 10;  c(:,:,1) = C\*log(double(1 + img(:,:,1)));  c(:,:,2) = C\*log(double(1 + img(:,:,2)));  c(:,:,3) = C\*log(double(1 + img(:,:,3)));  out = uint8(c);  end  %power\_law.m  function [out] = power\_law(img)  C = 10;  gamma = 10;  d(:,:,1) = C\*(img(:,:,1).^gamma);  d(:,:,2) = C\*(img(:,:,2).^gamma);  d(:,:,3) = C\*(img(:,:,3).^gamma);  out = uint8(d);  end  %reverse.m  function [out] = reverse(img)  b(:,:,1) = 255 - img(:,:,1);  b(:,:,2) = 255 - img(:,:,2);  b(:,:,3) = 255 - img(:,:,3);  out = uint8(b);  end  %lab2.m  img = imread('灰度变换.bmp');  subplot(3,3,1);  imshow(reverse(img));  title('反转变换');  subplot(3,3,2);  imshow(logarithmic(img));  title('对数变换');  subplot(3,3,3);  imshow(power\_law(img));  title('幂律变换'); |      1. 对图像进行γ较正,分别取值为0.4,0.6,0.8。   通过定义函数gamma\_correction对输入图像按比例进行γ较正   |  | | --- | | %gamma\_correction.m  function [out] = gamma\_correction(img,rate)  img = double(img) / 256.0;  out = uint8(img.^(1/rate).\*256.0);  end  %lab2.m  gamma\_img = imread('伽马校正.bmp');  subplot(3,3,4);  imshow(gamma\_correction(gamma\_img,0.4));  title('gamma with 0.4');  subplot(3,3,5);  imshow(gamma\_correction(gamma\_img,0.6));  title('gamma with 0.6');  subplot(3,3,6);  imshow(gamma\_correction(gamma\_img,0.8));  title('gamma with 0.8'); |      1. 显示一幅图像的灰度直方图。   定义函数get\_histogram对图像的每一个像素点进行统计，得到图像的灰度直方图   |  | | --- | | % get\_histogram.m  function [out] = get\_histogram(img)  [height,width,~] = size(img);  histogram = zeros(1,256);  for i=1:height  for j=1:width  k= img(i,j);  histogram(k+1)= histogram (k+1)+1;  end  end  out = histogram;  end  %lab2.m  subplot(3,3,7);  bar(get\_histogram(img));  title('灰度直方图'); | |
| **小结** |
| 通过本次实验我学会了图像校正的基本原理，通过编程实现图像反转、对数和指数变换、分段线性变换和伽马校正等图像增强算法。加深了对图像处理中数字图像灰度变换和伽马校正基本原理与方法的印象。 |
| **成绩**  **指导教师：** |

《数字图像处理》课程实验报告

实验题目： 实验三 数字图像的噪声去除

姓名： 向天翼 学号： 201836020389

班级： 18计算机联合班 组别：

合作者：

指导教师： 张星明

|  |
| --- |
| **实验概述** |
| 【实验目的及要求】  实验目的：   1. 学会用滤波器去除图像的噪声   实验要求：   1. 用均值滤波器去除图像中的噪声； 2. 用中值滤波器去除图像中的噪声； 3. 用K均值滤波器去除图像中的噪声； 4. 比较三种方法的处理结果。   【实验环境】  Windows 10 |
| **实验内容** |
| 【实验过程】  一、实验步骤：   1. 根据BMP格式，将图像内容读入内存数组； 2. 用均值滤波器去除图像中的噪声； 3. 用中值滤波器去除图像中的噪声； 4. 用K均值滤波器去除图像中的噪声； 5. 将三种处理方法的结果与原图比较；注意三种处理方法对边缘的影响。   二、实验数据：   |  |  | | --- | --- | |  |  | | 原始图像 | 加噪声后的图像 |   三、实验主要过程：   1. 将BMP图像内容读入内存数组，用均值滤波器去除图像中的噪声；   根据均值滤波的公式，定义函数mean\_filtering实现对图像做均值滤波，代码如下：   |  | | --- | | % mean\_filtering.m  function [out] = mean\_filtering(img)  res1 = img;  k = 1;  [m,n,~] = size(img);  for i = 1:m  for j = 1:n  i1 = max(1,i-k);  i2 = min(m,i+k);  j1 = max(1,j-k);  j2 = min(n,j+k);  tmp = img(i1:i2,j1:j2);  s = sum(tmp(:)) \* (1.0/(4\*(i2-i1)\*(j2-j1)+1));  res1(i,j) = uint8(s);  end  end  out = res1;  end  %lab3.m  origin = imread('原始图像.jpg');  noise = imread('噪声图像.bmp');  subplot(2,3,1);  imshow(origin);  title('原始图像');  subplot(2,3,2);  imshow(noise);  title('噪声图像');  subplot(2,3,3);  imshow(mean\_filtering(noise));  title('均值滤波'); |      1. 用中值滤波器去除图像中的噪声   根据中值滤波的公式，定义函数median\_filtering实现对图像做中值滤波，代码如下：   |  | | --- | | %median\_filtering.m  function [out] = median\_filtering(img)  res1 = img;  k = 1;  [m,n,~] = size(img);  for i = 1:m  for j = 1:n  i1 = max(1,i-k);  i2 = min(m,i+k);  j1 = max(1,j-k);  j2 = min(n,j+k);  tmp = img(i1:i2,j1:j2);  v = tmp(1:end);  v = sort(v);  res1(i,j) = v(ceil((i2-i1)\*(j2-j1)/2));  end  end  out = res1;  end  %lab3.m  origin = imread('原始图像.jpg');  noise = imread('噪声图像.bmp');  subplot(2,3,1);  imshow(origin);  title('原始图像');  subplot(2,3,2);  imshow(noise);  title('噪声图像');  subplot(2,3,3);  imshow(median\_filtering(noise));  title('中值滤波'); |      1. 用K均值滤波器去除图像中的噪声   K均值滤波先对当前窗口内所有像素值与窗口中心像素值的差的绝对值进行升序排序，并取第二个到第K+1个的索引（第一个为到中心像素到自己的距离，为0，舍去）。再计算窗口内对应索引处的像素平均值作为滤波后图像中心像素点的像素值。根据K均值滤波的公式，定义函数median\_filtering实现对图像做K均值滤波，代码如下：   |  | | --- | | % k\_mean\_filtering.m  function [output] = k\_mean\_filtering(img,n,k)  [height,width,~] = size(img);  half\_window\_size = floor(n/2);  output = uint8(zeros(height - 2\*half\_window\_size,width - 2\*half\_window\_size));  for i=1+half\_window\_size:height-half\_window\_size  for j=1+half\_window\_size:width-half\_window\_size  mask = img(i - half\_window\_size:i+half\_window\_size, j -half\_window\_size: j + half\_window\_size);  center = img(i,j);  [~,index] = sort(reshape(abs(uint8(zeros(n,n))+center-mask),1,n\*n));  arraified\_mask = reshape(mask,1,n\*n);  k\_top\_index = index(2:k+1);  k\_top = uint8(zeros(1,k));    for l=1:k  k\_top(l)=arraified\_mask(k\_top\_index(l));  end  output(i - 1,j - 1) = mean(k\_top);  end  end  end  %lab3.m  origin = imread('原始图像.jpg');  noise = imread('噪声图像.bmp');  subplot(2,3,1);  imshow(origin);  title('原始图像');  subplot(2,3,2);  imshow(noise);  title('噪声图像');  subplot(2,3,3);  imshow(k\_mean\_filtering(noise,3,5));  title('k均值滤波'); |      1. 将三种处理方法的结果与原图比较；注意三种处理方法对边缘的影响。   对比结果如下：    根据对比结果可发现均值滤波结果较暗，黑边较多；中值滤波线条变粗，丢失了一些细节信息；而K均值滤波相对前两者效果较好。 |
| **小结** |
| 通过本次实验我学会了通过编程实现用滤波器去除图像中的噪声。包括用均值滤波器、中值滤波器以及K均值滤波器去除图像中的噪声，并比较三种方法的处理结果。加深了我对数字图像基本平滑方法的印象。 |
| **成绩**  **指导教师：** |

《数字图像处理》课程实验报告

实验题目： 实验四 图像的空间域锐化（拉普拉斯算子）

姓名： 向天翼 学号： 201836020389

班级： 18计算机联合班 组别：

合作者：

指导教师： 张星明

|  |
| --- |
| **实验概述** |
| 【实验目的及要求】  实验目的：   1. 了解数字图像的空间域锐化； 2. 了解数字图像的拉普拉斯算子。   实验要求：   1. 理解图像的空间域锐化原理； 2. 熟悉拉普拉斯算子的公式和实现； 3. 运用拉普拉斯算子对图像进行空间域锐化。   【实验环境】  Windows 10 |
| **实验内容** |
| 【实验过程】  一、实验步骤：   1. 将BMP图像内容读入内存数组； 2. 运用拉普拉斯算子对图像进行空间域锐化； 3. 将锐化后的图像和原图像进行对比。   二、实验数据：    三、实验主要过程：   1. 将BMP图像内容读入内存数组，运用拉普拉斯算子对图像进行空间域锐化；   定义函数laplacian\_sharpening实现对给定图像用给定掩模进行锐化，代码如下：   |  | | --- | | % laplacian\_sharpening.m  function [out] = laplacian\_sharpening(img,f)  [r,c,~] = size(img);  out = zeros(r,c);  img = im2double(img);  for x=1:r  for y=1:c  if (x==1||y==1||x==r||y==c)  out(x,y)=img(x,y);  else  out(x,y)=f(1,1)\*img(x-1,y-1)+f(1,2)\*img(x-1,y)+f(1,3)\*img(x-1,y+1)...  +f(2,1)\*img(x,y-1)+f(2,2)\*img(x,y)+f(2,3)\*img(x,y+1)...  +f(3,1)\*img(x+1,y-1)+f(3,2)\*img(x+1,y)+f(3,3)\*img(x+1,y+1);  end  end  end  end  %lab4.m  img = imread('梯度算子.bmp');  f1 = [0,1,0;1,-4,1;0,1,0];  f2 = [1,1,1;1,-8,1;1,1,1];  img = im2double(img);  subplot(2,3,1);  imshow(img);  title('原图');  subplot(2,3,2);  imshow(laplacian\_sharpening(img,f1));  title('F1拉普拉斯滤波');  subplot(2,3,3);  imshow(img-laplacian\_sharpening(img,f1));  title('F1锐化后');  subplot(2,3,4);  imshow(img);  title('原图');  subplot(2,3,5);  imshow(laplacian\_sharpening(img,f2));  title('F2拉普拉斯滤波');  subplot(2,3,6);  imshow(img - laplacian\_sharpening(img,f2));  title('F2锐化后'); |      1. 将锐化后的图像和原图像进行对比。   对比结果如第一部分结果图。根据对比结果可发现锐化后图像在灰度值变化较快的地方均有增强，且八邻域掩膜(F2)的增强效果比四邻域掩膜(F1)更强。 |
| **小结** |
| 通过本次实验学会了通过编程实现拉普拉斯算子和用拉普拉斯算子对图像进行空间域锐化。加深了我对空间域锐化原理的理解以及对数字图像基本锐化方法的印象。 |
| **成绩**  **指导教师：** |

《数字图像处理》课程实验报告

实验题目： 实验五 频率域低通和高通滤波

姓名： 向天翼 学号： 201836020389

班级： 18计算机联合班 组别：

合作者：

指导教师： 张星明

|  |
| --- |
| **实验概述** |
| **【实验目的及要求】**  **实验目的：**   1. 学会频域低通滤波方法； 2. 学会频域高通滤波方法。   **实验要求：**   1. 学会傅立叶变换方法； 2. 使用布特沃斯和高斯滤波器进行低通滤波； 3. 使用布特沃斯和高斯滤波器进行高通滤波。   【实验环境】  Windows 10 |
| **实验内容** |
| 【实验过程】  一、实验步骤：   1. 将图像内容读入内存数组，并用布特沃思低通滤波器进行滤波: 2. 用高斯低通滤波器进行滤波: 3. 用布特沃思高通滤波器进行滤波: 4. 用高斯高通滤波器进行滤波:   二、实验数据：    三、实验主要过程：   1. 将图像内容读入内存数组，并用布特沃思低通滤波器进行滤波:。   定义函数butterworth\_lowpass实现对图像进行布特沃思低通滤波。该函数先对图像进行傅里叶变换得到频率域的图像，再对频率域的图像进行布特沃思低通滤波操作，最后反傅里叶变换回空间域图像并返回。本例中取n=2,测试G0在不同取值（10,20,40,80）下滤波效果。代码如下：   |  | | --- | | % butterworth\_lowpass.m  function [y] = butterworth\_lowpass(img,d0,nn)  f=double(img);  g=fft2(f);  g=fftshift(g);  [M,N] = size(g);  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+(d/d0)^(2\*nn));  end  y(i,j)=h\*g(i,j);  end  end  y=ifftshift(y);  y=ifft2(y);  y=uint8(real(y));  end  % lab5.m  noise = imread('sample5-1.bmp');  subplot(2,4,1);  imshow(noise);  title('噪声图像');  subplot(2,4,5);  imshow(butterworth\_lowpass(noise,10,2));  title('Butterworth lowpass G0=10 n=2');  subplot(2,4,6);  imshow(butterworth\_lowpass(noise,20,2));  title('Butterworth lowpass G0=20 n=2');  subplot(2,4,7);  imshow(butterworth\_lowpass(noise,40,2));  title('Butterworth lowpass G0=40 n=2');  subplot(2,4,8);  imshow(butterworth\_lowpass(noise,80,2));  title('Butterworth lowpass G0=80 n=2'); |     对比可发现D0越小，结果越模糊，去噪效果越好；D0越大，结果越清晰，去噪效果越差。   1. 用高斯低通滤波器进行滤波。   通过定义函数gauss\_lowpass实现对图像进行高斯低通滤波，本例同样取不同G0值对比滤波结果。代码如下：   |  | | --- | | % gauss\_lowpass.m  function [y] = gauss\_lowpass(img,d0)  f=double(img);  g=fft2(f);  g=fftshift(g);  [M,N] = size(g);  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=exp(-(d^2)/(2\*(d0^2)));  end  y(i,j)=h\*g(i,j);  end  end  y=ifftshift(y);  y=ifft2(y);  y=uint8(real(y));  end  %lab5.m  noise = imread('sample5-1.bmp');  subplot(2,4,1);  imshow(noise);  title('噪声图像');  subplot(2,4,5);  imshow(gauss\_lowpass(noise,10));  title('Gauss lowpass G0=10');  subplot(2,4,6);  imshow(gauss\_lowpass(noise,20));  title('Gauss lowpass G0=20');  subplot(2,4,7);  imshow(gauss\_lowpass(noise,40));  title('Gauss lowpass G0=40');  subplot(2,4,8);  imshow(gauss\_lowpass(noise,80));  title('Gauss lowpass G0=80'); |      1. 用布特沃思高通滤波器进行滤波   通过定义函数butterworth\_highpass实现对图像进行布特沃思高通滤波，本例同样取n=2,不同G0值对比滤波结果。代码如下：   |  | | --- | | % butterworth\_highpass.m  function [y] = butterworth\_highpass(img,d0,nn)  f=double(img);  g=fft2(f);  g=fftshift(g);  [M,N] = size(g);  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+(d0/d)^(2\*nn));  end  y(i,j)=h\*g(i,j);  end  end  y=ifftshift(y);  y=ifft2(y);  y=uint8(real(y));  end  %lab5.m  noise = imread('sample5-1.bmp');  subplot(2,4,1);  imshow(noise);  title('噪声图像');  subplot(2,4,5);  imshow(butterworth\_highpass(noise,10,2));  title('Butterworth highpass G0=10 n=2');  subplot(2,4,6);  imshow(butterworth\_highpass(noise,20,2));  title('Butterworth highpass G0=20 n=2');  subplot(2,4,7);  imshow(butterworth\_highpass(noise,40,2));  title('Butterworth highpass G0=40 n=2');  subplot(2,4,8);  imshow(butterworth\_highpass(noise,80,2));  title('Butterworth highpass G0=80 n=2'); |      1. 用高斯高通滤波器进行滤波   通过定义函数gauss\_highpass实现对图像进行高斯高通滤波，本例同样取不同G0值对比滤波结果。代码如下：   |  | | --- | | % gauss\_highpass.m  function [y] = gauss\_highpass(img,d0)  f=double(img);  g=fft2(f);  g=fftshift(g);  [M,N] = size(g);  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-exp(-(d^2)/(2\*(d0^2)));  end  y(i,j)=h\*g(i,j);  end  end  y=ifftshift(y);  y=ifft2(y);  y=uint8(real(y));  end  %lab5.m  noise = imread('sample5-1.bmp');  subplot(2,4,1);  imshow(noise);  title('噪声图像');  subplot(2,4,5);  imshow(gauss\_highpass(noise,10));  title('Gauss highpass G0=10');  subplot(2,4,6);  imshow(gauss\_highpass(noise,20));  title('Gauss highpass G0=20');  subplot(2,4,7);  imshow(gauss\_highpass(noise,40));  title('Gauss highpass G0=40');  subplot(2,4,8);  imshow(gauss\_highpass(noise,80));  title('Gauss highpass G0=80'); | |
| **小结** |
| 本次实验我学会两种简单的频域低通和高通滤波方法，包括通过编程实现傅立叶变换方法，使用布特沃斯和高斯滤波器进行低通滤波以及使用布特沃斯和高斯滤波器进行高通滤波。加深了我对数字图像频率域低通和高通滤波的印象。 |
| **成绩**  **指导教师：** |

《数字图像处理》课程实验报告

实验题目： 实验六 数字图像复原

姓名： 向天翼 学号： 201836020389

班级： 18计算机联合班 组别：

合作者：

指导教师： 张星明

|  |
| --- |
| **实验概述** |
| 【实验目的及要求】  实验目的：   1. 了解Fourier变换算法的实现； 2. 了解反变换的算法实现； 3. 掌握频域逆滤波图像复原的方法； 4. 掌握频域维纳滤波图像复原的方法。   实验要求：   1. 用Fourier变换算法对图像作二维Fourier变换； 2. 用Fourier反变换算法对图像作二维Fourier反变换； 3. 频域逆滤波和维纳滤波图像复原。   【实验环境】  Windows 10 |
| **实验内容** |
| 【实验过程】  一、实验步骤：   1. 将图像读入内存数组，并用用快速Fourier变换算法，对图像作二维FFT变换得到G(u,v)； 2. 进行频率域逆滤波和维纳滤波，其中   逆滤波退化函数：  维纳滤波退化函数：   1. 进行傅里叶反变换得到。   二、实验数据：    三、实验主要过程：   1. 对图像进行频率域逆滤波：   通过定义函数inverse\_filter对输入图像进行频率域逆滤波。首先对原图像进行傅里叶变换，得到的结果再根据公式进行逆滤波操作，最后把结果进行傅里叶反变换并输出。由于逆滤波算法中值存在零点问题，因此这里当小于阈值时将置为一个超参数0.78，使用该参数能使逆滤波算法发挥较好的作用。源代码实现如下：   |  | | --- | | % inverse\_filter.m  function [output] = inverse\_filter(img)  f = double(img);  g = fft2(f);  g = fftshift(g);  [M,N] = size(g);  k = 80;  for u=1:M  for v=1:N  h(u,v) = exp(-k\*(((u-M/2)^2+(v-N/2)^2)^(5/6)));  if(h(u,v)<0.0001)  f(u,v)=g(u,v)/0.78;  else  f(u,v)=g(u,v)/h(u,v);  end  end  end  y = ifftshift(f);  y = ifft2(y);  output = uint8(real(y));  end  %lab6.m  a = imread('sample6-1.bmp');  subplot(1,3,1);  imshow(a);  title('原图像');  subplot(1,3,2);  imshow(inverse\_filter(a));  title('逆滤波'); |      1. 对图像进行频率域维纳滤波：   通过定义函数wiener\_filter对输入图像进行频率域维纳滤波。首先对原图像进行傅里叶变换，得到的结果再根据公式进行逆滤波操作，最后把结果进行傅里叶反变换并输出。源代码实现如下：   |  | | --- | | % wiener\_filter.m  function [output] = wiener\_filter(img)  f = double(img);  g = fft2(f);  g = fftshift(g);  [M,N] = size(g);  k = 0.0025;  for u=1:M  for v=1:N  h(u,v) = exp(-k\*(((u-M/2)^2+(v-N/2)^2)^(5/6)));  f(u,v) = (1/h(u,v))\*(h(u,v)^2/(h(u,v)^2+k))\*g(u,v);  end  end  y = ifftshift(f);  y = ifft2(y);  output = uint8(real(y));  end  %lab6.m  a = imread('sample6-1.bmp');  subplot(1,3,1);  imshow(a);  title('原图像');  subplot(1,3,2);  imshow(wiener\_filter(a));  title('维纳滤波'); | |
| **小结** |
| 通过本次实验学会了通过编程实现傅里叶变换与反傅里叶变换，并将其运用到实现频率域逆滤波和维纳滤波的图像复原算法的实现上。加深了我对数字图像频率域图像复原方法的理解。 |
| **成绩**  **指导教师：** |

《数字图像处理》课程实验报告

实验题目： 实验七 人脸皮肤颜色分层

姓名： 向天翼 学号： 201836020389

班级： 18计算机联合班 组别：

合作者：

指导教师： 张星明

|  |
| --- |
| **实验概述** |
| 【实验目的及要求】  实验目的：   1. 学习使用颜色分层方法 2. 对身份证标准图像实现肤色检测和背景分离   实验要求：   1. 理解彩色图像的分层原理和方法 2. 利用颜色分层方法对彩色图像进行分层处理 3. 实现对身份证标准图像的背景分离和皮肤区域提取。   【实验环境】  Windows 10 |
| **实验内容** |
| 【实验过程】  一、实验步骤：   1. 根据BMP格式，将彩色图像内容读入内存数组； 2. 对图像中背景和皮肤区域计算背景和肤色平均值(a值)，并设置分层阈值 W； 3. 利用图像分层函数对彩色图像进行扫描和计算处理； 4. 根据背景平均值和分层阈值去掉身份证标准图像背景区域； 5. 根据肤色平均值和分层阈值提取身份证标准图像皮肤区域；   二、实验数据：    三、实验主要过程：  通过定义函数LayerExtraction实现对输入图像给定的像素点进行去除与提取。这个函数有五个参数，包括原图像、目标像素位置，阈值以及控制去除或提取的控制量。若控制量为1，则函数将扫描整个图像，并根据给定位置处像素值以及输入的阈值对原图像进行颜色提取，反之则执行颜色去除。无论是提取操作或去除操作，被去除的部分都将以灰色填充。此处去除背景时阈值设为0.2549\*3，而执行提取人脸皮肤区域时阈值设为0.2549\*2.2。  源代码实现如下：   |  | | --- | | % LayerExtraction.m  function [output] = LayerExtraction(img,pos\_x,pos\_y,w,isAccept)  if(isAccept~=0 && isAccept~=1)  error('参数错误！isAccept应该为0或1！');  end  img = double(img);  a = [0, 0, 0];  a(1) = img(pos\_x, pos\_y, 1);  a(2) = img(pos\_x, pos\_y, 2);  a(3) = img(pos\_x, pos\_y, 3);  a = a / 255;  B = img / 255.0;  m = size(img);  for i=1:m(1)  for j=1:m(2)  if(isAccept == 0)  if(abs(B(i, j, 1) - a(1)) <= w / 2 && abs(B(i, j, 2) - a(2)) <= w / 2 && abs(B(i, j, 3) - a(3)) <= w / 2)  B(i, j, 1) = 0.5;  B(i, j, 2) = 0.5;  B(i, j, 3) = 0.5;  end  else  if(abs(B(i, j, 1) - a(1)) > w / 2 || abs(B(i, j, 2) - a(2)) > w / 2 || abs(B(i, j, 3) - a(3)) > w / 2)  B(i, j, 1) = 0.5;  B(i, j, 2) = 0.5;  B(i, j, 3) = 0.5;  end  end  end  end  %lab7.m  img = imread('sample7-1.bmp');  subplot(1,3,1);  imshow(img);  title('原图');  subplot(1,3,2);  imshow(LayerExtraction(img,10,10,0.2549\*3,0));  title('去除背景');  subplot(1,3,3);  imshow(LayerExtraction(img,80,120,0.2549\*2.2,1));  title('只保留皮肤'); | |
| **小结** |
| 通过本次实验我理解了彩色图像的颜色分层原理和方法，并学会了通过编程实现颜色分层算法对人脸身份证彩色图像进行分层处理，从而实现对身份证标准图像的背景分离和皮肤区域提取。加深了我对数字图像颜色分层原理，背景分离和皮肤区域提取方法的理解与应用。 |
| **成绩**  **指导教师：** |