数字图像处理第一次作业

摘要

本文主要介绍了 bmp 文件的格式和具体内容以及 matlab 在灰度级变换，求取图像均值与方差，插值放大，及偏移和旋转变换等一些仿射变换当中的作用，并尝试自己编写程序，基于相同的原理采用不同的方法对图像进行灰度级变换以及插值放大。同时，对最近邻插值、双线性插值、三次插值处理后的结果进行了较为详细的对比与分析，并在最后附源代码。

姓名：生明旸

班级：自动化 64

学号：2160504100

提交日期：2019 年 3 月 3 日

1. Bmp 图像格式简介,以 7.bmp 为例说明：

简介：BMP 文件格式，又称为 Bitmap（位图）或是 DIB(Device-Independent Device，设备无关位图)，是 Windows 系统中广泛使用的图像文件格式。它采用位映射存储格式，除了图像深度可选以外，不采用其他任何压缩。图像的扫描方式一般为从左到右，从下到上，以行优先。

BMP 文件中存储的数据可分为四类：

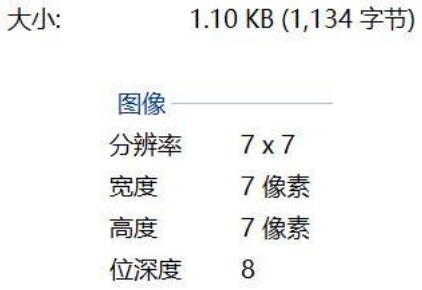
①BMP 文件头：包含了文件的格式、大小等信息

②位图信息头：提供了图像尺寸等信息

③调色板（可选）：映射表，建立了像素点存储数据与颜色间的映射关系

④位图数据：即为图像像素点存储的数据

以 7.bmp 文件为例，查询其基本信息如下：



根据基本信息可知，图像为 8 位的，所以包含调色板；图像宽和高均为 7。

BMP 文件头所占大小为固定的 14Bytes

位图信息头所占大小为固定的 40Bytes

因为每一个像素点的数据大小为 8bits，所以共有 2^8=256 种颜色，每个颜色占用 4Bytes，所以调色板所占大小为 1024Bytes

位图数据所占大小为图像宽×图像高×位深度：7×7×8bits=49Bytes，但考虑到 BMP 文件采用了数据对齐，要求每行数据的长度为 4 的倍数，所以实际所占大小为 7×8=56Bytes

14+40+1024+56=1134Bytes，与查询的 7.bmp 文件大小一致。

2. 把 lena 512*512 图像灰度级逐级递减 8-1 显示：

（1） 问题分析：

原图像为 8—bit 深的图像，共有 256 个颜色级。位深减小是通过改变像素数据的范围实现的。但需注意一点，当位深减小时要改变调色板，否则无法显示正确的图像。改变调色板可以有两种方式。一种是利用 imshow()函数中的可选参数改变，另一种是将原数据除以 2^{a-b} 向下取整后再乘 2^{a-b}（a 为原图像位深，b 为所需位深）。后者方法由于不是真正的改变位深，所以效果不如前者。本实验选择前一种方法。

（2） 实验过程：

首先利用 imread()函数读入图片，然后将原像素数据除以 2 并向下取整数以得到降低灰度后的像素数据，最后利用 imshow()函数的参数给定像素数据范围以实现降低灰度级的要求。

（3） 实验结果：

8—bit

7—bit

6—bit

5—bit

4—bit

3—bit

2—bit

1—bit

（4） 结果分析：

由图可见，当位深从 4 变化到 1 时，图像有明显的变化，一些细节随着位深的减小而消失。当位深在 5 到 8 时图像间没有明显的差异。

3. 计算 lena 图像的均值方差：

（1） 实验过程：

利用 mean2()和 std2()得到原图像的均值和标准差，然后将标准差平方得到方差。

（2） 实验结果：

均值 mean = 99.0512；方差 var = 2796.04

4. 把 lena 图像用近邻、双线性和双三次插值法 zoom 到 2048*2048：

（1） 问题分析：

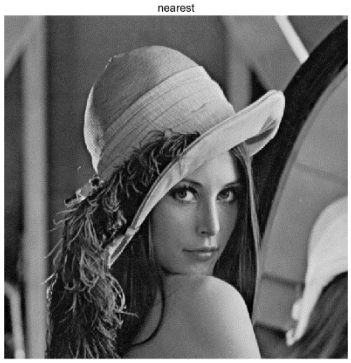
可以采用 matlab 自带的函数，也可以自己编写放大函数。下面介绍编写放大函数的思路。

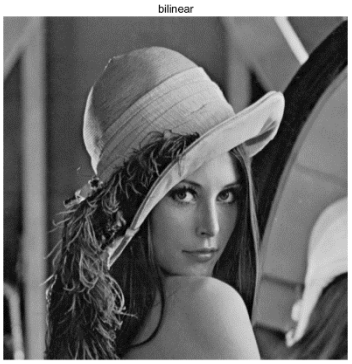
首先考虑到图像边界问题需要扩充原图像变为 513*513，否则无法满足放大后的图像为 2048*2048 的要求。扩充原图像的数据采用原图像最后一行和一列的数据。然后利用 matlab 的插值方法，进行插值。

（2） 实验过程：

利用 imresize(Image,[rows,cols],'method')函数，实现不同插值方法下的 zoom。其中，Image 为待处理图像，rows 和 cols 分别为处理后的图像宽和高，'method'为所用的插值方法。

（3） 实验结果：（自己编写的函数结果以图片形式保存在文件夹中）





（4） 结果分析：

通过将边缘放大可发现，最近邻插值方法所得的边缘不如双线性和三次插值方法的平滑，说明最近邻插值方法效果不如后两者好。双线性和三次插值方法差异不大，

考虑到算法复杂度，三次插值计算一个新像素点需要 16 个旧像素点，在一般情况下采用双线性插值即可。双线性插值与三次插值差异如下图所示：

自己编写的函数放大图片后在边界上与 matlab 自带的函数差异不大。如下图所示：

5. 把 lena 和 elain 图像分别进行水平 shear（参数可设置为 1.5，或者自行选择）和旋转 30 度，并采用近邻、双线性和双三次插值法 zoom 到 2048*2048：

（1） 问题分析：

先将 lena 和 elain 分别进行水平偏移变换、旋转变换,再将变换后的图像利用最近邻内插法、双线性内插法、双三次内插法放大为 2048*2048 的图像,并进行显示。

（2） 实验过程：

先创建变换矩阵 T，然后利用 affine2d()创建函数所需的仿射矩阵，最后利用 imwarp()函数实现变换。

（3） 实验结果：（zoom 后的图片未展示）

（4） 结果分析：

如前所述，将边缘经局部放大后可以发现，三次插值与双线性插值对边缘的影响差距不大，但显著优于最近邻插值方法。

附录：

实验源代码

第二题：

```
Image_L = imread('lena.bmp');
Image_L = double(Image_L);
image1 = Image_L;
for i = 1:7
    image1 = floor(image1./2);
    figure(i)
    imshow(image1,[0,2^(8-i)-1])
end
```

第三题：

```
average = mean2(Image_L); %均值
var = (std2(Image_L))^2; %方差
```

第四题：

```
Image_L = imread('lena.bmp');
Image_LN = imresize(Image_L,[2048 2048],'nearest');
Image_LL = imresize(Image_L,[2048 2048],'bilinear');
Image_LC = imresize(Image_L,[2048 2048],'bicubic');

%自己的方法
% A = my_zoom(Image_L,2048,2048,'nearest');
% A = uint8(A);
% imwrite(A,'lena_nearest.bmp')
% A = my_zoom(Image_L,2048,2048,'linear');
% A = uint8(A);
% imwrite(A,'lena_linear.bmp')
% A = my_zoom(Image_L,2048,2048,'cubic');
% A = uint8(A);
% imwrite(A,'lena_cubic.bmp')

%子函数 my_zoom()
% function image = my_zoom(Raw_Image,length,width,c) %放大函数,Raw_Image 为原图像,length 为放大后图像宽,width 为放大后图像高,c 为插值方法
%                                                    %'linear'为双线性插值,'cubic'为三次插值
% [m,n] = size(Raw_Image);
% Raw_Image = [Raw_Image,Raw_Image(:,n)]; %考虑到边界问题，在原图基础上添加一行一列数据
% Raw_Image = [Raw_Image;Raw_Image(m,:)];
% [x,y] = meshgrid(1:1:m+1,1:1:n+1);
```

```
% [x1,y1] = meshgrid(1:n/width:n+1-n/width,1:m/length:m+1-m/length);
% image = interp2(x,y,Raw_Image,x1,y1,c);
% end
```

第五题：

```
Image_E = imread('elain1.bmp');
Image_L = imread('lena.bmp');
T1 = [1,1.5,0;0,1,0;0,0,1];
T2 = [1,0,0;1.5,1,0;0,0,1];
tform1 = affine2d(T1);
tform2 = affine2d(T2);
E_shearH = imwarp(Image_E,tform1);
E_shearV = imwarp(Image_E,tform2);
E_rotate = imrotate(Image_E,30);
%Elain.bmp zoom 到 2048×2048
E_shearN = imresize(E_shearH,[2048 2048],'nearest');
E_shearL = imresize(E_shearH,[2048 2048],'bilinear');
E_shearC = imresize(E_shearH,[2048 2048],'bicubic');
E_rotateN = imresize(E_rotate,[2048 2048],'nearest');
E_rotateL = imresize(E_rotate,[2048 2048],'bilinear');
E_rotateC = imresize(E_rotate,[2048 2048],'bicubic');

L_shearH = imwarp(Image_L,tform1);
L_shearV = imwarp(Image_L,tform2);
L_rotate = imrotate(Image_L,30);
%lena.bmp zoom 到 2048×2048
L_shearN = imresize(L_shearH,[2048 2048],'nearest');
L_shearL = imresize(L_shearH,[2048 2048],'bilinear');
L_shearC = imresize(L_shearH,[2048 2048],'bicubic');
L_rotateN = imresize(L_rotate,[2048 2048],'nearest');
L_rotateL = imresize(L_rotate,[2048 2048],'bilinear');
L_rotateC = imresize(L_rotate,[2048 2048],'bicubic');
```

参考文献：

【1】张德丰. MATLAB 数字图像处理. 北京：机械工业出版社, 2009.