

数字图像处理

第二次作业



摘要

本报告通过了解图像灰度级概念，图像均值方差的概念，以及一些图像变换相关的概念，利用Matlab实现以上功能，实现了理论与实践的相结合，对这些概念的把握更加深刻，可以不通过计算机来计算简单的图像变换了。

杨致远

电信钱51

2150405061

2018/3/22

## 任务1—图像灰度级变换

**实验介绍：**

通过查资料，灰度级的意思是，例如256（2^8）级就是从黑到白共有256种亮度变化，而128级就是将原来的256种亮度差异减小，将这256种亮度重新划分成128种亮度。

所以在第二题当中，要实现图像的灰度级降低，就是将原图中的每个像素点的值重新划分，比如原来是0或1值，现在均为0值；原来是2或3值的，现在均为1值。但是灰度图0代表黑，255代表白，所以不能直接缩小，灰度值计算缩小范围后，还得扩大回原来的0~255范围，在这里我尝试了几种缩小的思路。

1. 一种量化的思路。四舍五入取整运算(n/s)\*s，n代表了原来的像素值，s代表一个要缩小的灰度级，例如: 要将256级分成128级，用0~255去除以2，这样的除法加上取整运算就是使原来的255个值现在都变成了偶数，但这种方法存在一个问题，就是出来的数会多一个，例如要分成2级，最终会产生0，128，256三个值，这样对后续的处理是不利的。
2. 我在这里采用的是先缩小在扩大的策略，先直接除，(floor)n/s，要向下取整，例如分成4级，s=64，这样除后的结果就产生了0，1，2，3这样4个数，正好满足现在4灰度级的要求，然后再扩大，0依然是0，3要变成255需要乘以85，再让1和2均乘以85，这样就达到了等差扩大的效果。

操作中还需要注意，读取的图片像素值都是uint8编码格式，后面的计算它会自动地四舍五入，所以方便起见需要进行double转化；另一个问题就是会存在255除后不整的情况，只需要在最后的结果加上uint8转换即可。

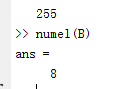
**结果展示：**



**结果讨论：**

由上图可以看出，在灰度级下降到32之前，肉眼几乎感觉不出降阶后图像发生的变化。但从16灰度级开始，肉眼明显能感觉到图像有稍许的不连续，与原图灰度渐变区对应位置灰度不连续的现象表现更加明显。随着灰度阶数的继续下降，图像开始出现大片的伪轮廓，最终的极端情况是退化为只有黑白两色的二值化图像。由此可以得出，图像采样的灰度阶数越高，肉眼看去更接近与现实情况。

同时我们还可以客观地验证，我的实验操作最后达到的颜色数量是否与级别数对应，我们以8灰度级为例，编写代码，发现此时的像素值只有8种，这样就验证了我的处理取得了成功。



## 任务2—图像均值方差的计算

**实验介绍：**

图像计算均值和方差与我们代数上的计算类似，图像数据读出来是一个矩阵，矩阵中每个值代表了图像的每个像素值，均值就是所有值相加再除以[长×宽]，方差与之类似。因为自己手写起来没有什么需要注意的地方，我就直接调用现有的库函数进行计算，用mean2()函数来计算均值，std2()函数来计算标准差。

注意事项：

1、用的函数是mean2()，不是mean()，前者是直接用来计算矩阵的，后者是用来计算数组的，同样也可以计算矩阵就是麻烦一些。

2、imread()读取的数据是uint8类型的，一般在使用时候还需要将数据转换为double类型，但是，根据使用说明，mean2()函数返回值就是singe 或者double类型，所以在这里我们无须进行数据类型的转换。

**结果展示：**



**结果讨论：**

图像的均值可反应图像整体的明暗程度，而方差可以反应图像整体的对比度情况。所以这个lena图片整体偏暗，对比度较为强烈。

## 任务3—近邻、双线性和双三次插值法的使用

**实验介绍：**

图像插值就是利用已知邻近像素点的灰度值来产生未知像素点的灰度值，以便由原始图像再生出具有更高分辨率的图像。典型方法包括最近邻、双线性、双三次三种方法。

最邻近插值法就是将待插值点像素的灰度值取为其相对于原图坐标中最邻近像素点的灰度值。这种方法比较简单，计算速度快，但效果不好，会产生严重的“棋盘格”效应。

双线性插值法是将待插值点像素的灰度值取为其在原图坐标中距离最近的4个像素点灰度值的加权和，而双三次法则是选取的邻近的16个点。

我在这里说一下最邻近插值法，最邻近我用的是将坐标（i,j）除以4，然后利用高斯取整，就能够将放大后的点归属到距离它最近的原坐标上了。

注意事项：

关于取整问题，目前有的是向下取整，向上取整，四舍五入取整。

**结果展示：**



**结果讨论：**

通过对处理后的图片进行放大观察，可以发现经过最邻近插值后的图像产生了“棋盘格”效应，而后两种插值方法表现的效果比较平滑。同时比较程序的运行时间发现三种方法运行时间逐级递增，这是因为运算复杂度增加了。

将自己写的代码实现结果与现有函数实现结果矩阵进行比较发现完全一致。

## 任务4—仿射变换

**实验介绍：**

二维仿射变换是一种二维坐标到二维坐标之间的线性变换，保持二维图形的“平直性”和“平行性”仿射变换可以通过一系列的原子变换的复合来实现，包括：平移（Translation）、缩放（Scale）、翻转（Flip）、旋转（Rotation）和错切（Shear）等。

二维仿射变换可以用一个3×3的矩阵来表示，其最后一行为(0 0 1)。该变换矩阵将原坐标(v,w)变换为新坐标(x,y)。设仿射矩阵为T，则新坐标与原坐标的关系为：

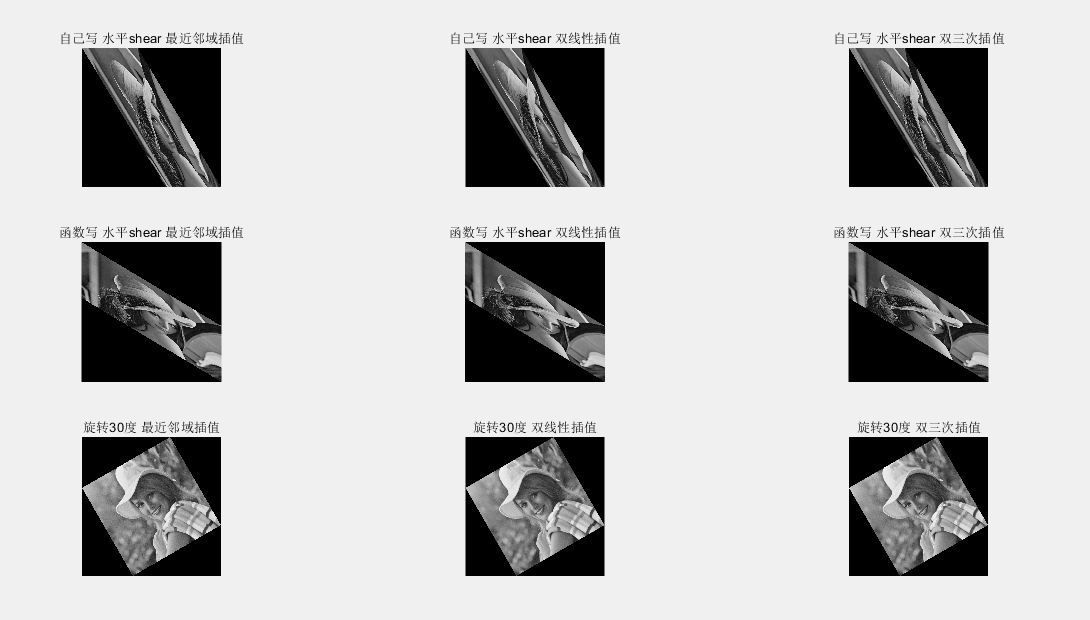
[x y 1]=[u v 1]\*T

T可以参照课本P51页的表2.2获知

注意事项：

Matlab里面的sin()，cos()函数括号里面都是弧度制。

**结果展示：**



**结果讨论：**

因为在计算时，存在前向映射以及反向映射，所以我自己写出来的和自带函数做出来的方向上有所差异。同时在实验中还应该注意仿射变换后可能导致图像超出原始尺寸边框，需要作扩充处理。

## 代码：

实验1%%把lena 512\*512图像灰度级逐级递减8-1显示

picture=imread('第一次作业\lena.bmp');

subplot(2,4,1);

imshow(picture);

title('256级原图像');

for i = 5:5

subplot(2,4,i+1);

a=2^i;

b=floor(double(picture)/a);

c=255/(2^(8-i)-1);

d=uint8(b\*c);

imshow(d);

title(sprintf("%d级灰度图像", 2^(8-i)));

end

实验2%%计算lena图像的均值方差

picture=imread('第一次作业\lena.bmp');

a=mean2(picture); %计算均值

fprintf('均值为： %8.5f\n',a)

b=std2(picture); %计算标准差

c=b^2; %计算方差

fprintf('方差为： %8.5f\n',c)

实验3%%把lena图像用近邻、双线性和双三次插值法zoom到2048\*2048

%%利用现有函数做的：

picture=imread('第一次作业\lena.bmp');

subplot(2,3,1);

picture1=imresize(picture,[2048,2048],'nearest'); %最近邻域插值;

imshow(picture1);

title('函数实现：最近邻域插值');

subplot(2,3,2);

picture2=imresize(picture,[2048,2048],'bilinear'); %双线性插值;

imshow(picture2);

title('函数实现：双线性插值');

subplot(2,3,3);

picture3=imresize(picture,[2048,2048],'bicubic'); %双三次插值;

imshow(picture3);

title('函数实现：双三次插值');

%%自己写实现，原图像是512\*512，要变成2048\*2048，要扩大4倍

subplot(2,3,4); %最近邻域插值;

for i=1:2048

for j=1:2048

picture4(i,j) = picture(ceil(i/4),ceil(j/4));

end

end

imshow(picture4);

title('自己实现：最近邻域插值');

实验4：%%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，并采用用近邻、双线性和双三次插值法zoom到2048\*2048

picture = imread('第一次作业\lena.bmp'); %读取图片

picture2 = imread('第一次作业\elain1.bmp');

%自己实现的函数：

A = [1 1.5 0;0 1 0;0 0 1]; %仿射变换矩阵，参数设置为1.5

theta=pi/6; %设置旋转角度30°

B = [cos(theta) sin(theta) 0;-sin(theta) cos(theta) 0;0 0 1]; %旋转变换矩阵

for i=1:512

for j=1:512

c=[i j 1]\*A; %错切变换

d=[i j 1]\*B; %旋转变换

picture31(round(c(1)),round(c(2)))=picture(i,j);

%picture32(round(d(1)),round(d(2)))=picture(i,j);

end

end

%现有函数实现

tform = affine2d(A); %仿射变换

AA = imwarp(picture,tform); %将几何变换应用于图形

rotate = imrotate(picture2,30); %旋转30度

subplot(3,3,1);

picture11 = imresize(picture31,[2048,2048],'nearest'); %最近邻域插值;

imshow(picture11);

title('自己写 水平shear 最近邻域插值');

subplot(3,3,2);

picture12 = imresize(picture31,[2048,2048],'bilinear'); %双线性插值;

imshow(picture12);

title('自己写 水平shear 双线性插值');

subplot(3,3,3);

picture13 = imresize(picture31,[2048,2048],'bicubic'); %双三次插值;

imshow(picture13);

title('自己写 水平shear 双三次插值');

subplot(3,3,4);

picture11 = imresize(AA,[2048,2048],'nearest'); %最近邻域插值;

imshow(picture11);

title('函数写 水平shear 最近邻域插值');

subplot(3,3,5);

picture12 = imresize(AA,[2048,2048],'bilinear'); %双线性插值;

imshow(picture12);

title('函数写 水平shear 双线性插值');

subplot(3,3,6);

picture13 = imresize(AA,[2048,2048],'bicubic'); %双三次插值;

imshow(picture13);

title('函数写 水平shear 双三次插值');

subplot(3,3,7);

picture21 = imresize(rotate,[2048,2048],'nearest'); %最近邻域插值;

imshow(picture21);

title('旋转30度 最近邻域插值');

subplot(3,3,8);

picture22 = imresize(rotate,[2048,2048],'bilinear'); %双线性插值;

imshow(picture22);

title('旋转30度 双线性插值');

subplot(3,3,9);

picture23 = imresize(rotate,[2048,2048],'bicubic'); %双三次插值;

imshow(picture23);

title('旋转30度 双三次插值');

## 参考文献

[1] 冈萨雷斯.数字图像处理（第三版）北京：电子工业出版社，2011

[2] 维基百科