

《数字图像处理》实验报告

一，实验目的

1. 编写程序，使图像（自行转换为灰度图）顺时针旋转 30 度，并进行适当填充。
2. 编写程序，使其基于最近邻和双线性插值将图像（自行转换为灰度图）分别放大 2 倍和 4 倍。
3. 编写程序，对以上图像（自行转换为灰度图）展开傅里叶变换，提取傅里叶变换图像（将频率原点移至图像中心）。

二，实验工具

MATLAB R2022b

三，实验原理

1. 图像获取

实验使用以下图片：



灰度图：



2. 图像旋转

图像旋转的本质利用的是向量的旋转。矩阵乘法的实质是进行线性变换，因此对一个向量进行旋转操作也可以通过矩阵和向量相乘的方式进行。因为图像都是通过二维矩阵存放的(单通道)，所以对图像进行旋转时要先设定一个像素作为旋转轴，然后其他的像素位置可以看作从旋转轴出发的向量。

如图中间的点为旋转轴，则旋转的实质就是将图中的各个向量进行旋转，然后将旋转前的位置的像素值赋值给旋转后的位置的像素值。

假设有二维向量 $v = [x; y]$ ，若要进行逆时针旋转角度 a 。则旋转矩阵 R 为

$$\begin{bmatrix} \cos(a) & -\sin(a) \\ \sin(a) & \cos(a) \end{bmatrix}$$

旋转后的向量：

$$v_2 = R * v$$

在正式处理过程中可以这么表示，原像素位置记为 p ，中心点记为 c ，旋转后像素位置记为 pp

则有：
$$(pp - c) = R \cdot (p - c)$$

即：

$$pp = R \cdot (p - c) + c$$

对于本次实验，要求将图像顺时针旋转 30 度，相当于将图像逆时针旋转 -30 度，所以，我们将 -30 度用弧度表示，代入到上述旋转矩阵中。在初步旋转后，发现结果图像有部分被原图边界截断，所以在旋转前应根据旋转角度计算新图像的大小。最终实验代码如下所示：

```
H1=imread('实验图像.bmp');
H1=rgb2gray(H1);
imshow(H1);

%求旋转矩阵
a=30/180*pi;
r=[cos(a),sin(a);sin(-a),cos(a)];

%求原图像大小
sz=size(H1);
h=sz(1);
w=sz(2);
c1=[h;w]/2;

%求变换后图像的大小
hh=floor(w*sin(a)+h*cos(a))+1;
ww=floor(w*cos(a)+h*sin(a))+1;
c2=[hh;ww]/2;

%旋转
im2=uint8(zeros(hh,ww,1));
for i=1:h
    for j=1:w
        p=[i;j];
        pp=round(r*(p-c1)+c2);
        if(pp(1)>=1&&pp(1)<=hh&&pp(2)>=1&&pp(2)<=ww)
            im2(pp(1),pp(2),1)=H1(i,j,1);
        end
    end
end
```

%对旋转图像插值，取周围灰度最小值

由于图像旋转计算得到的新图像的坐标往往不是整数，程序会对其进行四舍五入。这就导致了旋转后的图像会有许多空白点。所以要对图像进行插值处理。本次实验将空白点附近的灰度最小值近似为空白点的灰度值。实验代码如下：

```

%对旋转图像插值，取周围灰度最小值
for i=1:hh
    for j=1:ww
        if(im2(i,j)==0&& i>1&& i<hh&& j>1&& j<ww)
            s=min([im2(i-1,j),im2(i+1,j),im2(i,j-1),im2(i,j+1)]);
            im2(i,j)=s;
        end
    end
end

%imshow(im2);

```

3. 图像插值

1) 最近邻插值

顾名思义，最近邻插值就是在目标像素点上插入离对应原像素点最近的点的像素值。因此，该算法方法较为单一，原理也较为简单，计算量也是最少的。但是，这个方法也有着很明显的缺点，那就是锯齿化高，在颜色变化程度越高以及放大率越大的图像上进行插值处理时，锯齿化现象越严重。

对于程序，我们可以直接将要插值的点的坐标四舍五入，最后得到的坐标就是改点的最近邻坐标（round 后的 x 和 y 需要进行时候为 0 的判断以避免报错的情况）。该方法实现较为简单，代码如下图所示：

```

%基于最近邻插值放大两倍
h1=h*2;
w1=w*2;
im3=uint8(zeros(h1,w1,1));
for i=1:h1
    for j=1:w1
        i1=round(i/2);
        j1=round(j/2);
        if(i1==0)
            i1=i1+1;
        end
        if(j1==0)
            j1=j1+1;
        end
        im3(i,j)=H1(i1,j1);
    end
end

%imshow(im3);

```

2) 双线性插值

双线性插值中最主要的算法思想有两点：第一是双，指的是在两个方向上（x，y）取值，一般在 x 方向上取两个值，在 y 方向上取两个值；第二是线性，将各方向取的两个值做线性函数的计算，一般以 x 和 y 方向上的权重计算实际值，类似于

一次函数，越靠近映射点权重越高，并且让整体权重和=1。

根据上述的算法思想，通过映射点周围的 4 个点来进行计算，使得新像素点的颜色能具有不同于最近邻插值算法的简单颜色变化。由此使得图像插值算法得到一定改进。

然而，双线性插值算法仍然具有其短板，简单线性变化会带来边缘模糊化，在颜色变化程度大的边缘处，会明显发现插值的数据在边缘显得不符。另外由于增加了线性权重与像素值相乘，会存在一定的低通滤波性，高频分量会被削弱，在颜色变化程度大的地方可能无法得到正确的像素值。

设 (i,j) 为新图像像素点对应原图像的实际像素点位置，实际像素点位置坐标是包含小数点的，而小数值即是它的方向权重， u 为水平方向上权重即 $i-x$ ， v 为垂直方向上权重即 $j-y$ 。

则实际像素点处的线性值即为：

$$u*v*img(x,y)+(1-u)*v*img(x+1,y)+u*(1-v)*img(x,y+1)+(1-u)*(1-v)*img(x+1,y+1)$$

实验代码如下：

```
%基于双线性插值放大四倍
h2=h*4;
w2=w*4;
im4=uint8(zeros(h2,w2,1));
for i=1:h2
    for j=1:w2
        i2=round(i/4);
        j2=round(j/4);
        if(i2==0)
            i2=i2+1;
        end
        if(j2==0)
            j2=j2+1;
        end
        u=i/4-floor(i/4);
        v=j/4-floor(j/4);
        if(i>=h2-4||j>=w2-4)
            im4(i,j)=H1(i2,j2);
        else
            im4(i,j)=u*v*H1(i2,j2)+u*(1-v)*H1(i2,j2+1)+(1-u)*v*H1(i2+1,j2)+(1-u)*(1-v)*H1(i2+1,j2+1);
        end
    end
end

%imshow(im4);
```

4. 傅里叶变换

二维离散傅里叶变换如下图：

$$F(u, v) = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x, y) e^{-j2\pi \left(\frac{ux}{M} + \frac{vy}{N} \right)}$$

逆变换：

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

https://blog.csdn.net/weixin_43627492

其中, f 表示原图像灰度值函数, F 表示傅里叶变换后的函数, x, y 分别表示原图像像素点的横纵坐标, u, v 分别表示频域的横纵坐标, M, N 分别表示图像的长和宽。

傅里叶变换可将空域的数字图像信号转化为频域的信号, 是一种处理数字图像的重要方式, 然而, 对于计算机来说, 傅里叶变换计算量巨大, 需要构建四重循环, 大大增加了程序运行时间。因此, 我们需要对傅里叶变换进行适当的简化。

可将傅里叶变换分离:

正变换:

$$F(x, v) = \frac{1}{N} \sum_{y=0}^{N-1} f(x, y) \exp[-j2\pi vy / N]$$

$$F(u, v) = \frac{1}{N} \sum_{x=0}^{N-1} F(x, v) \exp[-j2\pi ux / N]$$

对应的, 也可以将逆变换分离。

为了更好地观察图像的频域特性, 还需要把频域图像平移至中心位置。

空域平移:

$$f(x - x_0, y - y_0) \Leftrightarrow F(u, v) \exp[-j2\pi(ux_0 + vy_0)/N]$$

频域平移:

$$f(x, y) \exp[j2\pi(u_0x + v_0y)/N] \Leftrightarrow F(u - u_0, v - v_0)$$

代码如下所示:

```
clear
clc
img=imread('实验图像.bmp');
subplot(2,2,1);imshow(img);title('原图');
f=rgb2gray(img);
f=im2double(f);
sz=size(f);
h=sz(1);
w=sz(2);
FF = ones(h,w);
Fs=ones(h,w);
Fsi=ones(h,w);
F=ones(h,w);
com = 0+1i; ..
```

```

%F=fft2(f); 傅里叶变换
for x=1:h
    for v=1:w
        s=0;
        for y=1:w
            s=s+f(x,y)*exp(-com*2*pi*v*y/w);
        end
        FF(x,v)=s;
    end
end

for u=1:h
    for v=1:w
        s=0;
        for x=1:h
            s=s+FF(x,v)*exp(-com*2*pi*u*x/h);
        end
        F(u,v)=s;
    end
end

F1=log(abs(F)+1);    %取模并进行缩放
subplot(2,2,2);imshow(F1,[]);title('傅里叶变换频谱图');
%Fs=fftshift(F);%将频谱图中零频率成分移动至频谱图中心
for x=1:h
    for v=1:w
        s=0;
        for y=1:w
            s=s+f(x,y)*[(-1)^(x+y)]*exp(-com*2*pi*v*y/w);
        end
        FF(x,v)=s;
    end
end

for u=1:h
    for v=1:w
        s=0;
        for x=1:h
            s=s+FF(x,v)*exp(-com*2*pi*u*x/h);
        end
        Fs(u,v)=s;
    end
end

S=log(abs(Fs)+1);    %取模并进行缩放
subplot(2,2,3);imshow(S,[]);title('频移后的频谱图');

```

```

%fr=real(iff2(iff2shift(Fs))); %频率域反变换到空间域，并取实部
for u=1:h
    for y=1:w
        s=0;
        for v=1:w
            s=s+F(u,v)*exp(com*2*pi*v*y/w);
        end
        FF(u,y)=s;
    end
end

for x=1:h
    for y=1:w
        s=0;
        for u=1:h
            s=s+FF(u,y)*exp(com*2*pi*u*x/h);
        end
        Fsi(x,y)=s;
    end
end
fr=real(Fsi);
ret=im2uint8(mat2gray(fr)); %更改图像类型
subplot(2,2,4);imshow(ret),title('逆傅里叶变换');

```

四，实验结果

旋转：



最近邻插值:



双线性插值:



傅里叶变换:

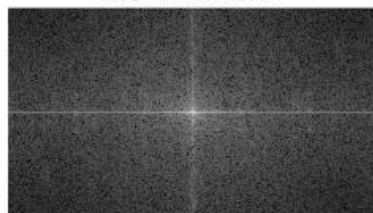
原图



傅里叶变换频谱图



频移后的频谱图



逆傅里叶变换



五，结果分析与总结

旋转后插值的图像比原图像模糊了一些，且有锯齿。运用最近邻插值法得到的图像锯齿较多，而用双线性插值法得到的图像锯齿减少，图像更加平滑，但物体的轮廓变得模糊。使用傅里叶变换时，由于程序中含有三个三级循环，所以程

序运行时间很长，远低于 MATLAB 本身自带的傅里叶变换函数。

本次实验，我更加深入的了解关于数字图像的旋转，插值和傅里叶变换的内容。学会了如何在 MATLAB 上对数字图像进行处理。

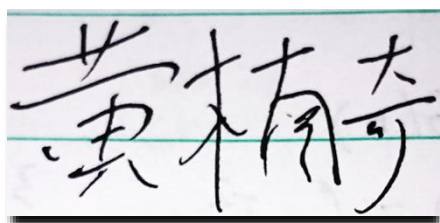
课后习题

- 结合授课内容和自身的理解，请尝试阐述傅里叶变换与傅里叶系数的物理含

义，可以上网查阅相关的资料。

答：傅里叶变换的物理含义是将一个信号分解为不同频率的正弦波组成的谱，从而揭示了信号的频率特性。傅里叶级数就是函数在某个函数空间中各个基底的投影。

电子签名：

A handwritten signature in black ink on a white background. The signature is written in a cursive style and appears to read '黄梦奇' (Huang Mengqi). The signature is positioned on a horizontal line.