DIP 第一次实验报告

一、 实验目的

内插是图像放大、收缩中广泛应用的基本工具。该实验分别利用最近邻插值、双线性插值和双立方插值来实现图像的放大与缩小。

二、 实验原理

a) 最近邻插值：将目标图像各点的像素值设为源图像中与其最近的点。

设（i+u, j+v）为待求象素坐标在源图像中的映射，则待求象素灰度的值 f(i+u, j+v)。则 i, j 为正整数, u, v 为大于零小于 1 的小数。根据 u,v 与 0.5 的关系，可以将源图像(i, j) ,(i+1, j) ,(i, j+1) ,(i+1, j+1)对应的值分配给该点。

b) 双线性插值：双线性插值是有两个变量的插值函数的线性插值扩展，其核心思想是在两个方向分别进行一次线性插值。

$$f(R_1) \approx \frac{x_2 - x}{x_2 - x_1} f(Q_{11}) + \frac{x - x_1}{x_2 - x_1} f(Q_{21}) \quad \text{where} \quad R_1 = (x, y_1),$$

$$f(R_2) \approx \frac{x_2 - x}{x_2 - x_1} f(Q_{12}) + \frac{x - x_1}{x_2 - x_1} f(Q_{22}) \quad \text{where} \quad R_2 = (x, y_2).$$

然后在 y 方向进行线性插值，得到

$$f(P) \approx \frac{y_2 - y}{y_2 - y_1} f(R_1) + \frac{y - y_1}{y_2 - y_1} f(R_2).$$

c) 双立方插值：

插值公式：

$$S(x) = \begin{cases} 1-(a+3)x^2+(a+2)|x|^3, & 0 \le |x| \le 1 \\ -4a+8a|x|-5ax^2+a|x|^3, & 1 < |x| \le 2 \end{cases}$$

计算公式：

$$F(i+v,j+u) = A * B * C$$

$$A = \begin{pmatrix} S(1+v) & S(v) & S(1-v) & S(2-v) \end{pmatrix}$$

$$B = \begin{pmatrix} f(i-1,j-1) & f(i-1,j) & f(i-1,j+1) & f(i-1,j+2) \\ f(i,j-1) & f(i,j) & f(i,j+1) & f(i,j+2) \\ f(i+1,j-1) & f(i+1,j) & f(i+1,j+1) & f(i+1,j+2) \\ f(i+2,j-1) & f(i+2,j) & f(i+2,j+1) & f(i+2,j+2) \end{pmatrix}$$

$$C = \begin{pmatrix} S(1+u) \\ S(u) \\ S(1-u) \\ S(2-u) \end{pmatrix}$$

等价于(可自行推导)

$$F(i+v,j+u) = \sum_{row=-1}^{2} \sum_{col=-1}^{2} f(i+row,j+col)\, S(row-v)S(col-u)$$

三、 代码实现

使用 MATLAB 实现三种插值效果，其中”main.m”是主函数，“myNearest.m“是最近邻插值函数，“myBilinear.m”是双线性内插函数，“myBicubic.m”是立方卷积函数。

四、 代码分析

a) main.m

i. 第一行读入图像，第二行将 RGB 图像转换为灰度图像以方便处理，而第三行则将图像统一转换为 double 型图像，方便数学处理。

```
im = imread([filename, '.jpg']);
im = rgb2gray(im);
im = im2double(im);
```

b) myNearest.m

i. 给出边界条件

```
if sid1(1) == 0
    sid1(1) = 1;
end
if sid1(2) == 0
    sid1(2) = 1;
end
if sid1(1) > h_1
    sid1(1)= h_1;
end
if sid1(2) > h_1
    sid1(2) = h_1;
end
if sid1 == floor(sid1)
    img_2(i,j) = img_1(sid1(1), sid1(2));
    continue;
end
```

ii. 寻找最近邻：

```
[row, column] = find( [sq1 sq2; sq4 sq3] ==
min(min([sq1 sq2; sq4 sq3])));
```

c) myBilinear.m

i. 考虑一般情况

```
if fR11(1) == 0 && fR11(2) == 0
          img_2(i,j) = img_1(1,1);
          continue;
       end  %(0,0)
       if fR11(1) == 0 && fR11(2) == w_1
          img_2(i,j) = img_1(1, w_1);
          continue;
       end %(0, w_1)
       if fR11(1) == h_1 && fR11(2) == w_1
          img_2(i,j) = img_1(h_1, w_1);
          continue;
       end %(h_1, 0)
       if fR11(1) == h_1 && fR11(2) == 0
          img_2(i,j) = img_1(h_1, 1);
          continue;
       end %(h_1, w_1)
       if fR11(1) == 0
          fR11(1) = 1;
          f_final = (img_1( 1, fR11(2)+1 ) -
img_1( 1, fR11(2) ) )* ( cen(2) - fR11(2) ) +
img_1(1,fR11(2));
          img_2(i,j) = f_final;
          continue;
       end  %(0,y)
       if fR11(2) == 0
          fR11(2) = 1;
          f_final = (img_1( fR11(1)+1 , 1 ) -
img_1( fR11(1), 1 ) )* ( cen(1) - fR11(1) ) +
img_1(fR11(1),1);
          img_2(i,j) = f_final;
          continue;
       end  %(x,0)

       if fR11(1) == h_1
          f_final = (img_1( h_1, fR11(2)+1 ) -
img_1( h_1, fR11(2) ) )* ( cen(2) - fR11(2) ) +
img_1( h_1,fR11(2) );
          img_2(i,j) = f_final;
          continue;
       end %(h_1,y)
       if fR11(2) == w_1
          f_final = (img_1( fR11(1)+1 , w_1 ) -
img_1( fR11(1), w_1 ) )* ( cen(1) - fR11(1) ) +
img_1(fR11(1),w_1);
```

```
            img_2(i,j) = f_final;
            continue;
        end  %(x,w_1)
```

ii. 一般情况：

1. X 线性化：

```
% X linear
f_R1 = img_1( fR11(1), fR11(2) ) + ( img_1( fR21(1), fR21(2) ) - img_1( fR11(1), fR11(2) ) )*( cen(1) - fR11(1) );
f_R2 = img_1( fR12(1), fR12(2) ) + ( img_1( fR22(1), fR22(2) ) - img_1( fR12(1), fR12(2) ) )*( cen(1) - fR12(1) );
```

2. Y 线性化：

```
% Y linear
f_final = f_R1 + ( f_R2 - f_R1 )*( cen(2) - fR11(2) );
```

d) myBicubic.m

i. 将图像零拓展，x,y 各 5 行（列）

```
img_2 = zeros(h_2, w_2);
img_1broad = zeros(h_1+5, w_1+5);
img_1broad(3:h_1+2, 3:w_1+2) = img_1;
img_1 = img_1broad;
```

ii. 直接计算：

```
for i = 1:h_2
    for j = 1:w_2
        x = floor(i / n)+2;
        u = i/n - floor(i/n);
        y = floor(j / n)+2;
        v = j/n - floor(j/n);
        img_2(i,j) = calculate(img_1,x,y,u,v);
    end
end
```

iii. S(x)函数：用于计算 S

```
function result=interpolation(x)

a = -0.5;
x = abs(x);
x2 = x*x;
x3 = x*x2;
if x <= 1
    result = 1 - ( a + 3 ) * x2 + ( a + 2 ) * x3;
elseif x <= 2
    result = -4 * a + 8 * a* x - 5 * a * x2  + a * x3;
else
    result = 0;
end
end
```

iv. Calculate 函数：用于计算最终的函数值

```
function sum = calculate(img_1, i,j,u,v)

sum = 0;
for row = -1:2
    for column = -1:2
        sum = sum + img_1(i+row, j+column)*interpolation(row-u)*interpolation(column-v);
    end
end
end
```

五、 实验结果与分析

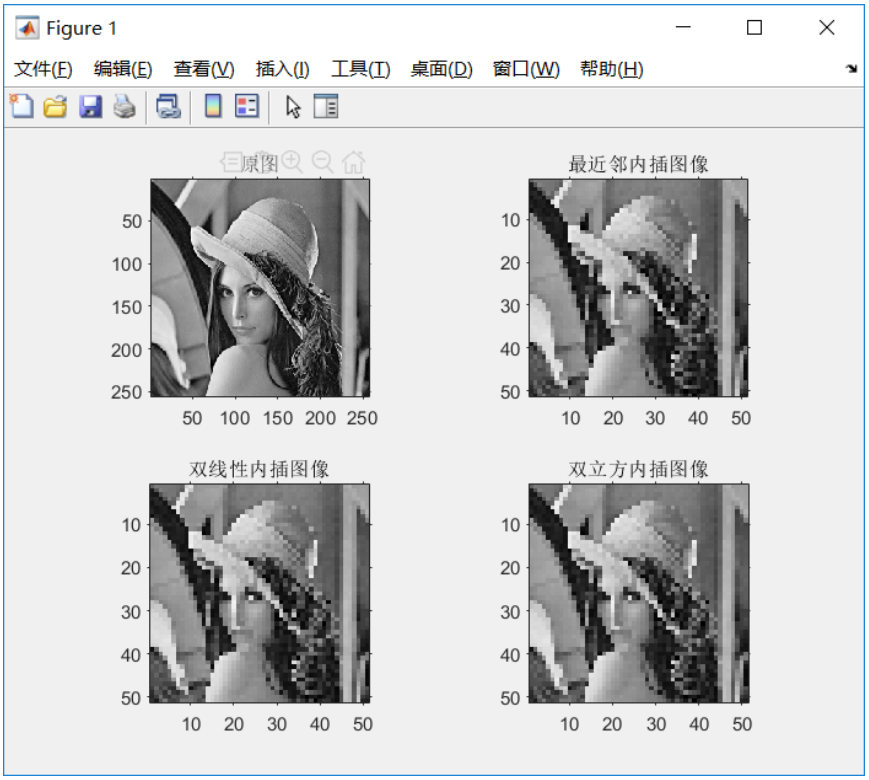
Figure 1
文件(F) 编辑(E) 查看(V) 插入(I) 工具(T) 桌面(D) 窗口(W) 帮助(H)
原图
最近邻内插图像
双线性内插图像
双立方内插图像

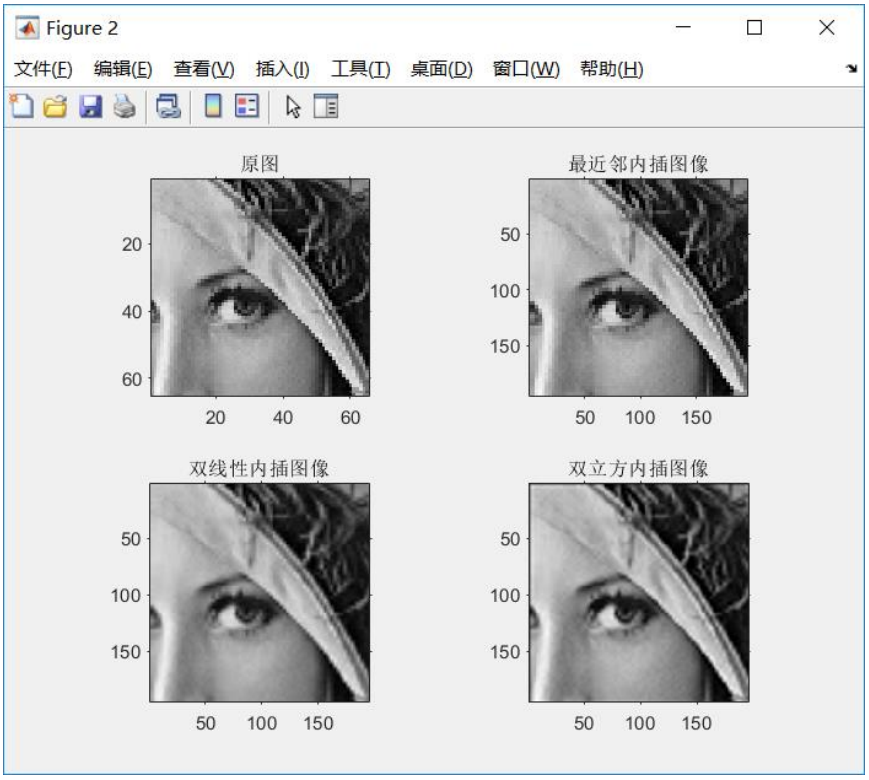
Figure 2
文件(F) 编辑(E) 查看(V) 插入(I) 工具(T) 桌面(D) 窗口(W) 帮助(H)
原图
最近邻内插图像
双线性内插图像
双立方内插图像

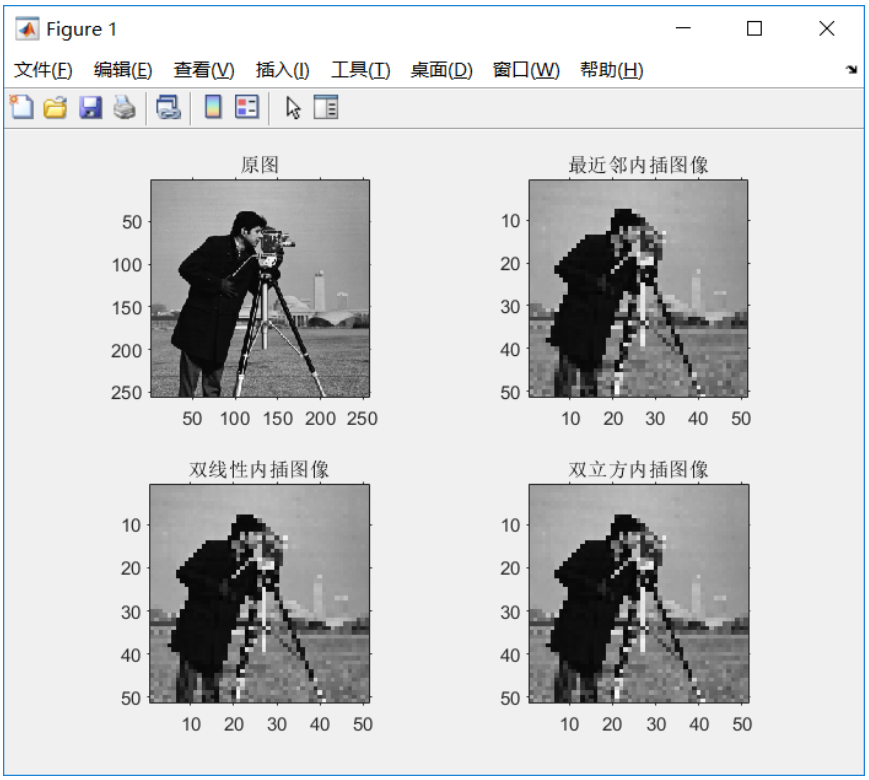
Figure 1
文件(F) 编辑(E) 查看(V) 插入(I) 工具(T) 桌面(D) 窗口(W) 帮助(H)
原图
最近邻内插图像
双线性内插图像
双立方内插图像

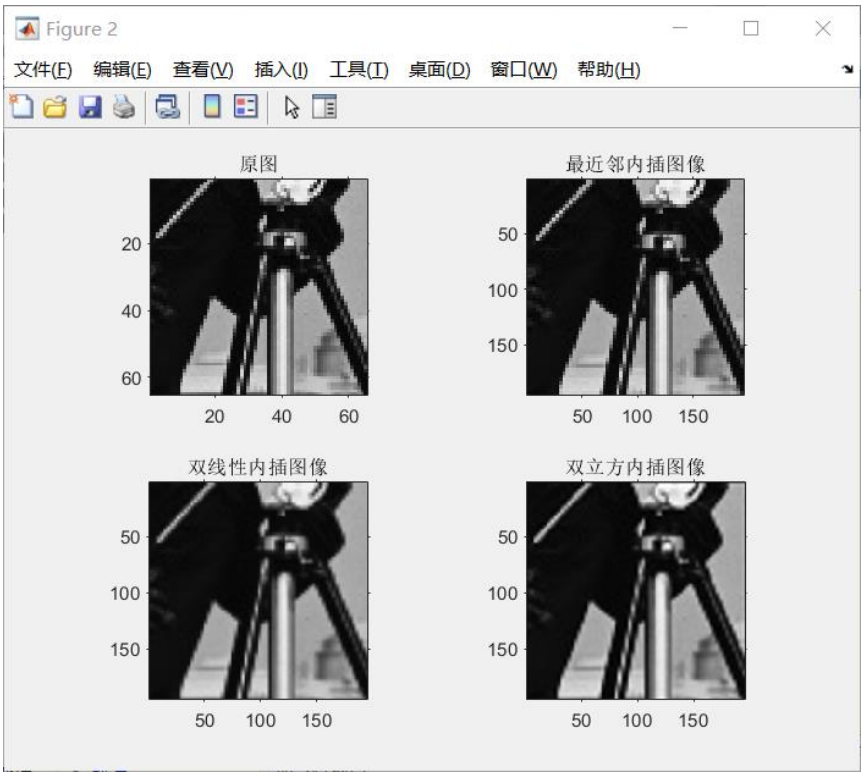
Figure 2
文件(F) 编辑(E) 查看(V) 插入(I) 工具(T) 桌面(D) 窗口(W) 帮助(H)
原图
最近邻内插图像
双线性内插图像
双立方内插图像

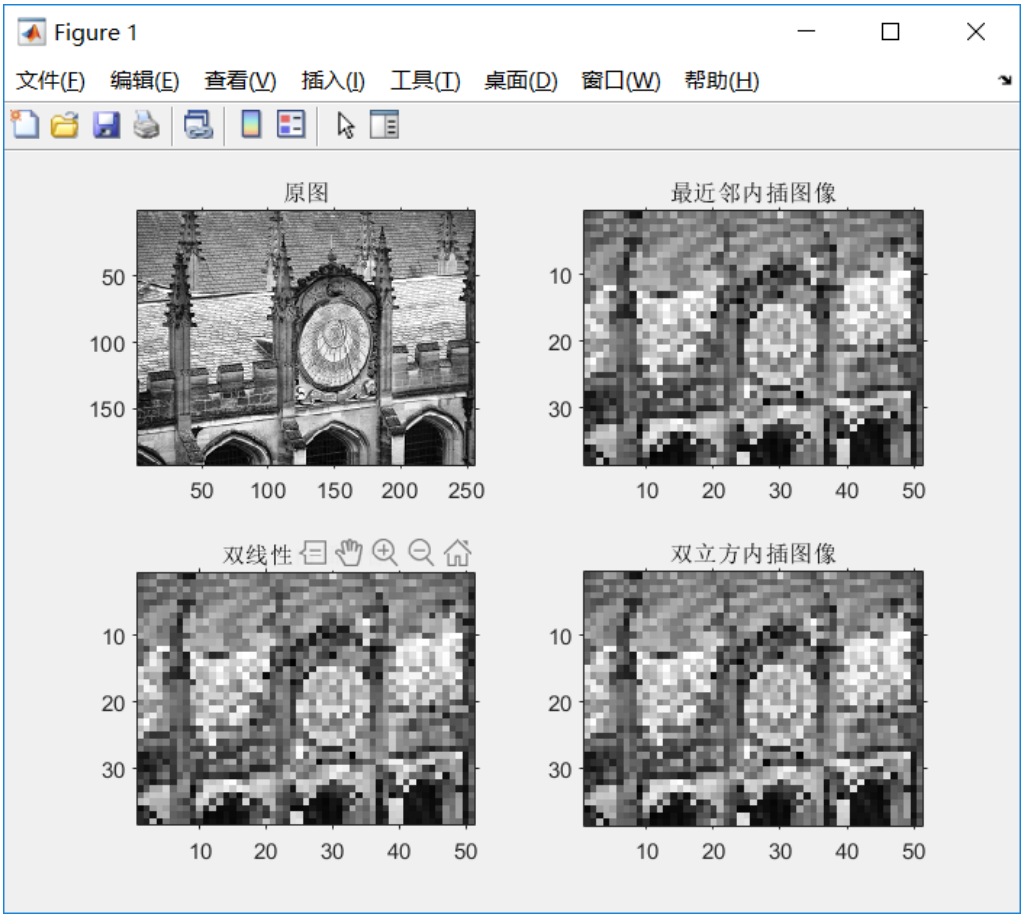
Figure 1
文件(F) 编辑(E) 查看(V) 插入(I) 工具(T) 桌面(D) 窗口(W) 帮助(H)
原图
最近邻内插图像
双线性
双立方内插图像

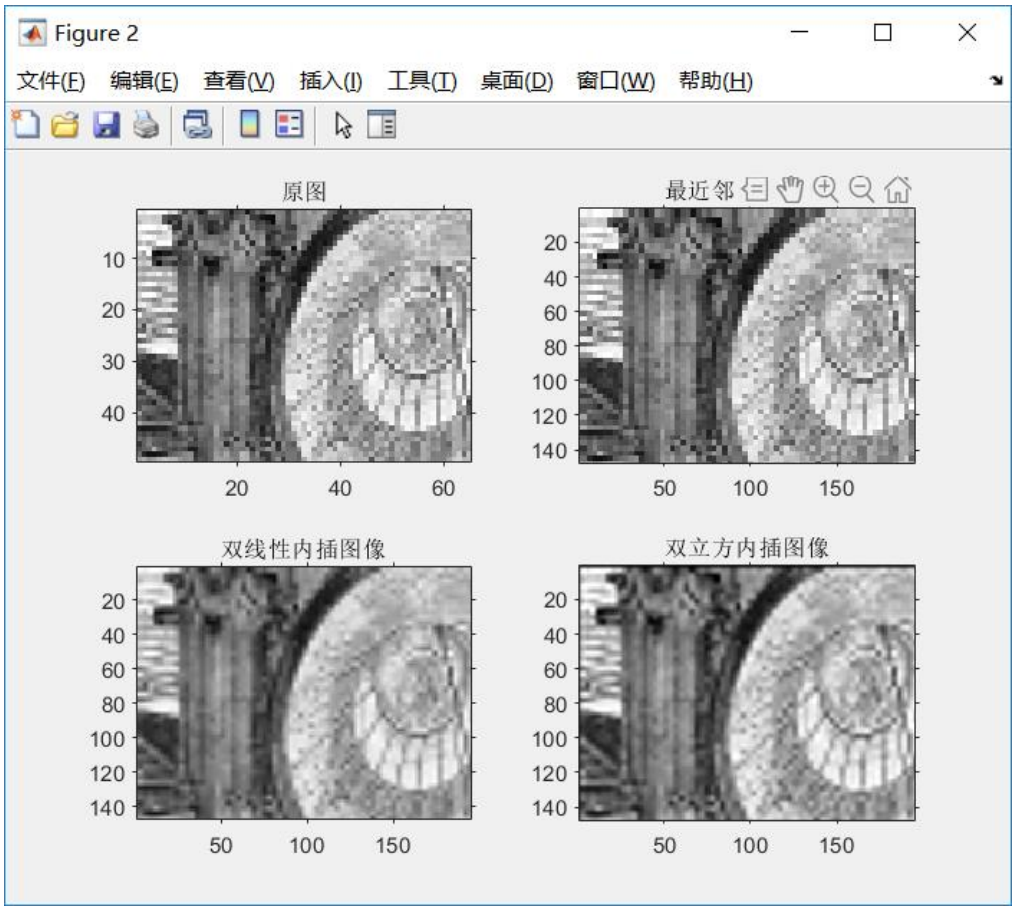
Figure 2
文件(F) 编辑(E) 查看(V) 插入(I) 工具(T) 桌面(D) 窗口(W) 帮助(H)
原图
最近邻
双线性内插图像
双立方内插图像

- 信息恢复程度:

图像缩小时，三种算法并没有非常明显的区别。
而图像放大时，效果由图像包含的信息多少而决定:
在细节信息包含不多的情况下，如"cameraman"，恢复图像信息的效果三种算法并没有多少差距。
而在细节信息丰富时，如“building”，“Lenna”，恢复图像信息的效果从好到坏排序为：双立方内插>双线性内插>最近邻内插。

- 图像平滑程度

此外，使图像轮廓更加平滑（模糊）的效果从好到坏的排序为：双立方内插>双线性内插>最近邻内插。这是由于算法本身操作像素块领域的大小决定的。