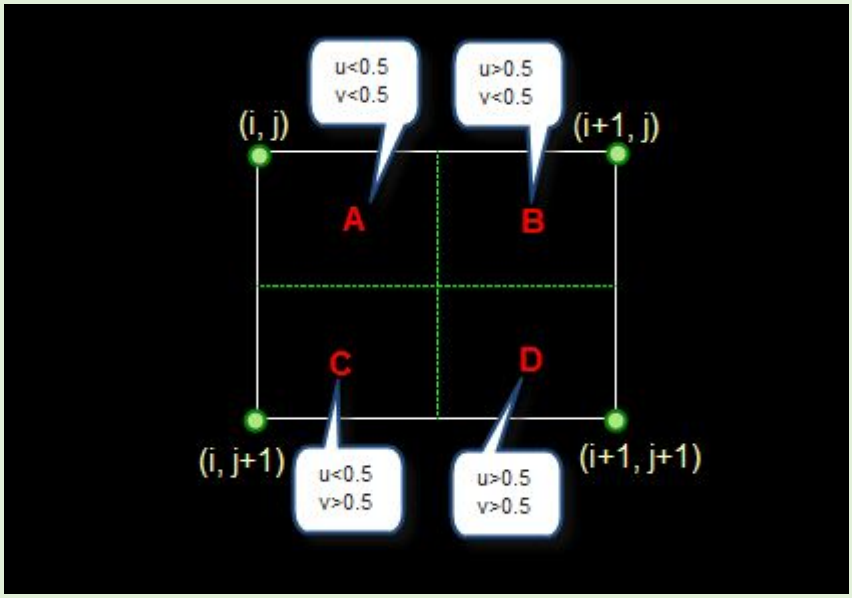
# 一、最近邻内插法

在待求象素的四邻象素中，将距离待求象素**最近**的邻象素灰度赋给待求象素。设 i+u, j+v(i, j 为正整数， u, v 为大于零小于 1 的小数，下同)为待求象素坐标，则待求象素灰度的值 f(i+u, j+v) 如下图所示：




如果(i+u, j+v)落在 A 区，即 u<0.5, v<0.5，则将左上角象素的灰度值赋给待求象素，同理，落在 B 区则赋予右上角的象素灰度值，落在 C 区则赋予左下角象素的灰度值，落在 D 区则赋予右下角象素的灰度值。

**例**：3X3 的 256 级灰度图，也就是高为 3 个象素，宽也是 3 个象素的图像，每个象素的取值可以是 0－255，代表该像素的亮度，255 代表最亮，也就是白色，0 代表最暗，即黑色。假如图像的象素矩阵如下图所示（这个原始图把它叫做源图，Source）：

234 38 22
67 44 12
89 65 63

这个矩阵中，元素坐标(x,y)是这样确定的，x 从左到右，从 0 开始，y 从上到下，也是从零开始，这是图象处理中最常用的坐标系，就是这样一个坐标：

```
  ---------------------＞X
  |
  |
  |
  |
  |
∨Y
```

如果想把这副图放大为 4X4 大小的图像

**第一步**：先把 4X4 的矩阵先画出来，如下所示，矩阵的每个像素都是未知数，等待着我们去填充（目标图,Destination）：

| | | | |
|---|---|---|---|
| ? | ? | ? | ? |
| ? | ? | ? | ? |
| ? | ? | ? | ? |
| ? | ? | ? | ? |

**第二步**：往这个空的矩阵里面填值，先填写目标图最左上角的象素，坐标为（0，0），该坐标对应源图中的坐标可以由如下公式得出：

srcX=dstX* (srcWidth/dstWidth) , srcY = dstY * (srcHeight/dstHeight)

套用公式，找到对应的原图的坐标了(0*(3/4),0*(3/4))=>(0*0.75,0*0.75)=>(0,0)

,找到了源图的对应坐标,就可以把源图中坐标为(0,0)处的 234 象素值填进去目标图的(0,0)这个位置了。

接下来,如法炮制,寻找目标图中坐标为(1,0)的象素对应源图中的坐标,套用公式:

(1*0.75,0*0.75)=>(0.75,0)

**象素的坐标都是整数,从来没有小数坐标**。这时候采用四舍五入的方法把非整数坐标转换成整数，即：

(1*0.75,0*0.75)=>(0.75,0)=>(1,0)

那么就可以再填一个象素到目标矩阵中了，同样是把源图中坐标为(1,0)处的像素值 38 填入目标图中的坐标。

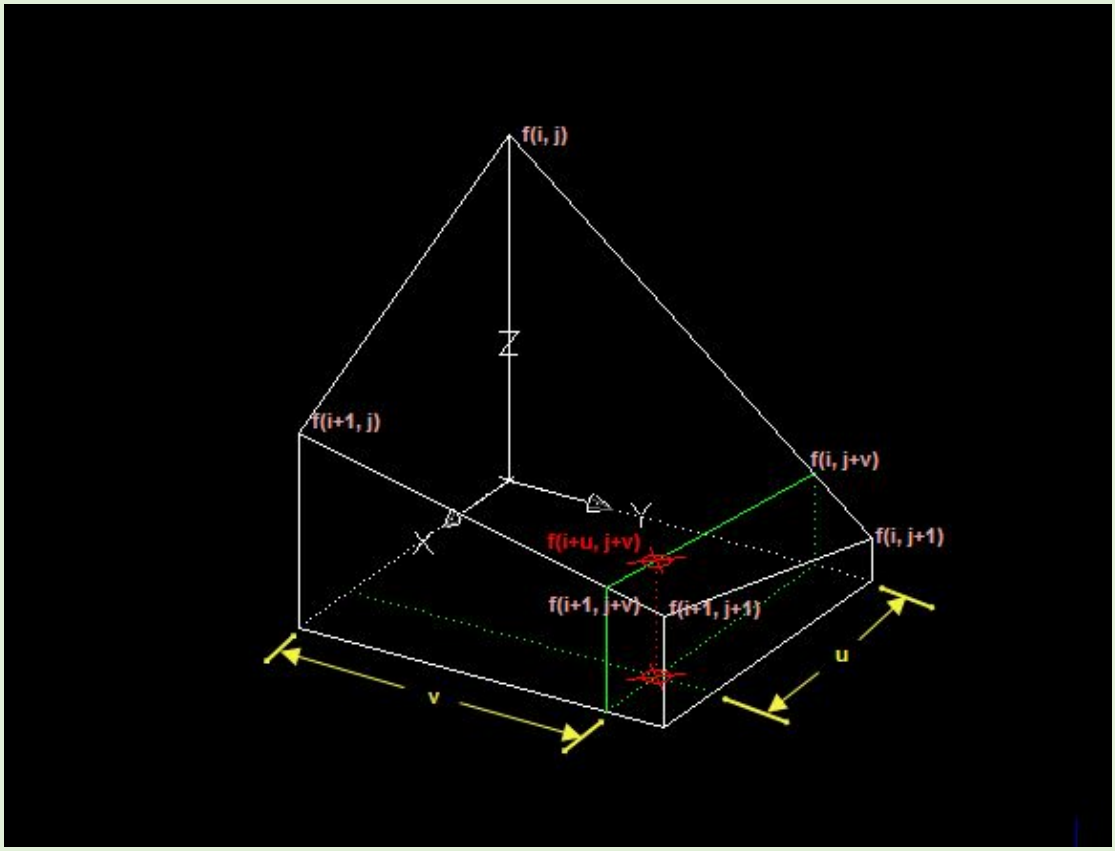
依次填完每个象素，像素矩阵如下所示：

| | | | |
|---|---|---|---|
| 234 | 38 | 22 | 22 |
| 67 | 44 | 12 | 12 |
| 89 | 65 | 63 | 63 |
| 89 | 65 | 63 | 63 |

这是一种最基本、最简单的图像缩放算法，效果也是最不好的，放大后的图像有很严重的马赛克，缩小后的图像有很严重的失真；

# 二、双线性内插法

双线性内插法是利用待求象素四个邻象素的灰度在两个方向上作线性内插，如下图所示：




对于 (i, j+v)，f(i, j) 到 f(i, j+1) 的灰度变化为线性关系，则有：

$$f(i, j+v) = [f(i, j+1) - f(i, j)] * v + f(i, j)$$

同理对于 (i+1, j+v) 则有：

$$f(i+1, j+v) = [f(i+1, j+1) - f(i+1, j)] * v + f(i+1, j)$$

从 f(i, j+v) 到 f(i+1, j+v) 的灰度变化也为线性关系，由此可推导出待求象素灰度的计算式如下：

$$f(i+u, j+v) = (1-u) * (1-v) * f(i, j) + (1-u) * v * f(i, j+1) + u * (1-v) * f(i+1, j) + u * v * f(i+1, j+1)$$

双线性内插值算法描述如下:

对于一个目的像素，设置坐标通过反向变换得到的浮点坐标为(i+u,j+v) (其中 i、j 均为浮点坐标的整数部分，u、v 为浮点坐标的小数部分，是取值[0,1)区间的浮点数)，则这个像素得值 f(i+u,j+v) 可由原图像中坐标为 (i,j)、(i+1,j)、(i,j+1)、(i+1,j+1)所对应的周围四个像素的值决定，即：

f(i+u,j+v) = (1-u)(1-v)f(i,j) + (1-u)vf(i,j+1) + u(1-v)f(i+1,j) + uvf(i+1,j+1)
公式 1

其中 f(i,j)表示源图像(i,j)处的的像素值，以此类推。

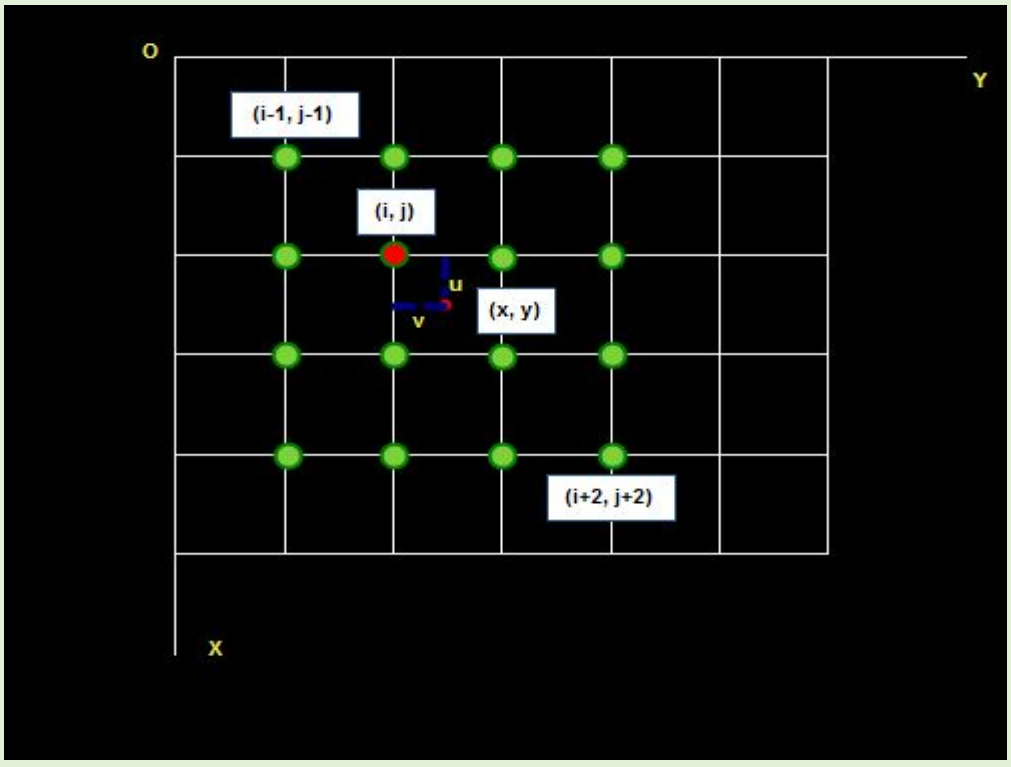
比如，象刚才的例子，现在假如目标图的象素坐标为（1，1），那么反推得到的对应于源图的坐标是（0.75 , 0.75），这其实只是一个概念上的虚拟象素,实际在源图中并不存在这样一个象素,那么目标图的象素（1，1）的取值不能够由这个虚拟象素来决定，而只能由源图的这四个象素共同决定：（0，0）（0，1）（1，0）（1，1），而由于（0.75,0.75）离（1，1）要更近一些，那么（1,1）所起的决定作用更大一些，这从公式 1 中的系数 uv=0.75×0.75 就可以体现出来，而（0.75,0.75）离（0，0）最远，所以（0，0）所起的决定作用就要小一些，公式中系数为(1-u)(1-v)=0.25×0.25 也体现出了这一特点。

原理参考 link:http://blog.csdn.net/andrew659/article/details/4818988

双线性内插法的计算比最邻近点法复杂，计算量较大，但没有灰度不连续的缺点，结果基本令人满意。它具有低通滤波性质，使高频分量受损，图像轮廓可能会有一点模糊。

# 三、双三次内插法

该方法利用三次多项式 S(x)求逼近理论上最佳插值函数 sin(x)/x,
待求像素(x, y)的灰度值由其周围 16 个灰度值加权内插得到，如下图：



待求像素的灰度计算式如下：

f(x, y) = f(i+u, j+v) = ABC

其中:

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

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

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

三次曲线插值方法计算量较大，但插值后的图像效果最好。