图像差值算法

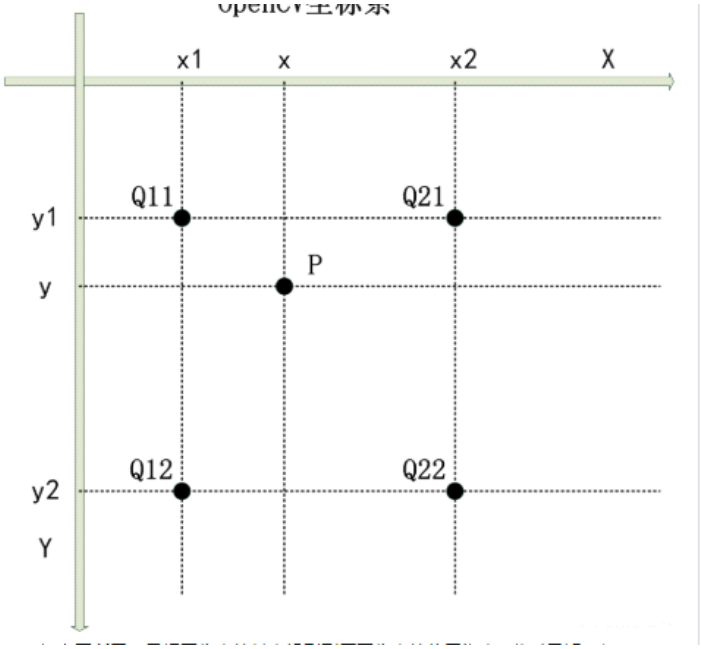
2020年4月20日 21:18

一、简介

在进行图像处理中，经常遇到需要将图片进行缩放的操作，需要对图片像素点修改。在扩大时，需要用一些特定的像素值来补充空缺的点，此时就需要用到图像插值算法。 常见的差值算法有最邻近插值法（Nearest-neighbor），双线性插值法（Bilinear），双三次插值（bicubic）

二、最邻近插值法

最近邻插值法，就是使用最近的点位进行插值补充那些在图像扩大变换时的空点位。



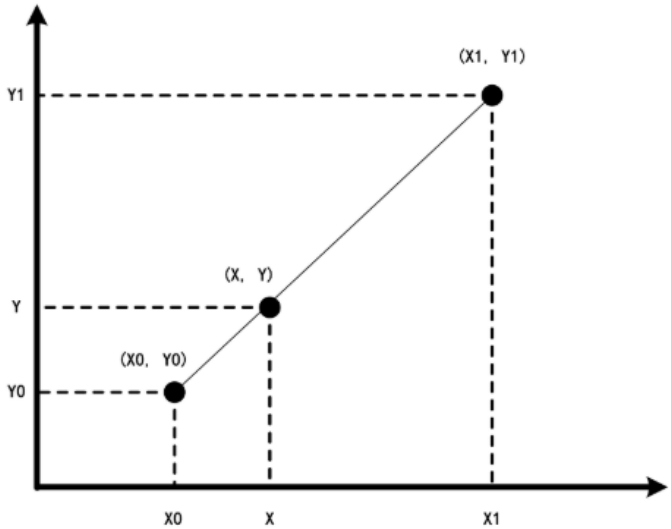
如图所示，点P为在图像放大过程中产生的空像素点，需要使用旁边的点位进行填补。此时p点距离Q11点的距离最近，直接使用Q11的值作为p点的值。

最近邻公式：

$$X_{src} = X_{dst} * (\mathrm{Width}_{src} / \mathrm{Width}_{dst})$$
$$Y_{src} = Y_{dst} * (\mathrm{Height}_{src} / \mathrm{Height}_{dst})$$

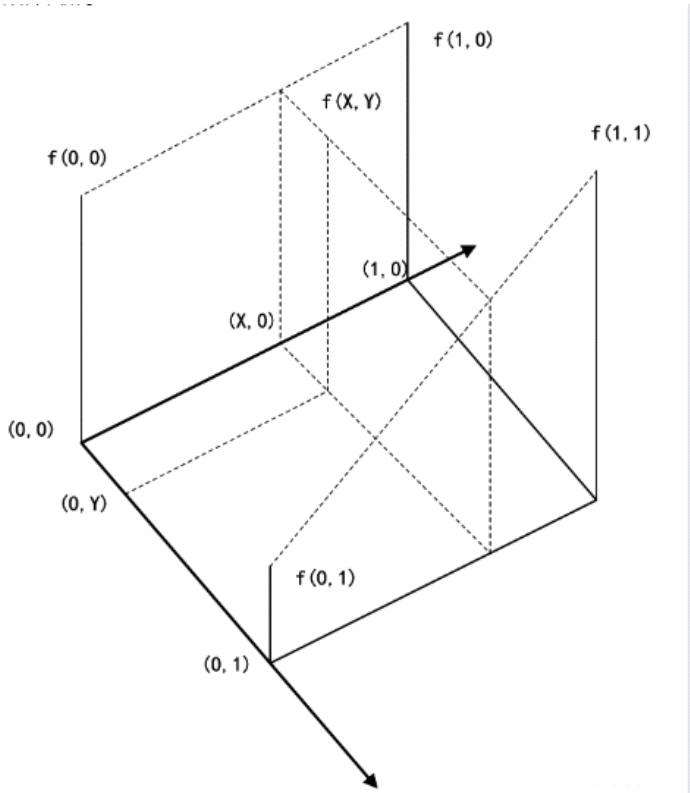
三、双线性插值

在opencv中默认的插值方法为双线性插值，先用下图解释下线性插值：



在二维平面内，如果想要求两点内一点的位置，可以在两个点的连接线上，求出线性方程，即可得到线上所有点位的值。设想一下，这个坐标系为就是一张图像，坐标轴的交点为图像中的像素点。在已知两点的情况下，可以求线上的交点的坐标值，用这个

值来作为图像的像素值。



双线性插值就如上图所示，在两个方向上做三次线性插值。

x方向上的插值为：

$$f(x, y_1) \approx \frac{x_2 - x}{x_2 - x_1} f(Q_{11}) + \frac{x - x_1}{x_2 - x_1} f(Q_{21}),$$
$$f(x, y_2) \approx \frac{x_2 - x}{x_2 - x_1} f(Q_{12}) + \frac{x - x_1}{x_2 - x_1} f(Q_{22}).$$

y方向上的插值：

$$\begin{aligned} f(x, y) &\approx \frac{y_2 - y}{y_2 - y_1} f(x, y_1) + \frac{y - y_1}{y_2 - y_1} f(x, y_2) \\ &= \frac{y_2 - y}{y_2 - y_1} \left(\frac{x_2 - x}{x_2 - x_1} f(Q_{11}) + \frac{x - x_1}{x_2 - x_1} f(Q_{21}) \right) + \frac{y - y_1}{y_2 - y_1} \left(\frac{x_2 - x}{x_2 - x_1} f(Q_{12}) + \frac{x - x_1}{x_2 - x_1} f(Q_{22}) \right) \\ &= \frac{1}{(x_2 - x_1)(y_2 - y_1)} \big(f(Q_{11})(x_2 - x)(y_2 - y) + f(Q_{21})(x - x_1)(y_2 - y) + f(Q_{12})(x_2 - x)(y - y_1) + f(Q_{22})(x - x_1)(y - y_1) \big) \\ &= \frac{1}{(x_2 - x_1)(y_2 - y_1)} \begin{bmatrix} x_2 - x & x - x_1 \end{bmatrix} \begin{bmatrix} f(Q_{11}) & f(Q_{12}) \\ f(Q_{21}) & f(Q_{22}) \end{bmatrix} \begin{bmatrix} y_2 - y \\ y - y_1 \end{bmatrix}. \end{aligned}$$

双线性插值在单个方向上为线性的，对于整幅图来说就是非线性的。

四、双三次插值

双三次插值又叫双立方插值，在这种方法中，函数f在点 (x,y) 的值可以通过矩形网格中最近的十六个采样点的加权平均得到，在这里需要使用两个多项式插值三次函数，每个方向使用一个。

$$\sum_{i=0}^{3} \sum_{j=0}^{3} a_{ij} x^i y^j$$

与双线性插值相比，能够保留更多的图像信息，是扩大后的图像更加细腻，但是由于运算比较复杂，消耗时间比双线性插值要多。

五、代码示例：

opencv函数

```
cv2.resize(InputArray src, OutputArray dst, Size, fx, fy, interpolation)
```

InputArray src	输入图片
OutputArray dst	输出图片
Size	输出图片尺寸
fx, fy	沿x轴，y轴的缩放系数
interpolation	插入方式

Interpolation 常用插值方法：

INTER_NEAREST	最近邻插值
INTER_LINEAR	双线性插值（默认设置）
INTER_CUBIC	4x4像素邻域的双三次插值

1. 最近邻差值：

```
# -*- coding:utf-8 -*-
import os
import cv2

if __name__ == '__main__':
    img = cv2.imread("img.jpg")
    h,w,n = img.shape
    print(h,w,n)   #252 250 3
    res = cv2.resize(img,(500,500),interpolation=cv2.INTER_NEAREST)
    cv2.imshow("img",img)
    cv2.waitKey(0)
    cv2.destroyAllWindows()
```

原图 | 效果图

2. 双线性插值

```
# -*- coding:utf-8 -*-
import os
import cv2

if __name__ == '__main__':
    img = cv2.imread("img.jpg")
```

```
    h,w,n = img.shape
    print(h,w,n)   #252 250 3
    res = cv2.resize(img,(500,500),interpolation=cv2.INTER_NEAREST)
    cv2.imshow("img",img)
    cv2.waitKey(0)
    cv2.destroyAllWindows()
```

原图 | 效果图

双三次插值

```
# -*- coding:utf-8 -*-
import os
import cv2

if __name__ == '__main__':
    img = cv2.imread("img.jpg")
    h,w,n = img.shape
    print(h,w,n)   #252 250 3
    res = cv2.resize(img,(500,500),interpolation=cv2.INTER_CUBIC)
    cv2.imshow("img",res)
    cv2.waitKey(0)
    cv2.destroyAllWindows()
```

原图 | 效果图