第二章作业

2025 年 3 月 8 日

1 插值方法及原理

1.1 最近邻插值

最近邻插值是一种零阶插值方法，其基本思想是在目标图像中每个像素对应的原始图像的连续坐标处，直接选择离该坐标最近的整数位置的像素值作为插值结果。设目标图像中某点对应原始图像的连续坐标为 (x', y')，通过将其分别四舍五入到最近的整数坐标 $\mathrm{round}(x')$ 和 $\mathrm{round}(y')$，得到目标像素的值，即

$$I'(x, y) = I\big(\mathrm{round}(x'), \mathrm{round}(y')\big).$$

这种方法由于只利用了单个像素的信息，因而计算量极小，适用于对实时性要求高的应用。然而，由于没有考虑周围其他像素的信息，导致在图像旋转等几何变换中容易出现块状效应、锯齿和马赛克现象，无法很好地保持图像的连续性和细节。

1.2 双线性插值

双线性插值法（Bilinear Interpolation）是一种常用的二维图像插值方法，其基本思想是在水平方向和垂直方向上分别进行线性插值，以估计待插值点的像素值。该方法假设图像在局部区域内变化平缓，可以近似看作一个平面，因此在图像连续性较好的情况下能够取得较为理想的效果。

具体来说，设待插值点在原始图像中的连续坐标为 (x', y')。为了进行插值，首先需要确定该点周围的四个整数坐标点，记为 (x_1, y_1)、(x_2, y_1)、(x_1, y_2) 以及 (x_2, y_2)，其中

$$x_1 = \lfloor x' \rfloor, \quad x_2 = \lceil x' \rceil, \quad y_1 = \lfloor y' \rfloor, \quad y_2 = \lceil y' \rceil.$$

然后定义水平和垂直方向上的偏移量为

$$dx = x' - x_1, \quad dy = y' - y_1.$$

插值过程首先在水平方向上进行，对位于 y_1 和 y_2 两条水平线上分别计算插值结果：

$$I_{y_1}(x') = I(x_1, y_1) + (x' - x_1)\big[I(x_2, y_1) - I(x_1, y_1)\big],$$

$$I_{y_2}(x') = I(x_1, y_2) + (x' - x_1)\big[I(x_2, y_2) - I(x_1, y_2)\big].$$

随后，在垂直方向上对这两个中间结果进行线性插值，得到最终的像素值：

$$I'(x, y) = I_{y_1}(x') + (y' - y_1)\big[I_{y_2}(x') - I_{y_1}(x')\big].$$

将上述步骤合并，可以写成一个统一的表达式：

$$I'(x,y) = (1-dx)(1-dy)I(x_1,y_1) + dx(1-dy)I(x_2,y_1) + (1-dx)dy\,I(x_1,y_2) + dx\,dy\,I(x_2,y_2).$$

这种方法既兼顾了计算效率，也能在一定程度上保持图像的平滑性，因此被广泛应用于图像旋转、缩放等几何变换中。不过，由于它仅利用了邻近四个像素的信息，对于存在较大灰度变化或高频细节的图像，可能会引入一定程度的模糊现象。

1.3 双三次插值

双三次插值是一种高阶插值方法，它考虑了目标像素周围 16 个邻近像素的信息，通过三次函数进行加权，从而在平滑性和细节保留上均表现较好。该方法基于三次卷积函数，常用的权重函数 $w(t)$ 通常定义为

$$w(t) = \begin{cases} (a+2)|t|^3 - (a+3)|t|^2 + 1, & \text{if } |t| \le 1, \\ a|t|^3 - 5a|t|^2 + 8a|t| - 4a, & \text{if } 1 < |t| < 2, \\ 0, & \text{if } |t| \ge 2, \end{cases}$$

其中参数 a 常取值为 -0.5 或 -0.75，用于控制插值的锐度与平滑性。在具体计算时，设目标像素对应的原始图像连续坐标为 (x', y')，并定义 $x_1 = \lfloor x' \rfloor$、$y_1 = \lfloor y' \rfloor$ 以及 $dx = x' - x_1$、$dy = y' - y_1$。则目标像素值通过下面的公式计算：

$$I'(x,y) = \sum_{i=-1}^{2} \sum_{j=-1}^{2} w(i-dx)\,w(j-dy)\,I(x_1+i, y_1+j).$$

该方法不仅能够提供比双线性插值更为平滑的结果，还能更好地保留图像边缘和细节，特别适用于存在丰富高频信息的图像。但由于涉及更大范围的像素和更复杂的计算，其运算量明显高于前两种方法，因此在实时应用中可能受到性能限制。

2 MATLAB 代码实现

```matlab
clc
clear
close all

%% 设置参数
img = imread('image.jpg');
img = rescale(img);
theta = -30;

%% 最近邻插值
tic
near = imrotate(img, theta, "nearest", "crop");
toc
figure
subplot(1,2,1)
imshow(img)
```

```
title("原始图像")
subplot(1,2,2)
imshow(near)
title("最近邻插值")

%% 双线性插值
tic
bil = imrotate(img, theta, "bilinear", "crop");
toc
figure
subplot(1,2,1)
imshow(img)
title("原始图像")
subplot(1,2,2)
imshow(bil)
title("双线性插值")

%% 双三次插值
tic
cub = imrotate(img, theta, "bicubic", "crop");
toc
figure
subplot(1,2,1)
imshow(img)
title("原始图像")
subplot(1,2,2)
imshow(cub)
title("双三次插值")

%% 对比
figure
subplot(2,2,1)
imshow(img)
title("原始图像")
subplot(2,2,2)
imshow(near)
title("最近邻插值")
subplot(2,2,3)
imshow(bil)
title("双线性插值")
subplot(2,2,4)
imshow(cub)
title("双三次插值")
```

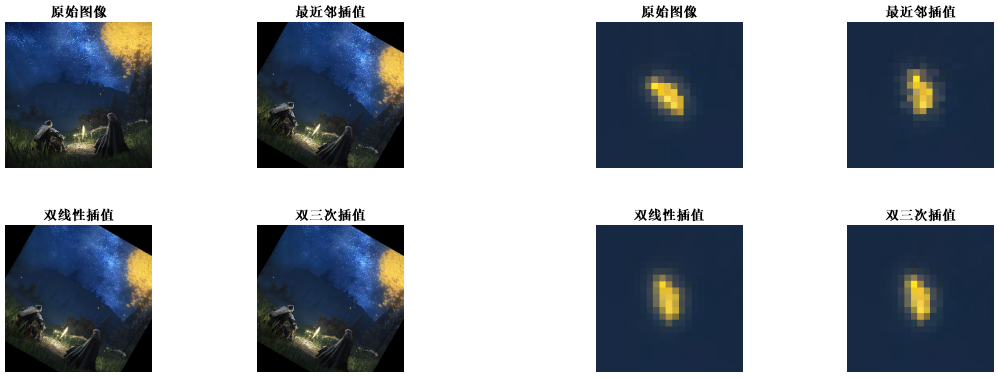
3 实验结果与分析

实验结果显示，对于一张 640×640 的图像，最近邻插值计算速度最快，用时 0.003828s；双线性插值次之，用时 0.006109s；双三次插值最慢，用时 0.009154s。

表 1: 插值算法处理耗时对比

算法类型	计算时间（秒）	相对耗时比例
最近邻插值	0.003828	1.00×
双线性插值	0.006109	1.60×
双三次插值	0.009154	2.39×

将原图像逆时针旋转 30 度后，分别使用最近邻插值、双线性插值和双三次插值三种方法进行插值处理，得到的结果如下图所示。从全图来看，最近邻插值算法呈现明显的锯齿效应，将同一区域放大后观



(a) 全图对比 (b) 局部放大

图 1: 插值算法对比：（a）原始图像与三种插值结果的全图对比；（b）局部放大对比

察，最近邻插值计算速度快但图像质量较差，容易产生锯齿。双线性插值能够显著改善图像的平滑性，但会导致一定程度的模糊。双三次插值在边缘保留和图像平滑性方面表现最佳，但计算复杂度较高。