目录

1 实验目的

本次实验通过利用 c 程序实现数字信号处理的相关功能，主要培养学生以下能力：

1、巩固学生对信号处理原理知识的理解，提高实际编程和处理数据的综合能力，初步培养在解决信号处理实际应用问题中所应具备的基本素质和要求。

2、培养研发能力，通过设计实现不同的信号处理问题，初步掌握在给定条件和功能的情况下，实现合理设计算法结构的能力。

3、提高文献整理和资料查询的能力。通过课下对相关图像知识的学习和理解，培养快速解决实际问题的能力，并在文献整理的过程中学会科技文献的写作，提高语言表达能力。

2 实验内容与过程

任意打开一个图像编辑器，一般都会有对图像进行放大、缩小、旋转等操作，这类操作改变了原图中各个区域的空间关系。现只对图像缩放进行研究。

首先，完成一个图像的缩放，需要对空间坐标进行变换。而它是用来描述目标图像像素从最开始的位置迭代到最终的位置。通俗来说，这个目的就是索引对应原图的位置.

值得注意的是：这里的放大和缩小不是我们生活中物体的放大和缩小。而是二维空间坐标(x,y)以$(0,0)$中心在水平方向和垂直方向缩放s_x倍和s_y倍.缩放也可以用放射矩阵来表示,即

$$\begin{pmatrix}\tilde{x}\\ \tilde{y}\\ 1\end{pmatrix}=\begin{pmatrix}s_x & 0 & 0\\ 0 & s_y & 0\\ 0 & 0 & 1\end{pmatrix}\begin{pmatrix}x\\ y\\ 1\end{pmatrix}.$$

其次，在知道原图的位置之后，我们还需要一个插值算法来完成图像每个像素的输出。这里分别介绍最近邻插值、双线性插值、双三次插值、Hermite 样条插值原理对目标像素输出的研究效果。

2.1 最近邻插值

2.1.1 插值原理

从计算量的角度来说，最近邻插值是最简单的插值。在图像缩放的这种算法中，每一个插值输出像素的值就是在输入图像中与其最临近的抽样点的值。数学

表达式为：

$$f\left(round\left(x\times s_y\right),round\left(y\times s_y\right)\right)=f\left(x\times s_y,y\times s_y\right).$$

其中，$f\left(x,y\right)$为原图像函数，$\left(x\times s_y,y\times s_y\right)$为原始图像坐标$\left(x,y\right)$以$(0,0)$中心在水平方向和垂直方向缩放了$s_x$倍与$s_y$倍，$round(\ )$为四舍五入符号。

举个例子：对于原始像素$(1,1)$，我们在水平方向和垂直方向都放大了 1.5 倍，即得到目标图像坐标为$(1.5,1.5)$，然后对其进行四舍五入，得到$(2,2)$，故对应目标图像的像素为$f\left(2,2\right).$

2.1.2 插值效果

基于 vs，我们编写代码，如实验代码 1 所示。由于最近邻插值对图像放大与缩小的影响几乎一致，我们主要讨论图像放大的情况。

- **图像放大：**我们设置图像的长和宽，同时放大 3 倍，如图 1 所示；

然后对局部区域进行放大，如图 2 所示。

由图 1 可以知道，我们对放大后的图像与原始图像进行统一缩放。放眼看去，两张图片似乎没有什么太大的区别。但是根据最近邻插值的公式

$$f\left(round\left(x\times s_y\right),round\left(y\times s_y\right)\right)=f\left(x\times s_y,y\times s_y\right).$$

它仅使用离待测采样点最近的像素的灰度值作为该采样点的灰度值，而没考虑其他相邻像素点的影响，因而重新采样后灰度值有明显的不连续性，图像质量损失较大，在图 2 中，图像明显会产生明显的马赛克和锯齿现象。

图 1 原始图像与 3X 图像

图 2 原始图像与 3X 图像局部放大图像

- **图片缩小**：图像缩小也会产生马赛克和锯齿现象，如图 3 所示。

图 3 原始图像与 0.5X 图像

2.2 双线性插值

由于最近邻插值只是利用原始图像区域中最近的一个像素的值当作输出的像素的值。为了充分利用周围的邻域像素的值，我们引进双线性插值.

2.2.1 插值原理

与最近邻插值不同的是，双线性插值也是基于相邻的四个像素值，但是对其最近的像素进行加权得到目标图像的像素值。而双线性插值如图 1 所示：

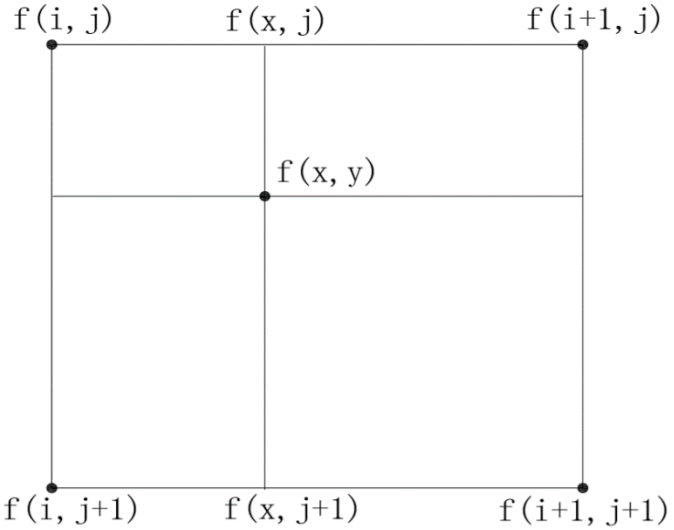

图 4 双线性插值示意图

双线性插值就是在水平方向进行两次线性插值，然后在垂直方向进行一次插值，具体地：

在 x 方向插值，根据 *lagrange* 插值公式得

$$f(x,j)=(i+1-x)\cdot f(i,j)+(x-i)f(i+1,j),$$
$$f(x,j+1)=(i+1-x)\cdot f(i,j+1)+(x-i)f(i+1,j+1).$$

在 y 方向插值，根据 *lagrange* 插值公式得

$$f(x,y)=(j+1-y)(i+1-x)f(i,j)+(j+1-y)f(i+1,j)+$$
$$(y-j)(i+1-x)f(i,j+1)+(y-j)(x-i)f(i+1,j+1).$$

通过这个式子，我们可以计算像素经过仿射变换后的像素了.

2.2.2 插值效果

基于 vs，我们编写代码，如实验代码 2 所示。

- **图像放大：**我们设置图像的长和宽，同时放大 3 倍，如图 5 所示；然后对局部区域进行放大，如图 6 所示。

图 5 原始图像与 3X 图像

图 6 原始图像与 3X 图像局部放大图像

- **图片缩小**：图像缩小也会产生模糊现象，如图 7 所示。

图 7 原始图像与 0.5X 图像

由图 5、图 6 和图 7 可见，双线性插值法效果比较好，但是计算量比较，程序运行时间也稍长，但缩放后图像质量高，基本克服了最近邻插值灰度值不连续的特点，因为它考虑了待测采样点周围四个直接邻点对该采样点的相关性影响。用此方法缩放后的输出图像与输入图像相比，在梯度变化较大的地方，边缘相对平滑.

2.3 双三次插值

为了更加精确的计算输出的像素值，我们引进双三次插值，即利用周围 16 个像素值，对当前位置输出的像素的值进行近似计算。

2.3.1 插值原理

目标图像中任一点的像素(x,y)的像素值，其由其周围 16 个点的颜色值加权得到。如图所示：

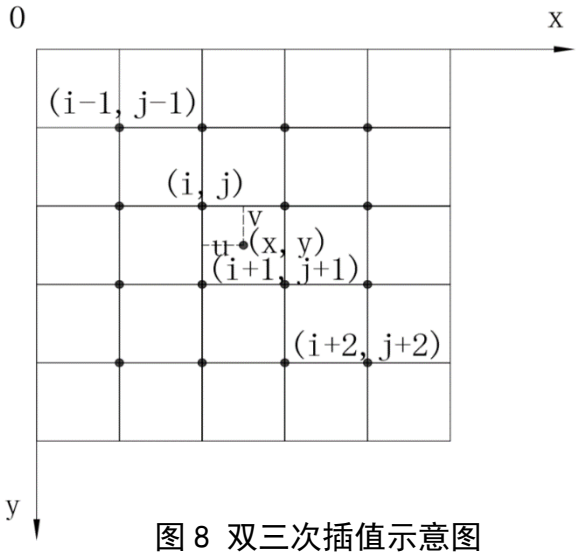

图 8 双三次插值示意图

本文中，采用的双三次插值的数学公式为：

$$f(x,y)=\sum_{i=0}^{3}\sum_{j=0}^{3}f\left(x_i,y_j\right)W\left(x-x_i\right)W\left(y-y_j\right).$$

其中，$W(x)$采用 WIKI-Bicubic interpolation[1]中给出的权重函数，即

$$W(x)=\begin{cases}(a+2)|x|^3-(a+3)|x|^2+1 & \text{for } |x|\le 1,\\ a|x|^3-5a|x|^2+8a|x|-4a & \text{for } 1<|x|<2,\\ 0 & \text{otherwise.}\end{cases}$$

当取值为-1 时，权重公式为：

$$W(x)=\begin{cases}|x|^3-2|x|^2+1 & \text{for } |x|\le 1,\\ -|x|^3+5|x|^2-8|x|+4 & \text{for } 1<|x|<2,\\ 0 & \text{otherwise.}\end{cases}$$

2.3.2 插值效果

基于 vs，我们编写代码，如实验代码 1 所示。

- **图像放大：**我们设置图像的长和宽，同时放大 3 倍，如图 9 所示；然后对局部区域进行放大，如图 10 所示。

图 10 原始图像与 3X 图像

图 11 原始图像与 3X 图像局部放大图像

- **图片缩小：**把图片缩小 0.5 倍，如图 12 所示。

图 12 原始图像与 0.5X 图像

双三次插值比最近邻插值和双线性插值的公式更加复杂的一个插值公式。该算法利用待采样点周围 16 个点的灰度值作三次插值，不仅考虑到 4 个相邻点的灰度影响，而且考虑到灰度值变化率对结果的影响。由图 11 和图 12 可以知道，三次运算可以得到更接近高分辨率图像的放大效果，但也导致了运算量的急剧增加。

2.4 三次 Hermite 样条插值

三次 Hermite 样条插值，即利用周围 16 个像素值，对当前位置输出的像素的值进行近似计算。

2.4.1 插值原理

当取值为-0.5 时，此时对应的是三次 Hermite 样条插值[2].

$$W(x)=\begin{cases}1.5|x|^3-2.5|x|^2+1 & \text{for } |x|\le 1,\\ -0.5|x|^3+2.5|x|^2-4|x|+2 & \text{for } 1<|x|<2,\\ 0 & \text{otherwise.}\end{cases}$$

通过这两个式子，我们可以计算像素经过仿射变换后的像素了.

2.4.2 插值效果

基于 vs，我们编写代码，如实验代码 1 所示。

- **图像放大**：我们设置图像的长和宽，同时放大 3 倍，如图 3 所示；然后对局部区域进行放大，如图 14 所示。

图 13 原始图像与 3X 图像

图 14 原始图像与 3X 图像局部放大图像

- **图片缩小：**把图片缩小 0.5 倍，如图 15 所示。

图 15 原始图像与 0.5X 图像

Hermite 三次插值也跟上述插值一样，是一种更加复杂的插值方式。该算法利用待采样点周围 16 个点的灰度值作三次插值，运算量明显增加，但图像也比较清晰。

3 实验总结

- **图像放大**

图 16 图像局部放大图像（左最近邻、双线性、右双三次）

- 图像缩小

图 17 图像局部放大图像（左最近邻、双线性、右双三次）

最近邻插值法的优点是计算量很小，算法也简单，因此运算速度较快。但它仅使用离待测采样点最近的像素的灰度值作为该采样点的灰度值，而没考虑其他相邻像素点的影响，因而重新采样后灰度值有明显的不连续性，由图 16 可以知道，图像质量损失较大，会产生明显的马赛克和锯齿现象。

双线性插值法效果要好于最近邻插值，只是计算量比较大，但缩放后图像质量高，基本克服了最近邻插值灰度值不连续的特点，因为它考虑了待测采样点周围四个直接邻点对该采样点的相关性影响。但是，此方法仅考虑待测样点周围四个直接邻点灰度值的影响，而未考虑到邻点之间的灰度值变化率的影响，因此具有低通滤波器的性质，从而导致缩放后图像的高频分量受到损失，图像边缘在一定程度上变得较为模糊。用此方法缩放后的输出图像与输入图像相比，仍然存在由于插值函数设计考虑不周而产生的图像质量受损与计算精度不高的问题。

立方卷积插值计算量最大，算法也是最为复杂的。在几何运算中，双线性内插法的平滑作用可能会使图像的细节产生退化，在进行放大处理时，这种影响更为明显。在其他应用中，双线性插值的斜率不连续性会产生不希望的结果。立方卷积插值不仅考虑到周围四个直接相邻像素点灰度值的影响，还考虑到它们灰度值变化率的影响。因此克服了前两种方法的不足之处，能够产生比双线性插值更

为平滑的边缘，计算精度很高，处理后的图像像素损失比较少，效果是最佳的。

总之，在进行图像缩放处理时，应根据实际情况对三种算法做出选择，既要考虑时间方面的可行性，又要对变换后图像质量进行考虑，这样才能达到较为理想的结果。

4 实验探索

图片的无极缩放就是图片可以按任意比例缩放，但是这种缩放不是漫无目的，我们约束图片的缩放是在一定的条件下进行缩放，不是任意比例缩放因为毫无意义。

来自加州大学圣地亚哥分校的英伟达[6]提出一种新颖的 LIIF 用于对自然图像进行连续表达，即用局部隐含函数学习连续图像表示法。作者称 LIIF 表示法在二维离散和连续表示法之间建立了一座桥梁，自然地支持大小不同的图像真值学习任务，并显著优于调整真值大小的方法。

4.1 原理

局部隐函数

作者引入了 LIIF 表达方式。 在 LIIF 表达中，每个连续图像 $I^{(i)}$ 可以表示为 2D 特征形式 $M^{(i)} \in R^{H \times W \times D}$ 。 f_θ 是全图像共享，它是一种 MLP，其形式如下：

$$s = f\left(z, x\right)$$

其中， z 为向量， x 表示连续图像域坐标， $s \in \mathcal{S}$ 表示预测信号(也就是 RGB 值)。基于上述定义，每个向量 z 可以表示为这样的映射过程： $f(z,\cdot): \mathcal{X} \to \mathcal{S}$ ，也就是说 $f(z,\cdot)$ 可以表示为连续图像，它将坐标映射到 RGB.

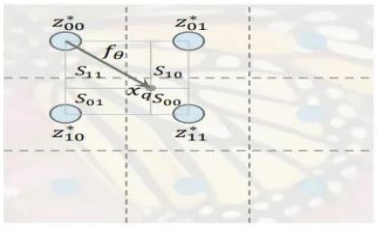

Figure 2: **LIIF representation with local ensemble.** A continuous image is represented by a 2D feature map and a implicit function f_θ shared by all the images. The signal is predicted by ensemble of the local implicit functions, which guarantees smooth transition between different areas.

Cell decoding

作者希望 LIIF 表示能够对图像进行任意分辨率表示，尽管上述方式可以在连续图像域进行插值输出。但是上述方式并非最优的，因为每个像素的面积信息却被忽略了。为解决该问题，作者提出了 cell decoding，见下图

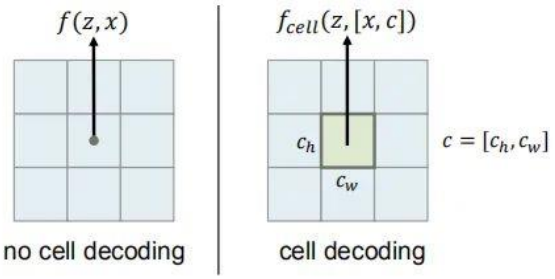

Figure 3: **Cell decoding.** With cell decoding, the implicit function takes the shape of the query pixel as an additional input and predicts the RGB value for the pixel.

连续图像表达学习

有了 LIIF 的定义与构成，数据准备与训练过程见下图。

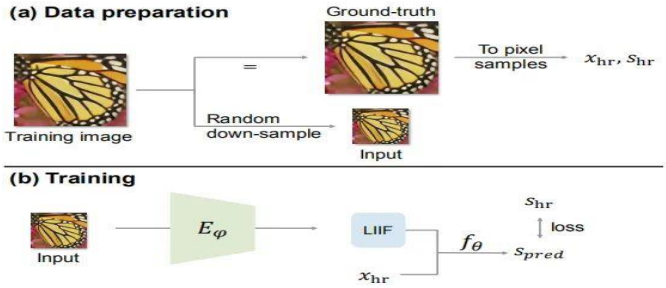

Figure 4: **Learning to generate continuous representation for pixel-based images.** An encoder is jointly trained with the LIIF representation in a self-supervised super-resolution task, in order to encourage the LIIF representation to maintain high fidelity in higher resolution.

在这里，我们需要训练两个东西：Encoder 与 LIIF。前者用于将图像转成特征并送入到 LIIF。Encoder 就简单了，

4.2 实验结果

图 3 图片放大 20x

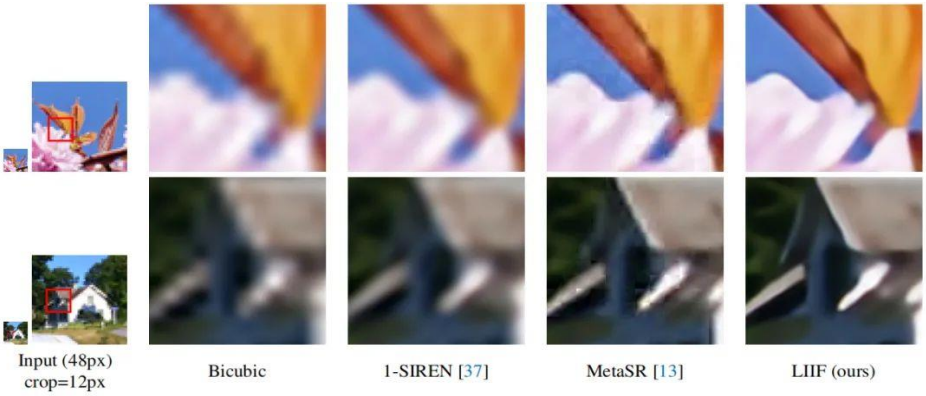

Figure 5: **Qualitative comparison of learning continuous representation.** The input is a 48 × 48 patch from images in DIV2K validation set, a red box indicates the crop area for demonstration (×30). 1-SIREN refers to fitting an independent implicit function for the input image. MetaSR and LIIF are trained for continuous random scales in ×1–×4 and tested for ×30 for evaluating the generalization to arbitrary high precision of the continuous representation.

图 4 30x 超分上的效果对比

虽然所提方法可以达到“无极放大”的目的与效果，但目前该方案的计算量还是比较大的。如何将其做到更轻量可能会是一个不错的尝试点。

5 实验感谢

6 参考文献

[1] https://en.wikipedia.org/wiki/Bicubic_interpolation.

[2] https://en.wikipedia.org/wiki/Cubic_Hermite_spline.

[3] https://en.wikipedia.org/wiki/Bilinear_interpolation

[4] https://en.wikipedia.org/wiki/Nearest-neighbor_interpolation

[5] Learning Continuous Image Representation with Local Implicit Image Function.

7 实验代码