基于二阶牛顿插值的图像自适应缩放设计及实现

王 康 杨瑞祺 杨依忠 解光军
(合肥工业大学电子科学与应用物理学院 安徽 合肥 230601)

摘 要 针对图像插值算法的不足会导致图像模糊及边缘不清晰等不良视觉效果,提出一种新的基于二阶牛顿插值的自适应缩放算法。根据插值点的位置对图像水平方向进行插值计算得到中间图像;根据中间图像计算出插值点的垂直方向、45 度方向和 135 度方向上的二阶差分等参数;判断插值点位置并计算得到目标像素图像。对算法进行硬件实现及验证,结果表明,该算法图像缩放效果更优,图像边缘保护更佳,其峰值信噪比(PSNR)和边缘保护指数(EPI)优于其他算法,且计算复杂度低,利于硬件实现。

关键词 插值算法 二阶牛顿插值 图像缩放 硬件实现

中图分类号 TP391 **文献标志码** A DOI:10.3969/j.issn.1000-386x.2020.09.021

DESIGN AND IMPLEMENTATION OF IMAGE ADAPTIVE SCALING BASED ON SECOND ORDER NEWTON INTERPOLATION

Wang Kang Yang Ruiqi Yang Yizhong Xie Guangjun
(*School of Electronic Science and Applied Physics, Hefei University of Technology, Hefei 230601, Anhui, China*)

Abstract The lack of image interpolation algorithm leads to poor visual effects such as blurred image and unclear edge. This paper proposes a novel adaptive scaling algorithm based on second order Newton interpolation. The intermediate image was obtained by interpolating the horizontal direction of the image according to the position of the interpolation point. The parameters second order difference in the vertical direction, the 45-degree direction and the 135-degree direction of the interpolation point were calculated according to the intermediate image, and the interpolation point position was judged and the image of target pixel was obtained. Finally, the algorithm was implemented and verified in hardware. The experimental results show that the image scaling effect and the edge protection of the image are better. The PSNR and EPI are better than other algorithms. Meanwhile, the complexity of calculation is low, which is good for hardware implementation.

Keywords Interpolation algorithm Second order Newton interpolation Image scaling Hardware implementation

0 引 言

图像缩放技术被广泛地应用于各类消费类电子产品、医学图像处理以及安防监控等领域中[1-3]。图像缩放处理算法的种类较多,根据算法是否对边缘进行处理,可以大致分为线性的图像插值算法和非线性的图像插值算法[4]。线性的插值方法如最邻近插值、双线性插值和双三次插值等经典的插值算法[5-6],在计算目标图像的像素时只通过其位置和邻域内的原始像素数据计算,不考虑边缘处的图像信息变化,缩放后效果会导致边缘模糊不清。葛晨阳等[7]设计了一种图像混合插值缩放的 IP 核,其相较于双三次插值硬件消耗少,但也没有对图像边缘进行处理。对于非线性的插值算法,研究较多的是基于边缘的图像缩放算法。边缘检测算法包括 Sobel[8]、Prewitt[9]和 Canny[10]以及基于形态学的边缘检测算法[11]等。Chen 等[12]提出的基于 Sobel 算子的边缘插值算法和尉成勇等[13]提出的改进边缘方向判定的图像插值算法分别通过 Soble 算子和 Canny 算子对图像边缘进行检测,然后对边缘进行

收稿日期:2019-06-24。国家自然科学基金项目(61674049)。**王康**,硕士生,主研领域:图像处理及实现。**杨瑞祺**,硕士生。**杨依忠**,副教授。**解光军**,教授。

一维的方向插值计算,非边缘采用线性插值的方法,这样会使边缘区域的非边缘像素参与边缘区域计算,导致得到的目标图像仍会有一些模糊。还有一些采用边缘自适应缩放的算法,如 Cha 等[14]提出了一种误差修正的边缘插值方法(EASE),但算法复杂度较高。王效灵等[15]和李自勤等[16]提出边缘自适应图像缩放算法(EASA),采用平均思想找到边缘位置,然后选取边缘方向上的点进行插值计算,该算法需要考虑插值点邻域内多个像素点,且不同方向的插值系数计算复杂。李春龙等[17]提出自适应立方卷积算法,在平缓区域采用双三次插值算法,在边缘区域根据插值点的位置进行两轮插值,虽然效果不错,但运算量大。齐敏等[18]通过数据融合的方式对不同区域的插值点进行插值计算,算法通过多次归一化方法求系数,但缩放效果并不理想,且计算过程繁杂。刘政林等[19]提出基于边缘的实时缩放算法,在边缘检测上考虑欠缺,导致边缘区域的缩放效果不佳。叶森等[20]提出一种基于三次拉格朗日的插值算法,但该算法只考虑了水平方向和竖直方向的边缘检测,忽略了 45 度和 135 度方向上的边缘检测,且计算复杂。此外,詹毅等[21]提出一个变指数变分模型的插值算法,范清兰等[22]提出一种基于 NSCT 的区域自适应图像插值算法,Lee 等[23]提出一种利用泰勒级数近似的边缘引导图像插值方法,以及近些年提出的基于深度学习的图像插值算法[24]等。尽管这些算法的缩放效果不错,但都比较复杂,难以用硬件实现。

基于上述分析,本文提出一种基于二阶牛顿插值的自适应缩放算法。对处于非边缘位置的像素点,根据插值点位置关系选择一组合适的计算方法得到目标像素值。对处于边缘位置的像素点,不是简单采用对边缘进行一维插值的方法,而是对多个方向及其相关度进行权重分配计算得到目标像素值。处理结果相比传统的缩放算法效果更好,且计算复杂度低,利于硬件实现。

1　算法设计

在对图像进行实时缩放处理时,通常对二维图像的水平方向和垂直方向分别进行插值处理。假设把分辨率为 $M \times N$ 原始图像缩放为分辨率为 $U \times V$ 的目标图像,可以先对水平方向进行缩放得到大小为 $U \times N$ 的中间图像,然后再对该中间图像垂直方向进行缩放得到目标图像。对水平方向进行缩放只需考虑一维方向的缩放处理,假设对水平方向进行插值处理,将一行像素序列看成距离为 1 的等距节点,则目标像素点与原始图像在水平方向上的位置关系如图 1 所示。其中:f 是目标像素点在水平方向上映射到原始图像中的位置;f_1、f_2、f_3 和 f_4 为 f 水平邻域内的四个原始像素点;Δx 是 f 和 f_2 之间的距离,取值范围为 $0 \leqslant \Delta x < 1$,且 $t = 1 + \Delta x$。

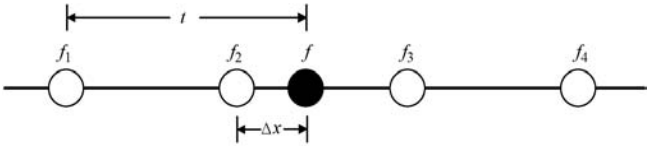


图 1　目标像素水平方向插值示意图

选取二阶牛顿插值算法来计算 f 的像素值时,可以用 f_1、f_2 和 f_3 三个原始像素点作为一组像素或 f_2、f_3 和 f_4 三个原始像素点作为一组来计算目标像素 f 的像素值,两种方式的插值公式分别为:

$$F_1 = f_1 + \Delta f_1 t + \frac{\Delta^2 f_1}{2!} t(t-1) \tag{1}$$

$$F_2 = f_2 + \Delta f_2 (t-1) + \frac{\Delta^2 f_2}{2!}(t-1)(t-2) \tag{2}$$

式中:Δf_1 和 Δf_2 分别为 f_1 和 f_2 的一阶差分;$\Delta^2 f_1$ 和 $\Delta^2 f_2$ 分别为 f_1 和 f_2 的二阶差分。

当 f 处于非边缘位置时,所有计算结果几乎一致,如果插值位置位于边缘区域时,考虑到其位置以及相关性,需要选取其中的一组来计算以得到更逼近边缘位置像素点的像素值。将 $t = 1 + \Delta x$ 代入式(1)和式(2)进行化简可得:

$$F_1 = \frac{1}{2}(\Delta x^2 - \Delta x) f_1 + (-\Delta x^2 + 1) f_2 + \frac{1}{2}(\Delta x^2 + \Delta x) f_3 \tag{3}$$

$$F_2 = \frac{1}{2}(\Delta x^2 - \Delta x) f_4 + (\frac{1}{2}\Delta x^2 - \frac{3}{2}\Delta x^2 + 1) f_2 + (-\Delta x^2 + 2\Delta x) f_3 \tag{4}$$

由插值理论可知,当 f 点处于边缘位置时,如果接近 f_2,则希望计算目标像素的像素值更接近 f_2 的像素值,如果接近 f_3,则希望计算目标像素的像素值更接近 f_3 的像素值,这样可以使插值图像的边缘宽度变窄,不会因边缘宽而导致插值图像的边缘模糊。从式(3)和式(4)可以看出,选取第一组像素时 f_1 所占权重和选取第二组像素时 f_4 所占权重一样,两像素点对计算目标像素的影响也一样。下面分别考虑两组计算公式中 f_2 和 f_3 的权重对目标像素的影响。

对式(3)中 f_2 的系数和式(4)中 f_2 的系数设定关于 Δx 的函数为 $y1$ 和 $h1$,如图 2 所示。对式(3)中 f_3 的系数和式(4)中 f_3 的系数设定关于 Δx 的函数为 $y2$ 和 $h2$,如图 3 所示。

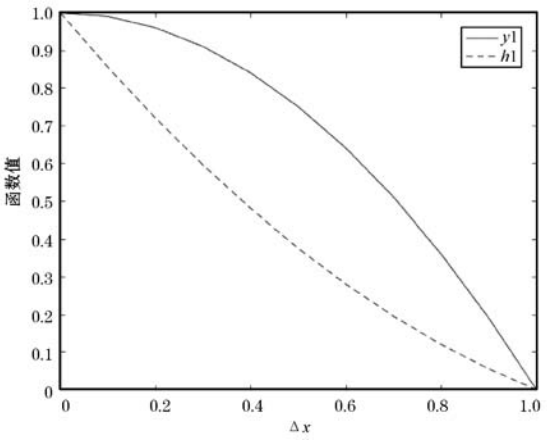


图 2 第一组 f_2 权重函数和第二组 f_2 权重函数

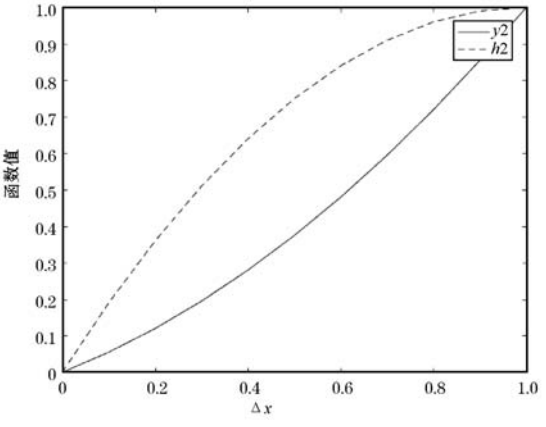


图 3 第一组 f_3 权重函数和第二组 f_3 权重函数

当 Δx 小于 0.5 时,从图 2 中可以看到 $y1$ 的值一直大于 $h1$ 的值,经求导得到 $y1$ 的导数始终小于 $h1$ 的导数,并且当 Δx 越小时,$y1$ 的导数越小,$h1$ 的导数越大。这说明当插值点 f 接近 f_2 点时,第一组公式 f_2 所占权重大于第二组,且第一组 f_2 的权重随着 Δx 增大缓慢减小。从图 3 中可以看出,$h2$ 的值此时一直大于 $y2$ 的值,说明当插值点 f 接近 f_2 时,第一组公式 f_3 所占权重小于第二组,且通过 $h2$ 和 $y2$ 的导数可知第一组 f_3 权重随着 Δx 增加缓慢变大。由此,当 Δx 小于 0.5 时,选取第一组公式计算得到的目标值更接近 f_2 的像素值。同理,当 Δx 大于 0.5 时,选取第二组公式计算得到的目标值更接近 f_3 的像素值。因此,对于水平方向缩放的插值计算公式可改写如下:

$$F=\begin{cases}\dfrac{1}{2}(\Delta x^2-\Delta x)f_1+(-\Delta x^2+1)f_2+\\ \qquad\dfrac{1}{2}(\Delta x^2+\Delta x)f_3 & \Delta x\leqslant 0.5\\ \dfrac{1}{2}(\Delta x^2-\Delta x)f_4+\left(\dfrac{1}{2}\Delta x^2-\dfrac{3}{2}\Delta x^2+1\right)f_2+\\ \qquad(-\Delta x^2+2\Delta x)f_3 & \Delta x>0.5\end{cases}\tag{5}$$

由水平缩放过后得到中间图像,需再对中间图像 45 度、135 度和垂直方向进行缩放处理以得到目标图像。其插值示意图如图 4 所示。

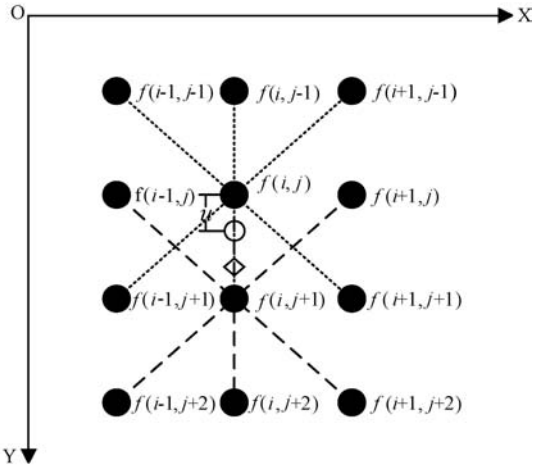


图 4 中间图像插值示意图

图 4 中空心圆形和空心菱形分别表示插值点距离像素点 $f(i,j)$ 小于 0.5 和大于 0.5 两种情况,实心圆点分别是插值点邻域内的各原始像素点。对水平缩放后的中间图像进行垂直方向上的缩放,假设空心圆为目标像素点所在位置,u 的值小于等于 0.5,根据水平方向的插值原理,在 $f(i,j)$ 的 3×3 邻域内分别选取垂直方向、45 度方向和 135 度方向各一组像素,计算其二阶差分分别为 $\Delta^2 f_1$、$\Delta^2 f_2$ 和 $\Delta^2 f_3$。将阈值 T 分别与 $\Delta^2 f_1$、$\Delta^2 f_2$ 和 $\Delta^2 f_3$ 的绝对值进行比较,如果三者中有小于 T 值的,则说明在该方向上是像素值变化较小的平缓区域。依次按照垂直方向、45 度方向和 135 度方向进行判断是否小于 T,直到首次符合条件时,就利用该方向上三个像素点进行二阶牛顿插值计算得到目标像素值。对于三者都大于 T 时,则说明该目标像素在三个方向都不处于平缓区域,需利用三个方向的像素点进行计算得到目标像素值。方法如下:

考虑到二阶差分越小的方向,其相关性越大,所占权重应该越高,对 $\Delta^2 f_1$、$\Delta^2 f_2$ 和 $\Delta^2 f_3$ 进行归一化处理,得到三个方向的权重系数,公式如下:

$$k_1=1-\frac{|\Delta^2 f_1|}{|\Delta^2 f_1|+|\Delta^2 f_2|+|\Delta^2 f_3|}\tag{6}$$

$$k_2=1-\frac{|\Delta^2 f_2|}{|\Delta^2 f_1|+|\Delta^2 f_2|+|\Delta^2 f_3|}\tag{7}$$

$$k_3=1-\frac{|\Delta^2 f_3|}{|\Delta^2 f_1|+|\Delta^2 f_2|+|\Delta^2 f_3|}\tag{8}$$

同时,根据式(5)计算三个方向的二阶牛顿插值结果,为简化计算,对于 45 度和 135 度方向上 Δx 的值直接选用垂直方向上 Δx 的值,即图 4 中 u 的值,计算得到三个方向的结果分别为 F_1、F_2 和 F_3,与各自方向上的权重系数相乘后再求和,因为三个方向权重系数之和为 2,所以最后需乘以 1/2 得到目标像素值,计算

公式如下:

$$N = \frac{1}{2}\left[\left(1 - \frac{|\Delta^2 f_1|}{|\Delta^2 f_1| + |\Delta^2 f_2| + |\Delta^2 f_3|}\right)F_1 + \left(1 - \frac{|\Delta^2 f_2|}{|\Delta^2 f_1| + |\Delta^2 f_2| + |\Delta^2 f_3|}\right)F_2 + \left(1 - \frac{|\Delta^2 f_3|}{|\Delta^2 f_1| + |\Delta^2 f_2| + |\Delta^2 f_3|}\right)F_3\right] \quad (9)$$

同理,假设目标像素位置处于图 4 中空心菱形的位置,即 u 大于 0.5,则选择 $f(i,j+1)$ 像素点 3×3 邻域的像素点,采用同样的方法计算目标像素点的像素值。

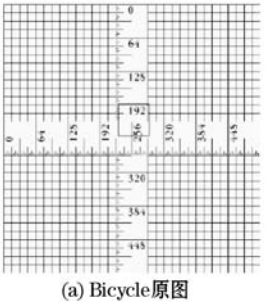
考虑到边缘位置的判断,通过设定一个阈值 T 作为参数。参数选取的大小会导致缩放效果不同,通过实验,选取不同的参数 T 对多组不同图像进行先放大两倍再缩小到原图像大小,然后求出不同阈值下缩放过后的峰值信噪比(Peak Signal-to-Noise Ratio, PSNR)和边缘保护指数(Edge Protection Index, EPI)。通过对比实际缩放后视觉效果,以及对 PSNR 和 EPI 比较分析,本文算法选取阈值 T 为 50。

2　实验结果分析

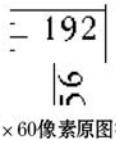
实验选取了 2 幅大小为 256×256 像素的灰度图像 House 和 Baboom,3 幅大小为 512×512 像素的灰度图像 Items、Ruler 和 Bicycle 作为原始图像。使用双线性插值算法[5]、双三次插值算法[6]、数据融合算法[18]、边缘插值算法[19]及本文算法对图像进行缩放处理。

2.1　缩放结果图对比

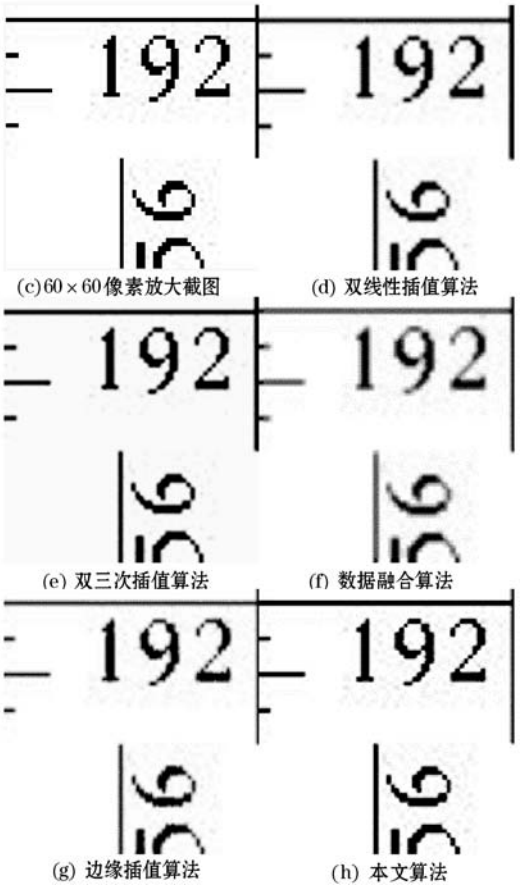
为了直观地对比不同算法处理的效果,以灰度图像 Ruler 的实验结果为例。截取图像处于边缘且大小为 60×60 像素的区域图像,将原图中 60×60 像素大小的原图截取放大,如图 5(c)所示,放大后可以清楚地看到原图像中区域的细节,使得对比效果更加直观,再分别用双线性插值算法、双三次插值算法、数据融合算法、边缘插值算法和本文算法对其放大后得到大小 160×160 像素的图像。



(a) Bicycle原图



(b) 60×60像素原图截取



(c) 60×60像素放大截图　(d) 双线性插值算法　(e) 双三次插值算法　(f) 数据融合算法　(g) 边缘插值算法　(h) 本文算法

图 5　Ruler 放大后不同算法结果

由图 5 可知,双线插值算法缩放后由于高频分量丢失而导致图片整体变得模糊,双三次插值的效果比双线性插值要好些,但由于两种算法都未考虑到对图像边缘进行处理,因此边缘处理效果都不好。数据融合算法同样由于高频分量丢失,导致放大后图像模糊。边缘插值算法在边缘处仅考虑了水平和垂直方向上像素的相关性,并未考虑 45 度和 135 度方向上的相关性,且没有考虑到插值点最近位置像素所占权重问题,因此处理后的结果也不尽如人意,甚至对部分图像处理的结果没有双三次插值的效果好。而本文算法在边缘处和细节丰富区域的处理更接近原图,虽然进行了放大处理,图像的细节和原图接近,说明其高频分量得到保持。同时边缘纹理和原图基本一致,说明其在边缘处理上效果也相对较好。因此,本文算法对图像的缩放总体效果更优。

为了更进一步比对不同算法处理效果,将 Bicycle 原图中 60×60 像素大小的原图截取图放大至更大比例的 4 倍,如图 6(c)所示,可以清楚地看到原图像中区域的细节,同时采用不同算法分别对 60×60 像素大小的截取图 4 倍放大后显示,结果如图 6(d-k)所示。可以看出,双线性插值、双三次插值和数据融合算法放大后边缘区域已经比较模糊,边缘插值算法放大后的图像出现一定程度的失真而本文算法在放大 4 倍后仍

能够保留边缘的信息且抑制了高频分量丢失导致图像模糊的情况。

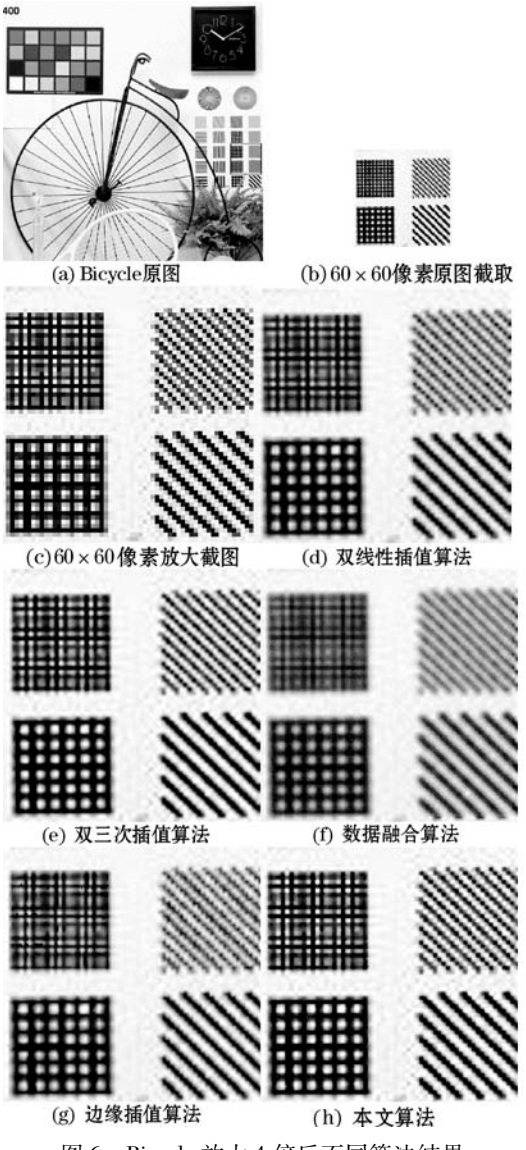


图 6 Bicycle 放大 4 倍后不同算法结果

2.2 客观评价指标对比

峰值信噪比(PSNR)是最常用的客观评价图像处理效果的量化指标,其计算公式如下:

$$PSNR(F_0,F) = 10\lg\frac{255^2}{\frac{1}{M\times N}\sum_{i=1}^{M}\sum_{j=1}^{N}(F(i,j)-F_0(i,j))^2} \quad (10)$$

式中:F_0代表大小为 $M\times N$ 像素的原始图像;F 表示处理过后的和原图像相同大小的图像。PSNR 越大,表示目标图像信息损失越少,图像越接近于原始图像,处理效果则越好。

从客观评价指标的角度说明边缘保护效果的好坏,通常用边缘保护指数(EPI)作为评价指标,其计算公式如下:

$$EPI = \frac{\sum[\,|p_s(i,j)-p_s(i+1,j)|+|p_s(i,j)-p_s(i,j+1)|\,]}{\sum[\,|p_o(i,j)-p_s(i+1,j)|+|p_o(i,j)-p_s(i,j+1)|\,]} \quad (11)$$

式中:p_s 表示处理过后的图像;p_o表示原始图像。EPI 值越接近 1,则边缘保护得越好。

实验选用了 House、Baboom、Items、Ruler 和 Bicycle 图像,用不同算法分别放大两倍再缩小还原,计算其峰值信噪比(PSNR)的值和边缘保护指数(EPI)的值,结果分别如表 1 和表 2 所示。从表 1 中可以看出,本文算法相对于其他 4 种算法,PSNR 均有大幅度的提升。从表 2 中可以看出,本文算法的 EPI 与其他 4 种算法相比更接近 1,对边缘的保护最好。可见,本文算法对图像的处理效果明显优于其他 4 种算法。

表 1 不同算法 PSNR

图像	双线性插值算法[5]	双三次插值算法[6]	数据融合算法[18]	边缘插值算法[19]	本文算法
House	0.591 0	0.782 8	0.672 5	0.824 5	1.041 7
Baboom	0.412 2	0.656 1	0.451 1	1.014 0	1.037 8
Items	0.507 1	0.725 5	0.548 5	0.779 8	1.033 5
Ruler	0.606 4	0.783 4	0.615 1	0.790 5	0.997 3
Bicycle	0.570 3	0.779 6	0.601 3	0.792 0	1.018 2
平均值	0.537 4	0.745 5	0.577 7	0.840 2	1.025 7

表 2 不同算法 EPI

图像	双线性插值算法[5]	双三次插值算法[6]	数据融合算法[18]	边缘插值算法[19]	本文算法
House	36.098 4	39.819 2	36.606 3	40.816 1	50.468 0
Baboom	31.519 2	33.468 3	31.780 4	41.074 8	45.089 2
Items	32.755 7	35.420 5	33.114 2	36.639 2	47.001 6
Ruler	31.060 3	31.131 9	31.166 3	34.489 9	38.487 3
Bicycle	33.438 1	35.655 4	33.802 7	37.550 2	44.330 5
平均值	32.974 3	35.099 1	33.294 0	38.114 0	45.075 3

3 算法硬件实现

3.1 硬件实现方案

硬件实现方案如图 7 所示,首先通过行数据缓存模块对输入的图像数据进行行缓存,采用 FPGA 的内部存储资源,用三组 line-buffer 来实现,因为在水平方向利用二阶牛顿插值需要用到三个像素点来进行计算,为了实现并行运算,采用三组 line-buffer 同时缓存一行像素数据。然后通过水平方向的插值计算模块完成对水平方向的插值计算,将计算得到的数据结果暂存到异步 FIFO 中,等待一行像素计算完成以后,再将异步 FIFO 中的数据送入到外部的 DDR3 存储器中,其原因是为了满足 DDR3 的高速要求,FIFO 模块采用两组进行乒乓操作。等一帧图像水平插值完成以后,再

从 DDR3 中读取数据。通过列数据缓存单元将用于插值计算的三列数据从 DDR3 中读出存入 line-buffer 中,对垂直方向、45 度方向和 135 度方向分别进行插值计算。最终将插值计算的结果存入输出缓存模块中,将数据输出。其中 DDR3 控制器直接调用 IP 核来实现 DDR3 的读写控制。

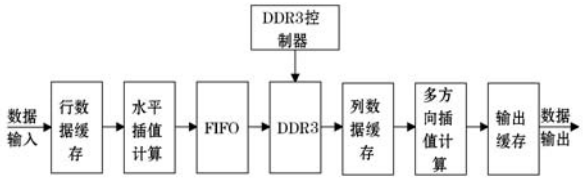


图 7　硬件实现方案

3.2　硬件设计

根据硬件实现方案,下面以最主要的水平插值模块和多方向插值模块为例进行硬件设计说明。

3.2.1　水平插值模块设计

首先使用 line-buffer 对图像的行数据进行缓存,坐标系数生成模块通过对缩放系数进行累加确定源像素地址和插值系数,然后根据像素地址从 line-buffer 中读取参与计算的三个相邻像素点,最后通过插值计算模块计算出目标像素值,从而完成水平方向的缩放处理。模块设计原理如图 8 所示。由于设计中缩放系数使用浮点数计算,需对其放大 256 倍处理,放大后缩放系数用 24 位二进制数表示,其中后 8 位为插值系数,用于确定各像素点权重,前 16 位整数部分用于确定目标像素位置。

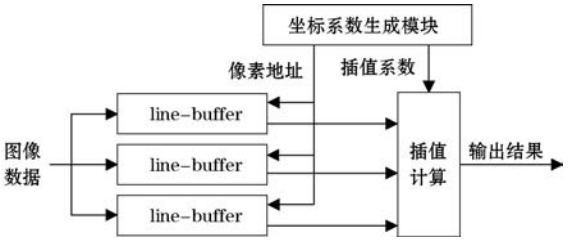


图 8　水平插值模块硬件设计原理图

3.2.2　多方向插值模块设计

多方向插值模块的设计原理和水平插值模块原理类似。先从列数据缓存器中读取目标像素 3 ×3 邻域内用于计算的 7 个像素点,分别计算出垂直方向、45 度方向和 135 度方向的二阶差分和二阶牛顿插值结果,将结果存入寄存器中。之后将二阶差分与设定的阈值比较作为后续多路选择器的判断条件,如果判断为非边缘区域,多路选择器会选择一个方向的插值结果作为最终插值结果;如果判断是边缘区域,则通过式(9)计算得到最终的插值结果。模块设计的仿真结果如图 9 所示,其中信号 T 为设定阈值,信号 k 为垂直方向插值系数,从列数据缓存模块读取的是相邻三列数据,在沿水平方向计算完成后,k 的值才会更新。输入的 7 个像素点信号分别是 f_00、f_01、f_02、f_11、f_20、f_21和 f_22,三个方向的二阶差分绝对值对应的信号分别为 r_h11、r_h22 和 r_h33;信号 F1、F2 和 F3 分别是用 24 位的二进制数表示的三个方向的牛顿二阶插值结果;信号 o_pixel 为最终输出的目标像素值。由于采用流水线的设计方法,最终输出的目标像素比输入像素延迟了两拍。

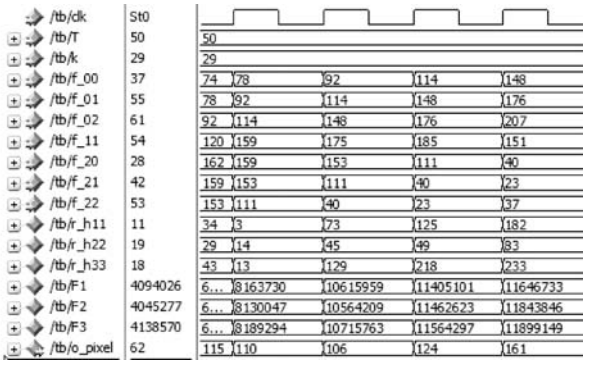


图 9　多方向插值模块仿真结果图

3.3　FPGA 实现及验证

硬件实现和验证所使用的 FPGA 开发板选用的是 ALTERA 公司 Stratix V 系列的 5SGXEA7N2F45C2N FPGA 开发板。在 Quartus II 15.0 软件平台上完成了设计及验证。FPGA 设计验证平台如图 10 所示,首先通过 Quantum Data980 给出视频图像的 RGB 数据、水平同步信号、垂直同步信号和使能信号到 FPGA,再将 FPGA 对数据进行缩放处理,再将 TTL 转 HDMI 桥接芯片将信号转换成 HDMI 标准的信号格式,最后输出到显示器上。FPGA 实现及验证装置如图 11 所示。

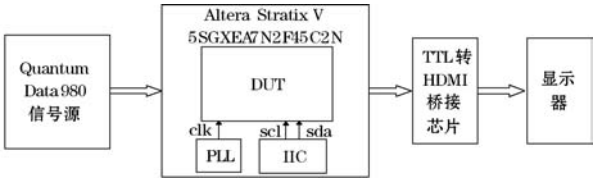


图 10　FPGA 设计验证平台

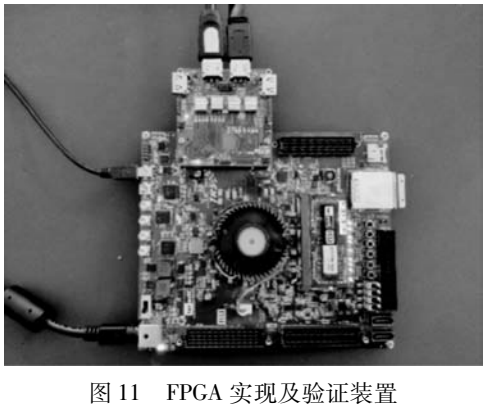
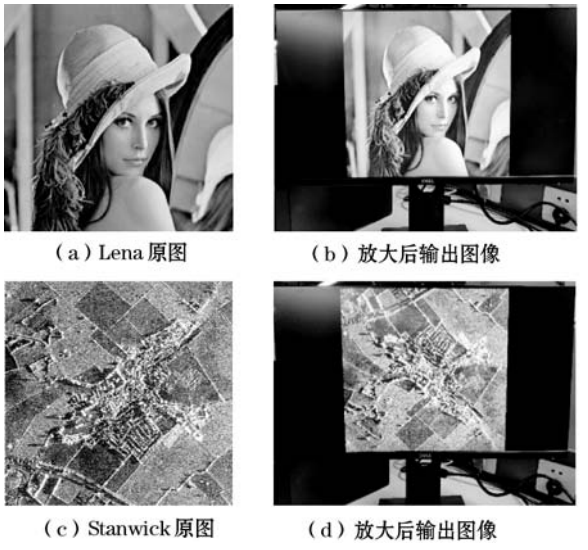


图 11　FPGA 实现及验证装置

实验选取了 2 幅 512 ×512 像素大小的图像进行验证,将其放大到 1 024 ×1 024 像素大小后输出到显

示器上,得到的结果如图 12 所示,可以看到输出结果效果良好。



(a) Lena 原图 (b) 放大后输出图像

(c) Stanwick 原图 (d) 放大后输出图像

图 12 硬件验证结果

4 结 语

本文提出一种基于牛顿二阶插值的边缘自适应缩放算法,对原始图像的水平方向根据插值点映射的位置利用二阶牛顿插值算法进行插值计算得到中间图像,着重考虑垂直方向、45 度方向和 135 度方向上的相关性,通过计算三个方向上的二阶差分并与阈值作比较,选择合适的算法计算得到目标像素值。对算法进行硬件的实现与验证,结果表明,无论从图像处理效果还是客观评价指标结果看,本文算法与其他相关算法相比,其缩放效果及对图像边缘的保护明显有优势,且算法复杂度不高,利于硬件实现。

参 考 文 献

[1] Chang Y M, Fan C P. Spatial-domain edgedirected interpolation based de-interlacing to up-scaling technology for 1080i full HD to progressive 8K Ultra HD[C]//2017 IEEE International Conference on Consumer Electronics-Taiwan (ICCE-TW), 2017.

[2] He H, Mandal S, Buehler A, et al. Improving optoacoustic image quality via geometric pixel superresolution approach [J]. IEEE Transactions on Medical Imaging, 2016, 35(3): 812 – 818.

[3] Seibel H, Goldenstein S, Rocha A. Eyes on the target: surper-resolustion and licenseplate recognition in low-quality surveillance videos[J]. IEEE Access, 2017, 5: 20020 – 20035.

[4] 钟宝江,陆志芳,季家欢. 图像插值综述[J]. 数据采集与处理,2016,31(6):1083 – 1096.

[5] 钟雪燕,夏前亮,陈智军. 基于 FPGA 的图像超分辨率的硬件化实现[J]. 现代电子技术,2017,40(17):44 – 46,50.

[6] 张阿珍,刘政林. 基于双三次插值算法的图像缩放引擎设计[J]. 微电子学与计算机,2007,24(1):49 – 51.

[7] 葛晨阳,郑南宁,任鹏举. 图像混合插值缩放 IP 核的 VLSI 设计[J]. 西安电子科技大学学报,2010,37(1):158 – 162.

[8] Zhang Y, Han X, Zhang H, et al. Edge detection algorithm of image fusion based on improved sobel operator[C]//2017 IEEE 3rd Information Technology and Mechatronics Engineering Conference, 2017.

[9] Ye H, Shen B, Yan S. Prewitt edge detection based on BM3d image denoising[C]//2018 IEEE 3rd Advanced Information Technology, Electronic and Automation Control Conference, 2018.

[10] Lee J, Tang H, Park J. Energy efficient canny edge detector for advanced mobile vision applications[J]. IEEE Transactions on Circuits and Systems for Video Technology, 2018, 28(4): 1037 – 1046.

[11] 王淑青,姚伟,陈进,等. 基于直方图均衡化与形态学处理的边缘检测[J]. 计算机应用与软件,2016,33(3):193 – 196.

[12] Chen W, Yu Y J, Shi H. An improvement of edge-adaptive image scaling algorithm based on sobel operator[C]//2017 4th International Conference on Information Science and Control Engineering, 2017.

[13] 尉成勇,邓燕妮. 基于改进到边缘方向判定方法的图像插值算法[J]. 科学技术与工程,2017,17(7):85 – 89.

[14] Cha Y, Kim S. The Error-Amended sharp edge(ease) scheme for image zooming[J]. IEEE Transactions on Image Processing, 2007, 16(6): 1496 – 1505.

[15] 王效灵,陈涛,汪颖,等. 边沿自适应图像缩放算法[J]. 浙江大学学报(自然科学版),2006,40(9):1507 – 1510.

[16] 李自勤,蒋涛,项铁铭. 基于 FPGA 的边缘自适应图像缩放算法[J]. 计算机工程,2013,39(5):253 – 256.

[17] 李春龙,潘海侠,王华峰. 自适应立方卷积图像插值算法[J]. 北京航空航天大学学报,2014,40(10):1463 – 1468.

[18] 齐敏,程恭,杜乾敏,等. 一种分区域多方向数据融合图像插值方法[J]. 数据采集与处理,2016,31(1):73 – 84.

[19] 刘政林,肖建平,邹雪城,等. 基于边缘的实时图像缩放算法研究[J]. 中国图象图形学报,2008,13(2):225 – 229.

[20] 叶森,丁勇,王翔,等. 基于三次拉格朗日插值的自适应图像缩放[J]. 小型微型计算机系统,2012,6(33):1278 – 1283.

[21] 詹毅,李梦. 图像插值的一个变指数变分模型[J]. 计算机应用,2017,37(7):2067 – 2070.

(下转第 138 页)

阈值隶属度函数,进而构造二维多阈值 α-型模糊散度作为选取最佳阈值的准则函数;其次,用线性递减和线性递增函数分别对 PSO 算法的"自我认知"和"社会认知"部分进行改进,以改善 PSO 算法易陷入局部最优,造成虚假收敛的问题;最后,为解决未加优化的二维多阈值分割算法耗时长的问题,用改进后的 IPSO 算法优化求解二维多阈值 α-型模糊散度的最优阈值,实现图像的多阈值分割。实验结果表明,本文算法对不同测试图像有较好的分割结果,提高了分割精度,同时节省了大量时间。

参 考 文 献

[1] 章毓晋. 图像分割[M]. 北京:科学出版社,2001.

[2] Li J F, Tang W Y, Wang J, et al. Multilevel thresholding selection based on variational mode decomposition for image segmentation[J]. Signal Processing, 2018, 147(1): 80 - 91.

[3] Tarabalka Y, Chanussot J, Benediktsson J A. Segmetation and classification of hyperspectral image using watered transformation[J]. Pattern Recognition, 2010, 43(7): 2367 - 2379.

[4] Matic T, Aleksi I, Hocenski Z, et al. Real-time biscuit tile image segmentation method based on edge detection[J]. ISA Transactions, 2018, 76: 246 - 254.

[5] 赵凤,张咪咪,刘汉强. 区域信息驱动的多目标进化半监督模糊聚类图像分割算法[J]. 电子与信息学报,2019,41(5):1106 - 1113.

[6] Otsu N. A threshold selection method from gray-level histograms[J]. IEEE Transactions on Systems Man and Cybernetics, 1979, 9(1): 62 - 66.

[7] 刘健庄,栗文青. 灰度图象的二维 Otsu 自动阈值分割法[J]. 自动化学报,1993,19(1):101 - 105.

[8] Wang D, Li H H, Wei X Y, et al. An efficient iterative thresholding method for image segmentation[J]. Journal of Computational Physics, 2017, 350: 657 - 667.

[9] Ye Z, Ye Y, Yin H. Qualitative and quantitative study of GAs and PSO based evolutionary intelligence for multilevel thresholding[C]//2017 10th International Symposium on Advanced Topics in Electrical Engineering(ATEE). IEEE, 2017.

[10] 吴虎胜,张凤鸣,吴庐山. 一种新的群体智能算法——狼群算法[J]. 系统工程与电子技术,2013,35(11):2430 - 2438.

[11] Civicioglu P, Besdok E. A conceptual comparison of the Cuckoo-search, particle swarm optimization, differential evolution and artificial bee colony algorithms[J]. Artificial Intelligence Review, 2013, 39(4): 315 - 346.

[12] 曹爽,安建成. 狼群优化的二维 Otsu 快速图像分割算法[J]. 计算机工程与科学,2018,40(7):79 - 84.

[13] 罗钧,杨永松,侍宝玉. 基于改进的自适应差分演化算法的二维 Otsu 多阈值图像分割[J]. 电子与信息学报,2019,41(8):2017 - 2024.

[14] Rupak C, Rama S, Garg M L. An improved PSO-based multilevel image segmentation technique using minimum cross-entropy thresholding[J]. Arabian Journal for Science and Engineering, 2019, 44: 3005 - 3020.

[15] Bhandari D. Fuzzy divergence, probability measure fuzzy events and image thresholding[J]. Pattern Recognition Letters, 1992, 13(12): 857 - 867.

[16] Zhao X, Turk M, Li W, et al. A multilevel image thresholding segmentation algorithm based on two-dimensional K - L divergence and modified particle swarm optimization[J]. Applied Soft Computing, 2016, 48: 151 - 159.

[17] 兰蓉,范九伦. 基于 α-型相对信息的模糊散度[J]. 工程数学学报,2010,27(4):715 - 719.

[18] 兰蓉,史露娜. 基于改进的算术平均算子的阈值分割算法[J]. 西安邮电大学学报,2017,22(2):44 - 52.

[19] Eberhart R C, Kennedy J. A new optimizer using particle swarm theory[C]//Proceeding of the 6th international symposium on micro machine and human science. IEEE, 1995: 39 - 43.

[20] Bustince H, Barrenechea E, Pagola M. Image thresholding using restricted equivalence functions and maximizing the measures of similarity[J]. Fuzzy Sets and Systems, 2007, 158(5): 496 - 516.

[21] Arbelaez P, Maire M, Fowlkes C. Contour detection and hierarchical image segmentation[J]. IEEE Transcation on Pattern Analysis and Machine Intelligence, 2011, 33(5): 898 - 916.

[22] Sezgin M, Sankur B. Survey over image thresholding techniques and comparative analysis of entropy evaluation[J]. Journal of Electronic Imaging, 2004, 13(1): 146 - 165.

(上接第 132 页)

[22] 范清兰,张云峰,包芳勋,等. 基于 NSCT 的区域自适应图像插值算法[J]. 计算机研究与发展,2018,55(3):629 - 642.

[23] Lee S J, Kang M C, Uhm K H, et al. An edge-guided image interpolation method using tayloy series approximation[J]. IEEE Transactions on Consumer Electronics, 2016, 62(2): 159 - 165.

[24] Liu P, Hong Y, Liu Y. A novel multi-scale adptive convolutional network for single image super-resolution[J]. IEEE Access, 2019, 7: 45191 - 45200.