

文章编号:1007-2780(2010)01-0130-04

FPGA 实现的视频图像缩放显示

孙红进

(南京工业大学 信息科学与工程学院,江苏 南京 210009,E-mail: sunhjin@yeah.net)

摘 要: 对几种常用的图像缩放算法进行了比较,在权衡了算法复杂度、缩放效果和 FPGA 逻辑资源等 3 大因素后,选择了双线性插值算法来实现图像缩放。重点介绍了双线性插值算法和该方法的 FPGA 硬件实现方法,包括图像数据缓冲单元、插值系数生成单元以及插值计算单元等。应用结果表明,双线性插值算法及其硬件实现模块达到了预期的效果。

关 键 词: 图像缩放;FPGA;双线性插值

中图分类号: TP301.6 **文献标识码:** A

FPGA Realization of Video Image Zooming Display

SUN Hong-jin

(College of Information Science and Engineering, Nanjing University of Technology,
Nanjing 210009, China, E-mail: sunhjin@yeah.net)

Abstract: Some commonly used image scaling algorithms were introduced, as well as a comparison among several algorithms. After weighing the algorithm complexity, zoom effects and FPGA logic resources, etc., the bilinear interpolation algorithm was chosen to achieve image scaling. This work focused on the bilinear interpolation algorithm and the method of FPGA hardware implementation, including image data buffer unit, interpolation coefficient generating unit, as well as the interpolation calculation unit. The realized results show that the bilinear interpolation algorithm and its hardware realizing modules achieve the desired results.

Key words: image scaling; FPGA; bilinear interpolation

1 引 言

图像是人类感知世界的视觉基础,是人类获取信息、表达信息的重要手段。现在研究较多的是数字图像,它是由连续函数或离散函数生成的抽象图像,能直接被计算机处理。图像缩放是数字图像处理的重要一环,在实际应用中经常要将图像进行放大或缩小。例如,等离子电视、液晶电视和发光二极管(LED)显示屏等显示器需将接收到的图像分辨率进行缩放,使之与显示器的物理

分辨率一致,才能在显示器上完整地显示视频图像。视频图像处理的详细内容请见参考文献[1,2]。目前,市场上有数字视频后处理芯片,这些芯片大都是从欧美、日、韩等地的公司进口的,国内最早进行视频处理芯片研究的是西安交通大学^[3],但这些芯片主要应用于高复杂度的系统,应用在低复杂度的系统则显得不合适,比如应用在需要单纯进行图像缩放的场合,则显得浪费,还可能带来不必要的麻烦。本文要实现的图像缩放是视频图像的实时缩放,而不是仅实现某个缩放比例

收稿日期:2009-06-26; 修订日期:2009-07-16

作者简介:孙红进(1985—),男,江苏南京人,硕士研究生,主要研究方向为数字信号处理、显示技术。

的图像缩放,适合于低复杂度的系统,成本较低。

2 算法原理

2.1 算法选择

图像缩放算法种类较多,几乎都是通过插值算法实现的,传统的插值算法有:最近邻插值、双线性插值和立方卷积插值等。最简单的插值方法是最近邻插值,与其他两种插值算法相比,最近邻插值具有简单快速的特点,但是对于近邻像素点的灰度值有较大改变,细微结构是粗糙的。双线性插值方法具有低通滤波器性质,使高频信息受损,图像轮廓模糊,但边缘处的过渡比较自然。与最近邻插值相比,它的计算量较大,但缩放后的图像质量高,不会出现像素值不连续的情况,在对图像边缘质量要求不是非常高的情况下,这种方法是完全可以接受的。立方卷积法计算量要比双线性插值大很多,但精度高,能保持较好的图像边缘细节。实际应用中,因为 FPGA 内的逻辑资源有限,而算法越复杂占用的逻辑资源就越多,对 FPGA 完成图像的实时缩放有重要影响^[4-8],所以实际应用中应选择合适的算法,并非精度越高越好。综上所述,本文选用双线性插值算法来实现图像缩放。

2.2 双线性插值算法

在双线性插值中,新创建的图像的像素值是由原图像位置在它附近的 (2×2) 个邻近像素的值通过加权平均计算得出的,然后由新图像的某个像素 (x, y) 映射到原始图像 (x', y') 处,对 x', y' 取整得到 (u, v) 并得到 $(u+1, v)$ 、 $(u, v+1)$ 和 $(u+1, v+1)$ 的值,利用双线性插值得到像素点 (x, y) 的值并写回新图像^[6,7]。图 1 为双线性插值的原理图。

设原始图像中相邻两行中相邻两列的 4 个像素点为 p_1 、 p_2 、 p_3 、 p_4 ,水平和垂直两个方向点距为

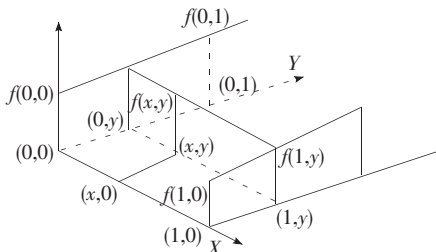


图 1 双线性插值原理图

单位长度 l 。4 点坐标分别为 $p_1(x, y)$ 、 $p_2(x+l, y)$ 、 $p_3(x, y+l)$ 、 $p_4(x+l, y+l)$,目标像素点的坐标为 $A(x+dx, y+dy)$, dx 、 dy 分别是目标像素点与原始图像中邻近点的水平和垂直坐标方向上的两个增量。那么,采用双线性插值法得到目标插值像素点 A 的灰度值为:

$$\begin{aligned} A(x+dx, y+dy) &= (1-dx)(1-dy)p_1 + \\ &dx(1-dy)p_2 + (1-dx)dy p_3 + dx dy p_4 \\ \text{考虑以 FPGA 来实现插值运算时,共需要 8} \\ \text{个乘法器、2 个减法器和 4 个加法器,占用了较多的} \\ \text{逻辑资源}^{[5]}, \text{所以将它改写为:} \\ A(x+dx, y+dy) &= p_1 + dx(p_2 - p_1) + \\ &dy(p_3 - p_1) + dx dy(p_1 - p_2 - p_3 + p_4) = \\ &p_1 + (p_2 - p_1)dx + (p_3 - p_1)dy + \\ &[(p_4 - p_3) - (p_2 - p_1)]dx dy \end{aligned} \quad (1)$$

3 算法的 FPGA 实现

FPGA 选用 ALTERA 公司型号为 EP1C6Q-240I7 的工业级 FPGA。整个算法主要由图像数据缓冲模块、系数生成和逻辑控制模块、双线性插值计算模块等 3 大模块来实现,如图 2 所示。

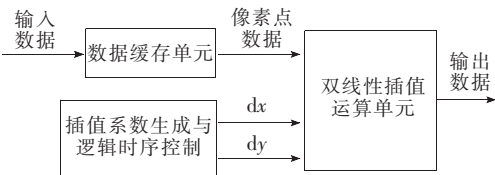


图 2 用 FPGA 实现双线性插值的系统框图

Fig. 2 System block diagram of realizing bilinear interpolation using FPGA

3.1 图像数据缓冲

数字视频信号的数据量很大,传输带宽特别高。在用硬件系统对数字视频图像进行实时处理时,需要建立一个图像数据的存储缓冲机制,为插值运算及时准确地提供图像数据。一般情况下,需要采用片外大容量高速存储器存储至少两帧的图像数据,利用“Ping-Pong”轮换机制实现图像数据缓冲。当需要使用一帧内像素点数据时,可以及时从一个存储器中读取;与此同时,连续的图像数据被存至另一个存储器中。但是,这样的帧存储器不仅增加了成本,还要在 FPGA 内利用逻辑模块设计帧存控制逻辑。为了降低成本,提高系统集成度,本文利用 FPGA 的内部逻辑资源为每

Fig. 1 Principle diagram of bilinear interpolation

一彩色通道设计了一组由 3 个双端口 RAM (DPRAM)组成的图像数据缓冲阵列,并将这样一个缓冲阵列称为行缓冲器^[5,9]。

3.2 插值系数生成

根据双线性插值运算公式及二维图像插值的 FPGA 实现方法,两个运算系数 dx 和 dy 需要准确及时地求出,提供给插值运算单元,并且只有当生成的系数在时序节拍上与图像数据存储、插值计算单元的时序配合一致时,才能有效地完成整幅图像的插值运算。因此,在生成系数的同时,必须能够给出系数在原图像中对应邻近像素点的坐标信息,即在 DPRAM 中的存储地址,以便图像数据缓冲模块可以同步地取出参与运算的 4 个图像数据^[5]。

首先需要确定缩放因子 S 。对于以单位长度“1”来表示相邻两个像素点的距离的模型进行缩放时,以 S 为间隔形成一个新采样格栅,重新对原图像在水平和垂直两个方向上进行重采样,新图像的像素点距离即为 S 。如果把分辨率为 640×480 的图像转化为 800×600 的图像,就需要放大

1.25 倍,则缩放因子为:
$$S=l/1.25=0.8$$

如果把分辨率为 $1\,024 \times 768$ 的图像转化为 800×600 ,将图像缩小 0.781 25 倍时,缩放因子为:
$$S=l/0.781\,25=1.28$$

上述两种情况下,目标图像在水平和垂直两个方向的缩放比例是一样的。若在水平和垂直两个方向上的缩放比例不一致,则缩放因子也就不相同,分别定义为 S_x 和 S_y 。那么,对应于新图像中坐标为 (m,n) 的像素点,系数 dx 和 dy 可以由公式(2)得到:

$$\begin{aligned} dx &= m \times S_x - [m \times S_x] \\ dy &= n \times S_y - [n \times S_y] \end{aligned} \tag{2}$$

式(2)中, $[\]$ 表示下取整。

4 双线性插值运算单元

在获取插值像素点相邻 4 点的图像数据和插值系数后,就可以进行双线性插值运算了。依据插值公式(1),设计了一种具有流水线特性的运算逻辑结构,如图 3。

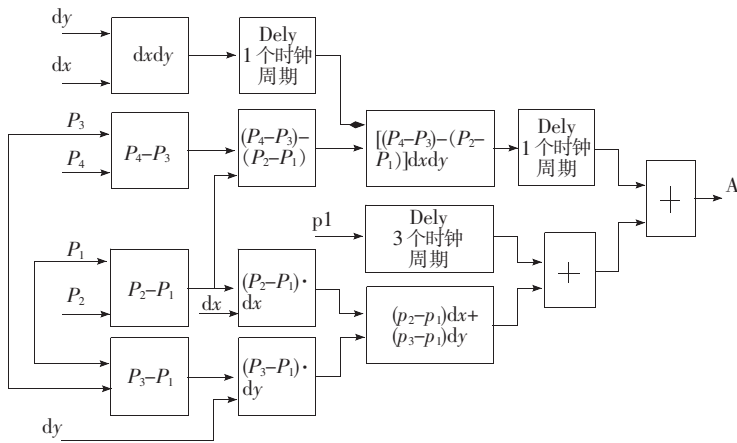


图 3 双线性插值运算单元

Fig. 3 Computing unit of bilinear interpolation

图中, p_1 、 p_2 、 p_3 和 p_4 由 DPRAM 实时输出, dx 和 dy 则由系数生成单元提供。逻辑实现过程中,要特别注意时序节拍的控制,力求时序同步。当某个运算步骤输出数据与其他运算的输出值在某个时钟不能同步到达下一个运算器时,在数据线上插入延时单元,使它们能达到同步,并且适当增加乘法器和加法器等的流水线延时。

5 功能实现

设计中,选用 ALTERA 公司型号为 EP1C6Q-240I7 的工业级 FPGA,采用 VHDL 语言对各逻辑模块进行设计,在开发环境 Quartus II 8.0 下完成了时序仿真和功能验证。此外,整个硬件模块在像素间距为 20 mm 的发光二极管(LED)显示屏中得到了成功应用,虽然从理论上讲,利用双

线性插值算法实现图像缩放存在边缘高频分量的损失,但从 LED 显示屏的显示效果来看,人眼完

全可以接受,所采用的算法适合于这一工程应用。显示效果如图 4 中的(a)、(b)、(c)所示。



图 4 图像缩放显示。(a)100%显示;(b)放大 1.5 倍显示;(c)缩小 0.7 倍显示。

Fig. 4 Image zoom display. (a) Display with 100%; (b) Display with enlarging by 150%; (c) Display with narrowing by 70%.

6 结 论

介绍了一些常用的图像缩放算法并对几种算法进行了比较,在权衡算法复杂度、缩放效果和 FPGA 逻辑资源等 3 大因素后,选择了双线性插值算法来实现图像缩放。从实现结果看,本算法

以及硬件实现模块达到了预期的效果。虽然该算法在理论上会造成被缩放图像边缘高频分量的损失,使图像边缘模糊,但在很多对图像质量要求不是很严格的场合,比如在 LED 显示屏上播放视频图像,所获得的图像质量是人眼完全可以接受的,效果良好,适合工程应用。

参 考 文 献:

- [1] 许录平,编著. 数字图像处理 [M]. 北京: 科学出版社,2007.
- [2] Castleman Kenneth R[美]. *Digital Image Processing* [M]. 北京:电子工业出版社,2008.
- [3] 葛晨阳, 郑南宁, 任鹏举, 等. 平板电视数字视频后处理芯片的设计与实现 [J]. 西安交通大学学报,2008,42(10): 5-9.
- [4] 吴均, 朱重光. 带锐度保持的斜投影 B2 样条图像缩放 [J]. 遥感学报,2002,6(2):108-112.
- [5] 盛磊. 面向特种 LCD 图像处理方法与 FPGA 实现研究 [D]. 合肥:合肥工业大学博士论文,2005:26-40.
- [6] 蔡希昌, 翟林培, 周九飞, 等. 基于 FPGA 的非 XGA 标准图像显示控制器的设计 [J]. 液晶与显示,2008,23(3): 301-306.
- [7] 徐伯庆, 孙国强, 陈离. 数字图像缩放的原理与硬件实现 [J]. 上海理工大学学报,2002,24(3):223-225.
- [8] Hearn Donald, Baker M P. 计算机图形学 [M]. 蔡士杰, 宋继强, 蔡敏, 译. 北京:电子工业出版社, 2007.
- [9] Feng Tao, Xie Wenlu, Yang Lianxing. An architecture and implementation of image scaling conversion[C]//The 4th International Conference on ASIC Proceedings, Shanghai, China; IEEE, 2001:409-410.