文章编号:1002-8692(2007)08-0018-03

基于移动窗口的图像缩放算法*

·论文·

田利波[1,2], 王瑞光[1]

(1. 中国科学院 长春光学精密机械与物理研究所, 吉林 长春 130033; 2. 中国科学院 研究生院, 北京 100039)

【摘 要】针对传统双线性插值法在缩小时需要预缩放,硬件实现复杂,成本高,提出了改进的基于移动窗口的缩放算法,其实现简单,成本低。用 Matlab 前期仿真表明缩放效果好。并对 LED 屏的'比例缩放'工程问题,详介了 FPGA 设计过程,给出了后期验证方案。

【关键词】图像缩放; 传统双线性插值法; 移动窗口; 现场可编程门阵列

【中图分类号】TN941; TP391.4 【文献标识码】A

Image Scaling Algorithm Based on Moving Window

TIAN Li-bo[1,2], WANG Rui-guang[1]

(1. Changchun Institute of Optics, Fine Mechanics and Physics, Chinese Academy of Sciences, Changchun 130033, China;
2. Graduate School of the Chinese Academy of Sciences, Beijing 100039, China)

【Abstract】 The traditional bilinear interpolation in zooming-out needs to pre-zoom and needs different hardware implement between zooming-out and zooming-in, so its cost is high and it is complicated. An improved image scaling algorithm based on moving window is presented, which is simple and low cost. Preliminary simulation using Matlab shows the good results and proves its feasibility. Subsequently, as for practical engineering problems of "zoom ratio" in LED display, the design process based on the FPGA is introduced, and a later stage verification scheme is given.

【Key words】 image-scaling; traditional bilinear interpolation; moving window; FPGA

1 引言

目前,LED 显示系统主要通过 PC 机的 DVI 接口取得视频源。LED 屏一般由几个等大的模块(单个模块一般为 128×64)拼装而成,在一般场合其分辨力为 256×256 到 1 024×768,在某些特殊场合(如体育馆)会更高。PC 机显卡出来的视频源需经 0.5~2 倍的缩放后才能送入 LED 屏显示。图像缩放常用插值算法有最近邻域插值法(Nearest Neighbor interpolation)、双线性插值算法(Bilinear interpolation)、双立方插值算法(Bicubic interpolation),其中双线插值法效果较好,硬件实现较容易,故常用在视频处理芯片中。本文就双线性插值法存在缩小时需要预缩放且缩放要分别进行、延时大[1]、占用硬件资源多提出:缩放时,目标像素点的产生只与相邻 4 个原像素点有关。文献[2-3]认为缩放因子小于 0.5 时,要先经预缩放后再缩小,如只是部分像素参与运算,效果可能较差,但经 Matlab 仿真后,笔者发现其缩放效果并不差。在缩放因子为 1~2 时,采用缩放成奇偶 2 场后再合成 1 帧的方法,减小了硬件实现复杂性和成本。

2 双线性插值法的改进

2.1 传统双线性插值法

双线性插值法是通过求像素 2×2 邻域内的权平均值,也就是选取与待插值的目标像素点(a+i, b+j)(其中 a 和 b 为整数; i 和 j 为浮点数)的水平和垂直方向坐标最近的 4 个原像素,然后根据插值点到各个原像素点的距离进行加权求和,即得到目标新像素值。

传统的双线性插值算法是:先把输入的源图像进行水平方向的缩放,然后写入行缓冲存储器中,再按垂直方向缩放,如遇到缩放因子小于 0.5 时,还要先压缩到压缩率为 0.5~1 之间,缩放工作过程如图 1 所示。帧缓冲存储器如用 FPGA 内的 RAM 实现,对于缓存 1 帧分辨力为 1 024×768 的图像则要 18 Mbit 的空间,这显然不可行,如用片外 SDRAM,则增加了系统设计的复杂度。可见双线性插值缩放的低成本实现是有困难的。

图 1 传统的双线性缩放结构框图

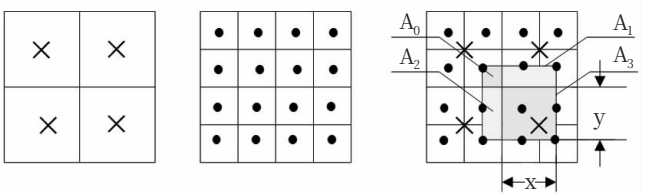
2.2 改进的双线性插值法

笔者对传统的双线性插值法进行了改进,硬件实现简单。其原理是:把像素点看成矩形像素块,则原图像 f(m, n)可看作由 m×n 个单位面积为 1.0 的像素块组成,如图 2a;按水平缩放因子 $Ratio_H$ 和垂直缩放因子 $Ratio_V$ 缩放成的目标图像为 f(M, N),其可看作由 M×N 个像素块组成,每个像块长宽分别为 m/M, n/N,即分别为 $1/Ratio_H$, $1/Ratio_V$,如图 2b。目标图像上每块像素矩形块的中心即为要插值的目标像素点在原图像中的位置,见图 2b,为

* 吉林省与中国科学院科技合作资金项目(2005SYH20010)

了选择与插值点相关性最大的4个原像素点，可用以插值点为中心的面积为1.0的窗口从原图像的最左上角沿行、列的方向移动到最右下角，把该移动的窗口所覆盖的相邻4个原像素块作为参与插值运算的4个点（见图2c）。覆盖在某个原像素块的面积记为$A_i(i=0,1,2,3)$，对应面积越大的原像素对插值点的贡献越大，即权重值越大，则目标新像素值I可用下式计算

$$f(a+i,b+j)=f(a,b)A_0+f(a,b+1)A_1+f(a+1,b)A_2+f(a+1,b+1)A_3 \quad (1)$$



（a）原图像2×2　（b）目标图像4×4　（c）目标像素的产生

注："×"表示原像素点，"·"表示目标像素点

图2　改进双线性插值法的原理示意图

式中：A_0, A_1, A_2, A_3分别表示移动窗口覆盖在原图像上的各像素块的面积，$A_0+A_1+A_2+A_3=1$。

$$\begin{cases} A_0=(1-x)(1-y) \\ A_1=x(1-y) \\ A_2=(1-x)y \\ A_3=xy \end{cases} \quad (2)$$

由式(1)和式(2)可推出

$$f(a+i,b+j)=(1-x)(1-y)f(a,b)+x(1-y)f(a,b+1)+(1-x)yf(a+1,b)+xyf(a+1,b+1) \quad (3)$$

笔者研究发现可变速求得第一行第一列的目标坐标，就可通过加法器依次得到每一点的目标像素坐标。由图2看出x=j+0.5, y=i+0.5，可很快求出A_0, A_1, A_2, A_3。当缩放因子变化时，本文算法不必计算窗口的大小，因为窗口的大小是固定的，这样减少了计算量。但插值点位于原图像的边缘时，窗口只覆盖到一个或两个原像素点，对此本文假设覆盖到原图像外的像素块值为0。当某一缩放比例时，插值点只覆盖到一个原像素点，如i或j等于0.5时，这时目标像素值和原像素值相同，采样具有最近邻特性，保留了高频成分。

当缩放因子小于0.5时，本文只计算与插值点相邻的相关性最大的4个原像素点，其他原像素点没有参与计算（见图3），且未采用预缩放。所谓预缩放就是在缩放因子小于0.5时，分步进行缩小，每步缩小一半，直到缩放到缩放因子在0.5~1之间时，再采用双线性插值法缩放。

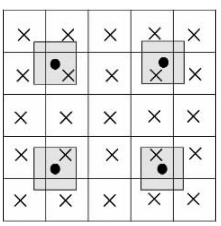


图3　缩放因子小于0.5时缩放示50意图

2.3　算法验证

本方法未保证每个原像素值参与运算，可能缩小时效果差，为此用Matlab仿真，验证其可行性。客观验证采用峰值信噪比（PSNR）[5]，选择分辨力为256×256的标准8位Lenna图，先以一定的缩放因子缩放得到一个目标图像，再用该缩放因子的倒数作为新的缩放因子把该目标图像缩放成原来大小的新目标图像，最后计算该图相对于原图的PSNR值。用Matlab编程仿真可得表1结果。其中传统算法直接调用Matlab中的库函数。在缩放因子为0.25~1时，改进算法的PSNR值略高于传统算法。从主观视觉验证效果上看，把Lenna图分别用传统算法和本文的改进算法缩小为1/4后，可看出本文算法与传统算法的效果几乎相同。由此可见前面所假设的"本文算法在缩小时效果差"不成立，验证了本文算法在减小硬件成本后，仍能达到很好的缩放效果。

表1　两种双线性插值算法的PSNR对比

Ratio	不同算法的PSNR/dB	
	传统算法	改进算法
0.25	23.050	24.686
0.50	27.423	28.757
0.75	31.350	32.600
1.25	34.534	35.929
1.50	34.061	36.953
2.00	32.469	36.944

3　硬件实现

3.1　系统实现结构

笔者的思路是：插值点只与相关性最大的相邻4个原像素点有关，一般这4个点在相邻2行，所以每次处理时，先根据要插值点的坐标反向变换得到其在原始图像中对应的浮点坐标(a+i, b+j)，从而得到所要缓存的2行原像素，可以用FPGA里RAM块配置成2个双端口或2个三端口RAM来缓存2行。缩放因子为0.25~1时，一些原像素点可能重复取数2次，即一个原像素时钟必须取数2次，否则，缓存的2行像素数据还没有处理完，下一行像素数据已到，如果采用双端口RAM，则用倍频时钟来读RAM，但来自PC机的DVI接口的视频源经TMDS解码后的像素时钟为80 MHz左右，倍频时钟高达160 MHz左右，对于后续处理有相当大的压力，所以配置成三端口RAM，可用原像素时钟读写该RAM，一个时钟周期可同时读出2个相同的原像素；但当缩放因子为1~2时，原像素可能重复取数4次，则至少需要4行来缓存原像素，本文采用"缩放成目标图像的奇场和偶场后再用奇偶场合成一帧"的技巧，这样只需缓存2行。

整个图像缩放系统结构如图4所示。

工作过程为：接收来自PC机的显卡的DVI接口的视频源，经TMDS解码成同步信号（Vsync）、行同步信号（Hsync）、像素时钟（Pclk）、数据使能信号（DE）和24 bit并行像素数据（Pdata）。解码后的信号送入行选择模块

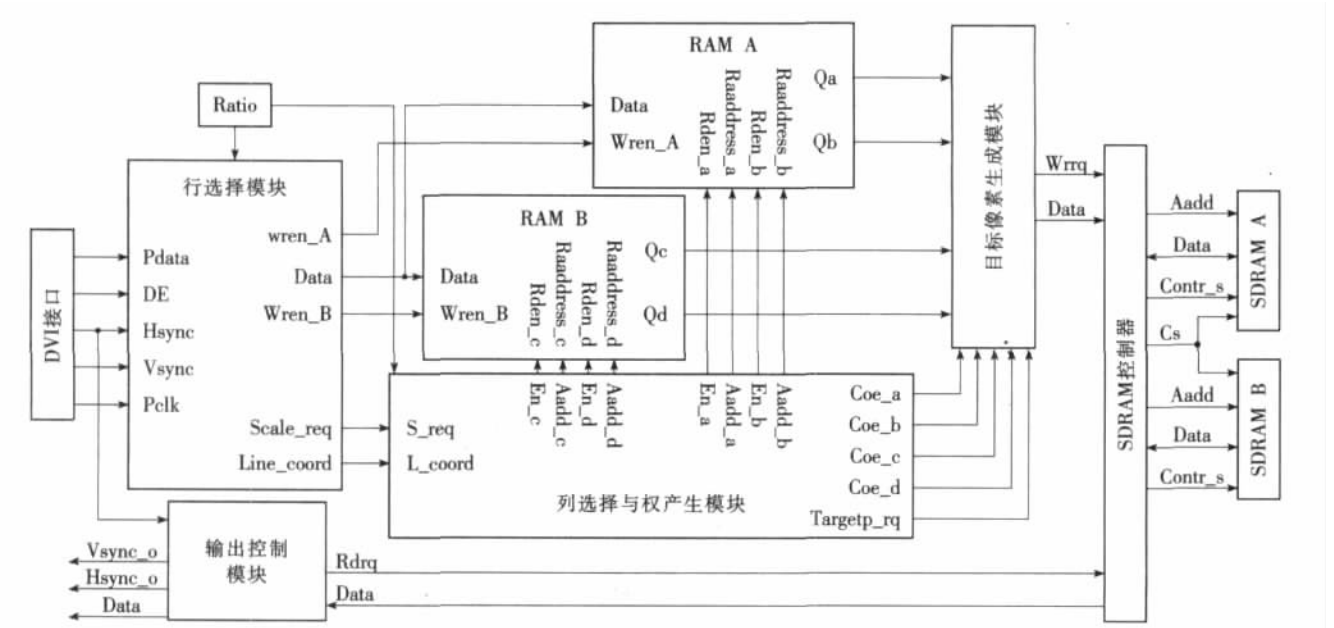


图 4　系统结构设计

中，该模块读取缩放比例寄存器中的缩放因子，求出目标图像的各行位置，然后把其反向变换为在原图像中对应的用浮点数表示的纵坐标，从而选择插值运算所需要的一行或两行原像素缓存在 RAM 中，并产生行缩放请求信号(Scale_req)。列选择与权产生模块接到请求信号后，根据缩放因子求出目标图像的各列的位置，然后把其反向变换为原图像中对应的用浮点数表示的横坐标，从而计算插值所要的原像素的列，这样把列当作读地址送入相应 RAM 中；同时算出与所选 4 个原像素所对应的权值；然后，发出目标像素产生请求(Targetp_rq)。目标像素生成模块响应请求后，产生的目标像素通过 SDRAM 控制器缓存入 SDRAM。SDRAM 至少要有缓存 2 帧的目标图像数据空间，这里在逻辑空间上把 SDRAM 分成 A，B 区，由控制器构成双通道进行交替读写，在写入时数据是不连续的，而从 SDRAM 读出时是连续的，这样可把缩放时不连续的图像数据连续输出。

3.2　系统实现与验证

FPGA 芯片采用 Altera 的 Cyclone 系列 EP1C12Q240C8N 型芯片，是一款高性价比的低端芯片，其 RAM 块共有 239 616 bit，足够配置成 2 个 1 024×24 大小的三端口 RAM。TMDS 解码芯片用 TI 的 TFP101A 芯片，采用 DVI1.0 标准，能支持 XGA 分辨力，输出像素时钟可达 86 MHz。

整个系统的验证环境如图 5 所示，来自 PC 机的 DVI 视频源经 TMDS 解码芯片解码成数字的 RGB 信号，再送入 FPGA 中进行缩放处理，最后把缩放后的目标图像送入显示控制系统中，由于控制显示在 LED 显示屏上。笔者主要进行 1 024×768@60~80 Hz 的视频源缩放，采用前述三端口 RAM 读写数据，验证板上的最高时钟不超过 100 MHz，普通的 4 层板 PCB 设计就可满足要求，从而降低板级设计要求。

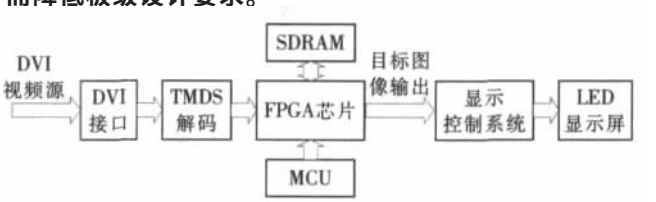


图 5　系统验证

整个系统采用 MCU 作为主控制器，上电复位后，设置 FPGA 的工作状态并读入缩放因子到 FPGA 中，外部输入 DVI 视频源后，开始缩放工作，最后显示在 LED 显示屏上，验证缩放效果。

4　小结

笔者就来自 DVI 接口的 1 024×768@60~80 Hz 分辨力的图像进行缩放，已在 Quartus II 5.1 环境中经过功能仿真验证，结果表明硬件实现结果与经 Matlab 仿真结果相同，缩放后的图像比较清晰。由于整个 LED 屏显示控制比较复杂，调试难度大，暂还未完成系统验证，下一步工作将进行系统验证。

参考文献

[1] 刘政林，李仕杰，邹雪城，等. LCD 定标器图像缩放引擎的设计[J]. 计算机工程与科学，2006，28(1)：39-40.

[2] 陈素琼，苏凯雄. 数字电视的多画面功能设计和实现[J]. 有线电视技术，2006，13(4)：42-45.

[3] 汪颖，陈涛. 视频图像缩放的设计及实现[J]. 电视技术，2004(6)：36-38.

作者简介：

田利波(1982-)，硕士生，主研 FPGA 设计及数字视频实时处理。

责任编辑：哈宏疆　　收稿日期：2007-06-11