

LOG 算子在 FPGA 中的实现

祁艳杰^{1,2}

- (1.电子科技大学 电子工程学院,四川 成都 610054;
- 2.太原科技大学 电子工程学院,山西 太原 030024)

摘要:介绍了一种高斯—拉普拉斯(LOG)算子在 FPGA 中的实现方案,并通过对一幅 bmp 图像的处理,论证了在 FPGA 中实现的 LOG 算子的图像增强效果。

关键词:高斯—拉普拉斯(LOG) FPGA Verilog

图像增强处理的目的是改善原图像的“视觉效果”(包括人和机器“视觉”),针对给定图像的应用场合,有目的地强调图像的整体或局部特性。图像增强技术是多种图像处理的重要环节,尤其是在视频图像的应用中,更是各厂商提高自己竞争力的焦点。

日前三洋电机(Sanyo)借助 Altera 公司的 Stratix FPGA 和 Nios II 嵌入式处理器,迅速、经济地实现了新的家庭影院系统,系统具有良好的特性和图像质量。三洋电机在最新的家庭影院产品中采用了 Altera FPGA 的处理器,实现了功能最丰富、视觉效果最佳的显示方案。Altera 的可编程解决方案支持这种业界最佳产品的快速开发,是 ASSP、昂贵的 ASIC 以及传统处理器等替代方案所无法实现的。

考虑到价格、产权等因素,为了提高现有高分辨图像采集卡的图像质量,借鉴 Sanyo 的经验,作者用 Altera 公司的 EP1S20F780C7 实现了对实时彩色视频信号的图像增强。本文仅对图像预处理(包括平滑、锐化)部分作详细的分析和实现,介绍 LOG 算子在实时彩色图像处理中的优点及其在 FPGA 中的实现,并给出经过 FPGA 处理后的图像效果,以证明这种算法的有效性。

1 高斯—拉普拉斯算子

高斯—拉普拉斯(LOG)算子可以在保持边界的情况下将噪声滤掉,是一种新型的去噪方法。

1.1 高斯算子

高斯滤波是一种根据高斯函数的形状来选择模板权值的线性平滑滤波方法,高斯平滑滤波对去除服从正态分布的噪声效果较好。二维高斯函数为:

$$G(x, y) = Ae^{-\frac{x^2+y^2}{2\sigma^2}} = Ae^{-\frac{r^2}{2\sigma^2}}$$

二维高斯函数具有旋转对称性,即滤波器在各个方向上的平滑程度是相同的。

高斯算子可以用 3×3 的模板 $\frac{1}{16} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$ 来实

现。空域高斯平滑滤波的实质是加权均值滤波,而对彩色图像处理来说,邻域平均值平滑可以在每个彩色平面的基础上进行,其结果与用 RGB 彩色向量执行平均是相同的^[1]。所以,用高斯滤波器进行平滑时,可以分别对 R、G、B 三个通道分开处理,不会影响图像的质量,简化了设计方法。

然而,理论和实验证明,虽然高斯滤波器具有良好的噪声抑制能力,但是对图像的平滑会造成图像中的细节信息损失,从而使处理后的图像产生模糊。

1.2 拉普拉斯算子

拉普拉斯(Laplace)锐化算子是一种二阶微分算子,一个二元图像函数 $f(x, y)$ 的拉普拉斯变换定义为:

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

它是一种各向同性,即旋转不变的算子。

由于拉普拉斯是一种微分算子,因此,它的应用强调图像中灰度慢变化的区域。这将产生一幅把图像中的浅灰色边线和突变点叠加到暗背景中的图像。将原始图像和拉普拉斯图像叠加在一起的简单方法可以保护拉普拉斯锐化处理的效果,同时又能复原背景信息。满足上述条件的前提下,常用的模板为:

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

由于可以通过分别计算每一分量图像的拉普拉斯去计算全彩色图像的拉普拉斯^[1],因而可以分别对 R、G、B 三个通道分开处理,不会影响图像的质量。拉普拉斯算子在增强图像边缘的同时,却增强了图像的噪声。

1.3 高斯—拉普拉斯(LOG)算子

拉普拉斯算子利用图像的二阶微分算子的零交叉

点来进行边缘增强处理,其缺点是对噪声十分敏感。为抑制噪声,可先作平滑滤波,然后再作二次微分,故有 LOG(Laplacian of Gaussians)算子。LOG 算子边缘增强处理方法是 Marr-Hildreth 提出的一种同时具有图像平滑功能和边缘增强功能的二阶微分算法。其处理过程是:首先利用二维高斯函数卷积对图像作最佳的平滑处理,然后再利用平滑图像的二维拉普拉斯函数进行边缘增强处理。

在本文对图像进行增强处理时,针对图像函数的离散性特点,采用上述给出的高斯算子和拉普拉斯算子进行卷积,得到 LOG 算子的 5×5 卷积模板进行卷积计算,LOG 算子模板为:

$$L = \frac{1}{16} \begin{bmatrix} 0 & -1 & -2 & -1 & 0 \\ -1 & 1 & 4 & 1 & -1 \\ -2 & 4 & 12 & 4 & -2 \\ -1 & 1 & 4 & 1 & -1 \\ 0 & -1 & -2 & -1 & 0 \end{bmatrix}$$

2 LOG 算子在 FPGA 中的实现

2.1 处理模块的总体实现

因为 LOG 算子可以在每个彩色平面的基础上分别进行处理,所以在接收到原始的彩色视频图像后,首先把原 24 位的彩色图像分离成 R、G、B 三个通道(每个通道都是 8 位)。在 FPGA 中图像处理模块的总体实现框图如图 1 所示。

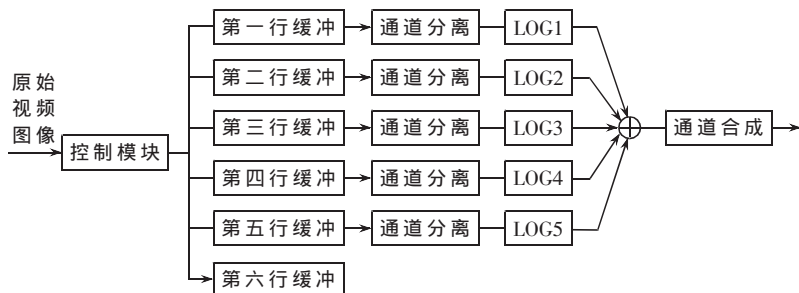


图 1 处理模块总体实现框图

图 1 中,经通道分离模块后,原 24 位图像数据分离成 R、G、B 三个通道,分别对每个通道进行 LOG 运算(如 2.2 节,实际上是对一行的 5 列进行卷积运算),然后分别相加,得出 R、G、B 经 LOG 运算后的结果,最后把 R、G、B 经通道合成模块,合成为 24 位的输出。

每个行缓冲器是用 FPGA 内部自带的 RAM 编程实现的,大小为 1024×24bit(文中假设处理的原始视频图像分辨率是 1024×768),由 6 个 M4K RAM 块组成。控制模块的作用是控制 FPGA 对行缓冲的访问。由于在此文中,要实现的 LOG 算子是 5×5 的卷积模板,所以需要有 6 个行缓冲。因为控制模块控制 FPGA 对其中的一个行缓冲进行写操作的同时,对另外 5 个行缓冲进行读操作,所以控制模块采用的处理数据的方法,实质上就是“乒乓操作”。通过这种处理技巧,提高了数据处理的速度。

通道分离模块的作用是把采集到的 24 位的原始图像数据分离成 3 个 8 位的数据,分离时,由高到低,每 8 位赋予一个通道。如设高 8 位为 R 通道,中间 8 位为 G 通道,则低 8 位为 B 通道。

通道合成模块的作用是把 R、G、B 三个通道的 3 个 8 位数据合成为 24 位的数据,以便于输出。

整个图像处理模块在 FPGA 中的占用资源为:逻辑单元 2 725 个,内部 RAM 为 148 368bit。

2.2 LOG 算子的实现

在 FPGA 中设计实现 LOG 算子时,需考虑两个重要指标:对 FPGA 逻辑资源的占用和图像处理的速度;在设计中采用串并转换、流水线操作等手段来提高图像处理速度并减少占用的逻辑资源。下面以第一行缓冲 LOG1 为例,说明 LOG 算子的实现。图 2 是 LOG 算子的实现框图。

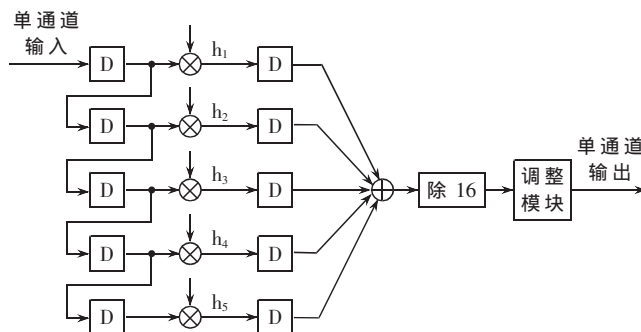


图 2 LOG 算子的实现框图

图 2 中,对于 5 行不同的行缓冲 h_1, h_2, h_3, h_4, h_5 具有不同的系数值。由 1.3 节叙述的 LOG 算子可知,对于第一行, h_1, h_2, h_3, h_4, h_5 分别为 0、-1、-2、-1、0; 第二行, h_1, h_2, h_3, h_4, h_5 分别为 -1、1、4、1、-1 等。

原始视频图像进入图像处理模块后,经过通道分离模块,24 位的原始图像数据分离成 R、G、B 三个通道,然后分别对每个通道进行处理,即分别让每个通道的数据通过 LOG 算子。所以 LOG 算子的输入是单通道(R 或 G 或 B)数据输入。

在图 2 中,串并转换、流水线操作等都是通过第一列 D 触发器实现的。单通道数据输入后,每个步骤数据通过 D 触发器打一个节拍,所以流水线操作类似于一个移位寄存器。由于要实现的 LOG 算子是 5×5 的卷积模板,所以第一列需要 5 个 D 触发器来实现。流水线建立起来后,即可实现 5 列的卷积运算。如图 1 所示,原始图像输入后有 5 行缓冲,每行又有 5 列,所以整个流水线建立起来后,即可实现 5×5 的卷积运算。

图 2 中的第二列 D 触发器是为了减少设计中输出波形的毛刺,以减小设计中可能会出现竞争与冒险。因每条通路到达加法器的延迟时间不一样,容易产生竞争和冒险。故用第二列 D 触发器来打一个节拍,在时钟的上升沿,把 5 条通路相乘后的结果送到加法器相加,

就可以减小竞争和冒险的几率。

由前所述,图像数据经过 LOG 算子卷积后,要除以一个系数 16,在 FPGA 中,把加法器输出的数据右移 4 位,即实现了除以 16 的功能。

调整模块的作用是调整经 LOG 算子运算之后的数据,使其范围在 0~255(8 位数据)之间。由于单通道数据经 LOG 算子卷积后,其位数不再是 8 位数据(超过 8 位),其值有可能为负数,或者超过 255。因此,数据经过调整模块后,为负数的数据赋为 0,超过 255 的数据赋为 255,强制数据在 0~255 之间,以保证图像的正常输出。

在 Quartus II 中对 LOG 算子进行仿真,其仿真波形图如图 3 所示。由输出波形可以看到,LOG 算子的毛刺很小,即竞争和冒险很小,且流水线的建立需要几个时钟的延迟,所以在采集实时视频信号时,输出会有延迟,即位置有所偏离。

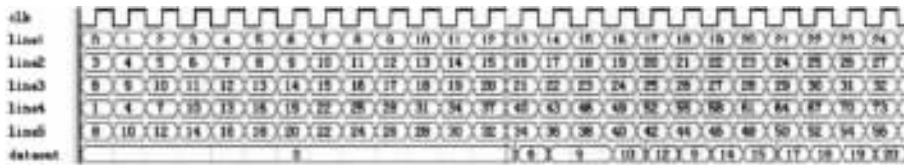


图 3 LOG 算子的 Quartus II 仿真波形

3 经 LOG 算子处理的图像效果及分析

经高分辨图像采集卡采集,然后经 LOG 算子增强之后的图像效果如图 4 所示。它示出了一幅 bmp 格式的灰度图像和它的处理图像。

由图可以看出,经 LOG 算子处理后,无论是在 MATLAB 的仿真中,还是在实际的图像采集卡处理中,图像的边缘部分都得到了加强,达到了既保持边缘又去除噪声的效果。同时在实际中采集彩色实时视频图像时,彩色图像的色彩没有突变,说明经 LOG 算子处理时,可以对彩色图像的三个图像分量分别做处理,不会影响图像效果。另一方面也可以看到,在 MATLAB 中仿真的结果和从实际的图像采集系统处理后的结果还是有区别的,



(a)原始图像 (b)在 MATLAB 中仿真的 (c)经图像采集卡采集并由 LOG 算子处理的图像

图 4 LOG 算子对灰度图像的处理效果图

图像在色彩方面稍微有变化,说明在用 Verilog 实现 LOG 算子时,某些处理方法影响了图像效果,但对彩色图像影响不大。

本文介绍了 LOG 算子在彩色视频图像处理中的优点,给出了在 FPGA 中的实现框图。整个 LOG 算子在 FPGA 中的实现,是用 Verilog 语言编程实现的。实践证明

了 LOG 算子方法的可行性和有效性。同时由于设计中综合运用了乒乓操作、并行操作、流水线操作等各种技术,使图像处理的速度非常快,在处理 VGA 输入、VGA 输出的彩色实时视频图像时,可以达到输入 60 帧/秒、输出 60 帧/秒的效果,而且

Altera 芯片价格低廉。本设计具有很高的实用价值。

参考文献

- [1] GONZALEZ R C, WOODS R E. Digital image processing, 阮秋琦, 阮宇智译. 北京: 电子工业出版社, 2003: 100-105, 258-259.
- [2] 吴继华, 王诚. Altera FPGA/CPLD 设计(高级篇). 北京: 人民邮电出版社, 2005: 19-22.
- [3] 夏宇闻. Verilog 数字系统设计教程. 北京: 北京航空航天大学出版社, 2003.
- [4] 蒋杰. 新型智能车辆视觉系统及图像处理技术的研究. 吉林大学硕士学位论文, 2003: 37-51.

(收稿日期: 2006-11-02)