文章编号:1671-4598(2022)07-0173-08 DOI:10.16526/j.cnki.11-4762/tp.2022.07.026 中图分类号:TP751 文献标识码:A

实时彩色图像自适应中值滤波算法的FPGA实现

韩玉鑫，王晓凯，陆金旺
（山西大学 物理电子工程学院，太原 030006）

摘要：图像在采集、压缩、传输和接收的过程中，会受到脉冲噪声的影响，给后续的边缘检测、图像分割或目标识别等造成干扰；传统中值滤波将图像的RGB分量转为YCbCr分量进行滤波，该转换方法会损失一定的色度信息，且未将彩色信息完全分离，导致滤波后的图像产生色彩偏移；针对上述问题，利用HSI彩色空间中的I和S色彩分量对椒盐噪声进行检测，并通过自适应中值滤波将椒盐噪声滤除，图像还原度更高；利用FPGA对算法进行了硬件加速，在xc7a100tfgg484-2芯片上运行频率达222 MHz，可实现2k@60 Hz视频的实时处理。

关键词：HSI彩色空间；自适应中值滤波；FPGA；椒盐噪声

Real-time Color Image Adaptive Median Filtering Algorithm Based on FPGA

HAN Yuxin，WANG Xiaokai，LU Jinwang
（School of Physics and Electronic Engineering，Shanxi University，Taiyuan 030006，China）

Abstract：In the process of image acquisition，compression，transmission and reception，it will be affected by the impulse noise，which will cause the interference to subsequent edge detection，image segmentation or target recognition. Traditional median filtering converts the RGB components of the image to the YCbCr components for filtering. This conversion method will lose a certain amount of chrominance information，and the color information is not completely separated，resulting in a color shift in the filtered image. Aiming at the above problems，the I and S color components in the HSI color space are used to detect the salt and pepper noise，and the adaptive median filter is used to filter the salt and pepper noise. The image reduction is higher. The algorithm of hardware is accelerated by the FPGA. The system runs at 222 MHz on the xc7a100tfgg484-2 chip，which can realize the real-time processing of the 2k@60 Hz video.

Keywords：HSI color space；adaptive median filter；FPGA；salt and pepper noise

0 引言

数字图像在采集、传输和处理的过程中，由于设备工作环境的恶劣，会引入一些随机、离散噪点，降低了图像质量和视觉效果[1-2]。现实环境中存在多种图像噪声，如高斯噪声、随机噪声、瑞利噪声等[3]，图像降噪应兼顾降噪效果和细节保留这两方面。

目前对椒盐噪声的滤除，有加权中值滤波算法[4]、开关中值滤波算法[5]、自适应中值滤波算法[6]和改进型中值滤波算法[7-10]，自适应中值滤波算法运算量大、复杂度高，虽然在PC端可以达到很好的滤波效果，但由于其串行计算的特点，很难满足对高分辨率、高速图像实时滤波的要求。FPGA（Field Programmable Gate Array）为半定制电路，可通过流水线等操作实现面积与速度的互换，可直接将算法映射为电路，从而可极大加快算法执行的效率[11-13]。目前彩色图像的中值滤波主要是将RGB彩色空间转换到YCbCr彩色空间，然后对Y分量进行滤波。由于YCbCr色彩空间会损失掉色度信息，因此降噪后的图像会产生失真。

鉴于此，结合HSI（Hue Saturation Intensity）色彩空间特点设计了一个基于FPGA的实时彩色图像自适应中值滤波系统，极大的增强了滤波的实时性。最终实验结果表明该系统可对2 k（2 560×1 440）@60 Hz彩色视频实时处理，且滤波后图像色彩还原度高，具有广阔的应用前景。

1 RGB-HSI模块

本节将介绍HSI彩色模型，并将几何推导算法和分段定义算法在FPGA硬件上实现。

收稿日期:2021-12-16； **修回日期**:2022-03-09。
基金项目:山西省重点研发计划(高新技术领域)基金项目(201803D121102)。
作者简介:韩玉鑫(1997-)，男，山西忻州人，硕士研究生，主要从事数字图像处理方向的研究。
通讯作者:王晓凯(1963-)，男，博士，教授，主要从事物联网与智能信息处理方向的研究。
陆金旺(1997-)，男，硕士研究生，主要从事图像处理方向的研究。
引用格式:韩玉鑫，王晓凯，陆金旺.实时彩色图像自适应中值滤波算法的FPGA实现[J].计算机测量与控制，2022，30(7)：173-180.

1.1 HSI 彩色模型

我们观察物体时，用其色调、饱和度和亮度来描述这个物体。HSI 彩色模型可从携带彩色信息（色调和饱和度）的图像中去除强度分量的影响。因此，HSI 模型是开发基于彩色描述图像算法的理想工具。如图 1 所示图（a）为基于圆形彩色平面的 HSI 彩色模型，图（b）为基于三角形彩色平面的 HSI 彩色模型。以图（a）为例，白色顶点与黑色顶点相连的轴为亮度轴，饱和度为像素点与亮度轴的距离，色调为像素点与红色轴的夹角，在亮度轴上饱和度为 0 且无色调。常见的 RGB-HSI 转换算法共有 5 种，算法的具体推导和公式可查阅文献［14］获得。由于 HSI 各分量间相对独立，故可以根据具体应用，将不同算法的 HSI 公式交叉使用。下节分别在 FPGA 上实现几何推导算法和分段定义算法，通过对两种方法转化后图像的质量进行对比，在转化精度与消耗硬件资源之间进行权衡，选择合适的算法。

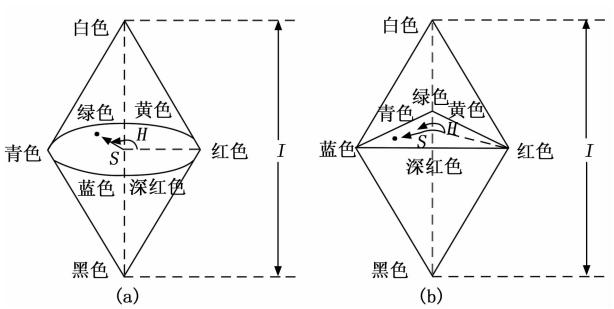

图 1 HSI 彩色模型

1.2 几何推导法硬件实现

由于 FPGA 只能进行整数计算，所以将浮点数转换为整数进行计算，由 8 位 2 进制数表示［0，1］，量化精度为 0.003 921。给定一幅 RGB 彩色图像，每个 RGB 像素的 H 分量可由下式得到：

$$H=\begin{cases}\theta, & B\leqslant G\\ 360-\theta, & B>G\end{cases} \tag{1}$$

式中，

$$\theta=\arccos\left\{\frac{(R-G)+(R-B)}{2\sqrt{(R-G)^2+(R-B)(G-B)}}\right\} \tag{2}$$

饱和度分量由下式给出：

$$S=255-\frac{3\times\min(R,G,B)}{R+G+B}\times 255 \tag{3}$$

最后强度分量由下式给出：

$$I=\frac{1}{3}(R+G+B) \tag{4}$$

在硬件中计算 arccos（x）比较困难，主要有以下几种计算方式：

1）通过 Xilinx 官方提供的 CORDIC IP 核，该 IP 核可计算 arctan（x），利用 arctan（x）与 arccos（x）转换关系即可完成对 arccos（x）的计算。由于在原公式的基础上加入了开方和除法等复杂运算，而 IP 核直接计算 arctan（x）需要 20 个时钟周期的延迟，所以增大了硬件资源的消耗和处理时延；

2）通过下式：

$$\arccos(x)=\frac{\pi}{2}-\left(x+\ldots+\frac{x^{2n+1}}{2n+1}\times\frac{(2n)!}{4^n(n!)^2}+\ldots\right)\ |x|<1 \tag{5}$$

上式为 arccos（x）的泰勒展开式，可通过上式选取合适的 n 逼近 arccos（x），但是当 $|x|$ 接近 1 时，上式收敛速度很慢，所需硬件乘法器、除法器和加法器急剧增加，处理时延也会进一步加大，无法满足图像地实时处理；

3）使用 ROM 查找表的方式实现，将提前计算好的数值从 ROM 中读取出来，只需要 2 个时钟周期就可以完成运算，可满足实时图像处理。

为符合处理实时图像的要求，选取 ROM 查找表方式。当 x 趋近于 1 时，由于 arccos（x）的导数趋于无穷大，所以 arccos（x）会产生较大误差。将 x 分别量化到［0，255］和［0，4 095］，通过这两种量化精度之间的对比，说明上述误差对 H 分量的影响。

乘法器实现主要分两种情况：在乘一个定值时，为节约硬件资源，采用 FPGA 擅长的移位运算；在乘一个变量时，为保证系统性能，采用 Multiplier IP 核实现。除法器实现也分为两种情况，当除数是一个定值时，采用下式实现：

$$quotient=divident\times\frac{2^n\times 2^m}{divisor}\div 2^{n+m} \tag{6}$$

其中：2^n 为大于除数的最小整数，m 控制计算的精度，m 越大精度越高。例如，计算 $x/3$，我们取 2^n 为 4，m 为 8，公式就变为 $x\times 341\div 1024$，乘法采用上述左移方式实现，由于除数为 2 的整数次幂，除法采用右移相应的位数即可实现。当除数是变量时，采用 Divider IP 核计算，计算结果将余数舍去，只取商即可。开平方的运算通过调用 CORDIC IP 核实现。

通过硬件转换与软件仿真结果对比发现，S 分量和 I 分量的误差均在 ± 1 以内，而 H 分量在某些像素点的误差达到 20。这是由于硬件计算开平方、除法和反余弦均有误差，而开平方的误差会引入到除法计算中，加大除法计算的误差，除法计算的误差又会被引入到反余弦中，这样会将误差逐级放大。在 1.4 节会通过具体图片的硬件仿真说明这种误差对色彩空间转换的影响。

1.3 分段定义法与几何推导法相结合

由于 HSI 各分量相互独立，所以可将分段定义算法与几何推导算法相结合，即 H 分量通过分段定义法求得，S 分量、I 分量通过几何推导算法求得，其中上节已完成几何推导法 S、I 分量的计算，这节主要完成对分段定义算法中 H 分量的计算，H 分量由下式给出：

$$H=\begin{cases}\theta, & \theta\geqslant 0\\ 360+\theta, & \theta<0\end{cases} \tag{7}$$

式中，θ 由下式给出：

$$\theta=\begin{cases}60\times\dfrac{G-B}{\mathrm{Max}-\mathrm{Min}},\mathrm{Max}=R\\60\times\dfrac{B-R}{\mathrm{Max}-\mathrm{Min}}+120,\mathrm{Max}=G\\60\times\dfrac{R-G}{\mathrm{Max}-\mathrm{Min}}+240,\mathrm{Max}=B\end{cases}\tag{8}$$

其中：Max＝max（R，G，B），Min＝min（R，G，B），由于分子的减法会存在负数的情况，所以使用补码进行减法运算。为节约硬件资源，将 Divider IP 核配置为无符号除法器，这就需要求一个补码的绝对值。当补码的符号位为 0 时，绝对值为其自身。当补码符号位为 1 时，可通过 2^n-complement 进行计算，其中 n 为补码的位宽。下一小节将通过具体图像进行硬件仿真，分别通过 MSE（mean squared error）、PSNR（peak signal to noise rate）、SSIM（structural similarity）等评价指标对 3 种方式生成的 HSI 进行对比分析。

1.4　硬件仿真对比结果

本节将通过由软件转换后的 HSI 和 FPGA 硬件转换后的 HSI 进行对比，分析上述各种转换方法的差异。如图 2 所示，其中图 a、b、c 分别为 MATLAB 使用几何推导算法转化后的 H、S、I 分量图，图 d、e、f、g、h 为通过 Modeisim 硬件仿真生成的 HSI 分量图。其中图 d、g 为几何推导算法 H 分量的转换结果，它们区别为图 d 中 arccos（x）的 x 量化精度为 0.003 92，而图 g 中 arccos（x）的 x 量化精度为 0.000 244 2。图 e、f 分别为几何推导算法 S、I 分量的转换结果，图 h 为分段定义算法 H 分量转换结果。

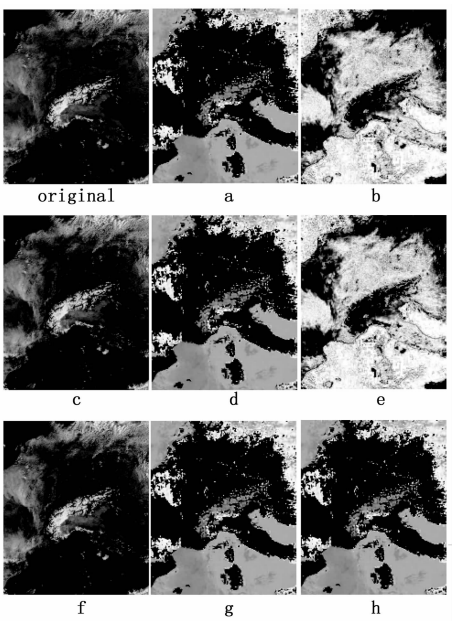

图 2　HSI 对比图

为了分析硬件转换的精度，分别计算软件与硬件转换结果之间的 MSE、PSNR、SSIM[15-16]，如表 1 所示，其中 H^1 为几何推导算法（arccos（x）中 x 的量化精度为 0.003 92）的色度，H^2 为几何推导算法（arccos（x）中 x 的量化精度为 0.000 244 2）的色度，H^3 为分段定义算法转换的色度，S、I 为几何推导法转换的饱和度和亮度。对比 3 个 H 分量，分段定义算法转换较为精确，结构相似性可达到 99.9%。而几何推导算法即使是将 arccos（x）的 x 量化精度提升至 0.000 244 2，相应的结构相似性仅仅从 99.68%提升至 99.72%，未达到分段定义算法的精度，不仅如此，几何推导法在计算 H 分量时，进行了复杂函数（开平方和反三角函数）的计算，这些会消耗大量的硬件逻辑资源，而分段定义法最复杂的计算为除法运算，所需硬件逻辑资源较少。分段定义算法 H 分量结构相似性达到 99.9%，几何推导算法 S、I 分量结构相似性均达到了 99.9%，因此本文采用分段定义法计算 H 分量，几何推导法计算 S 分量和 I 分量。

表 1　各评价标准对比

颜色分量	MSE	PSNR	SSIM
H^1	$5.086\,36\times10^{-5}$	42.935 93	0.996 80
H^2	$3.827\,59\times10^{-5}$	44.170 75	0.997 16
H^3	$4.969\,24\times10^{-6}$	53.037 10	0.999 38
S	$7.483\,82\times10^{-6}$	51.258 77	0.999 41
I	$5.140\,57\times10^{-6}$	52.889 89	0.999 39

2　滤波窗口和排序模块

本节将介绍 3×3、5×5 和 7×7 滤波窗口的获取，以及窗口内最小值、最大值和中值的获取，并将其在硬件上实现。

2.1　自适应滤波窗口的生成

本系统自适应中值滤波支持最大滤波窗口为 7×7，为实现实时图像处理，此模块直接生成 7×7 的滤波窗口，3×3、5×5 滤波窗口可直接从 7×7 的滤波窗口得到。为保护图像细节，我们采用扩展邻近像素点的方式对边界进行填充。

我们采用 RAM 存储图像前 6 行的数据，而输入的数据作为第 7 行，这样可以节省一行 RAM 硬件资源。由于图像分辨率为 400×480，HSI 分量需要 25 bit 来存储，其中 H 为 9 bit、S 为 8 bit、I 为 8 bit，因此我们需要调用 6 个位宽为 25 bit、存储深度为 400 的 Simple Dual Port RAM。RAM 存储时序如图 3 所示，由于 RAM 读取数据需要一个时钟周期，所以要将输入像素数据延时一个时钟周期，延时后的像素数据与 6 个 RAM 读出的数据构成一个“七行一列”的像素数据。为防止 RAM 读写冲突，输入的像素数据在 RAM1 读出数据后写入 RAM1 中，相应的 RAM1 读出的数据在 RAM2 读出数据后写入 RAM2 中，以此类推。连续缓存 7 个“七行一列”的像素数据就可以生成 7×7 的滤波窗口。因为需要等 3 行像素数据的缓存，所以第一个 7×7 的滤波窗口与第一个像素相差 1 200 个有效像素时钟。

2.2　排序模块

FPGA 是以并行计算为主，通过硬件描述语言实现电路的映射，与单片机顺序操作有很大区别。FPGA 实现算

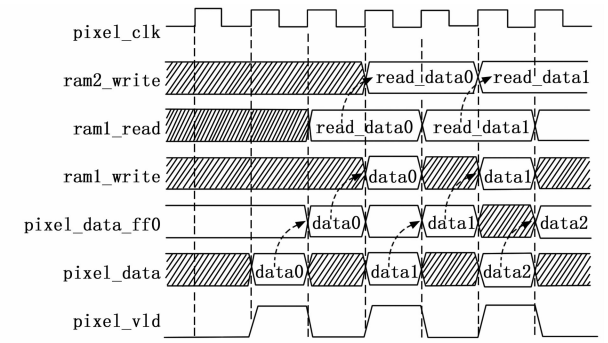

图 3 RAM 存储时序

法通过触发器和逻辑门电路组合形成电路，更加接近硬件底层的实现，其可在一个时钟周期内完成多个任务，达到并行加速的目的[17]。由于要达到处理实时图像的目的，比较矩阵排序器[18]可以满足实时的要求，该排序器原理为将每一个元素与序列中所有的元素进行比较，大于被比较元素则得分为 1，否则得分为 0，当所有元素比较完之后，将所得分数进行累加，获得该元素在序列中最终索引值[18]。具体实现步骤为，将序列中的所有元素分别作为一个矩阵的行值和列值，除去对角线上的元素，将每列值分别与每个行值进行比较，并记录每对行列值的比较情况，比较完成后，对矩阵每一行的所得分进行相加，即可得到该行对应元素的最终索引值，但是随着输入元素的增加，FPGA 硬件逻辑资源的消耗呈指数增加[18]。由于本文最大输入元素为 49，故本文将 Priyadarshan Kolte 和 Roger Smith 等人提出的快速中值滤波算法[19]与上述算法相结合，采用流水线方式设计，最大时延为 9 个时钟周期。下面举例说明该算法排序过程。

假设有一待排序数组 {10，10，50，40，30}，设 D0＝10、D1＝10、D2＝50、D3＝40、D4＝30，考虑到数组中有相同元素，而且还要进行累加运算。相同元素排序按照原数据谁在前谁优先的原则，即在每个数据与其它数据比较时，根据数据在原数组中的位置，比较器的类型要发生变化[20]。例如 Dm 与 Dn 比较时，如果 $m>n$，则选用“≥”比较器，如果 $m<n$，则选择“＞”比较器，这样可以避免数组中存在相同元素时的排序问题。排序过程如表 2 所示，累加值为 4、2、0 分别为该数组的最大值、中值、最小值。用时序逻辑实现数据间的比较和输出，组合逻辑实现得分的累加，需两个时钟周期完成数据排序。

表 2 数据比较结果表

	D0	D1	D2	D3	D4	累加值
D0(10)	×	0	0	0	0	0
D1(10)	1	×	0	0	0	1
D2(50)	1	1	×	1	1	4
D3(40)	1	1	0	×	1	3
D4(30)	1	1	0	0	×	2

注：对角线为自身元素，不进行比较

对于 7×7 滤波窗口计算过程如图 4 所示（箭头所指的方向为升序方向）。

（1）将 7×7 滤波窗口中每列像素按照箭头方向做升序排列；

（2）在（1）的基础上将每行像素按照箭头方向做升序排列，则像素点 11 为最小值像素，像素点 77 为最大值像素；

（3）在（2）的基础上把 45 度对角线方向上的 5 列像素按箭头方向做升序排列；

（4）在（3）的基础上，将像素点 27、35、43、51（“＋”所在位置的像素点），像素点 36、44、52（“－”所在位置的像素点），像素点 37、45、53、61（“＊”所在位置的像素点），按照箭头方向做升序排列，形成新的 7×7 滤波窗口；

（5）在（4）处理后的 7×7 滤波窗口中，取像素点 37、44、51 的中值，即为 7×7 滤波窗口内的中值。

其中 sort5、sort6、sort7 采用上述的比较矩阵排序法（sortx 表示对 x 个元素进行排序），排序需要 2 个时钟周期。sort3 直接采用逻辑门实现，处理需要 1 个时钟周期。sort4 则通过调用 sort3，然后将第 4 个元素插入到合适位置，处理需要 2 个时钟周期。由于 3×3、5×5 窗口排序与 7×7 窗口类似，此处不再赘述。

3 自适应中值滤波

自适应中值滤波算法是根据不同噪声浓度来自适应地调节滤波窗口[21]，通过滤波窗口内的极值点来判别噪声与信号。此算法实现需要两个进程，将其中变量表示为：M_{max} 表示 M_{xy} 允许的最大滤波窗口，I_{min} 表示 M_{xy} 中的最小亮度值，I_{max} 表示 M_{xy} 中的最大亮度值，I_{med} 表示 M_{xy} 中亮度值的中值，I_{xy} 表示坐标（x，y）处的亮度值。

自适应中值滤波两个进程如下：

（1）进程 A，$A_1=I_{med}-I_{min}$，$A_2=I_{med}-I_{max}$，如果 A_1

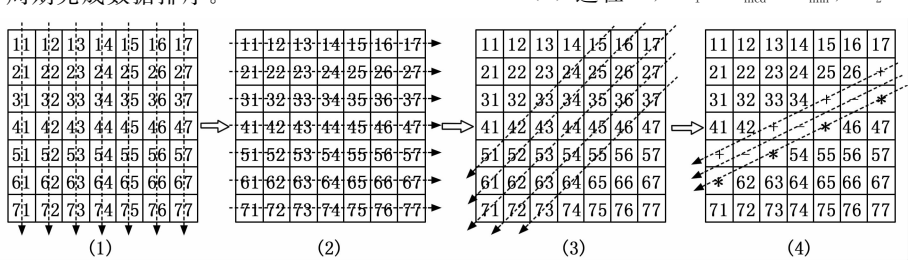

图 4 7×7 窗口中值获取流程

>0 且 $A_2<0$，则转到进程 B 判断原像素点是否为噪声点，否则增大窗口尺寸，如果当前窗口尺寸$\leqslant M_{max}$，则重复进程 A，否则输出 I_{med}。

(2) 进程 B，$B_1=I_{xy}-I_{min}$，$B_2=I_{xy}-I_{max}$，如果 $B_1>0$ 且 $B_2<0$，则输出 I_{xy}，否则输出 I_{med}。

传统自适应中值滤波算法由于只将局部极值点作为噪声的判定依据，所以此过程很容易造成噪声的误判。由于椒盐噪声集中在 HSI 彩色模型的亮度轴上，其饱和度是接近于 0 的，故本文在传统自适应中值滤波算法的基础上，加入 S 分量进行噪声的判别，可以减少图像细节的丢失。进程 A 不变，进程 B 调整为：如果 $B_1=0$ 或 $B_2=0$ 并且 S 接近 0，则输出 I_{med}，否则输出 I_{xy}。

4　HSI-RGB 模块

本模块的输入为滤波后 HSI 分量，目的是实现彩色图像的还原。本节包括了 HSI-RGB 的转换公式，以及在硬件上的具体实现，同时结合 RGB-HSI 模块，将一幅图片通过 RGB-HSI 和 HSI-RGB 之后，与原图像进行对比，分析其还原精度以及硬件逻辑资源消耗情况。

4.1　HSI-RGB 转换公式

公式的选取取决于 H 的值，在原色分割中有 3 个相隔 120° 的扇区（见图 1）。当 H 的值在 RG 扇区时，即 $0\leqslant H<120$，RGB 分量由以下公式[22]给出：

$$B=I\times\frac{255-S}{255} \tag{9}$$

$$R=I\times\left[255+\frac{S\cos H}{\cos(60-H)}\right]\times\frac{1}{255} \tag{10}$$

$$G=3I-(R+B) \tag{11}$$

H 值在 GB 扇区时，即 $120\leqslant H<240$，RGB 分量由以下公式[22]给出：

$$H=H-120 \tag{12}$$

$$R=I\times\frac{255-S}{255} \tag{13}$$

$$G=I\times\left[255+\frac{S\cos H}{\cos(60-H)}\right]\times\frac{1}{255} \tag{14}$$

$$B=3I-(R+G) \tag{15}$$

H 值在 BR 扇区时，即 $240\leqslant H<360$，RGB 分量由以下公式[22]给出：

$$H=H-240 \tag{16}$$

$$G=I\times\frac{255-S}{255} \tag{17}$$

$$B=I\times\left[255+\frac{S\cos H}{\cos(60-H)}\right]\times\frac{1}{255} \tag{18}$$

$$R=3I-(B+G) \tag{19}$$

上述的公式具体实现将在下节详细讨论。

4.2　硬件实现

为减少硬件逻辑资源的消耗，可添加扇区标志位，只需实现如下公式即可：

$$a=I\times\frac{255-S}{255} \tag{20}$$

$$b=I\times\left[255+\frac{S\cos H}{\cos(60-H)}\right]\times\frac{1}{255} \tag{21}$$

$$c=3I-(a+b) \tag{22}$$

通过扇区标志位，根据不同扇区公式将结果分别赋值给 RGB 即可。对于三角函数的计算，我们采用 1.2 节中 ROM 查找表的方式实现。与之不同的是，由于分子分母同时存在余弦函数，可以使用串行方式分别计算两个余弦函数，但是这样会使输出多延时两个时钟周期，所以我们采用同时计算的方式。通过使用 Dual Port ROM，这样不仅可以同时完成两个余弦函数的计算，而且所用的 ROM 资源与计算单个余弦函数相同，可节省一半的 ROM 资源。可以利用余弦函数的性质，仅仅需要计算［0，90］即可，将计算结果量化到［0，255］，这样只需要一个存储宽度为 8 bit、存储深度为 91 的 ROM 即可。下节将 1.4 节中生成的 HSI 分量通过本模块进行图像的还原，并与原图像进行比较，分析其还原准确度。

4.3　还原精度分析

本节将 1.4 节中生成的 HSI 图像通过硬件进行还原，将生成图像与原图进行对比，进一步说明分段定义算法和几何推导算法相结合的转换精度。如图 5 所示，其中图 a 为原始图像，图 b（arccos（x）中 x 的量化精度为 0.003 92）和图 c（arccos（x）中 x 的量化精度为 0.000 244 2）为经过几何推导算法转换后复原的图像，图 d 为经过几何推导算法和分段定义算法相结合转换后复原的图像，从视觉上难以分辨差别。我们将从 MSE、PSNR、SSIM 分析复原图像与原始图像之间的差别。

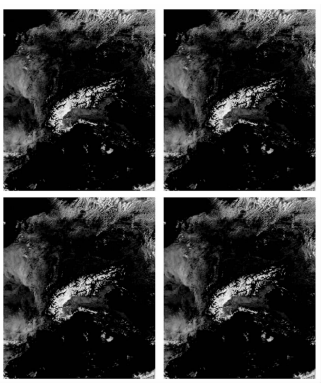

图 5　原图与复原的图像

如表 3 所示，c 图的还原精度较 b 图而言有所提升，但是这种提升是以牺牲较多逻辑资源为代价得到的，结构相似度提高了大约 0.7%。相比之下，d 图的结构相似度为 99.859%，比 b 图提升了 2.872%，比 c 图提升了 2.254%。d 图的峰值信噪比为 46.87，比 b 图高 9.6，比 c 图高 8.38。不仅如此，几何推导算法和分段定义算法相结合所用硬件逻辑资源比几何推导算法少的多，具体如表 3 所示，其中 LUT 减少了 37.2%，LUTRAM 减少了 82%，FF 减少了 59%，DSP 减少了 100%。进一步说明了该方法不仅转换精度高，而且硬件逻辑资源消耗较少，适合在 FPGA 硬件中实现。

表 3 复原图像质量以及所用硬件资源

	评价指标			硬件逻辑资源的消耗					
	MSE	PSNR	SSIM	LUT	LUT RAM	FF	BRAM	DSP	BUFG
b	$1.873\,71\times10^{-4}$	37.272 96	0.969 87	891	37	1237	0.5	2	1
c	$1.414\,92\times10^{-4}$	38.492 68	0.976 05	924	39	1255	0	2	1
d	$2.055\,33\times10^{-5}$	46.871 19	0.998 59	580	7	515	0	0	1

5 系统性能分析

系统结构框图如图 6 所示，RGB 格式数据先经过 RGB2HSI 模块，将 RGB 格式转化为 HSI 格式，处理所需 14 个时钟周期，其次通过滤波窗口生成模块，处理所需 1 200个（以 400×480 分辨率为例，延时 3 行像素数据 400×3=1 200）时钟周期，接着经过比较排序模块，处理所需 9 个时钟周期，然后通过自适应中值滤波模块，处理所需 1 个时钟周期，最后将处理后的 HSI 数据通过 HSI2RGB 模块将 HSI 格式转化为 RGB 格式，完成对原彩色图像的还原，处理所需 27 个时钟周期。由于多个模块采用面积换取速度的思想，使用流水线方式实现各模块，故系统处理时间达到了 1 251 个时钟周期，但这也使得系统支持时钟频率更高。以 Xilinx 公司的 xc7a100tfgg484-2 芯片为例，当系统时钟为 222 MHz 时，Worst Negative Slack（WNS）为 0.025 ns，Worst Hold Slack（WHS）为 0.043 ns，Worst Pulse Width Slack（WPWS）为 1.446 ns，所有的端点都满足时序要求，该系统可以稳定运行。系统最高可对 2 k（2 560×1 440）@60 Hz 视频进行实时处理，所需像素时钟为 2 560（1 440×60=221 MHz，满足系统要求，此时系统时延为 14+2 560×3+9+1+27=7 731 个时钟周期（1×10^{-6}/222 s），处理时延为 347 90 ns。

通过 MATLAB 将原图像分别添加不同概率的椒盐噪声，分别将不同概率噪声的图像通过硬件仿真，观察其降噪效果。硬件仿真通过 Modelsim 进行仿真，在激励文件中，按照 bmp 文件格式将图片数据读入，并且按照 RGB 时序输入到系统中，将处理后的图像数据以 bmp 文件的格式写入。如图 7 所示，第一行为添加不同概率椒盐噪声的原图像，第二行图为 YCbCr 彩色空间自适应中值滤波后的图像，第三行为 HSI 彩色空间自适应中值滤波后的图像。椒盐噪声的概率分别为 20%、30%、40%、50%、60%、70%。从中可得出，HSI 彩色空间自适应滤波后的图像色彩较为鲜艳，色彩还原度好。当噪声概率小于 60%时，降噪后的图像细节保留完整，且无明显的噪声点，当噪声达到 70%时，降噪后的图像出现明显的噪声点。

为了对比不同噪声密度降噪效果，分别计算两种色彩空间滤波后图像与原图像之间的 MSE、PSNR、SSIM。如图 8 所示，当椒盐噪声概率达到 70%时，降噪效果明显下降，HSI 色彩空间降噪后图像与原图的结构相似度为 76.7%。在噪声概率小于 60%时，HSI 色彩空间降噪后图像与原图的结构相似度均可达到 90%以上，而 YCbCr 色彩空间降噪后图像与原图的结构相似度仅达到 80%以上。当噪声概率为 10%时，HSI 色彩空间降噪后图像与原图的结构相似度可达到 99%，而 YCbCr 仅为 86%。当噪声概率小于 90%时，HSI 彩色空间的各项指标（MSE、PSNR、SSIM）均优于 YCbCr 彩色空间。HSI 彩色空间降噪后图像细节保留完好，未出现图像模糊，达到了在滤除噪声的同时减少细节损失的目的。如表 4，为 2 种不同的彩色空间（YCbCr 和 HSI）采用相同自适应滤波所消耗的硬件资源，由于 HSI 为非线性转换，而 YCbCr 为线性转换，故 HSI 所消耗硬件资源会更多。但是 HSI 彩色空间滤波效果提升较大，很好地保留了图像细节，对后续的图像处理影响较小。

表 4 消耗硬件资源对比

色彩空间	FPGA 硬件资源			
	LUT	LUTRAM	FF	BRAM
YCbCr	18 437	44	10 903	3
HSI	19 812	78	12 802	3.50

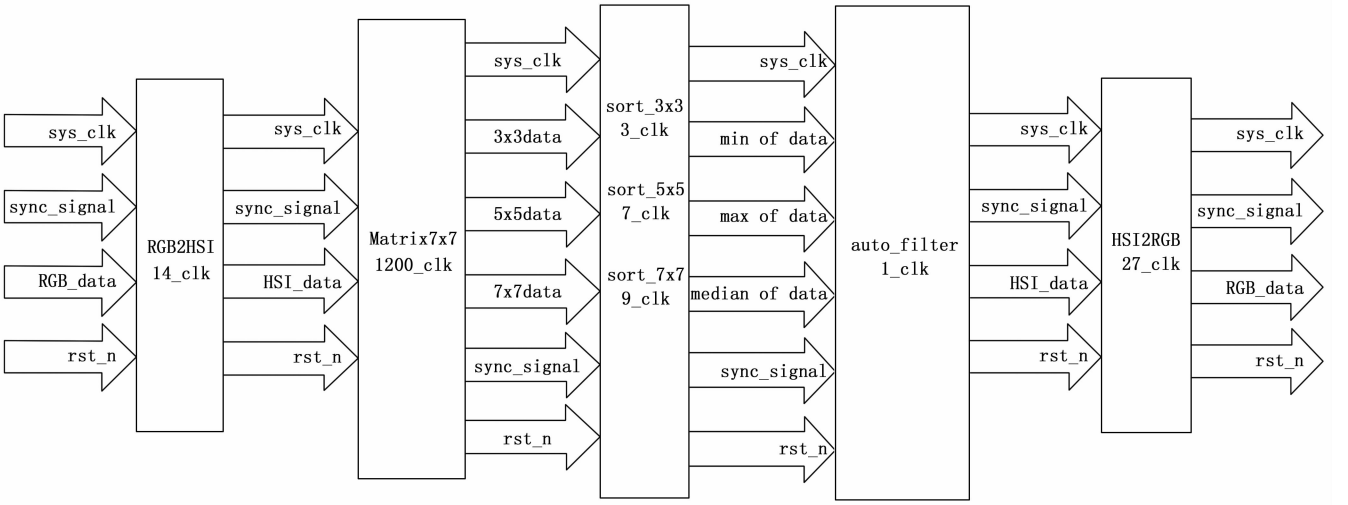

图 6 系统结构框图

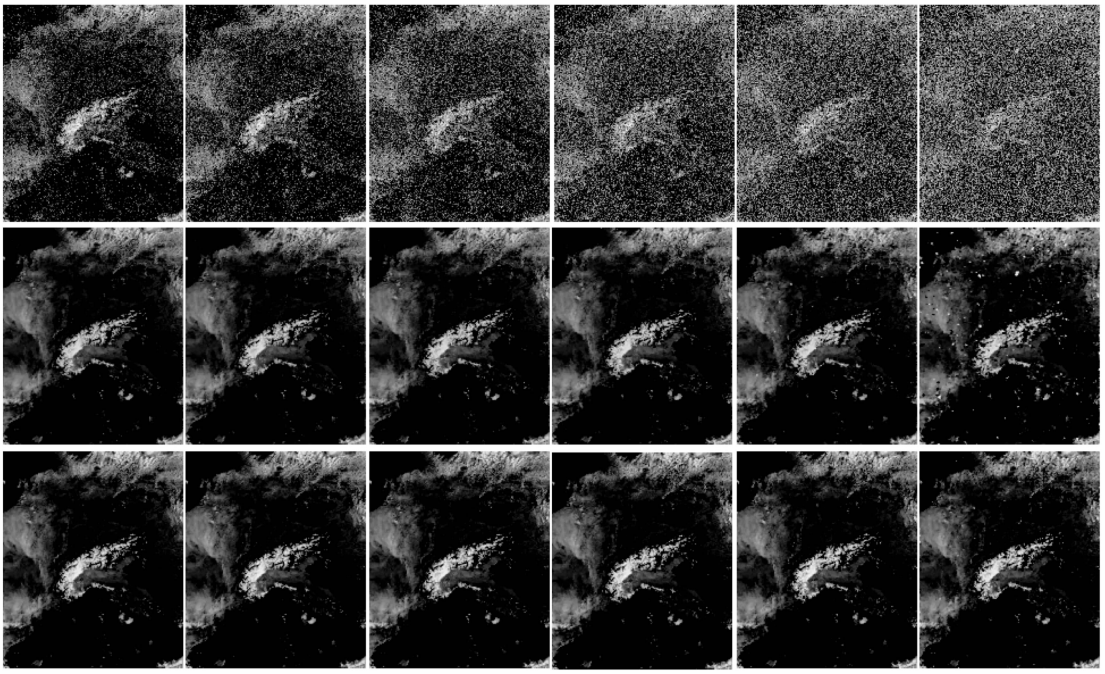

图 7　降噪前后图像对比

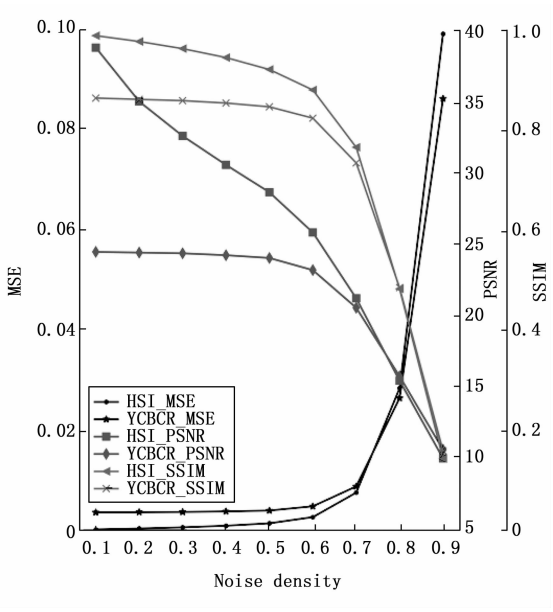

图 8　滤波后图像 MSE、PSNR、SSIM

6　结束语

实时彩色图像自适应中值滤波的 FPGA 实现具有重大的实践意义和研究价值。本文在深入研究 HSI 彩色空间的基础上，结合两种不同的 RGB-HSI 转换算法，在硬件上达到 RGB-HSI 高精度的转换与还原。结合并行排序算法和快速中值滤波算法，在减少硬件资源消耗的同时，完成对 7×7 窗口内元素的快速排序。该系统在硬件资源、精度和速度等方面满足 FPGA 的设计理念，可广泛应用于光学条纹图像的相位分析、智能监控等领域。

参考文献：

[1] 周林鑫，余飞鸿. 图像降噪算法的 FPGA 实现方法 [J]. 电子测量技术，2020，43 (24)：135－140.

[2] 帅慕蓉，廖秀英，程　辉，等. 一种改进均值的自适应中值滤波算法 [J]. 测绘通报，2019 (3)：53－56，90.

[3] 陈家益，战荫伟，曹会英，等. 消除椒盐噪声的基于纹理特征的决策滤波 [J]. 电子测量与仪器学报，2019，33 (3)：126－135.

[4] OLIINYK V，IEREMEIEV O，DJUROVIC I. Center Weighted Median Filter Application to Time Delay Estimation in Non-Gaussian Noise Environment [C] // Ukraine Conference on Electrical and Computer Engineering (UKRCON)，2019：985－989.

[5] VISHAGA S，DAS S L. A survey on switching median filters for impulse noise removal [C] // ICCPCT，2015：1－6.

[6] HWANG H，HADDAD R A. Adaptive median filters：new algorithms and results [C] // IEEE Transactions on Image Processing，1995：499－502.

[7] 马　祥，杜忠华，蔡　雨，等. 融合梯度信息的改进中值滤波算法研究 [J]. 传感器与微系统，2021，40 (3)：48－51.

[8] 唐　超，左文涛，李小飞. 结合修剪均值与高斯加权中值滤波的图像去噪算法 [J]. 计算机工程，2021，47 (9)：210－216.

[9] 王　拓，王洪雁，裴炳南. 一种消除椒盐噪声的迭代自适应中值滤波算法 [J]. 电光与控制，2019，26 (2)：23－27.

[10] 邓廷权，董天祯，谢　巍，等. 自适应中心加权的彩色图像中值滤波方法 [J]. 控制与决策，2013，28 (9)：1372－1376.

[11] JIN S，KIM D，NGUYEN T，KIM D，et al. Design and Implementation of a Pipelined Datapath for High-Speed Face Detection Using FPGA [J]. IEEE Transactions on Industrial Informatics，2012，8 (1)：158－67.

[12] 朱 捷，朱小娟，贺 明. 基于 FPGA 的实时彩色图像中值滤波设计 [J]. 计算机测量与控制，2007 (6)：798－800.
[13] 胡越黎，计惠杰，吴 频，等. 图像的中值滤波算法及其 FPGA 实现 [J]. 计算机测量与控制，2008 (11)：1672－1675.
[14] 丁博文. 基于 FPGA 的一种色空间转换算法的设计与实现 [D]. 南京：东南大学，2017.
[15] HORE A，ZIOU D. Image quality metrics：PSNR vs. SSIM [C] //20th International Conference on Pattern Recognition (ICPR)，2010：23－26.
[16] DE ROSAL IGANTIUS MOSES SETIADI. PSNR vs SSIM：imperceptibility quality assessment for image steganography [J]. Multimedia Tools and Applications，2021，80 (6)：1－22.
[17] 郭诚欣，陈 红，孙 辉，等. 基于现代硬件的并行内存排序方法综述 [J]. 计算机学报，2017，40 (9)：2070－2092.
[18] 吕伟新，李清清，娄俊岭. FPGA 比较矩阵排序法及在中值滤波器中的应用 [J]. 电子器件，2012，35 (1)：34－38.
[19] KOLTE P，SMITH R，SU W. A fast median filter using AltiVec [C] // IEEE International Conference on Computer Design. 1999：384－391.
[20] 师廷伟，金长江. 基于 FPGA 的并行全比较排序算法 [J]. 数字技术与应用，2013 (10)：126－127.
[21] TANG J，WANG Y，CAO W，et al. Improved Adaptive Median Filtering for Structured Light Image Denoising [C] // International Conference on Information，Communication and Networks (ICICN)，2019：146－149.
[22] 阮秋琦. 数字图像处理 (第三版) [M]. 北京：电子工业出版社，2017：259－261.

(上接第 172 页)

复杂场景的解耦合和高并发性能。使用了新增的自整定 PID 算法，适应性好，解决了控制系统快速性、小超调量和小误差性之间的矛盾，更适合复杂航天器的真空外热流试验系统。

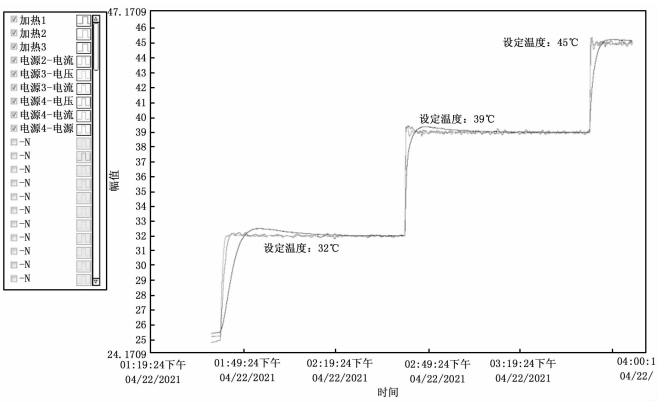

图 15 阶梯控温试验结果

5 结束语

基于虚拟仪器的航天器外热流系统采用模块化设计，通过 LXI 总线集成数字万用表、矩阵开关和程控电源等设备，结合 LabVIEW 环境开发软件，实现了大通道温度采集和电源智能控制，使用改善的软件框架和故障处理机制，稳定安全地在真空环境下模拟了航天器表面接收的外热流；引进自整定控制算法，大幅提升试验效率，提高控温精度。该系统可实现多种复杂航天器的真空外热流试验，节约人力成本，具有使用方便灵活和系统稳定性强等优点。

参考文献：

[1] 孙日明，刘吴月，张荣春. 分布式航天器真空热试验测控系统设计 [J]. 计算机测量与控制，2015，23 (10)：3270－3273.
[2] 韩智慧，王生生，赵建华. 真空热试验通用软件平台工作模式与关键技术 [J]. 计算机测量与控制，2015，23 (8)：2932－2935.
[3] 周武能，田林林. 基于模糊 PID 算法的无线分布式温度控制系统 [J]. 控制工程，2014，21 (3)：3112－3113.
[4] 张景川，谢吉慧，王奕荣，等. 航天器真空热试验测控系统应用现状及发展趋势 [J]. 航天器环境工程，2012，29 (3)：263－267.
[5] 冯 尧，郄殿福，谢吉慧，等. 航天器真空热试验测控系统软件结构设计与建模 [J]. 航天器环境工程，2010，27 (5)：616－620.
[6] 宋智罡，郁其祥，王益明，等. 基于 LabVIEW 的 PID 参数自适应模糊控制器设计 [J]. 机械设计与制造，2003，21 (4)：11－13.
[7] 孙 娜，刘晓博，李江忠. LabVIEW 和 MATLAB 混合编程在 PID 参数自整定算法仿真中的应用 [J]. 江苏电器，2006，12 (5)：26－29.
[8] 李 宁，郑艺华. 基于 LabVIEW 的 PID 自整定温控系统研究 [J]. 青岛大学学报 (工程技术版)，2019，34 (2)：27－32.
[9] 关艳翠，赵硕伟，杨振元. 基于 LabVIEW 的自整定 PID 控制器设计与分析 [J]. 山东工业技术，2017，15 (2)：217－219.
[10] 梁九生，吴清文，黄 涛，等. 基于 LabVIEW 和 HP VEE 的程控电源系统设计 [J]. 测控技术，2009，21 (9)：37－40.
[11] 顾 苗，刘晓雷，李 娜，等. LabView 平台下电源测控系统的实现 [J]. 装备环境工程，2012，9 (3)：23－26.
[12] 徐 昕，等. LXI 总线接口的设计及其在军用测试系统中的应用 [J]. 计算机测量与控制，2009，17 (10)：1996－2000.
[14] 郭 赣. 真空热试验的温度测量系统 [J]. 航天器环境工程，2009，26 (1)：33－36.
[15] 孙兴华，韩 放，裴一飞. 真空热试验热响应测试程序设计 [J]. 航天器环境工程，2010，27 (6)：720－722.
[16] 顾 苗，刘劲松. 真空热试验中闭环温度控制参数分析 [J]. 航天器环境工程，2010，27 (5)：611－615.
[17] 孙 宇，刘高同，顾志飞，等. 热真空试验中的自校正 PID 控制策略 [J]. 航天器环境工程，2016，33 (3)：333－336.
[18] 杨 帆，徐皓吉，蒋晓峰. 一种基于 LabVIEW 的数据存储与回放技术 [J]. 测控技术，2017，36 (7)：124－127.
[19] 董延军，李兴生. LabVIEW 多核编程技术在局域网文件传输中的应用 [J]. 测控技术，2014，33 (9)：113－115.
[20] 冀常鹏，孙 巍. 变论域自适应模糊 PID 控制系统仿真与应用 [J]. 测控技术，2018，37 (10)：119－123.
[21] 张文霞，钱祥忠. 基于模糊 PID 控制的双向 DC/DC 变换器研究 [J]. 测控技术，2015，34 (12)：74－77.