Received September 9, 2020, accepted September 19, 2020, date of publication September 23, 2020, date of current version October 2, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3026189

An FPGA-Based Simple RGB-HSI Space Conversion Algorithm for Hardware Image Processing

SHUAIQING ZHI[1], YANI CUI[1], JIAXIAN DENG[1], AND WENCAI DU[2]
[1]College of Information and Communication Engineering, Hainan University, Haikou 570228, China
[2]Institute of Data Science, City University of Macau, Taipa 999078, China

Corresponding author: Yani Cui (cyn0213@163.com)

This work was supported in part by the Hainan Provincial Key Research and Development Plan under Grant ZDYF2019014, in part by the National Natural Science Foundation of China and Macau Science and Technology Development Joint Fund under Grant 61961160706 and Grant 0066/2019/AFJ, in part by the Hainan Association for Science and Technology Plans to Youth Research and Development Innovation under Grant QCXM201912, and in part by the Program of and the Scientific Research Foundation of Hainan University under Grant KYQD(ZR)1859.

ABSTRACT In this paper, a new, low-complexity, easy-to-implement hardware method for color space conversion between the red-green-blue (RGB) and the hue-saturation-intensity (HSI) color spaces called the simple RGB-HSI space conversion (S-SC) algorithm is proposed, which aims to provide more rapid computing due to the need for fewer operations. In the S-SC algorithm, we reconstruct the model of space conversion between the RGB color space and the HSI color space (RGB-HSI) by inverting the conversion from the HSI color space to the RGB color space of the traditional geometric derivation algorithm. As a result, the nonlinear model-realized RGB-HSI color space conversion by the geometric derivation algorithm is transformed into a linear conversion model, which can avoid complicated calculations such as trigonometric and inverse trigonometric functions in the color space conversion process. The model can effectively reduce the computational complexity of the algorithm and facilitate hardware implementation at the same time. To evaluate the performance of the S-SC algorithm, we first compare the S-SC algorithm with the geometric derivation algorithm from the computational complexity perspective. On this basis, we compare the S-SC algorithm with five other RGB-HSI color space conversion algorithms from the perspectives of error and conversion effect. Finally, we use the field programmable gate array (FPGA) hardware platform to analyze and verify the timing sequence and logical resource consumption and verify the effectiveness of the proposed algorithm with experimental results. We show that the S-SC algorithm achieves good performance in terms of conversion accuracy, logical unit resource occupancy, and output timing.

INDEX TERMS RGB-HSI space conversion, FPGA implementation, low computational complexity, low logic resource consumption.

I. INTRODUCTION

Color spaces are essential in research fields such as image processing and computer vision. Frequently used color space types include the red-green-blue (RGB) color space, hue-saturation-intensity (HSI) color space, hue-saturation-value (HSV) color space, and Luma-blue-difference-Chroma-red-difference-Chroma (YCbCr) color space. Attributed to a color description based on the additive color principle and facilitating color data collection and display, the RGB color space is the most basic, commonly used, and hardware-oriented space. However, since the red, green, and blue components of the RGB color space are not independent of each other, interference between them occurs during image processing. This circumstance is disadvantageous to image processing [1]–[3]. The HSI color space, based on the human visual system, is used for color description, which is convenient for researchers to process color information from subjective consciousness. Moreover, since the HSI color space components are independent of each other and can be

The associate editor coordinating the review of this manuscript and approving it for publication was Liang-Bi Chen.

processed separately, the HSI space has been widely utilized in image processing [4]–[6]. In addition, relevant studies have also proven that compared with other color spaces such as HSV and YCbCr, the HSI color space is the most suitable for visible light image processing with the best processing effect [7].

Due to the reasons mentioned above, researchers have done much work on visible light image processing based on the HSI color space. A fast haze removal method based on the polarimetric imaging technique by combining the HSI color space and retinex theory was proposed in [8]. The method in this reference could recover the information data in an image with a fast process. In addition, the Gaussian function was utilized in [9] to fit the intensities of image color blocks based on the HSI color space and reconstruct the image by combining the chrominance information in the HSI color space and the intensity information in the fitting space. The algorithm in [9] possesses practical application value in the digital reconstruction of artworks. A deep convolutional neural network (CNN) model to enhance low-light sensor images based on the HSI color space model was proposed in [10], and the experimental results show that the proposed algorithm significantly enhanced the brightness and contrast of the image and avoided color distortion and over enhancement.

Furthermore, an entropy-based contrast enhancement method based on the HSI color space was proposed in [11]. The author processed the color saturation component while enhancing the image contrast, which made the processed image much smoother. However, since the images captured through a visible light camera are generally under the RGB color space model, the methods mentioned above first utilized the RGB-HSI color space conversion algorithm to realize color space conversion.

There are many algorithms for RGB-HSI color space conversion, among which the geometric derivation algorithm is the most classical and the most widely used RGB-HSI algorithm [12]. However, in the process of RGB-HSI conversion, the geometric derivation algorithm requires nonlinear calculations such as trigonometric and inverse trigonometric functions. These nonlinear calculations result in high algorithmic complexity and low conversion efficiency. For this reason, based on the geometric derivation algorithm, researchers have proposed a variety of improved RGB-HSI conversion algorithms. These include the coordinate transformation method, segmentation definition method, Bajon approximation algorithm, and standard module arithmetic method [13]. Among those algorithms, the geometric derivation algorithm achieves the highest resolution and the highest accuracy of reverse reduction [14]. In addition, researchers have realized relevant color space conversion algorithms based on the field programmable gate array (FPGA) hardware and carried out performance tests in practical application scenarios simultaneously [15], [16]. The RGB-HSI color space conversion based on FPGA hardware was realized in [17]. The experimental results showed that the reconstructed RGB image, obtained by color space conversion of RGB-HSI by utilizing FPGA hardware, was almost the same as the original image. It not only had the advantage of fast parallel data processing but also maintained a high conversion accuracy. However, the method proposed in [17] was not further optimized for the conversion between the RGB color space and the HSI color space. This resulted in a waste of logic resources such as slice registers and Look-Up Tables (LUTs), and LUTs can be regarded as one of the most important resources in FPGA hardware. In [7], based on the Swenson algorithm proposed in [18], the log-hybrid method was used to optimize the trigonometric function in the color space conversion from the HSI color space to the RGB color space, and the algorithm was realized on FPGA hardware. However, the algorithm proposed in [7] utilized digital signal processor (DSP) modules and did not optimize the color space conversion from the RGB color space to the HSI color space, leading to the large consumption of logic resources such as slice LUTs, which was proven by experiments.

According to the above, in the process of RGB-HSI color space conversion, nonlinear calculations frequently exist, such as trigonometric functions and inverse trigonometric functions. These complicated nonlinear calculations, if not optimized, will consume a large number of logic resources that are not abundant in FPGA hardware. To solve this problem, a new, easy-to-implement hardware method for RGB-HSI color space conversion named the simple RGB-HSI space conversion (S-SC) algorithm is proposed in this paper. By reconstructing the mapping relationship between the R, G, and B components in the RGB color space and the H, S, and I components in the HSI color space, the nonlinear calculation in the traditional color space conversion algorithm is avoided, which reduces the computational complexity and improves the conversion efficiency. On this basis, the effectiveness of the proposed algorithm is verified with simulation results.

The main contributions of this paper can be summarized as follows:

- We propose an optimized RGB-HSI color conversion algorithm, called the S-SC algorithm, based on the geometric derivation algorithm. In the S-SC algorithm, complex nonlinear calculations, such as square, square root, cosine, and arc-cosine calculations, can be eliminated, which can reduce calculational complexity and improve the conversion efficiency.
- We study the conversion accuracy and computational complexity of the S-SC algorithm together with the geometric derivation algorithm for comparison.
- We also investigate the performance of the S-SC algorithm on the Lena image process together with five other RGB-HSI color space conversion algorithms for comparison.
- We present the implementation of the S-SC algorithm on the FPGA platform. Moreover, we compare the logic resource consumption of the S-SC algorithm with the algorithms proposed in [17] and [7].

The results indicate that the S-SC algorithm achieves good performance in terms of conversion accuracy, logical unit resource occupancy, and output timing.

The remainder of this paper is organized as follows: the RGB2HSI model and HSI2RGB model of the geometric derivation algorithm and the S-SC algorithm are studied in Section II. The conversion error analysis and computational complexity analysis are performed in Section III and Section IV, respectively. To evaluate the performance of the S-SC algorithm, the processing results of the S-SC algorithm and five other RGB-HSI color space conversion algorithms are presented in Section V. The hardware implementation performance of the S-SC algorithm is presented in Section VI. Finally, the paper is concluded in Section VII.

II. THE DERIVATION OF THE S-SC ALGORITHM

A. THE RGB-HSI CONVERSION OF THE GEOMETRIC DERIVATION ALGORITHM

1) THE CONVERSION FROM THE RGB COLOR SPACE TO THE HSI COLOR SPACE

The essential conversion from the RGB color space to the HSI color space (hereafter written as RGB2HSI) is the conversion from the unit cube of the Cartesian coordinate system in the RGB color space to a cylindrical polar coordinate-based bipyramid model [19]. The process of the RGB2HSI conversion can be decomposed into the following steps:

- Peel off the I component (Intensity) from the RGB color space.
- Reduce the three-dimensional RGB color space to a two-dimensional space.
- Based on the above two points, decompose the chroma into the H component (Hue) and S component (Saturation).
- Calculate the H component through the dot product formula of the vector.
  The models of the RGB color space and HSI color space are shown in Fig. 1.

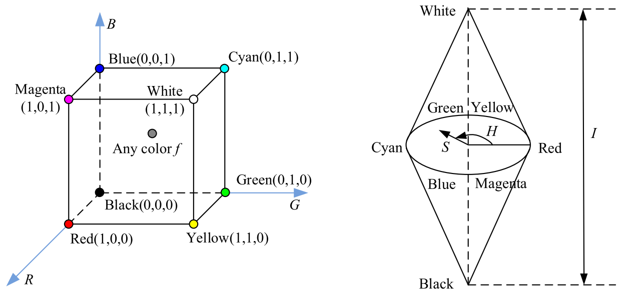
According to [7], the RGB2HSI conversion in the geometric derivation algorithm is written as:

$$\begin{cases} I = \dfrac{R+G+B}{3} \\ S = 1 - \dfrac{3\min(R,G,B)}{R+G+B} \\ H = \begin{cases} \theta & (G \geq B) \\ 2\pi - \theta & (G < B) \end{cases} \end{cases} \tag{1}$$

where $H \in [0,\ 2\pi)$, S and $I \in [0,\ 1)$, and θ is expressed as:

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

According to (1)-(2), R, G, and B represent the red, green, and blue components in the normalized RGB color space, respectively. H, S, and I represent the hue, saturation, and intensity components in the HSI color space, respectively. It can be directly shown from (2) that inverse-trigonometric and square root calculations are required while calculating the parameter θ. The terms in the denominator of (2) can be rewritten as:



(a) Normalized RGB color space model (b) HSI space model

FIGURE 1. The models of the RGB color space and the HSI color space.

$$J = (R-G)^2 + (R-B)(G-B) \tag{3}$$

Since the R, G, and B components are all 8-bit vectors, the range of J is in [0, 65025], and values of J are all integer type. In this case, since the enormous scope of the data would cause the consumption of many logic resources of the FPGA hardware, the lookup table method is truly hard to realize. In addition, when realizing square root calculations on FPGA hardware, it is difficult to use iterative methods to approximate the actual value [20], which leads to high time complexity for the whole system. The pseudocode of the RGB2HSI conversion of the geometric derivation algorithm is shown in Algorithm 1.

Algorithm 1 The RGB2HSI Conversion of the Geometric Derivation Algorithm

```
I: Calculate number of pixels $N$ of the input RAW RGB Image Format.
II: for $i = 0$ to $i < N$ do
III: Separate the $R(i)$, $G(i)$, and $B(i)$ components from an input RGB image.
IV: Search for the minimum value in the $R(i)$, $G(i)$, and $B(i)$ components.
V: Calculate the $H(i)$, $S(i)$, and $I(i)$ components according to (1)-(2).
VI: end for
Output: The $\boldsymbol{H}$, $R'$ $\boldsymbol{S}$, $G'$ and $\boldsymbol{I}$ components.
```

2) THE CONVERSION FROM THE HSI COLOR SPACE TO THE RGB COLOR SPACE

The sector where the H component is located determines the conversion from the HSI color space to the RGB color space (hereafter written as HSI2RGB). The process of the HSI2RGB conversion in the geometrically derived method can be separated into the following three conditions.

- In the case where $H \in [0,\ 2\pi/3)$, the B component is the minimum of the original image pixel, and the formulas for the HSI2RGB conversion can be written as:

$$\begin{cases} B = I\,(1-S) \\ R = I\left(1+\dfrac{S\cos H}{\cos(\pi/3-H)}\right) \\ G = 3I-(B+R) \end{cases} \tag{4}$$

- In the case where $H \in [2\pi/3, 4\pi/3)$, the R component is the minimum of the original image pixel, and the formulas for the HSI2RGB conversion can be written as:

$$\begin{cases} R = I\,(1-S) \\ G = I\left(1+\dfrac{S\cos(H-2\pi/3)}{\cos(\pi-H)}\right) \\ B = 3I-(R+G) \end{cases} \tag{5}$$

Let $H = H - 2\pi/3$(5) can be rewritten as:

$$\begin{cases} R = I\,(1-S) \\ G = I\left(1+\dfrac{S\cos(H)}{\cos(\pi/3-H)}\right) \\ B = 3I-(R+G) \end{cases} \tag{6}$$

- In the case where $H \in [4\pi/3,\ 2\pi)$, the G component is the minimum of the original image pixel, and the formulas for the HSI2RGB conversion can be written as:

$$\begin{cases} G = I\,(1-S) \\ B = I\left(1+\dfrac{S\cos(H-4\pi/3)}{\cos(5\pi/3-H)}\right) \\ R = 3I-(G+B) \end{cases} \tag{7}$$

Let $H = H - 4\pi/3$(7) can be rewritten as:

$$\begin{cases} G = I\,(1-S) \\ B = I\left(1+\dfrac{S\cos(H)}{\cos(\pi/3-H)}\right) \\ R = 3I-(G+B) \end{cases} \tag{8}$$

Although the formulas for the HSI2RGB conversion, as shown in (4)-(8), are determined by the position of the H component, regardless of the sector in which the H component is located, the cosine term must be calculated. If the cosine term cannot be optimized, this part would cost a large amount of logic resources and cause many difficulties in the process of timing sequence optimization during FPGA implementation. The pseudocode of the HSI2RGB conversion of the geometric derivation algorithm is shown in Algorithm 2.

B. THE OPTIMIZATION OF THE RGB-HSI COLOR SPACE CONVERSION IN THE S-SC ALGORITHM

1) THE OPTIMIZATION OF THE H COMPONENT IN RGB2HSI CONVERSION

According to part A, the position of the H component determines the HSI2RGB conversion formulas. The H component is located in the sectors $[0,\ 2\pi/3)$, $[2\pi/3, 4\pi/3)$ and $[4\pi/3,\ 2\pi)$. Further, all the formulas listed in (4)-(8) are similar except for changes in form. In brief, the following

Algorithm 2 The HSI2RGB Conversion of the Geometric Derivation Algorithm

```
I: Input number of pixels $N$ and the $\boldsymbol{H}$, $R'$ $\boldsymbol{S}$, $G'$ and $\boldsymbol{I}$ components.
II: for $i = 0$ to $i < N$ do
III: Calculate the $R(i)$, $G(i)$, and $B(i)$ components according to (4)-(8).
IV: Integrate the $R(i)$, $G(i)$, and $B(i)$ components into pixels value RGB($i$).
V: end for
Output: The recovered RGB image.
```

two expressions must be calculated regardless of the position of the H component:

$$I\,(1-S) \tag{9}$$

and

$$I\left(1+\frac{S\cos H}{\cos(\pi/3-H)}\right) \tag{10}$$

where (10) is the only one that is affected by the H component. In this case, the inverse transformation can be achieved if the term $\cos(H)/\cos(\pi/3-H)$ and the position of the H component are known. To facilitate the FPGA hardware implementation, reduce the computational complexity and improve the efficiency of the algorithm, the expression $\cos(H)/\cos(\pi/3-H)$ is regarded as the whole integral component used to replace the hue information of the image rather than directly utilizing (1) to calculate the H component through the RGB2HSI conversion. Briefly, the H component in the geometric derivation algorithm is replaced by:

$$H' = \frac{\cos H}{\cos(\pi/3-H)} \tag{11}$$

On this basis, the HSI2RGB inverse transformation formula in the geometric derivation algorithm is used to obtain the RGB2HSI transformation formula. The details are as follows.

a) When $H \in [0,\ 2\pi/3)$, the B component is the minimum of the original pixel. We know that $R = I(1+SH')$ based on (4), and then (12) can be elicited.

$$H' = \frac{R-I}{IS} \tag{12}$$

The expressions of the S and I components in (1) can be rewritten as (13)-(14).

$$S = 1-\frac{3}{R+G+B}\min(R,G,B) = \frac{R+G-2B}{R+G+B} \tag{13}$$

$$3I = R+G+B \tag{14}$$

We can obtain (15) by substituting (14) into (13).

$$S = \frac{R+G-2B}{3I} \tag{15}$$

Then, (15) can be rewritten as:

$$3IS = R + G - 2B \tag{16}$$

and (17) can be derived based on (14).

$$3\left(R - I\right) = 2R - G - B \tag{17}$$

Then, (18) can be derived by substituting (16)-(17) into (12).

$$H' = \frac{R - I}{IS} = \frac{2R - G - B}{R + G - 2B} \tag{18}$$

Similarly, the following two conditions can be derived as well.

b) When $H \in [2\pi/3,\ 4\pi/3)$, the R component is the minimum of the original image pixel.

$$H' = \frac{2G - B - R}{G + B - 2R} \tag{19}$$

c) When $H \in [4\pi/3,\ 2\pi)$, the G component is the minimum of the original image pixel.

$$H' = \frac{2B - R - G}{R + B - 2G} \tag{20}$$

According to (18)-(20), the formulas of the S-SC algorithm in the RGB2HSI conversion can be rewritten as (21), and H is replaced by $H'H'$ as the hue component of the image.

$$H' = \begin{cases} \dfrac{2G - B - R}{G + B - 2R} & w = R \\ \dfrac{2B - R - G}{B + R - 2G} & w = G \\ \dfrac{2R - G - B}{R + G - 2B} & w = B \end{cases} \tag{21}$$

where w represents the minimum value among the R, G, and B components.

$$w = \min\left(R, G, B\right) \tag{22}$$

According to 2) in part A, the formulas for the HSI2RGB conversion are determined by the position of the H component. However, according to (6)-(8), utilizing $H'H'$ to replace H as the hue component would cause the loss of sector information. In other words, with the H component changing in the range $[0,\ 2\pi)$, the H' component H' would lose the sector information. In this case, the HSI2RGB conversion cannot be achieved directly, and a necessary consideration for this issue is to construct a model to preserve the sector information. If the number of sectors is equal to three, the sector in which the H component is located is decided by the minimum value among the R, G, and B components. The sector information can be preserved through two-bit binary number encoding as:

$$H_{flag} = \begin{cases} 00 & w = R \\ 01 & w = G \\ 10 & w = B \end{cases} \tag{23}$$

By substituting (23) into (21), the formulas of the hue of the RGB2HSI conversion in the S-SC algorithm can be written as:

$$H' = \begin{cases} \dfrac{2G - B - R}{G + B - 2R} & H_{flag} = 00 \\ \dfrac{2B - R - G}{B + R - 2G} & H_{flag} = 01 \\ \dfrac{2R - G - B}{R + G - 2B} & H_{flag} = 10 \end{cases} \tag{24}$$

As seen in (24), in this way, the nonlinear calculation steps during the RGB2HSI conversion of the geometric derivation algorithm, such as the inverse trigonometric function and squared root function, can be eliminated.

2) THE OPTIMIZATION OF THE I COMPONENT IN THE RGB2HSI CONVERSION

To avoid extra logic resource consumption under the condition of utilizing the divider IP core to calculate the constant fraction 1/3, in the S-SC algorithm, we utilize the approximation method to approach 1/3. Because the constant 4096 can be rewritten as 2^{12}, the constant fraction 1/3 can be approximated by binary shift processing of the integer 1365. In this way, the approach can not only reduce the consumption of logical resources but also reduce the complexity of the timing sequence. The optimized calculation formula of the I component in the RGB2HSI conversion is shown as:

$$I' = \frac{1365}{4096}\left(R + G + B\right) \tag{25}$$

According to (24)-(25), the rest of the optimized formulas of the RGB2HSI conversion can be rewritten as:

$$H' = \begin{cases} \dfrac{2G - B - R}{G + B - 2R} & H_{flag} = 00 \\ \dfrac{2B - R - G}{B + R - 2G} & H_{flag} = 01 \\ \dfrac{2R - G - B}{R + G - 2B} & H_{flag} = 10 \end{cases} \tag{26}$$

$$S' = \begin{cases} \dfrac{G + B - 2R}{R + G + B} & H_{flag} = 00 \\ \dfrac{R + B - 2G}{R + G + B} & H_{flag} = 01 \\ \dfrac{R + G - 2B}{R + G + B} & H_{flag} = 10 \end{cases} \tag{27}$$

The pseudocode of the RGB2HSI conversion in the S-SC algorithm is shown in Algorithm 3.

3) THE OPTIMIZATION OF THE HSI2RGB CONVERSION

According to part 2), the sector information of the H component can be preserved by introducing the encoding variable H_{flag} On this basis, the optimized formulas of the HSI2RGB conversion in the S-SC algorithm are as follows.

- In the case where H_{flag} equals 00, the input R component is the minimum, and the optimized formulas of the

Algorithm 3 The RGB2HSI Conversion of the S-SC Algorithm

```
I: Calculate number of pixels $N$ of the input RAW RGB Image Format.
II: for $i = 0$ to $i < N$ do
III: Separate the $R(i)$, $G(i)$, and $B(i)$ components from an input RGB image.
IV: Search for the minimum value $w(i)$ in the $R(i)$, $G(i)$, and $B(i)$ components.
V: Decide the value of $H_{flag}(i)$ by $w(i)$.
VI: Calculate the $H'(i)$, $S'(i)$, and $I'(i)$ components according to (25)-(27).
VII: end for
Output: $\boldsymbol{H_{flag}}$ and the $\boldsymbol{H'}$, $R'\boldsymbol{S'}$, $G'$ and $\boldsymbol{I'}$ components.
```

HSI2RGB conversion can be written as:

$$\begin{cases} R' = I'\left(1 - S'\right) \\ G' = I'\left(1 + S'H'\right) \\ B' = 3I' - \left(R' + G'\right) \end{cases} \tag{28}$$

- In the case where H_{flag} equals 01, the input G component is the minimum, and the optimized formulas of the HSI2RGB conversion can be written as:

$$\begin{cases} G' = I'\left(1 - S'\right) \\ B' = I'\left(1 + S'H'\right) \\ R' = 3I' - \left(G' + B'\right) \end{cases} \tag{29}$$

- In the case where H_{flag} equals 10, the input B component is the minimum, and the optimized formulas of the HSI2RGB conversion can be written as:

$$\begin{cases} B' = I'\left(1 - S'\right) \\ R' = I'\left(1 + S'H'\right) \\ G' = 3I' - \left(B' + R'\right) \end{cases} \tag{30}$$

It is known from (29)-(31) that the equations for the above three conditions are similar except for the equation of each component needing to be rescheduled according to the value of H_{flag}. To be brief, (31) needs to be calculated regardless of the value of H_{flag}.

$$\begin{cases} f_1 = I'\left(1 - S'\right) \\ f_2 = I'\left(1 + S'H'\right) \\ f_3 = 3I' - (f_1 + f_2) = I'\left(1 + S' - S'H'\right) \end{cases} \tag{31}$$

Therefore, during the HSI2RGB conversion, f_1, f_2, and f_3 can be calculated during the first step based on (25)-(27). The values of the reconstructed $R'R'$, G', G' and B' components of the recovered RGB image can be obtained based on the value of H_{flag} In this way, the nonlinear calculation step of the trigonometric function during the HSI2RGB conversion in the geometric derivation algorithm is no longer needed. The pseudocode of the HSI2RGB conversion of the S-SC algorithm is shown in Algorithm 4.

Algorithm 4 The HSI2RGB Conversion of the S-SC Algorithm

```
I: Input number of pixels $N$, $\boldsymbol{H_{flag}}$, and the $\boldsymbol{H'}$, $R'\boldsymbol{S'}$, $G'$ and $\boldsymbol{I'}$ components.
II: for $i = 0$ to $i < N$ do
III: Calculate the $R'(i)$, $G'(i)$, and $B'(i)$ components according the value of $H_{flag}(i)$ as given by (31).
IV: Integrate the $R'(i)$, $G'(i)$, and $B'(i)$ components into pixels value RGB$(i)$.
V: end for
Output: The recovered RGB image.
```

III. THE ERROR ANALYSIS OF THE S-SC ALGORITHM

Based on Section II, MATLAB 2019a software is used to analyze the error of the S-SC algorithm and the geometric derivation algorithm for the RGB2HSI and HSI2RGB conversions, respectively, and the conversion accuracy is verified according to the experimental results.

The input data of the R, G, and B components are affected by the AD module inside the visible light camera. During the FPGA implementation of the S-SC algorithm, the real output data of the R, G, and B components are all integer data in the range [0x00, 0xFF]. In this case, the input original R, G, and B data of the S-SC algorithm are all 8-bit vectors. In other words, the input original R, G, and B data are all integers in the range $[0, 255]$. In this case, an experimental method is designed and utilized that makes the R, G, and B components traverse all possible input data cases in the range $[0, 255]$ under the condition that satisfies the value of H_{flag}.

A. THE ERROR ANALYSIS OF THE H COMPONENT

1) THE ANALYSIS OF THE H COMPONENT

- In the case where $H \in [0,\ 2\pi/3)$, (32) can be derived by combining (1) and (18).

$$\Delta H = H' - H = \frac{2R - G - B}{R + G - 2B} - \arccos\left(\frac{(R-G) + (R-B)}{2\sqrt{(R-G)^2 + (R-B)(G-B)}}\right) \tag{32}$$

According to (32), the error of the H component in the RGB2HSI conversion can be calculated. Similarly, the error of the H component can be derived in the other two conditions as well.

- In the case where $H \in [2\pi/3,\ 4\pi/3)$.

$$\Delta H = \frac{2G - B - R}{G + B - 2R} - \arccos\left(\frac{(R-G) + (R-B)}{2\sqrt{(R-G)^2 + (R-B)(G-B)}}\right) \tag{33}$$

- In the case where $H \in [4\pi/3,\ 2\pi)$.

$$\Delta H = \frac{2B-R-G}{R+B-2G} - \arccos\left(\frac{(R-G)+(R-B)}{2\sqrt{(R-G)^2+(R-B)(G-B)}}\right) \tag{34}$$

Sequentially, the error calculations in (32)-(34) can be calculated through MATLAB 2019a, and the results are shown in Fig. 2. For the supplementary explanation to Fig. 2, parts (a), (b), and (c) represent the error ΔH_a when $H \in [0,\ 2\pi/3)$, the error ΔH_b when $H \in [2\pi/3,\ 4\pi/3)$, and the error ΔH_c when $H \in [4\pi/3,\ 2\pi)$, respectively.

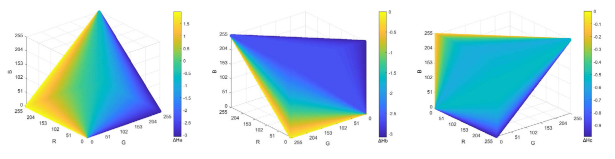

(a) The value of ΔH_a (b) The value of ΔH_b (c) The value of ΔH_c

FIGURE 2. The error ΔH of the H component.

According to part (a) in Fig. 2 and the experimental results, in the case where $H \in [0,\ 2\pi/3)$, the values of ΔH_a are in the range $[-3.0793,\ 1.9849]$ When the input data of the R, G, and B components are equal to 1, 255, and 0, respectively, ΔH_a possesses the minimum value, which equals -3.0793. When the input data of the R, G, and B components are equal to 255, 1, and 0, respectively, ΔH_a possesses the maximum value, which equals 1.9849.

According to part (b) in Fig. 2, in the case where $H \in [2\pi/3,\ 4\pi/3)$, the values of ΔH_b are in the range $[-3.0861,\ -0.1095]$ When the input data of the R, G, and B components are equal to 0, 1, and 255, respectively, ΔH_b possesses the minimum value, which equals -3.0861. When the input data of the R, G, and B components are equal to 0, 255, and 1, respectively, ΔH_b possesses the maximum value, which equals -0.1095.

According to part (c) in Fig. 2, in the case where $H \in [4\pi/3,\ 2\pi)$, the values of ΔH_c are in the range $[-0.9917,\ -0.1027]$ When the input data of the R, G, and B components are equal to 255, 0, and 1, respectively, ΔH_c possesses the minimum value, which equals -0.9917. When the input data of the R, G, and B components are equal to 1, 0, and 255, respectively, ΔH_c possesses the maximum value, which equals -1.027.

2) THE ANALYSIS OF THE I COMPONENT

According to (1) and (25), the error calculation formula of the I component in the RGB2HSI conversion can be written as:

$$\Delta I = I' - I = -\frac{R+G+B}{12288} \tag{35}$$

According to (35), the error value of the I component in the RGB2HSI conversion is inversely proportional to the sum of the values of the R, G, and B components. When the input data of the R, G, and B components are all equal to 255, the error of the component I is the minimum, which equals -0.0622. In this case, compared with the geometric derivation algorithm, the error of the I component of the S-SC algorithm in the RGB2HSI conversion is the maximum.

When the input data of the R, G, and B components are all equal to 0, the error of the I component is the maximum, which equals 0. In this case, compared with the geometric derivation algorithm, the error of the I component of the S-SC algorithm in the RGB2HSI conversion is the minimum. The calculation result is shown in Fig. 3.

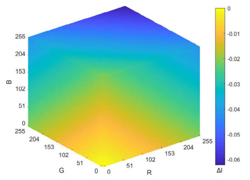


FIGURE 3. The error ΔI of the I component.

In addition, since the calculation formulas of the S component do not differ between the S-SC algorithm and the geometric derivation algorithm and the S component is directly determined by the input data of the R, G, and B components, the error of the S component can be regarded as 0 during the color space conversion.

B. THE ERROR ANALYSIS OF EACH COMPONENT OF THE RECONSTRUCTED RGB IMAGE IN THE S-SC ALGORITHM

In the previous section, we analyze the error of the S-SC algorithm in the HSI2RGB conversion. In this section, we take the case of $H \in [0,\ 2\pi/3)$ as an example, so that (36) can be derived from (4) and (30).

$$\begin{cases} \Delta B = B' - B = \Delta I\left(1 - S'\right) \\ \Delta R = R' - R = \Delta I\left(1 + S'H'\right) \\ \Delta G = G' - G = \Delta I\left(1 + S' - S'H'\right) \end{cases} \tag{36}$$

Fig. 4 shows the calculation results of the errors ΔR, ΔG and ΔB, and parts (a), (b), and (c) represent the error calculation results of the reconstructed R, G, and B components, respectively.

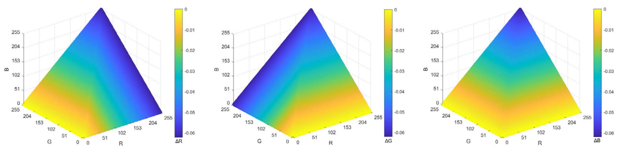

(a) The value of ΔR (b) The value of ΔG (c) The value of ΔB

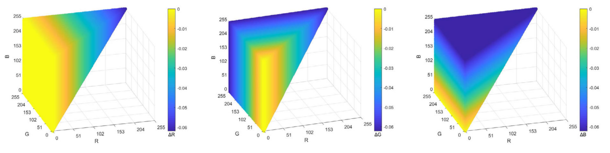
FIGURE 4. The error of the R, G, and B components of the reconstructed RGB image when $H \in [0,\ 2\pi/3)$.

According to Fig. 4 and the experimental results, in the case where $H \in [0,\ 2\pi/3)$, the ΔR, ΔG and ΔB

values are in the ranges $[-0.0623, -2.4414 \times 10^{-4}]$, $[-0.0623, -2.4414 \times 10^{-4}]$ and $[-0.0620, 0]$, respectively.

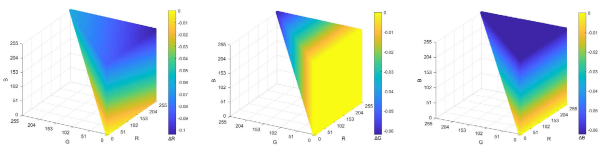
- When the input data of the R, G, and B components are equal to 255, 2, and 1, respectively, ΔR possesses the minimum value, which equals -0.0623. Further, when the input data of the R, G, and B components are equal to 1, 255, and 0, respectively, ΔR possesses the maximum value, which equals -2.4414×10^{-4}
- When the input data of the R, G, and B components are equal to 4, 255, and 3, respectively, ΔG possesses the minimum value, which equals -0.0623. Furthermore, when the input data of the R, G, and B components are equal to 1, 252, and 0, respectively, ΔG possesses the maximum value, which equals -2.4414×10^{-4}.
- When the input data of the R, G, and B components are equal to 255, 255, and 4, respectively, ΔB possesses the minimum value, which equals -0.0620. Furthermore, when the input data of the B component is equal to 0, ΔB possesses the maximum value, which equals 0.

Similarly, the error of the R, G, and B components of the reconstructed RGB image can be calculated in the same way in the cases where $H \in [2\pi/3, 4\pi/3)$ and $H \in [4\pi/3, 2\pi)$. The calculation results are shown in Fig. 5 and Fig. 6.



(a) The value of ΔR (b) The value of ΔG (c) The value of ΔB

FIGURE 5. **The error of the *R*, *G*, and *B* components of the reconstructed RGB image when $H \in [2\pi/3, 4\pi/3)$.**



(a) The value of ΔR (b) The value of ΔG (c) The value of ΔB

FIGURE 6. **The error of the *R*, *G*, and *B* components of the reconstructed RGB image when $H \in [4\pi/3, 2\pi)$.**

According to Fig. 5, Fig. 6, and the experimental results, in the case where $H \in [2\pi/3, 4\pi/3)$, the values of ΔR and ΔG are all in the range $[-0.0623, -2.4414 \times 10^{-4}]$, and ΔB is in the range [-0.0620, 0]. In the case where $H \in [4\pi/3, 2\pi)$, the values of ΔR and ΔG are all in the range $[-0.0623, -2.4414 \times 10^{-4}]$, and ΔB is in the range $[-0.0620, 0]$.

According to the above assertions, compared with the geometric derivation algorithm, the R, G, and B component error of the RGB image reconstructed by the S-SC algorithm are smaller. Combining the analyses in Section II part C) and this section, it can be proven that the S-SC algorithm achieves a high conversion accuracy and low complexity. In addition, due to high computational complexity and high resource consumption, brought the floating data operation in the FPGA hardware, it is necessary to convert the floating data into integer data when realizing the S-SC algorithm (software) in FPGA hardware [21], [22]. Therefore, from the perspective of FPGA implementation, the error of the R, G, and B components of the reconstructed RGB image can be regarded as integer values of 0. Briefly, the error of the three components of the reconstructed RGB image can be ignored as opposed to those of the geometric derivation algorithm.

IV. THE COMPUTATIONAL COMPLEXITY OF THE S-SC ALGORITHM

According to parts A and B of Section II, we obtain the RGB2HSI conversion equations of the geometric derivation algorithm and S-SC algorithm. As shown by pseudocodes 1-4, since the geometric derivation algorithm and the S-SC algorithm have the same number of iterations, we cannot judge their time complexity by the difference in iterations. In addition, because the S-SC algorithm is designed to be implemented on FPGA hardware, in this section, we analyze the time complexities of the two algorithms from the perspective of FPGA-based calculations by combining the traditional time complexity methods. In addition, considering the calculation characteristics of the RGB2HSI and HSI2RGB conversions, we decided to choose Taylor series expansions to expand the trigonometric function and inverse trigonometric function of the geometric derivation algorithm.

A. THE TIME COMPLEXITY ANALYSIS OF THE RGB2HSI CONVERSION

Assuming that the number of pixels in the input image is N, according to the pseudocode of the RGB2HSI conversion process of the S-SC algorithm shown in Algorithm 3, it can be seen that the conversion process of the S-SC algorithm mainly includes the following 4 parts.

- Separate the values of the R, G, and B components from each pixel (Algorithm 3 Step III).
- Calculate the minimum values of the separated R, G, and B components for each pixel (Algorithm 3 Step IV).
- Preserve the H component sector information (Algorithm 3 Step V).
- Calculate the values of the H', S', and I' components (Algorithm 3 Step VI).

For the above four steps, the time complexities of steps III, IV, and V are $O(3N)$, $O(4N)$, and $O(N)$, respectively. Step VI possesses a time complexity of $O(6N)$, including $O(3N)$ for calculating the H' component, $O(2N)$ for calculating the S' component, and $O(N)$ for calculating the I' component.

Therefore, the time complexity of the RGB2HSI conversion in the S-SC algorithm is $O(3N + 4\,N + N + 6N) = O(14N)$.

For the time complexity of the RGB2HSI conversion in the geometric derivation algorithm, according to the descriptions in parts A and B in Section II, the RGB2HSI conversion formulas are not different between the geometric derivation algorithm and S-SC algorithm except for the calculation of the H component. Therefore, as Algorithm 1 shows, the RGB2HSI conversion in the geometric derivation algorithm is similar to that of the S-SC algorithm, and it mainly includes the following three steps.

- Separate the values of the R, G, and B components from each pixel (Algorithm 1 Step III).
- Calculate the minimum values of the separated R, G, and B components for each pixel (Algorithm 1 Step IV).
- Calculate the values of the H, S, and I components (Algorithm 1 Step V).

Among the steps, the time complexities of step III, step IV, and the I component calculation in step V are the same as those of the S-SC algorithm, which are O($3N$), O($4N$), and O(N), respectively.

It can be seen from (1)-(2) that the conversion formula for calculating the H component in the geometric derivation algorithm includes the arccosine function. To calculate its time complexity, we use a Taylor series expansion to expand the inverse trigonometric function. In this part, we take case $H \in [0, 2\pi/3)$ as an example to analyze the case at point 0. At this time, the three-term Taylor series expansion of the arccosine function can be expressed as:

$$\arccos(x) = \frac{\pi}{2} - x - \frac{x^3}{6} + \mathrm{O}\left(x^3\right) \tag{37}$$

We can obtain (38) by incorporating (37) into (1).

$$H = \frac{\pi}{2} - X - \frac{X^3}{6} + \mathrm{O}\left(X^3\right) \tag{38}$$

where

$$X = \frac{(R-G)(R-B)}{2\sqrt{(R-G)^2 + (R-B)(G-B)}} \tag{39}$$

Then, the time complexity of the geometric derivation algorithm for calculating the H component is O($7N$). For the calculation of the S component, its time complexity in the geometric derivation algorithm is O(3N), which differs from that of the S-SC algorithm. Therefore, the time complexity of step V of Algorithm 1 is O($7N + 3N + N$) = O($11N$).

Therefore, the time complexity of the RGB2HSI conversion in the geometric derivation algorithm is O($3N + 4\,N + 11\,N$) = O($18N$).

Compared with the geometric derivation algorithm, the time complexity of the RGB2HSI conversion in the S-SC algorithm is reduced by O($4N$), which is not a significant gap. This is because we use a three-term Taylor series expansion to expand the H component when calculating its time complexity in the RGB2HSI conversion of the geometric derivation algorithm. In summary, the more Taylor series expansion items of the H component that are in the geometric derivation algorithm, the higher the time complexity, and the more prominent the advantage of the S-SC algorithm.

B. THE TIME COMPLEXITY ANALYSIS OF THE HSI2RGB CONVERSION

Similarly, we assume that the number of pixels in the input image is N. According to the pseudocode of the HSI2RGB conversion in the S-SC algorithm shown in Algorithm 4, it can be seen that the HSI2RGB conversion of the S-SC algorithm mainly includes the following two steps:

- Calculate the values of the $R'R'$, G', G' and B' components of each reconstructed pixel (Algorithm 4 Step III).
- Integrate the reconstructed $R'R'$, G', G' and B' components into a reconstructed RGB image (Algorithm 4 Step IV).

For the above two steps, the time complexities of steps III and IV are O($5N$) and O($3N$), respectively. In step III, the time complexity O($5N$) includes O($3N$) for calculating the reconstructed R' component, O(N) for calculating the G' component, and O(N) for calculating the B' component.

Therefore, the time complexity of the HSI2RGB conversion in the S-SC algorithm is O($5N + 3\,N$) = O($8N$).

For the time complexity of the HSI2RGB conversion in the geometric derivation algorithm, we also take $H \in [0, 2\pi/3)$ as an example. According to Algorithm 2, the HSI2RGB conversion of the geometric derivation algorithm mainly contains the following two steps.

- Calculate the values of the reconstructed R, G, and B components of each pixel (Algorithm 2 step III).
- Integrate the reconstructed R, G, and B components into a reconstructed RGB image (Algorithm 2 Step IV).

For the above two steps, the time complexities of step IV and the calculation of the reconstructed G component in step III are the same as those in the S-SC algorithm, which equal O($3N$) and O(N), respectively.

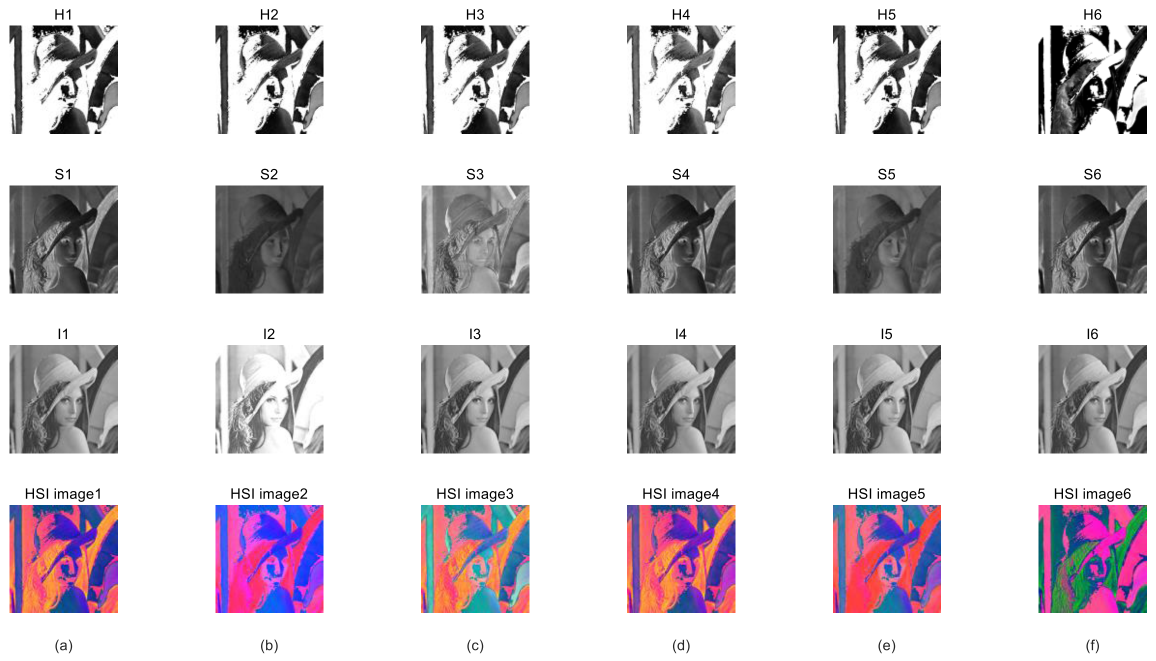
It can be seen from (4) that the HSI2RGB conversion of the geometric derivation algorithm contains the trigonometric function in case $H \in [0, 2\pi/3)$. To calculate the time complexity, we also use a Taylor series expansion to expand (4), and (4) can be rewritten in a three-term Taylor series expansion form as:

$$Z = I\left(1 + \frac{S\left(1 - \frac{Y_1^2}{2!} - \frac{Y_1^4}{4!} + O\left(Y_1^4\right)\right)}{1 - \frac{Y_2^2}{2!} - \frac{Y_2^4}{4!} + O\left(Y_2^4\right)}\right) \tag{40}$$

where Y_1 and Y_2 can be expressed as:

$$\begin{cases} Y_1 = H \\ Y_2 = \dfrac{\pi}{3} - H \end{cases} \tag{41}$$

Then, the time complexity of the geometric derivation algorithm for calculating the reconstructed R component is O($6N$). For the calculation of the reconstructed B component, its time complexity in the geometric derivation algorithm is O(2N), which differs from that of the S-SC algorithm. In the HSI2RGB conversion of the geometric derivation algorithm, the time complexity of step III is O($6N + N + 2N$) = O($9N$).



From left to right: (a) Geometric derivation algorithm, (b) Coordinate transformation method, (c) Segment definition method, (d) Bajon approximation algorithm, (e) Standard module arithmetic method, and (f) S-SC algorithm.

FIGURE 7. The HSI images and the images of their corresponding components through the RGB2HSI conversion.

Therefore, the time complexity of the HSI2RGB conversion in the geometric derivation algorithm is $O(3N + 9\,N) = O(12N)$.

Compared with the geometric derivation algorithm, the time complexity of the HSI2RGB conversion in the S-SC algorithm is reduced by $O(4N)$, which is not a significant gap. This is also because we use a three-term Taylor series expansion to expand the trigonometric function when calculating its time complexity in the HSI2RGB conversion of the geometric derivation algorithm. In conclusion, the more Taylor series expansion terms of the trigonometric function that are in the algorithm, the higher the time complexity, and the more prominent the advantage of the S-SC algorithm.

V. SIMULATION EFFECTS OF THE S-SC ALGORITHM

In this section, we use MATLAB 2019a with an AMD RYZEN 3970X and 64G memory to verify the effectiveness of the S-SC algorithm by comparing it with five classical RGB-HSI color space conversion algorithms, namely, the geometric derivation algorithm, the coordinate transformation method, the segment definition method, the Bajon approximation algorithm, and the standard module arithmetic method.

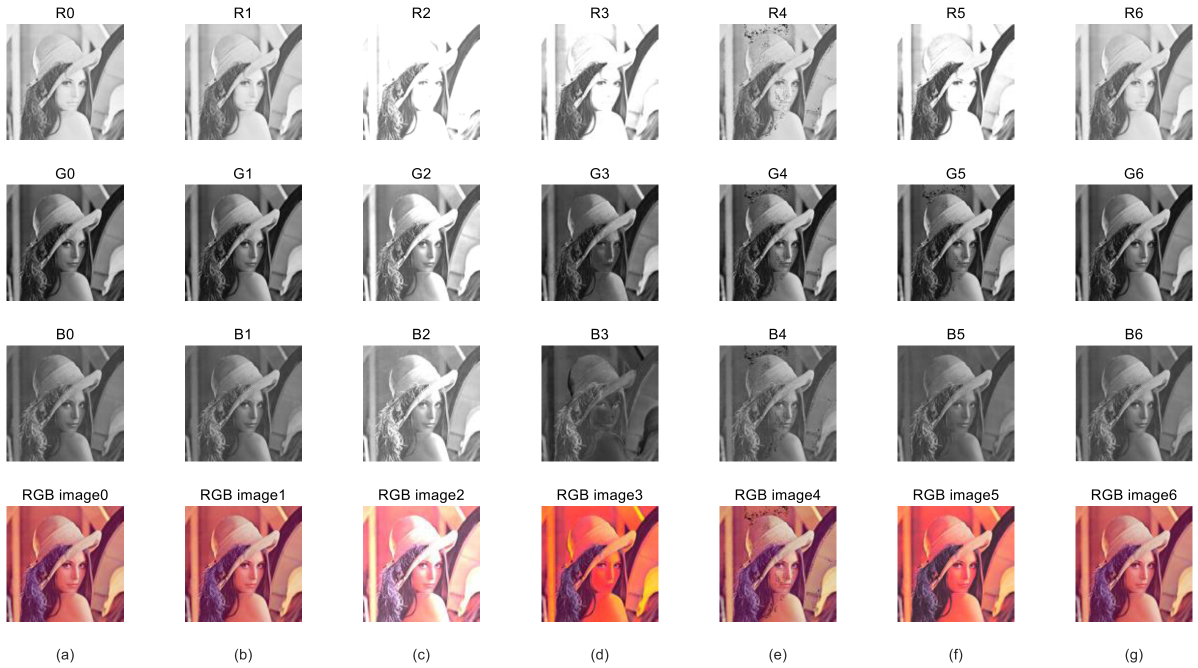
A. THE COMPARISON OF CONVERSION PERFORMANCES

The Lena image is one of the most commonly used images in digital image processing because it possesses high recognition and pronounced features. Therefore, in this section, we utilize the Lena image with 512×512 resolution as the original RGB image for the performance verification of the S-SC algorithm. First, the above six RGB-HSI conversion algorithms are utilized to convert the RGB Lena image into the HSI image through RGB2HSI conversion. Fig. 7 shows the HSI image and the images of the corresponding H, S, and I components.

According to Fig. 7, in the images of the H components, the conversion result from the S-SC algorithm differs from those of the other RGB-HSI conversion algorithms, and the conversion results obtained by the other five algorithms are relatively close. The reason for the above issue is that (11) is utilized as the H component for the hue information in the S-SC algorithm.

In the images of the S components, the result obtained by the S-SC algorithm is close to those obtained by the geometric derivation algorithm and the Bajon approximation algorithm but far from those obtained by the coordinate transformation method, the segment definition method, and the standard module arithmetic method. In terms of the I components, except for the coordinate transformation method, the conversion results of the S-SC algorithm are close to those of the other four algorithms.

Although these differences can be seen directly by the human eye, they do not clearly describe the specific differences between the aforementioned six algorithms. In this case, the reconstructed RGB Lena image and the images of its corresponding R, G, and B components are obtained by the HSI2RGB conversion based on the HSI images obtained by the six RGB-HSI conversion algorithms in Fig. 7. Further, the



From left to right: (a) original image, (b) geometric derivation algorithm, (c) coordinate transformation method, (d) segment definition method, (e) Bajon approximation algorithm, (f) standard module arithmetic method, and (g) S-SC algorithm.

FIGURE 8. The reconstructed RGB images and the images of their corresponding components after the HSI2RGB conversion.

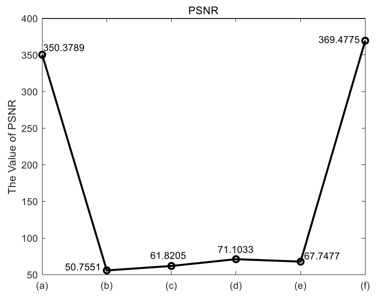
images of the reconstructed RGB and its corresponding components are compared with the images of the original Lena image and its corresponding components, and the results are shown in Fig. 8.

According to Fig. 8, the reconstructed Lena images generated by the S-SC algorithm and the other five RGB-HSI conversion algorithms, and the original Lena image are evaluated subjectively. The reconstructed Lena images generated by the S-SC algorithm and the geometric derivation algorithm are most similar to the original image. In contrast, there are obvious differences between the original image and those generated by the other four algorithms.

To further verify the performance of the S-SC algorithm, based on a subjective evaluation, two commonly used objective evaluation indexes, the peak signal-to-noise ratio (PSNR) and structural similarity (SSIM) [23], [24], are utilized.

Fig. 9 shows the PSNR values of the six RGB-HSI algorithms. The higher the PSNR value is, the less noise the image contains. According to Fig. 9, we can see that among the six RGB-HSI conversion algorithms, the PSNR value of the S-SC algorithm is the largest, equaling 369.4775, and it is slightly larger than that of the geometric derivation algorithm and much larger than that of the other four RGB-HSI conversion algorithms. Briefly, the experimental results show that the reconstructed RGB image of the S-SC algorithm possesses the lowest noise and is the most similar to the original image.

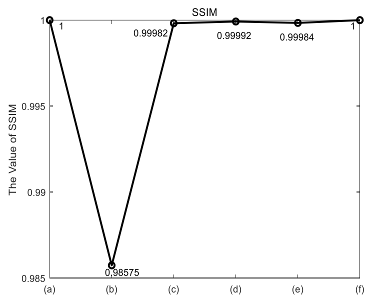
Fig. 10 shows the SSIM values of the six RGB-HSI conversion algorithms. The SSIM is an index to measure the similarity of two images and an all-reference image quality



From left to right: (a) Geometric derivation algorithm, (b) Coordinate transformation method, (c) Segment definition method, (d) Bajon approximation algorithm, (e) Standard module arithmetic method, and (f) S-SC algorithm.

FIGURE 9. The PSNR results for the reconstructed RGB Lena images.

evaluation index. It measures the similarity of images in terms of brightness, contrast, and structure. The value range of SSIM is between 0 and 1, and if the value is 1, it means the two images are the same. The higher the value of SSIM is, the higher the similarity and the lower the distortion of the image, which means that the evaluated image is closer to the original image. It can be seen from Fig. 10 that the SSIM value of the S-SC algorithm equals 1, which is the same as that of the geometric derivation algorithm, and slightly larger than those of the other four algorithms. The experimental results indicate that the structure of the reconstructed RGB Lena image of the S-SC algorithm is almost the same as the structure of the original image, and the degree of distortion is almost equal to 0.



From left to right are the SSIM results of the reconstructed RGB image by (a) Geometric derivation algorithm, (b) Coordinate transformation method, (c) Segment definition method, (d) Bajon approximation algorithm, (e) Standard module arithmetic method, and (f) S-SC algorithm

FIGURE 10. **The SSIM results for the reconstructed RGB Lena images.**

In conclusion, according to Fig. 7 to Fig. 10, the results show that the S-SC algorithm possesses a high performance in HSI-RGB conversion. The RGB image reconstructed by the S-SC algorithm is the most similar to the original RGB image. Since the trigonometric function and inverse trigonometric function parts are designed to be optimized, the image of the H component generated by the S-SC algorithm is different from those of all other algorithms.

B. THE COMPARISON OF COMPUTATIONAL COMPLEXITIES

Based on the comparisons with and verification of the conversion performance of the S-SC algorithm in the previous section, in this section, we simulate and verify the computational complexity of the S-SC algorithm. For the above six algorithms for processing the Lena graph, the running times are shown in Tables 1-3.

TABLE 1. **The time consumption of The RGB2HSI conversion.**

Algorithm	(a)	(b)	(c)	(d)	(e)	(f)
Time(s)	0.0731	0.0468	0.0784	0.0776	0.0742	0.0634

From left to right: (a) Geometric derivation algorithm, (b) Coordinate transformation method, (c) Segment definition method, (d) Bajon approximation algorithm, (e) Standard module arithmetic, and (f) S-SC algorithm.

TABLE 2. **The time consumption of The HSI2RGB conversion.**

Algorithm	(a)	(b)	(c)	(d)	(e)	(f)
Time(s)	0.0215	0.0211	0.0209	0.0212	0.0211	0.0152

From left to right: (a) Geometric derivation algorithm, (b) Coordinate transformation method, (c) Segment definition method, (d) Bajon approximation algorithm, (e) Standard module arithmetic, and (f) S-SC algorithm.

It can be seen in Table 1 that in the RGB2HSI conversion, except for the coordinate conversion method, the time consumption of the S-SC algorithm is the lowest. As Table 3 shows, the total time consumption of the coordinate transformation method is lower than that of the S-SC algorithm in the RGB2HSI and HSI2RGB conversions. However, it can

TABLE 3. **The total time consumption of The RGB2HSI and The HSI2RGB conversions.**

Algorithm	(a)	(b)	(c)	(d)	(e)	(f)
Time(s)	0.0946	0.0679	0.0993	0.0988	0.0953	0.0786

From left to right: (a) Geometric derivation algorithm, (b) Coordinate transformation method, (c) Segment definition method, (d) Bajon approximation algorithm, (e) Standard module arithmetic, and (f) S-SC algorithm.

be seen from Fig. 9 and Fig. 10 that although the coordinate transformation method requires less time, the PSNR and SSIM values of the coordinate transformation method are the lowest among the six RGB-HSI conversion algorithms, which means that the coordinate transformation method achieves the worst restoration effect compared to those of other RGB-HSI algorithms.

Furthermore, it can be seen in Table 2 that in the HSI2RGB conversion, the time consumption of the S-SC algorithm is the lowest. This is because the S-SC algorithm uses (11) as the hue information of the image. This makes it unnecessary to perform nonlinear calculations such as the cosine function in the HSI2RGB conversion, and only linear calculations are required, thereby reducing the running time of the S-SC algorithm.

Through combining the analysis mentioned in Section IV and in this section, it can be proven that compared with the geometric derivation algorithm, the S-SC algorithm has lower time complexity in both the RGB2HSI and HSI2RGB conversions. Compared with other traditional RGB-HSI conversion algorithms, the S-SC algorithm possesses the advantages of a high restoration degree and low operation time consumption.

VI. THE FPGA IMPLEMENTATION AND PERFORMANCE ANALYSIS OF THE S-SC ALGORITHM

After verifying the performance of the S-SC on the software platform, in this section, the XC7A100T-2fgg484 development board of the Artix-7 series produced by the Xilinx company is utilized as the hardware platform for the verification of the S-SC algorithm. First, the hardware logic timing sequence and hardware conversion effect of the S-SC algorithm are simulated and verified based on the Integrated Software Environment 14.7 (ISE 14.7) software. The logic resource consumption of the S-SC algorithm in the FPGA implementation is analyzed by comparing it with the algorithms proposed by [7] and [17]. Second, the reconstructed RGB images generated by the S-SC algorithm (FPGA) and S-SC algorithm (software) are compared and analyzed.

A. THE FPGA IMPLEMENTATION AND RESOURCE CONSUMPTION OF THE S-SC ALGORITHM IN THE RGB2HSI CONVERSION

To ensure the strict consistency of the timing sequence, it is necessary to delay the earlier output signals during the FPGA implementation. The RGB2HSI conversion structure of the S-SC algorithm is shown in Fig. 11.

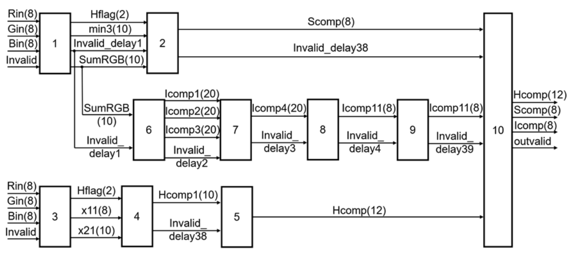


FIGURE 11. The FPGA structure flowchart for the RGB2HSI conversion in the S-SC algorithm.

As Fig. 11 shows, it is necessary to delay the signals of the *H* and *S* components to ensure that Hcomp, Scomp, and Icomp can be output in a strict timing sequence. According to the logic timing sequence simulation of signals in Fig. 12, there indeed exist 39 clock delays between the input data Rin, Gin, and Bin and the output data Hcomp, Scomp, and Icomp, which proves the correctness of the structure in Fig. 11.

For the FPGA implementation of the RGB2HSI conversion in the S-SC algorithm, the amounts of logic resources consumed based on the Artix7-100t board are shown in Table 4.

TABLE 4. The Logic resource consumption of The RGB2HSI conversion in The S-SC algorithm (Artix7-100T).

Name of logic resource	No. used	No. total	Occupation
Slice LUTs	1103	63400	1.74%
Slice Registers	2089	126800	1.65%
Memory	68	19000	0.36%

As Table 4 shows, the S-SC algorithm consumes few logic resources in the RGB2HSI conversion. The number of consumed LUTs is 1.74% of all available LUTs, and the number of consumed registers is 1.65% of all available registers.

B. THE FPGA IMPLEMENTATION AND RESOURCE CONSUMPTION OF THE S-SC ALGORITHM IN THE HSI2RGB CONVERSION

For the FPGA implementation of the HSI2RGB conversion in the S-SC algorithm, the designed structure of the HSI2RGB conversion is shown in Fig. 13, and the timing sequence simulation of the HSI2RGB conversion is shown in Fig. 14.

As Fig. 14 shows, there exist seven clock delays between the input data Hcomp, Scomp, and Icomp and the output data Rout, Gout, and Bout.

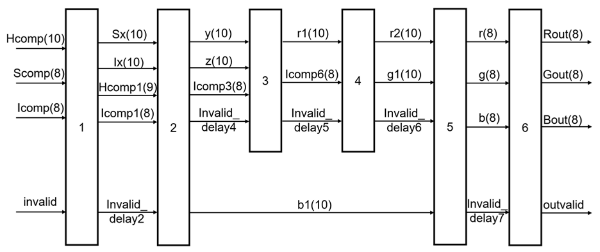


FIGURE 13. The FPGA structure flowchart for the HSI2RGB conversion in the S-SC algorithm.

According to the simulation result in Fig. 14, the delay between the input data and the output data is 140 ns, and the S-SC algorithm for the HSI2RGB conversion is proven in real-time.

For the FPGA implementation of the HSI2RGB conversion in the S-SC algorithm, the amounts of logic resources consumed are shown in Table 5.

TABLE 5. The Logic resource consumption of The HSI2RGB consumption in The S-SC algorithm (Artix7-100T).

Name of logic resource	No. used	No. total	Occupation
Slice LUTs	283	63400	0.45%
Slice Registers	209	126800	0.16%
Memory	3	19000	0.02%

As Table 5 shows, the HSI2RGB conversion of the S-SC algorithm consumes few logic resources. Through combining Table 5 with Table 4, it is shown that we have approximately 62000 LUTs and 124400 registers to achieve the subsequent RGB image procession based on the HSI color space, which may contain complex processing modules such as multiple DSP modules and IP cores. Therefore, we can perform the whole image processing algorithm with one board instead of two, and we can also reduce the cost and raise the stability of signals.

C. THE ANALYSIS OF LOGIC RESOURCE COMSUMPTION IN THE RGB2HSI CONVERSION AND THE HSI2RGB CONVERSION

Based on the above sections, in this part, through the comparison of the experimental results of the algorithms proposed in [17] and [7], we analyze the amounts of logic resources consumed by the S-SC algorithm.

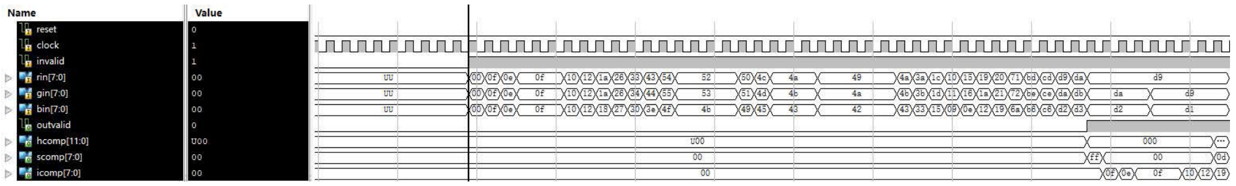


FIGURE 12. The timing sequence simulation result for the input data Rin, Gin, and Bin and the output data Hcomp, Scomp, and Icomp.

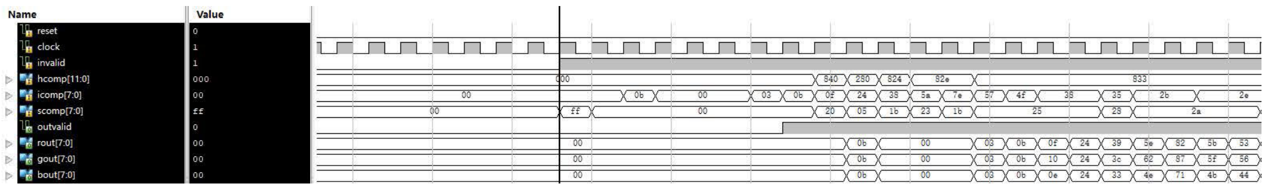


FIGURE 14. The timing sequence simulation result for the input data Hcomp, Scomp, and Icomp and the output data Rout, Gout, and Bout.

TABLE 6. The comparation of logic resource consumption between The S-SC algorithm (Artix-7) and the algorithm proposed in [7] (Virtex5).

Logic slice Utilization/Total	The RGB2HSI conversion in [7]	The RGB2HSI conversion in the S-SC algorithm	The HSI2RGB conversion in [7]	The HSI2RGB conversion in the S-SC algorithm
No. Slice LUTs / All	2947/28800	1103/63400	7396/28800	283/63400
No. DSPs / All	Unknown	0/240	3/48	0/240
No. Slice Registers / All	20/28800	2089/126800	268/28800	209/126800
No. LUT-FF pairs / All	17/2950	913/1962	263/7401	100/360
No. Bonded IOBs / All	51/480	57/285	51/480	57/285
No. BUFGCTRLs / All	1/32	1/32	1/32	1/32

TABLE 7. The comparation of logic resource consumption between The S-SC algorithm (Artix-7) and the algorithm proposed in [17] (Spartan6).

Logic slice Utilization/Total	The RGB2HSI conversion in [17]	The RGB2HSI conversion in the S-SC algorithm	The HSI2RGB conversion in [17]	The HSI2RGB conversion in the S-SC algorithm
No. Slice LUTs / All	4959/9112	1103/63400	3006/9112	283/63400
No. Occupied Slices / All	1540/2278	612/15850	998/2278	110/15850
No. Slice Registers / All	44/18224	2089/126800	28/18224	209/126800
No. Bonded IOBs / All	85/232	57/285	85/232	57/285

Further, we analyze the S-SC algorithm's realizability and efficiency, and Tables 6 and 7 show the comparisons.

According to Tables 6 and 7, the consumed slice LUTs in the algorithms proposed in [7] and [17] are significantly more than those in the S-SC algorithm regardless of the RGB2HSI conversion or the HSI2RGB conversion. In addition, the algorithm proposed in [7] utilizes the DSP modules, which leads to the high consumption of Slice LUTs. The details are shown in Table 8.

TABLE 8. Slice LUTs occupancy of the three algorithms in the RGB2HSI conversion and the HSI2RGB conversion.

Usage of slice LUTs resource	Proportion
The S-SC algorithm compared with [7] in the RGB2HSI conversion	37.43%
The S-SC algorithm compared with [17] in the RGB2HSI conversion	22.24%
The S-SC algorithm compared with [7] in the HSI2RGB conversion	3.83%
The S-SC algorithm compared with [17] in the HSI2RGB conversion	9.41%

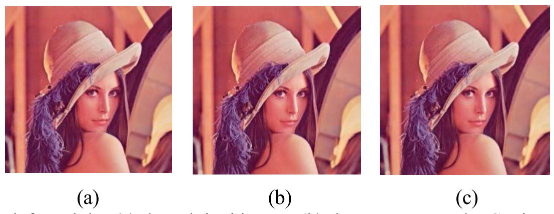
According to Tables 6 to 8, although the S-SC algorithm has slightly higher consumption of register and flip-flop resources, the consumption of Slice LUTs is significantly lower than that in algorithms proposed in [7] and [17]. The reason for the S-SC algorithm has a slightly higher consumption of register, and flip-flop resources can be split into the following two points.

On the one hand, a slightly higher cost of register resources is needed to ensure the strict timing sequence of the whole system. On the other hand, from the perspective of the parallel data stream and the sufficient register resources of the FPGA hardware itself, a certain cost of flip-flop resources can help the entire line of three-channel RGB image data be output in a strict timing sequence after being cached in the First Input First Output (FIFO). In this case, not only can the timing sequence of the R, G, and B component data be ensured, but the complexity of the whole system can also be decreased.

In conclusion, compared with the algorithms proposed in [7] and [17], the S-SC algorithm consumes many fewer LUTs with the cost of slightly more register and flip-flop resources. In this way, the S-SC algorithm can leave many more logic resources for the subsequent image processing algorithm than the other two algorithms proposed in [7] and [17] can if the types of FPGA boards are all the same. Furthermore, the whole image processing approach based on the S-SC algorithm can be achieved with one FPGA board instead of two, and it can help improve the stability of signals while the whole system runs.

D. THE COMPARISON AND ANALYSIS OF THE RECONSTRUCTED RGB IIMAGE EFFECT OF THE S-SC ALGORITHM BASED ON SOFTWARE AND FPGA HARDWARE

Based on the above analyses, the effects of the reconstructed RGB Lena image generated by the S-SC algorithm (software) and S-SC algorithm (FPGA) are compared and analyzed in this section. The reconstructed RGB Lena images are shown in Fig. 15.



From left to right: (a) the original image, (b) the reconstructed RGB image of the S-SC algorithm (software), and (c) the reconstructed RGB image of the S-SC algorithm (FPGA).

FIGURE 15. The effects of reconstructed RGB Lena images through the S-SC algorithm (software) and S-SC algorithm (FPGA).

The reconstructed RGB images generated by the S-SC algorithm (software) and the S-SC algorithm (FPGA) are evaluated and analyzed objectively, and the evaluation results are shown in Table 9.

TABLE 9. Evaluation of reconstructed RGB images of the S-SC algorithm (software) and S-SC algorithm (FPGA).

Evaluation index	The S-SC algorithm (software)	The S-SC algorithm (FPGA)
PSNR	369.4775	92.9305
SSIM	1	1

Combining Table 9 with Fig. 9 and Fig. 10, the SSIM index of the reconstructed RGB image generated by the S-SC algorithm (FPGA) possesses the same value as that generated by the S-SC algorithm (software). In addition, compared with Fig. 9 and Fig. 10 in Section IV, it can be found that the reconstructed RGB image generated by the S-SC algorithm (FPGA) not only obtains a higher value of the PSNR index than those generated by four RGB-HSI conversion algorithms, including the coordinate transformation method, the segmentation definition method, the Bajon approximation algorithm, and the standard module arithmetic method, but it also achieves a higher value of the SSIM index.

In summary, at the cost of a certain degree of register and flip-flop resource usage, the S-SC algorithm (FPGA) consumes significantly fewer slice LUTs resources than the algorithms proposed in [7] and [17]. In this case, it can be said that the S-SC algorithm (FPGA) provides sufficient logic resources for the follow-up works of FPGA-based hardware image processing. On the other hand, the reconstructed RGB image generated by the S-SC algorithm (FPGA) achieves a better effect than the above four RGB-HSI conversion algorithms, including the coordinate transformation method, the segmentation definition method, the Bajon approximation algorithm, and the standard module arithmetic method, which are achieved based on the software platform.

VII. CONCLUSION

In this paper, an FPGA-based simple RGB-HSI space conversion algorithm, named the S-SC algorithm, is proposed. The S-SC algorithm aims to solve the problems of high computational complexity and great difficulty in FPGA hardware implementation caused by nonlinear calculations in the geometric derivation algorithm, such as trigonometric and inverse trigonometric functions. The S-SC algorithm utilizes the inverse transformation process of the geometric derivation algorithm to reconstruct the mapping relationship between the R, G, and B components in the RGB color space and the H, S, and I components in the HSI color space. Based on ensuring the conversion accuracy, the complexity of the algorithm is effectively reduced, and the hardware realizability of the S-SC algorithm is improved. First, on this basis, the S-SC algorithm's conversion accuracy is verified by comparing the error with that of the geometric derivation algorithm. Second, the S-SC algorithm's performance is verified by comparing it with those of five commonly used RGB-HSI space conversion algorithms, such as the geometric derivation algorithm and the coordinate transformation method. Third, the hardware realizability of the S-SC algorithm is verified by comparing it with those of the algorithms proposed in [7] and [17].

The experimental results show that the S-SC algorithm not only possesses high conversion accuracy but it can also significantly decrease the consumption of slice LUTs at the cost of a few more register and flip-flop resources, and it can provide sufficient logic resources for subsequent research based on FPGA hardware.

REFERENCES

[1] H. Endo and A. Taguchi, "Color image enhancement by using hue-saturation gradient," in *Proc. Int. Symp. Intell. Signal Process. Commun. Syst. (ISPACS)*, Taipei, China, Dec. 2019, pp. 1–2.
[2] S.-L. Lee and C.-C. Tseng, "Color image enhancement using histogram equalization method without changing hue and saturation," in *Proc. IEEE Int. Conf. Consum. Electron. Taiwan (ICCE-TW)*, Taipei, China, Jun. 2017, pp. 305–306.
[3] G. Hou, Z. Pan, B. Huang, G. Wang, and X. Luan, "Hue preserving-based approach for underwater colour image enhancement," *IET Image Process.*, vol. 12, no. 2, pp. 292–298, Feb. 2018.
[4] S. Chen, R. Feng, Y. Zhang, and C. Zhang, "Aerial image matching method based on HSI hash learning," *Pattern Recognit. Lett.*, vol. 117, pp. 131–139, Jan. 2019.
[5] Y. Guo, Z. Zhang, H. Yuan, and S. Shao, "Single remote-sensing image dehazing in HSI color space," *J. Korean Phys. Soc.*, vol. 74, no. 8, pp. 779–784, Apr. 2019.
[6] N. Nakajima and A. Taguchi, "A novel color image processing scheme in HSI color space with negative image processing," in *Proc. Int. Symp. Intell. Signal Process. Commun. Syst. (ISPACS)*, Kuching, India, Dec. 2014, pp. 029–033.
[7] U. A. Nnolim, "Design and implementation of novel, fast, pipelined HSI2RGB and log-hybrid RGB2HSI colour converter architectures for image enhancement," *Microprocessors Microsyst.*, vol. 39, nos. 4–5, pp. 223–236, Jun. 2015.
[8] W. Zhang, J. Liang, L. Ren, H. Ju, Z. Bai, and Z. Wu, "Fast polarimetric dehazing method for visibility enhancement in HSI colour space," *J. Opt.*, vol. 19, no. 9, Sep. 2017, Art. no. 095606, doi: 10.1088/2040-8986/aa7f39.
[9] F. Qin, Y. Weiping, Y. Jia, L. Hongning, L. Yanlin, L. Xiankui, L. Yichao, and Z. Jiping, "Research on image reconstruction of traditional Chinese painting art based on radiance information matching," *Acta Optica Sinica*, vol. 34, no. 10, pp. 337–345, 2014, doi: 10.3788/AOS201434.1033001.
[10] S. Ma, H. Ma, Y. Xu, S. Li, C. Lv, and M. Zhu, "A low-light sensor image enhancement algorithm based on HSI color model," *Sensors*, vol. 18, no. 10, p. 3583, Oct. 2018.
[11] S. E. Kim, J. J. Jeon, and I. K. Eom, "Image contrast enhancement using entropy scaling in wavelet domain," *Signal Process.*, vol. 127, pp. 1–11, Oct. 2016, doi: 10.1016/j.sigpro.2016.02.016.

[12] R. C. Gonzalez, S. L. Woods, and S. L. Eddins, *Digital Image Processing*, 2nd ed. New York, NY, USA: Prentice-Hall, 2002.

[13] J. Bajon, M. Cattoen, and L. Liang, "Identification of multicoloured objects using a vision module," in *Proc. 6th Int. Conf., Robot Vis. Sensory Control (RoViSeC)*, Paris, France, 1986, pp. 21–30.

[14] H. Liu, *Comparative Studies on the Conversion Methods Between RGB and HSI Color Models*. China Sciencepaper, Apr. 2008. [Online]. Available: http://www.paper.edu.cn/releasepaper/content/200804-1063

[15] S. Khan, P. Manwaring, A. Borsic, and R. Halter, "FPGA-based voltage and current dual drive system for high frame rate electrical impedance tomography," *IEEE Trans. Med. Imag.*, vol. 34, no. 4, pp. 888–901, Apr. 2015.

[16] M. Ravi, A. Sewa, T. G. Shashidhar, and S. S. S. Sanagapati, "FPGA as a hardware accelerator for computation intensive maximum likelihood expectation maximization medical image reconstruction algorithm," *IEEE Access*, vol. 7, pp. 111727–111735, 2019.

[17] G. Saravanan, G. Yamuna, and S. Nandhini, "Real time implementation of RGB to HSV/HSI/HSL and its reverse color space models," in *Proc. Int. Conf. Commun. Signal Process. (ICCSP)*, Melmaruvathur, India, Apr. 2016, pp. 0462–0466.

[18] R. Swenson, "A real-time high performance universal colour transformation hardware system," Ph.D. dissertation, Dept. Electron. Eng., Univ. Kent, Canterbury, U.K., 2000.

[19] M. Shakeri, M. H. Dezfoulian, H. Khotanlou, A. H. Barati, and Y. Masoumi, "Image contrast enhancement using fuzzy clustering with adaptive cluster parameter and sub-histogram equalization," *Digit. Signal Process.*, vol. 62, pp. 224–237, Mar. 2017.

[20] N. K. Quang, N. T. Hieu, and Q. P. Ha, "FPGA-based sensorless PMSM speed control using reduced-order extended Kalman filters," *IEEE Trans. Ind. Electron.*, vol. 61, no. 12, pp. 6574–6582, Dec. 2014.

[21] G. Licciardo, C. Cappetta, and L. Di Benedetto, "Design of a convolutional two-dimensional filter in FPGA for image processing applications," *Computers*, vol. 6, no. 2, p. 19, May 2017.

[22] U. A. Nnolim, "FPGA-based hardware architecture for fuzzy homomorphic enhancement based on partial differential equations," *Int. J. Image Graph.*, vol. 17, no. 4, p. 170022, 2017, doi: 10.1142/S021946781750022X.

[23] X. Shang, J. Liang, G. Wang, H. Zhao, C. Wu, and C. Lin, "Color-Sensitivity-Based combined PSNR for objective video quality assessment," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 29, no. 5, pp. 1239–1250, May 2019.

[24] A. Hore and D. Ziou, "Image quality metrics: PSNR vs. SSIM," in *Proc. 20th Int. Conf. Pattern Recognit.*, Istanbul, Turkey, Aug. 2010, pp. 2366–2369.

SHUAIQING ZHI received the B.E. degree in electrical engineering and automation from Southwest Jiaotong University (SWJTU), Chengdu, China, in 2018. He is currently pursuing the M.S. degree with the College of Information and Communication Engineering, Hainan University, Haikou, China.

His research interests include hardware image processing and control engineering.

YANI CUI received the B.S. and Ph.D. degrees in communication and information systems and information and communication engineering from Hainan University, Haikou, China, in 2011 and 2017, respectively.

She is currently a Lecturer with the College of Information and Communication Engineering, Hainan University. Her research interests include intelligent systems, collaborative control, and image processing.

JIAXIAN DENG received the Ph.D. degree in information and communication engineering from Xidian University, Xi'an, China, in 2004.

Since 2005, he has been a Professor with the College of Information and Communication Engineering, Hainan University. His research interests include data compression and decompression, data encryption and decryption, parallel data procession, and hardware image processing.

WENCAI DU received the B.S. degree from Peking University, Beijing, China, in 1978, the M.S. degree from Hehai University, Nanjing, China, in 1986, the M.S. degree from the Faculty of Geo-Information Science and Earth Observation, University of Twente, Enschede, The Netherlands, in 1996, and the Ph.D. degree from the University of South Australian, in 2000.

From March 2001 to March 2002, he conducted Postdoctoral Research with the Technion-Israel Institute of Technology (IIT), Israel. He is currently the Dean of the Institute of Data Science, City University of Macau, Taipa, Macau, China. He has authored or coauthored 18 books and over 80 science publications. His research interests include several aspects of information technology and communication, including computer network and maritime communications.

Dr. Du is also a member of the Editorial Board of the *Invertis Journal of Science and Technology*, India. He has taken services on many professional conferences, including Conference Chair of the IEEE/ACIS ICIS 2011, the Conference Co-Chair of the IEEE/ACIS SNPD 2010, U.K., the Conference Chair of the IEEE/ACIS SERA 2009, and the Program Chair of the IEEE/ACIS SNPD 2009, Daegu, South Korea.