Circuits, Systems, and Signal Processing (2021) 40:1278–1294
https://doi.org/10.1007/s00034-020-01517-4

# Implementation of a Novel, Fast and Efficient Image De-Hazing Algorithm on Embedded Hardware Platforms

**Prathap Soma**[1] · **Ravi Kumar Jatoth**[1]

Received: 31 December 2019 / Revised: 27 July 2020 / Accepted: 31 July 2020 /
Published online: 18 August 2020

## Abstract

Improving the visibility of hazy images is desirable for robot navigation, security surveillance, and other computer vision applications. The presence of fog significantly damages the quality of the captured image, which does not only affect the reliability of the surveillance system but also produce potential danger. Therefore, developing as well as implementing a simple and efficient image de-hazing algorithm is essential. The reconfigurable computing devices like Field Programmable Gate Array and Digital Signal Processing (DSP) processors are used to implement these image processing applications. Several strategies are available for configuring these reconfigurable devices. In this paper, two approaches for hardware implementation of image de-hazing algorithm are presented. The pixel wise and gray image-based de-hazing algorithm is proposed in this paper. The key advantage of this proposed method is to estimate accurate transmission map. It eliminates the computationally complex step of refine transmission map as well as halos & artifacts in the recovered image and achieves faster execution without noticeable degradation of the quality of the de-hazed image. The proposed method is initially verified in MATLAB and compared with the existing four state-of-art methods. This algorithm is implemented on two different hardware platforms, i.e., DSP Processor (TMS320C6748) with floating pointing operations and Zynq-706 fixed-point operations. The performance comparison of hardware architectures is made with respect to Average Contrast of the Output Image, Mean Square Error, Peak Signal to Noise Ratio, Percentage of Haze Improvement and Structural Similarity Index (SSIM). The results obtained show that Zynq-706-based hardware implementation processing speed is 1.33 times faster when compared to DSP processor-based implementation for an image dimensions of 256 × 256.

✉ Prathap Soma
prathap.soma@student.nitw.ac.in

Ravi Kumar Jatoth
ravikumar@nitw.ac.in

[1] Department of Electronics and Communication Engineering, NIT Warangal, Warangal, India

**Keywords** Field programmable gate array (FPGA) · Digital signal processing (DSP) Processor · Zynq · Image de-hazing · Hardware descriptive language (HDL)

## 1 Introduction

Haze is a phenomenon of absorption or scattering of particles in the atmosphere. It influences the attenuation of radiance along the path toward the camera and captured images are suffered from contrast and quality. It limits the visibility of distant scenes. In recent times, significant advancements has been made in de-hazing techniques based on the atmospheric scattering model of the hazy image. The image de-hazing techniques are categorized: Based on image enhancement and physical model. The image enhancement methods include dynamic range compression and shadow compensation methods [1,7,12], linear transformation [10], structure-preservation [20] and dark channel prior [11] whereas, polarization-based [22] image de-hazing is an example for the physical model method.

According to [28], the methods [1,7,12] work well to enhance low-light areas while maintaining the color information and image data, without producing visual halos. But it has disadvantages in the dark areas and they are not being enough enhanced. In the polarization method, the haze can be removed by considering many hazy images taken from the dissimilar degree of polarization. In this case, more than one image is needed to recover the de-hazed image. Based on this, Narasimhan et al. [18,19] proposed a haze image model to estimate the haze properties of an image. This method can be used only when the environment has a thin haze. Schechner et al. [22] observed that air-light is partially polarized when it is scattered with atmospheric particles. Based on the above concept, they proposed a new haze removal algorithm using a polarizer at various angles. Therefore, the polarization method is inefficient for removing the haze from an image. Later, Kopf et al. [14] suggested a haze removal algorithm using depth information. Jobson et al. [13] used a multi-scale retinex method to increase the visualness of hazy images. Tan et al. [24] used to maximize the contrast of the de-haze image to achieve high contrast haze-free image. Since the resultant de-hazed image has a higher contrast than a hazy image, this method is better for the thickest haze regions with severe color distortions. Fattal et al. [9] proposed a refined image formation model with an assumption that the surface shading and the scene transmission are locally uncorrelated to estimate the thickness of the haze. Next, He et al. [11] proposed a well-used method called dark channel prior, which produces good outcomes and efficiently work compared with enhancement methods [1,7,10,12,20,28] for image de-hazing. Meng et al. [16] suggested an effective regularization method for de-hazing where the haze-free image can be restored with the help of inherent boundary constraints. Tang et al.[25] started an investigation to find out the best feature combination for image de-hazing on different hazy features in a random forest. The random forest is an over-fitting problem that has inherent limitations. Cai et al. [6] proposed a trainable end-to-end system called DehazeNet. This system works using Convolutional Neural Networks (CNN) based on deep architecture [4]. Recently, an active contour on a region based on image segmentation with a variational level set formulation for de-hazing is proposed [2]. This method minimizes the energy function to preserve the edges and it is an iterative structure. Thus, the computational complexity of this method is more. Hence, it is difficult to implement on hardware. A fast and memory efficient dehazing algorithm for real-time computer vision applications is proposed in [23] but not discussed about its hardware implementation. Bai et al. [5] proposed a real-time single image de-hazing system on DSP Processor, which required 4mSec time for execution. Hence, for real-time applications, de-hazing is still a challenging task.

Digital image/video processing algorithms are difficult to implement on a processor due to a few factors like a huge amount of data present in an image, and the complex operations need to be performed on the image. A single image of size 256 × 256 can be considered, to perform a single operation (3x3 convolution/masking) it requires about 0.2 million computations (addition/subtraction, multiplication, padding, and shifting) without considering the overhead of loading and recovering pixel values. This paper presents two different implementation strategies to carry out the goal of hardware implementation. One method is on Zynq-706 FPGA and another on DSP processor (TMS320C6748).

This paper is organized as follows. Section 2 describes the proposed method. In Sect. 3, the hardware architectures for the implemented de-hazing algorithm are described in detail. Section 4 illustrates the discussion of simulation results. Conclusions are provided in Sect. 5.

## 2 Proposed Method

Based on atmospheric scattering theory, McCartney [15] has described a haze image model in terms of light attenuation and air-light. This model is commonly used to define the haze image formation and is given as

$$I(x) = J(x)t(x) + A[1 - t(x)] \tag{1}$$

where $x$ is the image pixel coordinate, $I(x)$ is input hazy image, $J(x)$ is output de-haze image, $A$ is air light, and $t(x)$ is transmission media. In (1), the product term $J(x)t(x)$ is known as direct attenuation, the second term $A[1 - t(x)]$ is an additive component and termed as air-light. Three unknown components are present in the above equation. The basic idea of de-hazing is restoring $J(x)$ from $I(x)$ by estimating the parameters $A$ and $t(x)$. Then, the final dehazed image can be represented in (2)

$$J(x) = A + \left(\frac{I(x) - A}{max(t(x), t_0))}\right) \tag{2}$$

Using method [11] de-hazing can be done. But, this method suffers from high computation complexity owing to, wide-ranging matrix multiplication/division, sorting, and floating-point operations. In low-speed processors, this method cannot meet the user timing requirements for the real-time image processing applications. Therefore, an efficient and low-complexity haze removal method using pixel-based and gray image-based is proposed for real-time applications.

Figure 1 represents the block diagram of the proposed algorithm based on the minimum number of computations.

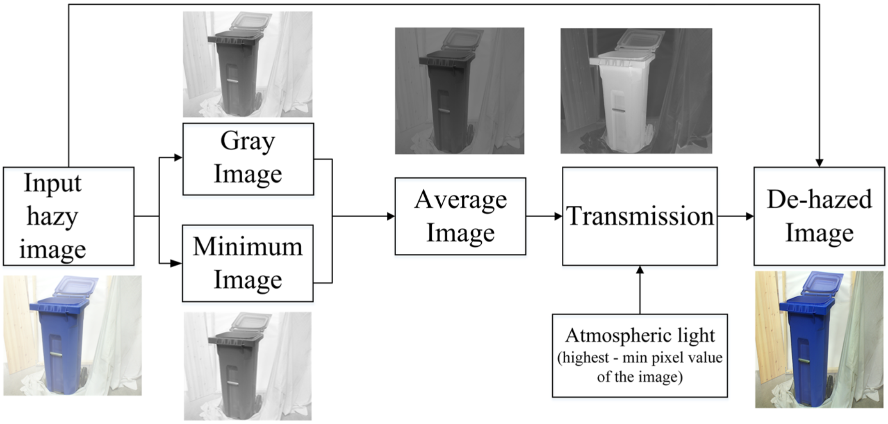


**Fig. 1** Block diagram of proposed algorithm

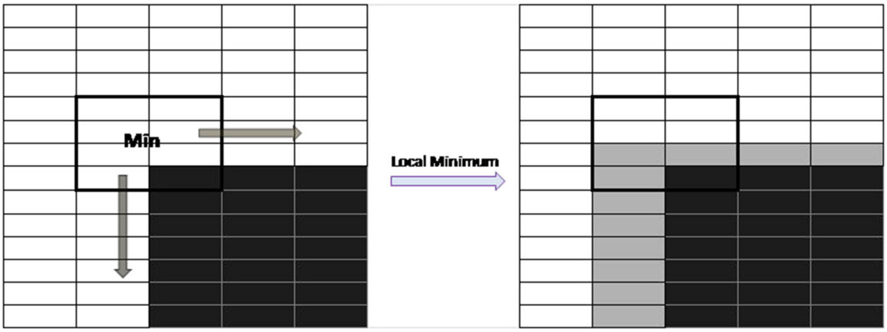


**Fig. 2** Halo effect

### 2.1 Estimation of Average Channel

The DCP method employed a minimum channel of the pixels as a dark channel of the image. As a result, it produces halos and artifacts in the de-hazed image. The effect of employing a minimum of pixels with a patch-based moment is represented in Fig. 2. The white region becomes shaded, resulting in will lead to producing undesired pixel intensity which causes a halo effect.

$$I_{average} = \frac{I_{dark} + I_{gray}}{2} \tag{3}$$

The $I_{dark}$ can be estimated using [11]. The $I_{gray}$ can be estimated using (4). The mean of these can be termed as $I_{average}$ and it significantly reduce the halos and artifacts present in the final dehazed image.

$$I_{gray} = 0.3 * R + 0.59 * G + 0.11 * B \tag{4}$$

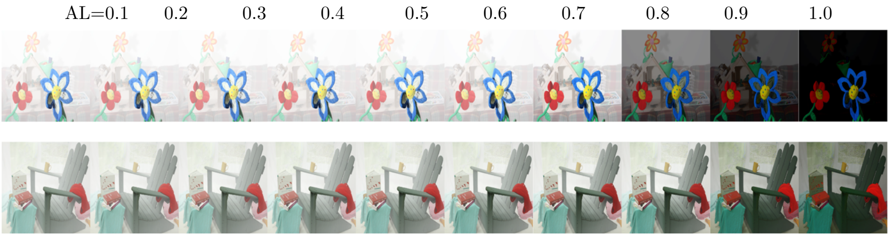
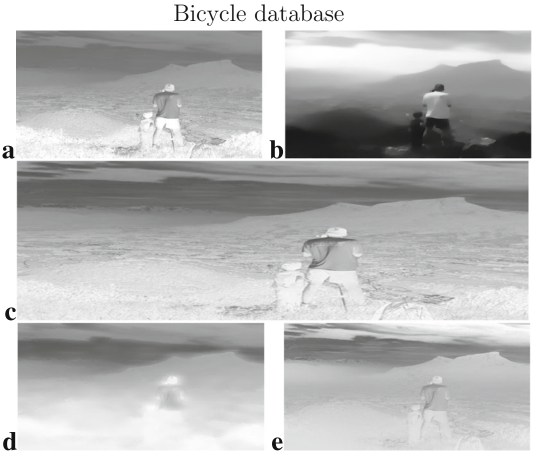


**Fig. 3** Variation of atmospheric light on hazy images (Flower, Table database)

**Fig. 4** Transmission depth map of different algorithms **a**.Zhu et al. **b**. Tarel et al., **c**.Our Proposed **d**. He et al., **e**. Tripathi et al.



## 2.2 Estimation of Atmospheric Light (AL)

It is noticed that the value of the air light is always closed to higher pixel value within the image. By conducting several simulation experiments in MATLAB with different values of atmospheric light ranging from 0 to 1 for floating-point datatype and 0 to 255 for unsigned integer format (uint8), the atmospheric light value is assumed as maximum pixel value within the dark image. It can be calculated as "$255 - min$" for uint8 format and "$1 - min$" for floating-point format. Figure 3 represents the effect of variation of atmospheric-light on de-hazed images, it is observed that the flower database produces accurate de-hazing output at AL=0.7, whereas table database at AL=0.9.

## 2.3 Estimation of Transmission Map

The transmission can be estimated by normalizing the average image with atmospheric light. Later, recovered scene radiance can be obtained as a de-hazed image. The transmission map of various algorithms are visually compared and is shown in Fig. 4. It is observed that our proposed method is estimated an accurate transmission map when compared with other existing algorithms.

### 2.4 Performance Evaluation Metrics

Evaluation metrics gives information about which scheme of the method can be preferable. In this paper, five performance metrics are used for evaluating the efficiency of the proposed algorithm.

#### 2.4.1 PSNR and MSE

PSNR and MSE are well-identified performance metrics for measuring the degree of error because these represent overall error content in the entire output image. PSNR is defined as the "logarithmic ratio of peak signal energy $(P)$ to the mean squared error (MSE) between output $N_i$ and input $M_j$ images". It can be expressed as

$$PSNR = 20 \times log \left[ \frac{max(P))}{\sqrt{MSE}} \right] \tag{5}$$

$$MSE = \frac{1}{kn} \sum_{i=0}^{k-1} \sum_{j=0}^{n-1} \left\| M_i - N_j \right\|^2 \tag{6}$$

where $P$ is the maximum value of the pixel in an image (typical value of $P = 255$), MSE is mean squared error, $k, n$ are the no. of rows, no. of columns of the image, respectively. Generally, the value of PSNR would be desirably high.

#### 2.4.2 Computation Time or Average Time Cost (ATC)

It represents the amount of time needed to complete an algorithm. The unit is in seconds. The average time cost of our proposed algorithm is estimated with the other four existing algorithms in 21 number of iteration.

#### 2.4.3 Average Contrast of Output Image (ACOI)

$$C = \frac{1}{MN} \left( max \left( \frac{L_{max} - L_{min}}{L_{max} + L_{min}} \right) \right) \tag{7}$$

Here, $L_{min}$ $L_{max}$ is the minimum, maximum luminance of an output image correspondingly. $M$, $N$ row, and column of an output image, respectively. Generally, the maximum value of it directs that an image is more quality.

The proposed algorithm is qualitatively and quantitatively compared in MATLAB 2017a with four existing state-of-art algorithms, namely Zhu et al. [31], Tarel et al. [26], He et al. [11] and Tripathi et al. [27] in terms of PSNR, ATC, ACOI, MSE, PHI, and SSIM [29] for an input image dimensions of 640 × 480. The comparison of quantitative metrics of the proposed algorithm is represented in Table 1 and corresponding visual comparison using qualitative metrics are represented in Fig. 5. It is observed that the proposed algorithm takes an optimum amount of execution time. The SSIM value indicates that the proposed method produces good quality of the

**Table 1** Comparison of quantitative metrics

| Database | Method | PSNR(dB) | ATC (s) | ACOI | MSE | PHI | SSIM |
|---|---|---|---|---|---|---|---|
| Men | Zhu et. al. | 22.19 | 1.478520 | 0.0107 | 1.52 | 12.1976 | 0.6342 |
| | Tarel et. al. | 22.54 | 2.885601 | 0.6080 | 1.53 | 14.2584 | 0.0236 |
| | He et. al. | 22.87 | 2.038838 | 0.1258 | 1.78 | 10.4658 | 0.7676 |
| | Tripathi et. al. | 22.16 | 1.628883 | 0.5399 | 1.69 | 27.0457 | 0.6749 |
| | Proposed | **23.54** | **1.155503** | **1.4617** | **1.34** | **27.3232** | **0.7736** |
| Play Table | Zhu et. al. | 22.99 | 1.44128 | 0.0123 | 1.63 | 12.976 | 0.7612 |
| | Tarel et. al. | 22.99 | 2.885601 | 0.0123 | 1.63 | 10.3668 | 0.0482 |
| | He et. al. | 22.99 | 2.038838 | 0.0123 | 1.63 | 7.4004 | 0.6813 |
| | Tripathi et. al. | 21.76 | 1.628883 | 0.4578 | 2.35 | 25.5258 | 0.7423 |
| | Proposed | **23.78** | **1.052765** | **1.2154** | **1.58** | **26.7324** | **0.8395** |
| Shelves | Zhu et. al. | 23.14 | 1.455901 | 0.0488 | 2.83 | 1.3772 | 0.7642 |
| | Tarel et. al. | 22.19 | 1.813378 | 0.0487 | 2.43 | 0.4644 | 0.0125 |
| | He et. al. | 22.62 | 2.024570 | 0.0046 | 1.93 | 1.5700 | **0.8431** |
| | Tripathi et. al. | 23.27 | 1.631953 | 0.1752 | 1.44 | 4.5346 | 0.2355 |
| | Proposed | **23.59** | **0.894809** | **1.4300** | **0.54** | **9.0500** | 0.8337 |
| Storage | Zhu et. al. | 22.76 | 1.418704 | 0.0329 | 1.61 | 3.2373 | 0.8166 |
| | Tarel et. al. | 22.57 | 1.906796 | 0.0341 | 1.76 | 6.6550 | 0.0917 |
| | He et. al. | 22.61 | 2.029526 | -0.0195 | 1.87 | 4.1131 | 0.8169 |
| | Tripathi et. al. | 22.82 | 1.619069 | 0.1238 | 1.53 | 5.9353 | 0.6730 |
| | Proposed | **23.16** | **1.148978** | **0.6168** | **0.31** | **10.6060** | **0.9139** |
| Shopvac | Zhu et. al. | 22.73 | 1.428657 | -0.0359 | 2.25 | 13.7933 | 0.8133 |
| | Tarel et. al. | 22.96 | 1.831700 | -0.0343 | 2.79 | 13.1588 | 0.5378 |
| | He et. al. | 22.58 | 2.020214 | -0.0686 | 2.53 | 18.9791 | 0.8152 |
| | Tripathi et. al. | 22.47 | 1.588451 | 0.1011 | 2.05 | **34.3493** | 0.53757 |
| | Proposed | **54.95** | **0.796819** | **2.0290** | **2.02** | 3.6349 | **0.9382** |
| Classroom | Zhu et. al. | 23.11 | 1.393388 | 0.0448 | 1.75 | 16.6764 | 0.7069 |
| | Tarel et. al. | 23.72 | 1.845369 | 0.0964 | 1.82 | 7.6522 | 0.2151 |
| | He et. al. | 22.58 | 2.029721 | 0.0193 | 1.61 | 13.1743 | 0.8049 |
| | Tripathi et. al. | 23.09 | 1.602608 | 0.3582 | 1.54 | 18.9678 | 0.2151 |
| | Proposed | **23.86** | **1.146288** | **1.8996** | **1.13** | **23.6419** | **0.9089** |
| Umbrella | Zhu et. al. | 23.50 | 1.525979 | 0.017 | 1.29 | 5.2107 | 0.7973 |
| | Tarel et. al. | 22.65 | 1.773976 | 0.0257 | 1.71 | 1.2454 | 0.0222 |
| | He et. al. | 23.78 | 1.964879 | -0.0265 | 1.95 | 6.6677 | 0.6674 |
| | Tripathi et. al. | 22.59 | 1.603365 | 0.2060 | 1.29 | 4.9916 | 0.1274 |
| | Proposed | **24.58** | **1.295478** | **2.5145** | 2.58 | **9.7845** | **0.8121** |
| Library | Zhu et. al. | 22.35 | 0.999786 | 3.8515 | 2.61 | 10.7208 | 0.6732 |
| | Tarel et. al. | **31.15** | 1.845405 | 3.0461 | 4.83 | 34.6196 | 0.1156 |
| | He et. al. | 16.74 | 2.388612 | 4.9670 | 2.11 | 20.1557 | 0.7420 |
| | Tripathi et. al. | 22.83 | 1.635244 | 2.4354 | 2.19 | 27.7894 | 0.1156 |
| | Proposed | 24.35 | **0.853564** | 1.4576 | 2.12 | **15.7623** | **0.7702** |

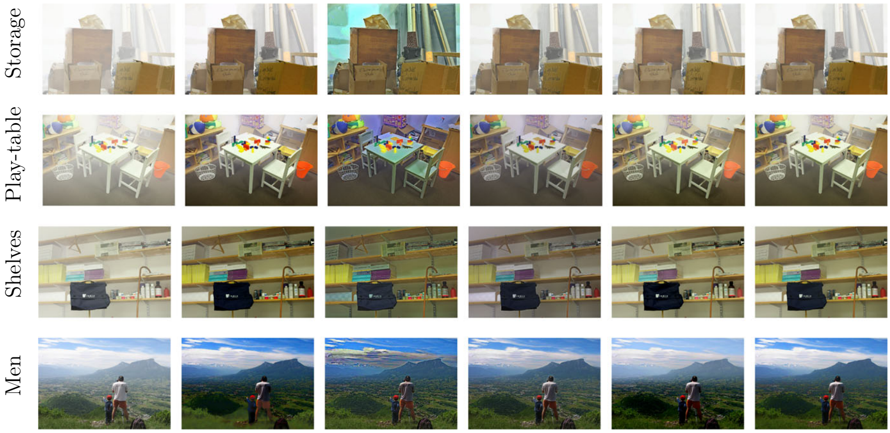


**Fig. 5** Simulation results of de-hazing Techniques (Input hazy image, output of Zhu et al., Tarel et. al., He et. al., Tripathi et. al., Proposed from left to right.)

de-hazed images. The PSNR and PHI values of our method are high which indicating that haze particle elimination, as well as visualization of a de-hazed image is good. Thereby, it eliminates hallos & artifacts in the recovered output image, reduces the time-consuming computational step of refining transmission in DCP, and produces accurate transmission map. The bold value in the Table 1 is indicating that the better evaluation metric value produced by the corresponding method over other methods.

## 3 Hardware Architecture

### 3.1 Implementation on Zynq-706 FPGA

Since the existing state-of-art algorithms are difficult to implement on the hardware platform, the 14-stage pipeline structure based on Zynq-706 [8] FPGA for single image haze removal is designed. The implemented algorithm takes full advantage of powerful parallel processing and the ability to perform the same on the hardware platform. Figure 6 represents the hardware architecture of the proposed de-hazing algorithm on Zynq FPGA. Initially, the design includes RAM inference using MIF (Memory Initialization File), which specifies the initial content of a memory block. Three .mif files are sufficient for storing the input image data. With the help of a counter, the data which is stored in the .mif file can be invoked. In the proposed design, Each .mif stores 256 × 256=65536 pixels data. Therefore, A 16-bit counter is used for extracting the address of 64Kb .mif data. Then, three 8-bit comparators are employed for finding minimum pixel among R, G and B, followed by a gray image of the input image using shift operations can be obtained using the following equation.

$$I_{graymodified} = 0.25 * R + 0.5 * G + 0.25 * B \tag{8}$$

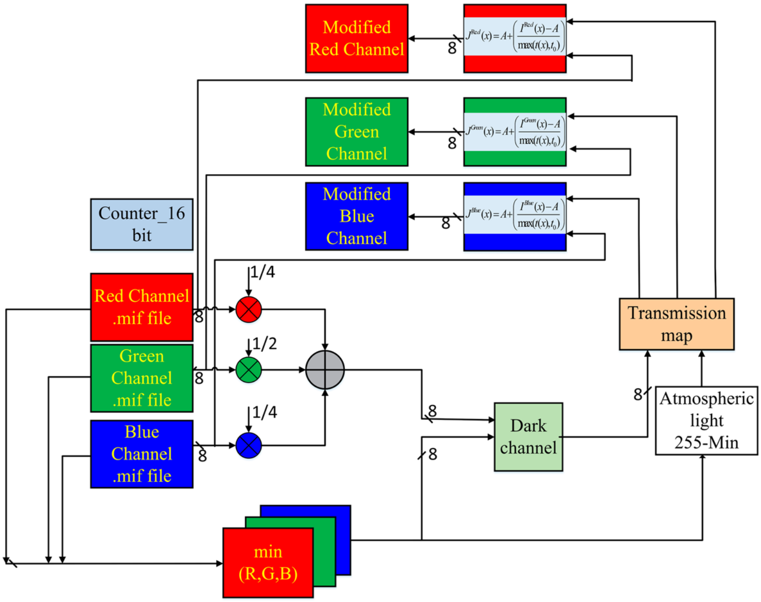

Modified Red Channel
$J^{Red}(x)=A+\left(\frac{I^{Red}(x)-A}{\max(t(x),t_0)}\right)$
8
Modified Green Channel
$J^{Green}(x)=A+\left(\frac{I^{Green}(x)-A}{\max(t(x),t_0)}\right)$
8
Modified Blue Channel
$J^{Blue}(x)=A+\left(\frac{I^{Blue}(x)-A}{\max(t(x),t_0)}\right)$
8
Counter_16 bit
Red Channel .mif file
Green Channel .mif file
Blue Channel .mif file
1/4
1/2
1/4
8
8
8
Transmission map
8
Atmospheric light 255-Min
8
8
Dark channel
min (R,G,B)

**Fig. 6** Proposed Hardware architecture of image de-hazing

The (8) is not the same as the traditional definition (4). Still, it makes almost no difference. It is much easier to be implemented in hardware design, to reduce the resource utilization, we employed a cut-off operation (shifting) instead of multiplication, thereby achieving a gray image of an input.

Later, a 10-bit adder is used and then truncating the LSB of the sum and carry bit so that an 8-bit data is given as input to further stages. Once the gray and dark channel is achieved, the mean of them using left shift operation is considered as the average channel of a hazy input image. Since the uint8 format of the image is employed, the value of atmospheric light calculated as the “255-minimum” pixel value of the input image. Moreover, the visibility content of the output is not much effected compared with the software platform. The normalization of the average channel to the atmospheric light results in a transmission map. An 8-bit divider circuit is employed for this operation. The above process is run in parallel for R, G, and B channels of the input image. Finally, using (2) modified pixel values are obtained as a de-haze image. The detailed 14-stage pipelining structure in the form of RTL of the proposed algorithm on the Zynq-706 platform using HDL is represented in Fig. 7. In order to obtain the output in a systematic manner, we introduced two new Intellectual Property (IP) cores, i.e., Virtual Input/ Output (VIO) and Integrated Logic Analyzer (ILA) these cores are responsible for verifying the de-haze algorithm. Figure 8 represents the top-level arrangement of IP cores with the de-haze algorithm.

Once the algorithm is implemented on hardware, verification of the algorithm can be done using VIO console. Figure 9 represents simulation results of the handwritten HDL code of image de-hazing. It can be observed that the first three signals within the waveform are input red, green and blue pixels, respectively, the corresponding gray,

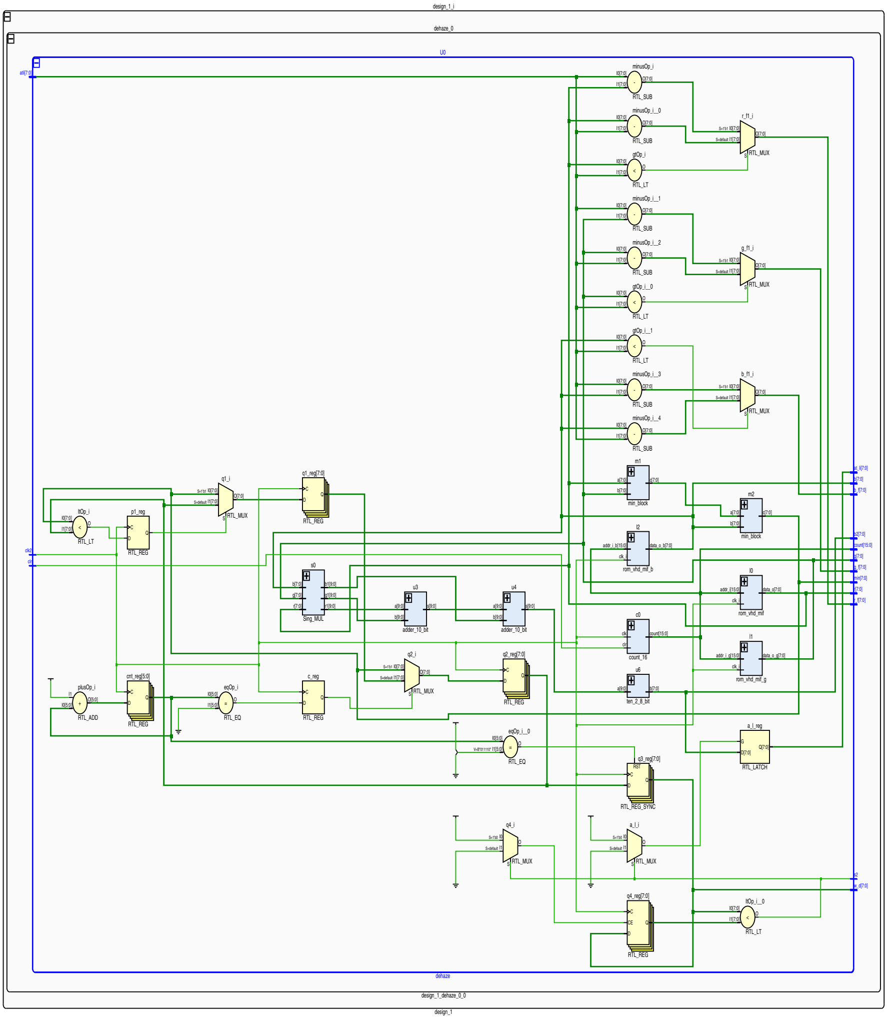


**Fig. 7** Schematic and elaborated design structure

dark channel and average channel of these pixels are indicated in the bottom of these waveforms. It is very hard to analyze for 256 × 256 = 65536 pixels. But, using the Tcl command the corresponding pixel values which are currently being executed on the FPGA can be exported to the spreadsheet, and qualitative analysis is verified in MATLAB.

### 3.2 Implementation on DSP Processor (TMS320C6748)

Based on DaVinci technology, an ideal core power for signal processing as well as image processing applications, the lowest cost DSP device has to choose. For which

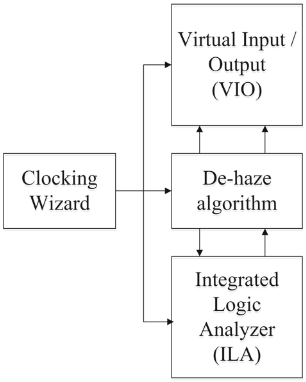
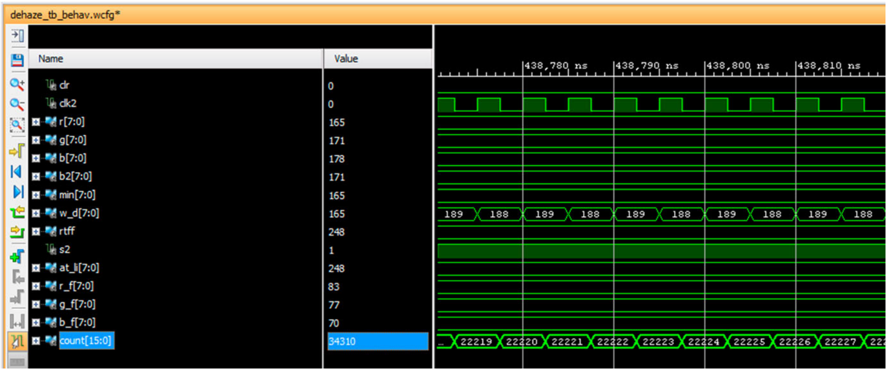


**Fig. 8** Test setup of de-hazing algorithm in Vivado


dehaze_tb_behav.wcfg*

| Name | Value |
|---|---|
| clr | 0 |
| clk2 | 0 |
| r[7:0] | 165 |
| g[7:0] | 171 |
| b[7:0] | 178 |
| b2[7:0] | 171 |
| min[7:0] | 165 |
| w_d[7:0] | 165 |
| rtff | 248 |
| s2 | 1 |
| at_li[7:0] | 248 |
| r_f[7:0] | 83 |
| g_f[7:0] | 77 |
| b_f[7:0] | 70 |
| count[15:0] | 34310 |

**Fig. 9** Simulation results of image de-hazing system

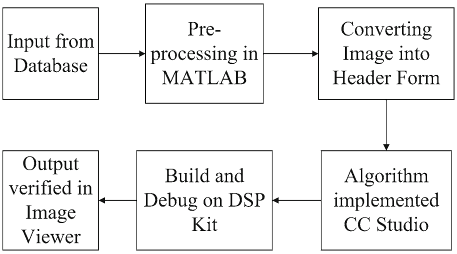

Input from Database
Pre-processing in MATLAB
Converting Image into Header Form
Algorithm implemented CC Studio
Build and Debug on DSP Kit
Output verified in Image Viewer

**Fig. 10** Flow of hardware implementation on TMS320C6748

various kinds of DSP processors have been compared. Finally, TMS320C6748 [30] is chosen to be the core device of the system. The TMS320C6748 evaluation module is chosen as the hardware platform of data processing, and Code Composer [21] V6.0 simulator is chosen to implement simulation.

The hardware implementation flow of the proposed algorithm on TMS320C6748 is presented in Figure 10. In initial step, an input image is taken from the standard database of size 256 by 256 for experimental verification. At the pre-processing stage, an image is cast into a standard datatype 'floating point (double precision)' format. The implementation of the de-haze algorithm in the code composer as follows. In

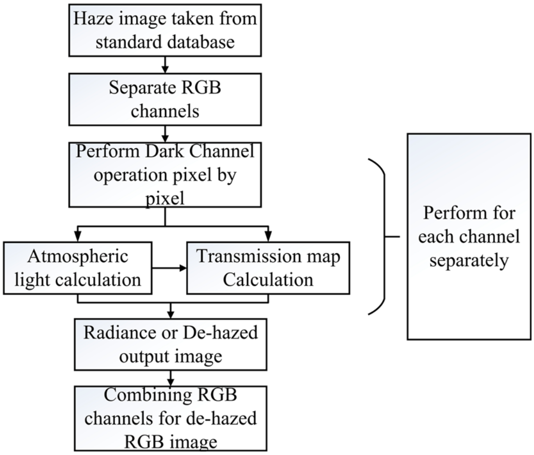

Haze image taken from standard database
Separate RGB channels
Perform Dark Channel operation pixel by pixel
Atmospheric light calculation
Transmission map Calculation
Radiance or De-hazed output image
Combining RGB channels for de-hazed RGB image
Perform for each channel separately

**Fig. 11** Basic Steps involved in hardware implementation of image De-hazing using TMS320C6748

the first step, we implemented a minimum filter for an image of M × N × 3 pixels. Using (4) gray image of the input image is calculated. The overall cost O(2 × M × N) time. Since the floating format of the image is employed, therefore, in transmission estimation each pixel value is escalating to 1. The value of atmospheric light assumed as the "1-minimum" pixel value of the image. Figure 11 represents basic steps involved for implementing image de-hazing on DSP TMS320C6748 processor. The process is applied on each color channel individually and combined at the final step.

## 4 Results and Discussion

The validity of the proposed algorithm, which is implemented on two hardware platforms is verified using a well-known standard database of Middlebury [3] and from the Flickr website. To speed up the algorithm while implementing it on Zynq FPGA, the "uint8" format for representing the pixels is employed. Consequently, there is a negligible amount of loss in information resulting in the variation of the percentage of haze improvement. Whereas the floating-point operations are added as an advantage for obtaining better results in DSP processor-based implementation. In these two implementations, a refined transmission map (Guided filter in case of DCP) was eliminated, and this technique is helpful for disallowing the artifacts and hallows. The intensity variation between the haze and de-hazed image can be observed in Fig. 12. Here, red, green and blue lines represent the variation of RGB pixels of the image. It is observed that the brightness or intensity value of the de-hazed image is less compared with haze image pixels.

Table 2 indicates the speed of operation of the above-mentioned techniques. It is observed that the processing time required for executing the haze removal algorithm is optimal in two hardware platforms. The Zynq-706 FPGA with HDL [17] based implementation took less amount of time when compared with DSP processor-based implementation. Since a 14-stage pipeline line structure is used, resulting in a number

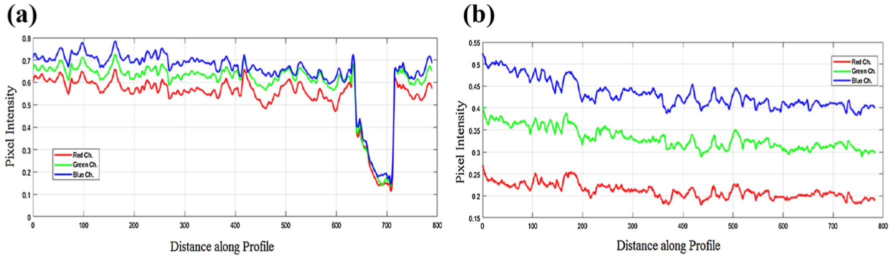


**Fig. 12** Sharp variation of brightness of the image pixels **a**. Haze image **b**.Dehazed image

**Table 2** Execution time on hardware platforms

| Parameter | On TMS320C6748 | On Zynq-706 Using HDL Code |
|---|---|---|
| Max. operating clock frequency (MHz) | 456 | 933 |
| Throughput | 333Mpixels/s | 385Mpixels/s |
| Execution time | 4.8 ns | 3.59 ns |

**Table 3** Utilization summary of hardware resources

| Resources | Utilization % (HDL) | Available resources | Resource utilized |
|---|---|---|---|
| LUT | 1.12 | 218600 | 2664 |
| LUTRAM | 0.3 | 70400 | 211 |
| FF | 1.63 | 218600 | 3562 |
| BRAM | 44.04 | 545 | 240 |
| IO | 0.55 | 362 | 2 |
| BUFG | 12.5 | 32 | 4 |

of clock pulses ($256 \times 256 \times 3 \times 14 = 2752512$) are required for de-haze operation theoretically.

Hardware resource utilization summary of the proposed algorithm on the Zynq platform is presented in Table 3, and it is noticed that the algorithm took less number of hardware resources. A very less number of other resources (LUT (Look Up Table), LUTRAM, FF (Flip-Flops), IO (Input Output), and MMCM) are utilized when compared with BRAM. Since storing the entire image or frame is not an efficient way of using on-chip memory (BRAM) resources on FPGA. Such a storage mechanism (.mif based) generally affects BRAM memory resources. This algorithm requires approximately 44 percentage of BRAM resources, which can be considered as a major challenging issue in hardware implementation. It can be overcome by storing a small portion of the image data inside the FPGA using line buffer instead of BRAMs, i.e., a portion of the image data or few selected lines of image data can be stored in a line buffer for manipulation and then, the remaining part of the image will be stored. Figure 13 and Table 4 represent qualitative and quantitative comparison of de-haze algorithm, respectively. Since the cutoff operation has employed for find-

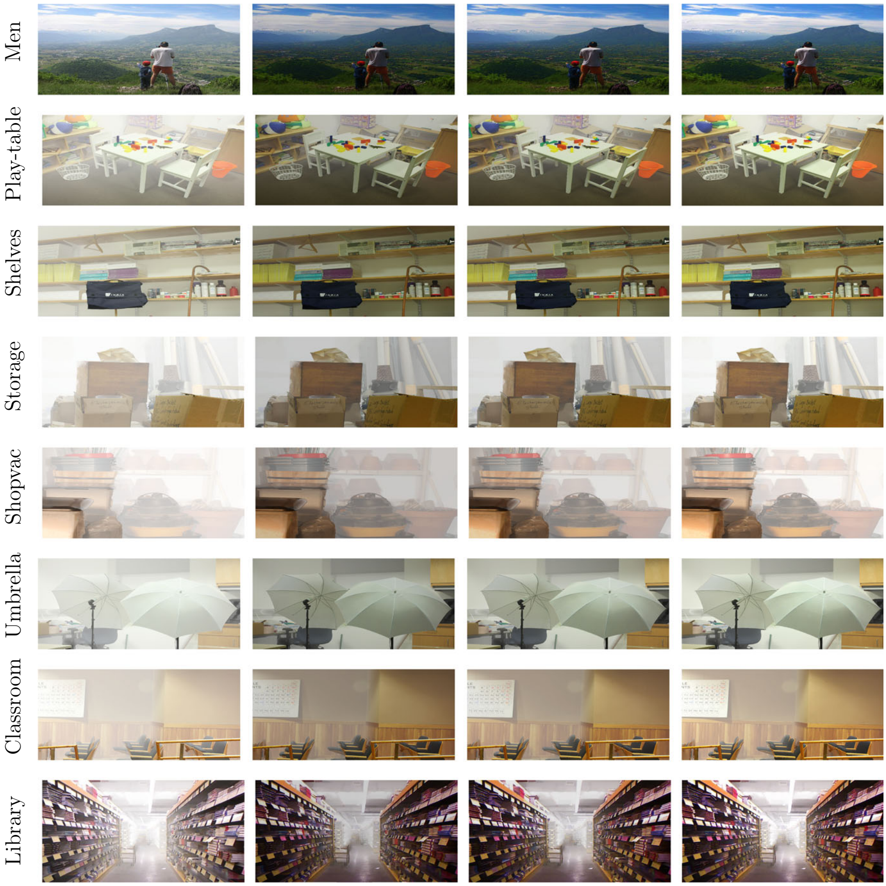


**Fig. 13** Implementation results of de-hazing system (Input hazy image, output on DSP processor, output on hand-written HDL based design, output of MATLAB from left to right)

ing the gray image in HDL-based implementation resulting in high MSE compared with processor-based implementation. PSNR and ACOI values are suggested that both hardware-based de-hazing algorithms produce a better result. When throughput is the user constraint, Zynq-based hardware platform is well suited and if the quality of the output image requirement is the user constraint then DSP processor-based hardware implementation is well suited for de-haze applications. The hardware test setup of the proposed algorithm is represented in Fig. 14.

## 5 Conclusion

In this paper, we have introduced a fast and efficient de-hazing algorithm well suited for implementing on any hardware platform. The entire work is carried out on i5 CPU 3 GHz and 8GB memory with OS window 8.1 with MATLAB 2017a. Vivado

**Table 4** Quantitative performance of de-haze algorithm on different hardware platforms

| Database | Method | ACOI | MSE | PSNR | PHI |
|---|---|---|---|---|---|
| Men | DSP | 0.4972 | 0.0706 | 29.8036 | 16.5585 |
| | HDL | 0.5000 | 0.0712 | 29.8222 | 15.4700 |
| Play Table | DSP | 0.4918 | 0.0963 | 28.9336 | 19.3389 |
| | HDL | 0.4887 | 0.1063 | 29.1463 | 14.3087 |
| Shelves | DSP | 0.4800 | 0.0358 | 29.2759 | 24.5587 |
| | HDL | 0.4984 | 0.0908 | 31.2973 | 23.4660 |
| Storage | DSP | 0.4725 | 0.0653 | 28.9938 | 37.8124 |
| | HDL | 0.4995 | 0.1034 | 29.9920 | 38.2875 |
| shopvac | DSP | 0.4952 | 0.0883 | 29.3367 | 38.7377 |
| | HDL | 0.2510 | 0.2847 | 26.7937 | 32.5731 |
| umbrella | DSP | 0.2036 | 0.2180 | 27.3732 | 16.8774 |
| | HDL | 0.2037 | 0.2142 | 27.4115 | 15.0265 |
| classroom | DSP | 0.5000 | 0.1152 | 28.7583 | 16.0200 |
| | HDL | 0.4514 | 0.2336 | 28.1334 | 08.3962 |
| Library | DSP | 0.4000 | 0.1241 | 27.3341 | 27.4581 |
| | HDL | 0.2608 | 0.2220 | 28.5963 | 11.7153 |

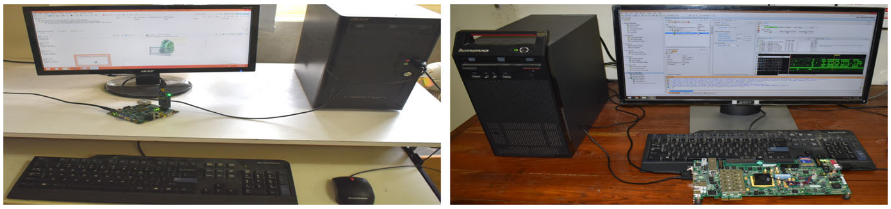


**Fig. 14** Hardware Platform Test Setup; Left: on DSP Processor TMS320C6748, Right: on Zynq 706 All programmable SoC

version of 2015.4 and Code composer studio v6.0 are used for implementation. A Zynq (xc7z045ffg900-2)-based hardware architecture, and DSP processor-based image de-hazing algorithms are implemented successfully. The RGB format always needs 24-bits to represent pixels. It will still affect the processing time and hardware resources. To overcome this issue, a YUV 4:2:2 format may be used. We consent to this issue in future research. The same algorithm can be extended for real-time video de-hazing in the future.

**Acknowledgements** The authors are thankful to SERB, Department of Science Technology, Govt. of India, for providing financial support under the grant of EEQ/2016/000556.

**Funding** This work was supported by the Science and Engineering Research Board (SERB) India, under the grant of EEQ/2016/000556.

**Data Availability Statement** Data sharing not applicable to this article as no datasets were generated or analyzed during the current study.

## Compliance with Ethical Standards

**Conflict of interest** The authors declare that they have no conflict of interest.

## References

1. F. Albu, C. Vertan, C. Florea, A. Drimbarean, One scan shadow compensation and visual enhancement of color images. in 2009 16th *IEEE International Conference on Image Processing (ICIP)* (pp. 3133-3136). IEEE (2009)
2. H. Ali, A. Sher, N. Zikria, LR ÜLGEN, a joint image dehazing and segmentation model. Turk J Electr Eng Comput Sci **27**(3), 1652–66 (2019)
3. C. Ancuti, CO. Ancuti, C. De Vleeschouwer, D-hazy: a dataset to evaluate quantitatively dehazing algorithms. in 2016 *IEEE International Conference on Image Processing (ICIP)* (pp. 2226-2230). IEEE (2016)
4. V. Andrearczyk, P.F. Whelan, Using filter banks in convolutional neural networks for texture classification. Pattern Recognit. Lett. **1**(84), 63–9 (2016)
5. L. Bai, Y. Wu, J. Xie, P. Wen, Real time image haze removal on multi-core dsp. Procedia Eng. **1**(99), 244–52 (2015)
6. B. Cai, X. Xu, K. Jia, C. Qing, D. Tao, Dehazenet: an end-to-end system for single image haze removal. IEEE Trans. Image Process. **25**(11), 5187–98 (2016)
7. A. Capra, A. Castrorina, S. Corchs, F. Gasparini, R. Schettini, Dynamic range optimization by local contrast correction and histogram image analysis. in: 2006 *Digest of Technical Papers International Conference on Consumer Electronics* (pp. 309–310). IEEE (2006)
8. L.H. Crockett, R.A. Elliot, M.A. Enderwitz, R.W. Stewart, *The Zynq Book: Embedded Processing with the Arm Cortex-A9 on the Xilinx Zynq-7000 All Programmable Soc* (Strathclyde Academic Media, Glasgow, 2014)
9. R. Fattal, Single image dehazing. ACM Trans. Graphics (TOG). **27**(3), 1–9 (2008)
10. G. Ge, Z. Wei, J. Zhao, Fast single-image dehazing using linear transformation. Optik. **126**(21), 3245–52 (2015)
11. K. He, J. Sun, X. Tang, Single image haze removal using dark channel prior. IEEE Trans. Pattern Anal. Mach. Intell. **33**(12), 2341–53 (2010)
12. S.C. Huang, F.C. Cheng, Y.S. Chiu, Efficient contrast enhancement using adaptive gamma correction with weighting distribution. IEEE Trans. Image Process. **22**(3), 1032–1041 (2012)
13. D.J. Jobson, Z.U. Rahman, G.A. Woodell, A multiscale retinex for bridging the gap between color images and the human observation of scenes. IEEE Trans. Image Process. **6**(7), 965–76 (1997)
14. J. Kopf, B. Neubert, B. Chen, M. Cohen, D. Cohen-Or, O. Deussen, M. Uyttendaele, D. Lischinski, Deep photo: model-based photograph enhancement and viewing. ACM Trans. Graphics (TOG). **27**(5), 1 (2008)
15. E.J. McCartney, *Optics of the Atmosphere: Scattering by Molecules and Particles* (Wiley, New York, 1976)
16. G. Meng, Y. Wang, J. Duan, S. Xiang, C. Pan, Efficient image dehazing with boundary constraint and contextual regularization. in *Proceedings of the IEEE International Conference on Computer Vision* (pp. 617–624) (2013)
17. J. Mermet, *Fundamentals and Standards in Hardware Description Languages* (Springer, Berlin, 2012)
18. S.G. Narasimhan, S.K. Nayar, Vision and the atmosphere. Int. J. Computer Vis. **48**(3), 233–54 (2002)
19. S. K. Nayar, S. G. Narasimhan, Vision in bad weather. in Proceedings of the Seventh IEEE International Conference on Computer Vision (Vol. 2, pp. 820–827). IEEE (1999)
20. M. Qi, Q. Hao, Q. Guan, J. Kong, Y. Zhang, Image dehazing based on structure preserving. Optik. **126**(22), 3400–6 (2015)
21. S. Qureshi, *Embedded Image Processing on the TMS320C6000TM DSP: Examples in Code Composer StudioTM and MATLAB* (Springer, Berlin, 2005)

22. Y. Y. Schechner, S. G. Narasimhan, S. K. Nayar, Instant dehazing of images using polarization. in Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001 (Vol. 1, pp. I–I). IEEE (2001)
23. P. Soma, R.K. Jatoth, H. Nenavath, Fast and memory efficient de-hazing technique for real-time computer vision applications. SN Appl. Sci. **2**(3), 1–10 (2020)
24. R.T. Tan, Visibility in bad weather from a single image. in 2008 *IEEE Conference on Computer Vision and Pattern Recognition* (pp. 1–8). IEEE (2008)
25. K. Tang, J. Yang, J. Wang, Investigating haze-relevant features in a learning framework for image dehazing. in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 2995–3000) (2014)
26. J.P. Tarel, N. Hautiere, Fast visibility restoration from a single color or gray level image. in 2009 *IEEE 12th International Conference on Computer Vision* (pp. 2201–2208). IEEE (2009)
27. A.K. Tripathi, S. Mukhopadhyay, Efficient fog removal from video. Signal Image Video Process. **8**(8), 1431–9 (2014)
28. W. Wang, B. Zhang, An improved visual enhancement method for color images. in *Fifth International Conference on Digital Image Processing (ICDIP 2013)* 2013 Jul 19 (Vol. 8878, p. 88780C). International Society for Optics and Photonics
29. Z. Wang, A.C. Bovik, H.R. Sheik, E.P. Simoncelli, Image quality assessment: from error visibility to structural similarity. IEEE Trans. Image Process. **13**(4), 1–14 (2004). https://doi.org/10.1109/TIP.2003.819861
30. P. Zahradnik, B. Simak, Implementation of Morse decoder on the TMS320C6748 DSP development kit. in 2014 6th *European Embedded Design in Education and Research Conference (EDERC)* (pp. 128–131). IEEE (2014)
31. M. Zhu, B. He, Q. Wu, Single image dehazing based on dark channel prior and energy minimization. IEEE Signal Process. Lett. **25**(2), 174–8 (2017)

**Publisher's Note** Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.