ASEAN Engineering Journal

Full Paper

SUB-0.4 W FULL-HD DARK CHANNEL PRIOR DEHAZING AT ~3.7 FPS ON EDGE DEVICES VIA AN HLS-BASED HARDWARE ACCELERATOR

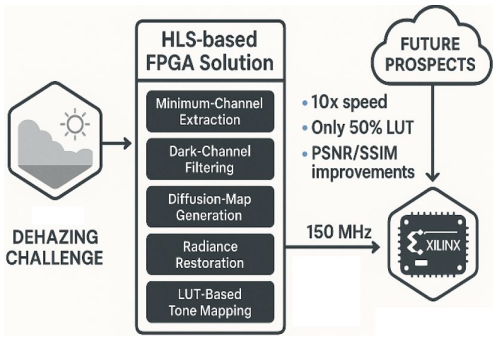
Van-Khoa Pham *, Tuan-Kiet Tran

Faculty of Advanced Education,Ho Chi Minh City University of Technology and Engineering,Ho Chi Minh City 700000, Vietnam

Article history
Received
09 June 2025
Received in revised form
25 January 2026
Accepted
25 February 2026
Published online
31 May 2026

*Corresponding author
khoapv@hcmute.edu.vn

Graphical abstract



Abstract

Fast single-image dehazing is essential for safety-critical, vision-driven edge systems, but common algorithms such as Dark Channel Prior (DCP) are often too compute-heavy for low-power platforms. This study presents a complete, sub-0.4 W FPGA accelerator for full-HD (1920×1080) DCP dehazing on the low-cost PYNQ-Z2. Built with Vitis HLS and Vivado, the design partitions the pipeline into five streaming hardware kernels—minimum-channel extraction, dark-channel filtering, diffusion-map generation, radiance recovery, and LUT-based tone mapping—coordinated by the on-board ARM cores via AXI4 interconnects. To improve restoration quality without inflating hardware cost, this study further introduces a spatially varying atmospheric-light map derived from an anisotropic diffusion matrix and implements it in fixed-point arithmetic. Operating at 100 MHz, the accelerator processes 1080p images in 0.27 s (≈3.7 fps), providing about a 10× speedup over an ARM-only implementation while consuming 27.6k LUTs and 55 DSPs. Image-quality evaluation reports 22.48 dB PSNR, 0.93 SSIM, and the best BRISQUE score (20.16) among the compared methods. Compared with prior ASIC designs and higher-end FPGA implementations, the proposed solution offers a stronger cost–throughput–resource balance, demonstrating the practicality of HLS-based FPGA acceleration for real-time edge dehazing under tight power budgets.

Keywords: Dark Channel Prior (DCP), Low-power Dehazing, Hardware-accelerated system, High-level synthesis (HLS), Peak Signal-to-Noise Ratio (PSNR).

1.0 INTRODUCTION

Low-latency image dehazing on resource-constrained edge devices presents significant challenges due to the computational demands of advanced algorithms. In safety-critical applications such as autonomous driving and surveillance, enhancing visibility under adverse weather conditions is essential for reliable image-based decision-making [1-4]. Among various dehazing techniques, the Dark Channel Prior (DCP) algorithm, introduced by He et al. [5], is notable for its effectiveness in restoring visibility from a single hazy image. However, its intensive per-pixel computational requirements limit its suitability for fast execution on conventional embedded CPUs, primarily due to high latency and energy consumption.

To address these limitations, Field-Programmable Gate Arrays (FPGAs) offer a compelling alternative for accelerating DCP on edge platforms. FPGAs provide a unique combination of reconfigurability and parallel processing capabilities, enabling the offloading of computationally intensive image processing tasks from the CPU. Unlike fixed-function Application-Specific Integrated Circuits (ASICs) or high-power Graphics Processing Units (GPUs), FPGAs are well-suited for deployment in power- and space-constrained environments, delivering high throughput via customized parallel datapaths [6]. Moreover, recent advancements in High-Level Synthesis (HLS) enable developers to describe algorithms using high-level languages such as C/C++, which significantly simplifies the design of complex FPGA-based accelerators [7]. Leveraging HLS makes it

feasible to implement the entire DCP dehazing pipeline in hardware, thus enabling high performance on edge devices.

This study focuses on the FPGA implementation and optimization of the DCP dehazing algorithm using a HLS approach. The motivation is to enable hardware-accelerated single-image dehazing on a low-cost edge device by accelerating the core computations in reconfigurable logic. This introduction has outlined the necessity for enhanced edge device performance for computer vision in foggy conditions and highlighted the potential of FPGA acceleration. The subsequent sections of this paper will review prior work on image dehazing and hardware acceleration, describe the proposed FPGA/HLS-based DCP architecture and its algorithmic enhancements, discuss the achieved performance and optimization results, and finally, conclude with the significance of this work for future edge computing applications.

2.0 METHODOLOGY

Fog and haze significantly degrade the performance of computer vision systems, particularly in intelligent transportation applications such as autonomous driving and traffic surveillance. Efficient fog removal or dehazing methods are critical for enhancing image clarity, object detection, and overall system safety. Recent research has explored both algorithmic enhancements based on the Dark Channel Prior method and hardware acceleration using FPGAs to meet high-throughput processing demands. The DCP-based approach, first introduced by He et al. [5], has become a foundational method for single-image dehazing due to its capability to estimate transmission maps by identifying pixels with low intensity in at least one color channel. However, standard DCP implementations are computationally intensive and not suitable for high-throughput applications in their raw form. Beyond classical DCP, new hardware schemes fuse multiple priors or even neural models. Ngo et al. [8] describe a VBI-accelerated FPGA dehazing engine that fuses the original image with a dehazed version based on a per-pixel haze estimation. This is achieved by performing computationally intensive tasks during the Vertical Blanking Interval (VBI) of the video output. In hardware, the accelerator processes 4K frames at real-time rates (hundreds of FPS) with efficient resource use. In the domain of learning-based approaches, Chen et al. [9] proposed a lightweight Convolutional Neural Network (CNN), SFA-Net, and co-designed an FPGA accelerator for it. The resulting implementation uses only 720 parameters and attains very high throughput (≈120 FPS) with 48% fewer logic resources than prior work. In each case, the emphasis is on simple, resource-efficient models and deeply pipelined logic so that low-latency dehazing is achieved on low-cost hardware.

To facilitate development and edge deployment, several works leverage high-level synthesis (HLS) and optimize for power. Shirai and Yamawaki [10] utilized HLS to prototype a haze removal pipeline on an FPGA. Although their HLS implementation did not achieve faster software execution times, it consumed over 25 times less dynamic power with minimal degradation in image quality [10]. This illustrates that the combination for HLS and bit-width reduction can dramatically lower power/area even if cycle count increases slightly. Similarly, the aforementioned Zynq-based systems achieve low latency by balancing the computational load between an ARM processor and FPGA logic.

Moreover, researchers have proposed various FPGA-based accelerators to achieve low latency and high throughput. An efficient FPGA implementation of DCP was developed to support low-latency video dehazing, achieving significant speed-ups compared to software-based solutions [11]. This approach is especially useful for high-throughput traffic monitoring and driver-assistance systems. Further improvements were proposed in [12], where the authors introduced MAPD, a mixed atmospheric prior-based dehazing technique. The method leverages atmospheric scattering models to better handle varying fog densities, and it was deployed on an FPGA to enable high-throughput video processing under different weather conditions. To address the memory and resource limitations of FPGA platforms, lightweight deep learning models, such as DCP-based U-Net architectures, have been proposed [13]. These models strike a balance between dehazing performance and hardware feasibility, enabling deployment on edge devices for smart traffic systems.

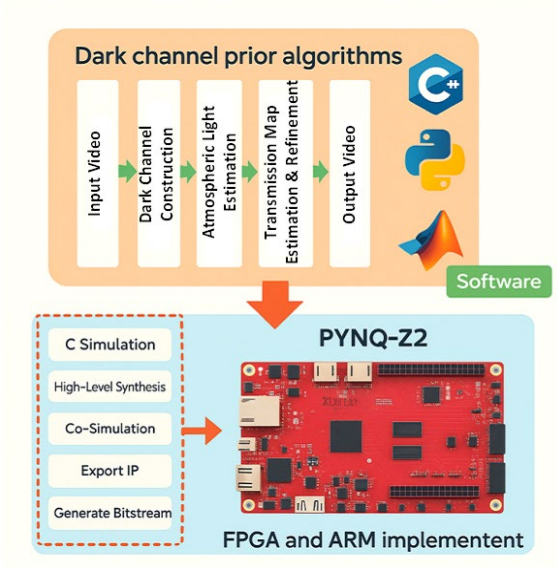


Figure 1 Software/Hardware Development Flow Diagram

The Dark Channel Prior (DCP) is based on the observation that, in most haze-free outdoor images, at least one color channel in a local patch contains very dark pixels; haze lifts these values due to airlight, so the dark channel becomes brighter. This provides a reliable cue for estimating transmission from a single image and recovering scene radiance using the atmospheric scattering model. Our pipeline follows the DCP flow in hardware-friendly stages: compute the per-pixel channel minimum, apply a local minimum filter to obtain the dark channel, estimate transmission, and restore the haze-free image, followed by a lightweight LUT-based tone adjustment. The main novelty is a diffusion-derived, spatially varying atmospheric-light factor instead of a single global airlight value. This spatial adaptation reduces bias from bright/sky regions and improves robustness under non-uniform haze, helping avoid over-dehazing and improving visual naturalness while remaining efficient for FPGA implementation.

This study presents a low-power Full-HD single-image dehazing accelerator based on the Dark Channel Prior (DCP) algorithm, implemented using FPGA-based High-Level Synthesis (HLS) on the PYNQ-Z2 board. The board's Xilinx Zynq-7020 SoC combines a dual-core ARM Cortex-A9 processor with programmable FPGA fabric, enabling the ARM system to manage control tasks while offloading intensive image processing to the FPGA. The development process utilized AMD Xilinx tools: Vitis HLS 2022.1 for C/C++ kernel implementation, Vivado for hardware integration, and the xfOpenCV library for efficient, reusable hardware blocks for operations such as filtering and color conversion. Figure 1 depicts a system architecture for implementing low-cost, sub-watt, edge-level image dehazing using FPGA technology based on the Dark-Channel Prior algorithm. Initially, the input video undergoes several software-level processing steps, including dark channel construction, atmospheric light estimation, and transmission map refinement, to generate the dehazed output. This algorithmic framework is developed and validated using software tools like C/C++, OpenCV, and MATLAB. Subsequently, the validated software algorithm is translated into hardware through an FPGA-based implementation workflow on a PYNQ-Z2 board. This process involves simulation, high-level synthesis, hardware-software co-simulation, IP core exportation, and finally, the generation of an FPGA bitstream. The result is an optimized, low-power hardware platform capable of performing edge-level image dehazing efficiently.

The conventional Dark Channel Prior algorithm relies on an interesting observation about the darkest channel in the original image. The image explains the concept behind the DCP algorithm for image dehazing. This observation posits that most non-sky pixels in hazy images contain very low intensity values in at least one color (RGB) channel, which provides information about the atmospheric light. By identifying these darkest pixel intensities locally—forming what is known as the 'dark channel'—the algorithm estimates the global atmospheric light and subsequently reduces haze. DCP not only removes haze but also improves image contrast and detail, benefiting tasks like object detection. The image also provides the mathematical definition of the dark channel as:

$$I^{dark}(x) = \min_{y \in \Omega(x)} (\min_{c}(I^{c}(y))) \qquad (1)$$

This formula computes the minimum intensity across all color channels and within a local patch $\Omega(x)$ around pixel x. A significant computational bottleneck in the DCP algorithm is the dark channel computation (Eq. 1), which involves nested loops to perform minimum operations across color channels within a local patch. For a 1920×1080 resolution image and a 3×3 patch, this results in 18.6 million inner comparisons per color channel, which is a computationally expensive operation for FPGAs in terms of cycle count and Block RAM (BRAM) usage. To address this bottleneck, the complex nested-loop structure is replaced by a single-pass 3×3 filter2D operation. This simplifies the process to one comparison per pixel, enabling high performance while maintaining the original mathematical intent.

The standard DCP algorithm was modified to better suit FPGA implementation and to improve dehazing in challenging regions. In the original DCP, a single atmospheric light value is used for the entire image. This work introduces an anisotropic diffusion matrix approach, wherein a spatially varying atmospheric light map is computed, acting as an 'adjustable fog factor'. This algorithmic innovation does incur some additional computations, but these were carefully optimized for hardware. The core steps of the proposed algorithm implemented are:

- Min Channel Calculation computes a grayscale "min" image where each pixel is the minimum of the R, G, B channels of the input foggy image. This corresponds to the dark channel's initial step. We implemented this as a hardware kernel IP_minMat, which scans through the 1920×1080 image and finds the per-pixel min.
- Dark Channel Filtering performs a 3×3 minimum filter on the "min" image to produce the dark channel image (each pixel is the minimum in a 3×3 neighborhood). In hardware IP_darkChannel, this was implemented using a 3×3 convolution with an all-ones kernel (via xf::cv::filter2D) to approximate a minimum filter, combined with scaling operations. The dark channel is also scaled as part of empirical tuning.
- Diffusion Matrix Computation computes a diffusion matrix from the min image and dark channel. This stage, implemented by IP_diffIm, creates a refined diffusion map. In practice, it takes the pixel-wise minimum of the min image and dark channel, then scales it by a factor of 0.6. The result highlights regions that should maintain higher airlight.
- Restoration restores the scene radiance using the transmission map and diffusion matrix. The IP_restoreOut kernel carries out: for each pixel, separate the RGB channels, apply the inverse of the haze model. As the Vitis HLS Vision library does not natively support floating-point division, this operation was implemented using custom loops and fixed-point arithmetic. This module was the most complex and resource-intensive, as it processes three channels and includes a division operation – it required careful pipelining and used more FPGA resources and power than the other kernels.
- Look-Up Table Adjustment is a final optional step, a post-processing LUT is applied to the output image to adjust contrast or brightness if needed. We derived this LUT from the diffusion results. The IP_LUT kernel uses Xilinx's xf::cv::LUT function, taking the restored image and applying a predefined 256-value lookup table. This step has relatively low cost in hardware.

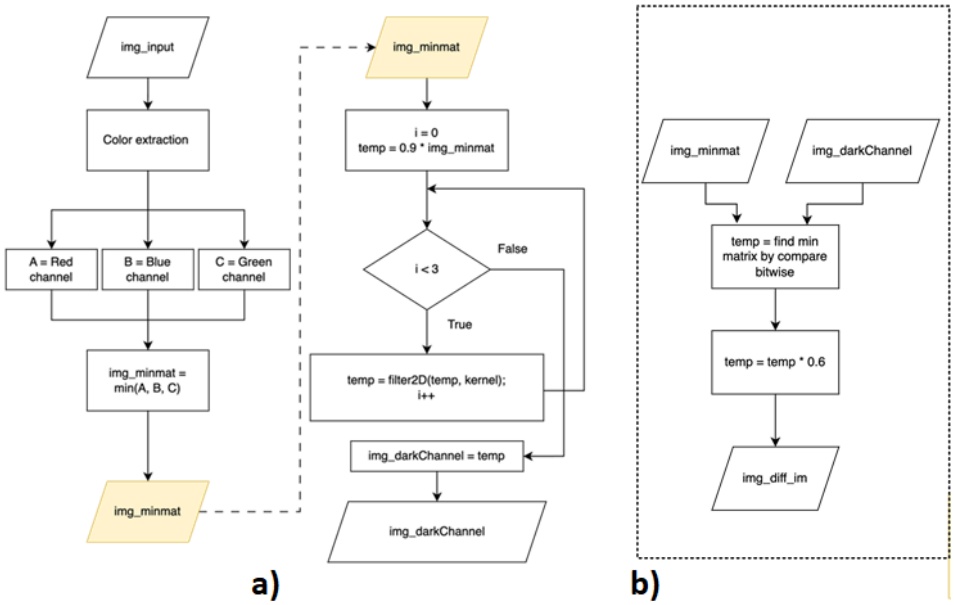
All five of these processing stages were implemented as separate HLS kernels (IP cores) to maximize modularity and allow each to be optimized independently. Table 1 maps these IP cores to their corresponding functions in software and the hardware library equivalents used. For example, the min and dark channel functions use simple loop logic or np.min in software, which we replaced with custom C++ loops and the xf::cv filtering in hardware; the color space conversions (RGB to gray, channel extract/merge) use provided xfOpenCV functions.

Table 1 Mapping of Software Functions to FPGA IP Cores and Hardware Library Equivalents

IP	Function	ARM/x86	FPGA
Minmat	Compute the per-pixel minimum across the three color channels to produce a single-channel image	np.min()	Using for loop + condition statement
Dark Channel	Multiple and add bitwise	Numpy bitwise add/multiply	xf::cv::convertScaleAbs()
	Filter 2D	cv2.filter2D()	xf::cv:: filter2d()
DiffIm	Multiple and add bitwise	Numpy bitwise add/multiply	xf::cv::convertScaleAbs()
	Find min 2 matrix	Numpy bitwise	xf::cv::min()
RestoreOut	Extract channel	Numpy array indexing	xf::cv::channelExtract()
	Merge channel	Numpy array indexing	xf::cv:: merge()
	Convert RGB to grayscale image	cv2.cvtColor	xf::cv:: rgb2gray()
	Calculate histogram	np.histogram	xf::cv:: calcHist()
	Duplicate matrix	Don't need on software	xf::cv:: duplicate()
	Triple matrix	Don't need on software	xf_triple()
	Find_LUT	Find_LUT	Find_LUT
LUT	Look up table	Using array index nested	xf::cv::LUT

To make the restoration stage feasible on the PYNQ-Z2 (Zynq-7020), we implement the radiance recovery using fixed-point arithmetic rather than floating point. This choice is primarily driven by hardware efficiency: fixed-point datapaths reduce DSP usage and critical-path delay compared with floating-point operators, enabling higher throughput under tight resource and power budgets. In addition, because the Vitis HLS vision primitives do not directly support the required floating-point division in this pipeline, the restoration step is implemented with custom fixed-point loops that are fully pipelined and optimized for FPGA execution.

We select fixed-point bit widths and scaling factors by tracking the dynamic range of the key terms in the haze model (input intensities, transmission-related factors, atmospheric-light modulation, and the restoration numerator/denominator) and allocating sufficient integer headroom to avoid overflow while reserving fractional precision to limit quantization error. To ensure numerical stability—especially in dense haze where transmission can become small—we clamp the effective denominator to a minimum threshold before division, which prevents noise amplification and halo artifacts. The restored outputs are then saturated to the valid 8-bit display range to avoid wrap-around artifacts. As a result, small brightness/tonal differences relative to the floating-point software reference can occur due to quantization and rounding, but these effects are bounded and can be further compensated by the final LUT-based tone adjustment stage.



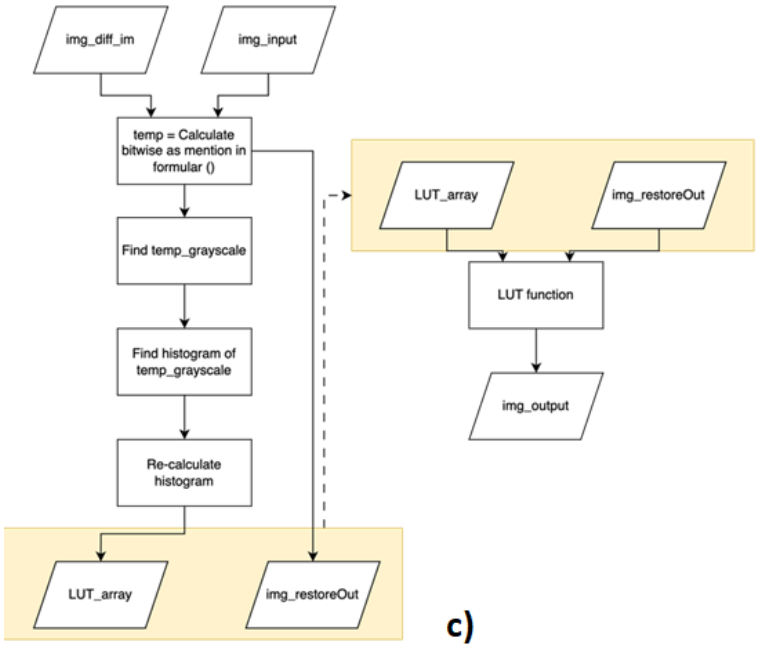


Figure 2 Proposed algorithms for (a) dark channel estimation, (b) diffusion matrix computation, and (c) image restoration

As depicted in Figure 2a, the modified dark channel estimation algorithm converts the incoming RGB video stream into a per-pixel minimum map (img_minmat) by splitting the color channels and selecting the minimum value among R, G, and B for each pixel. Instead of a computationally intensive large sliding-window minimum operation, this design processes the img_minmat map through three consecutive 3×3 filter2D convolutions, where each pass updates a temporary image and scales it by a factor of 0.9. Because the convolutions are separable and fully pipelined, only a few line buffers sit on-chip, and an initiation interval of one clock is maintained. After the third pass the result emerges as img_darkChannel, ready for subsequent masking while the original frame is still streaming, giving a memory-light, latency-hidden way of realising the core Dark Channel Prior on FPGA.

The diffusion matrix algorithm as shown in Figure 2b compares the freshly produced img_darkChannel and the img_minmat pixel-by-pixel to keep the darker of the two, thereby suppressing bright foreground objects that would otherwise distort the haze estimate. The retained value is multiplied by 0.6 to leave head-room for a later histogram stretch, forming img_diff_im. In the same pipeline stage, the algorithm computes the restoration formula by combining this mask with the original pixel stream, yielding an intermediate haze-free image while simultaneously converting it to grey-scale and accumulating a 256-bin histogram in a small dual-port BRAM. By the time the last pixel of the frame arrives, both a preliminary restored image (img_restoreOut) and a normalised histogram are ready, all without ever stalling the dataflow.

As shown in Figure 2c, the incoming frame (img_input) is combined with a pre-computed difference image to form a temporary stream on which an efficient, bit-wise approximation of the dark-channel minimum is performed; the result is converted to grayscale, its histogram is accumulated line-by-line, and that histogram is then reshaped into a cumulative distribution that implicitly encodes both the atmospheric light estimate and the transmission function. Those statistics are compacted into a 256-entry look-up table (LUT_array) that consumes only a few BRAM blocks. In the second stage, the freshly generated LUT is streamed back into a fully pipelined mapping unit that, on every clock cycle, reads one original pixel and produces one restored pixel (img_output) via simple table lookup, allowing the design to sustain full video rate while the histogram engine idles until the next frame. By separating the once-per-frame statistical pass from the per-pixel mapping pass and keeping all state on-chip, the architecture avoids external frame buffers, achieves deterministic sub-line latency, and fits comfortably within the logic, BRAM, and DSP budgets of modest Zynq-class FPGAs, all while freeing the on-board ARM cores for higher-level vision tasks.

These HLS IP cores were each synthesized to an RTL core with AXI interfaces. We adopted a software-driven sequential architecture for the data flow between these IPs, due to on-board memory limitations. In an ideal scenario, one could stream data directly from one IP to the next using AXI-Stream interfaces and achieve a fully pipelined hardware dataflow. However, on the PYNQ-Z2, the on-chip BRAM was not large enough to buffer full 1920×1080 frames between stages, and we opted to use external DDR memory as buffering between IPs. Thus, each IP was configured with AXI4 master ports for reading/writing image data to the PS DDR memory, and AXI4-Lite slave ports for control and configuration. All IP cores were integrated in Vivado IP integrator, connected to a common AXI interconnect. The ARM processor (PS) controls the sequence: it allocates a block of DDR memory for each intermediate image as shown in Figure 3, then triggers each IP in turn via writing to the AXI-Lite control registers. Each IP reads its input image from DDR and writes the output image back to DDR over the high-performance AXI bus. Although this introduces some off-chip memory latency between pipeline stages, it simplifies synchronization and ensures that each IP can run nearly independently.

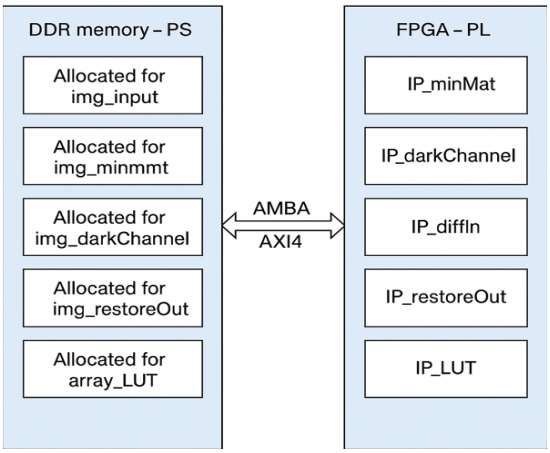


Figure 3 AMBA AXI4 interface enabling communication between PS and PL

It is important to distinguish between intra-kernel pipelining and inter-kernel scheduling in our implementation. Each HLS IP is internally optimized with pipelined loops so that, once started, it can sustain a high-throughput pixel processing rate (i.e., the kernel itself operates as a streaming engine). However, at the system level, the five kernels are executed sequentially because intermediate results are stored as full-resolution frames in external DDR and then consumed by the next stage. This orchestration is controlled by the ARM processing system, which allocates DDR buffers for each intermediate image and triggers each IP via AXI4-Lite control registers.

We adopt this DDR-buffered execution model mainly due to PYNQ-Z2 on-chip memory constraints: the available BRAM is insufficient to buffer multiple Full-HD (1920×1080) intermediate frames or to fully fuse the pipeline into a single on-chip streaming dataflow. Using external DDR buffering therefore simplifies synchronization and modular verification—each kernel can run nearly independently and write a complete output frame before the next stage begins. The trade-off is that repeated DDR read/write traffic increases memory latency and can become the dominant throughput limiter, particularly when multiple accelerators share the same DDR ports and AXI interconnect. In an ideal scenario, a fully streaming AXI-Stream architecture with line buffers would reduce off-chip transfers and improve end-to-end FPS, but it requires tighter cross-kernel synchronization and significantly more on-chip buffering than is practical on the PYNQ-Z2.

3.0 RESULTS AND DISCUSSION

Before deploying the modified Dark Channel Prior algorithm on an FPGA, it is essential to first validate its functionality on x86 or ARM platforms. On the PYNQ-Z2 board, this initial validation is performed on the Processing System (PS), which features a dual-core ARM Cortex-A9 CPU. Once the algorithm's performance is confirmed on the ARM platform, it is translated into C++ using the Xilinx Vision Library to prepare for hardware synthesis using Vitis HLS. Subsequently, the algorithm's output quality is evaluated using three widely recognized Image Quality Assessment (IQA) metrics: Structural Similarity Index Measure (SSIM), Peak Signal-to-Noise Ratio (PSNR), and Blind/Referenceless Image Spatial Quality Evaluator (BRISQUE). Higher SSIM and PSNR values indicate better visual fidelity, while lower BRISQUE scores suggest improved perceptual quality [14].

The Table 2 presents a comparative analysis of various image dehazing methods. Among the methods, Dehaze-Former exhibits the highest PSNR (37.86) and SSIM (0.99), indicating superior image quality and structural fidelity. However, it does not have the best BRISQUE score (21.29), suggesting slightly lower perceptual quality compared to others. Interestingly, this study achieves a strong balance, with a high SSIM of 0.93 (matching FFA-Net), a competitive PSNR of 22.48, and the lowest BRISQUE score (20.16), indicating that it delivers the best perceptual quality among all. Traditional methods like FVR, DCP, and BCCR show significantly lower PSNR and SSIM scores, reflecting inferior dehazing effectiveness. While FFA-Net achieves a high PSNR (31.10) and SSIM (0.93), its BRISQUE score (23.79) suggests it may not produce the most visually pleasing results. Overall, this study stands out for its effective trade-off between structural similarity and perceptual quality.

Table 2 Comparison of Image Quality Metrics (PSNR, SSIM, BRISQUE) for Software Evaluation of Various Dehazing Methods.

Method	PSNR	SSIM	BRISQUE
DCP [5]	17.33	0.84	20.67
FVR [15]	15.49	0.77	27.13
BCCR [16]	15.06	0.77	17.76
CAP [17]	18.28	0.75	21.57
NLD [18]	18.24	0.78	20.77
DehazeNet [19]	22.98	0.89	21.24
FFA-Net [20]	31.10	0.93	23.79
Dehaze-Former [21]	37.86	0.99	21.29
This Study	22.48	0.93	20.16

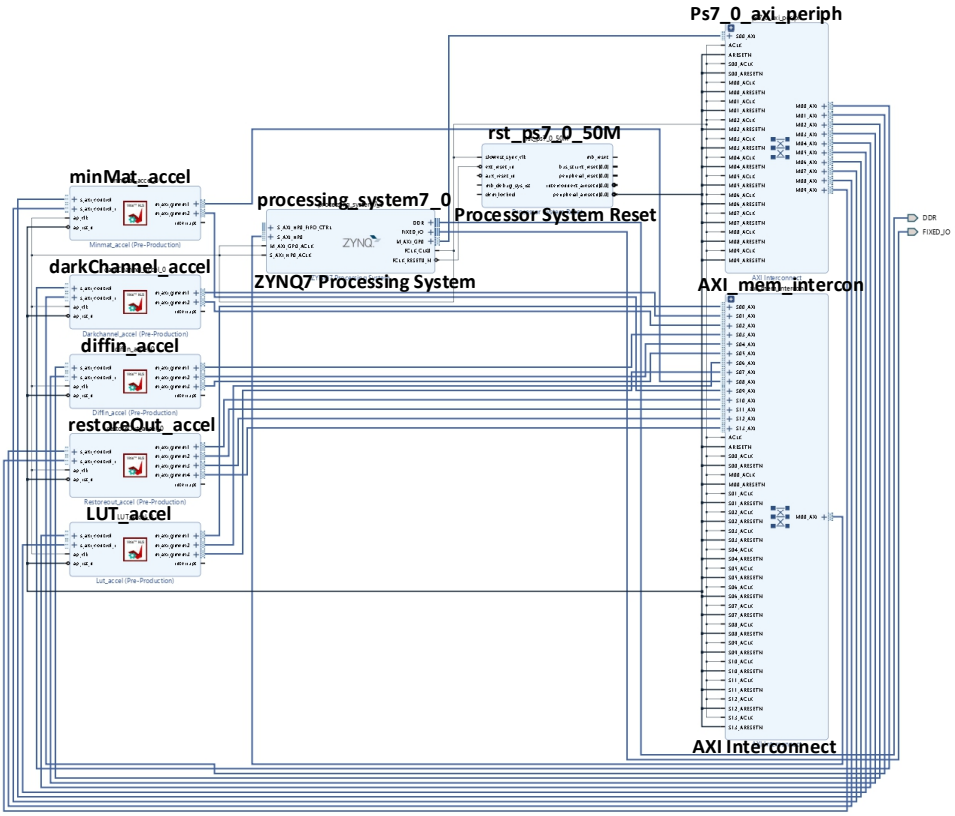


Figure 4 Vivado-generated block diagram of the dehazing accelerator chain on the Zynq-7000 SoC

The system-level integration on Vivado was automated with Tcl scripts. As illustrated in Figure 4, the resulting hardware design connects all five accelerators in parallel to the Zynq PS's memory ports. While this parallel connection enables potential concurrent operation, the current implementation executes them sequentially to mitigate memory contention. The entire FPGA logic utilization was well within the Zynq-7020 resources. The Minmat, DarkChannel, DiffIm, RestoreOut, and LUT IP blocks are connected to a shared AXI4 bus fabric that links them to the Zynq processing-system's high-performance DDR ports for data and to a lightweight GP0 → AXI-Lite cross-bar for control. Starting at the top of the pipeline, Minmat streams the incoming RGB frame and writes a single-channel image whose value at each pixel is simply the darkest of its R, G, B components. DarkChannel performs a sliding-window minimum on that map to produce the dark-channel image and extract the scene's atmospheric-light estimate. With those two results, DiffIm calculates a rough transmission map and the per-pixel difference. RestoreOut then reconstructs haze-free radiance, clamping the denominator to prevent noise amplification. Finally, LUT applies a programmable lookup table for tone-mapping or colour-correction, yielding a display-ready frame as

shown in Figure 4. The five IP cores are connected to the shared DDR interface but are invoked sequentially by the ARM processor.

The floorplan shown in Figure 5a illustrates the physical layout of the various components, including the FPGA fabric, the PS block, the memory interfaces, and the GPIO connections, all of which have been carefully orchestrated to achieve the desired system-level functionality and performance. According to the FPGA resource utilization presented in Figure 5b, the design utilized approximately 50% of Look-Up Tables (LUTs), 30% of Flip-Flops (FFs), and 11% of Block RAMs (BRAMs) available on the Zynq-7020, along with 55 out of 220 DSP slices. The most demanding module was the restoreOut IP, which used several DSPs for arithmetic. If a higher frame rate or resolution is required, one could duplicate certain IP cores to process tiles of the image in parallel. Alternatively, moving to a larger Zynq such as Ultrascale+ MPSoC would easily allow multiple pixel processing engines to be instantiated for higher throughput.

In addition to these quantitative assessments, visual inspection confirms that the dehazed images are noticeably clearer, though slightly darker, than the original images. The qualitative image results were generated using Python on the ARM processor of the PYNQ-Z2 board, with sample outputs presented in Figure 6. The dehazing algorithm was implemented on both ARM/x86 processors and FPGA, and tested using 1920×1080 Full HD sample images. A noticeable outcome was that the FPGA-generated images appeared slightly darker than those processed by the software implementations. This variation highlights an area for improvement in future iterations of the design. Beyond performance metrics such as runtime and power efficiency, the accuracy of image output between platforms is also critical. This discrepancy primarily arises from differences in data representation: FPGA hardware implementations typically operate using 8-bit integers for grayscale images and, for instance, 32-bit registers to process components of RGB images, whereas software implementations often utilize 32-bit floating-point values for each pixel component. This difference inherently introduces some variation in output, which is generally considered acceptable in many applications [22].

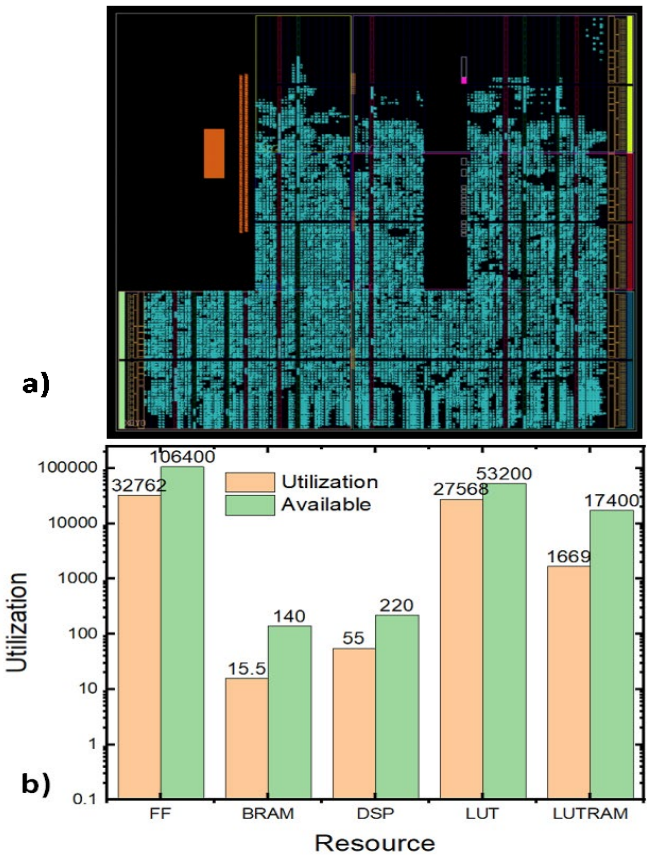


Figure 5 Proposed Dehazing Algorithm Implementation: (a) Post-placement and routing view on the PYNQ-Z2 FPGA; (b) Hardware Utilization

The hardware-accelerated implementation on the FPGA platform demonstrated a substantial performance improvement compared to the software-based solution. On the PYNQ board, which features a dual-core ARM Cortex-A9 processor running at 650 MHz, the DCP algorithm—executed in Python using a single core—required several seconds to process a single 1920×1080 The ARM/x86 processor executes pixel operations sequentially, the FPGA leverages pipelined processing to handle many pixels concurrently within each clock cycle. Further synthesis of the design at higher clock frequencies revealed that the image, resulting in a throughput of less than 0.3 frames per second (FPS). In contrast, the FPGA-based implementation processes the same Full HD image in approximately 0.27 seconds, achieving a throughput of roughly 3.7 FPS at a clock frequency of 100 MHz. This represents an approximate order-of-magnitude improvement in processing speed.

implementation remains stable up to approximately 150 MHz, where the throughput increases to about 5.5 FPS. At 200 MHz, the throughput exceeds 7 FPS. While this is still below the typical real-time video processing rate of 30 FPS, it constitutes a significant performance gain for a resource-constrained edge platform such as the PYNQ-Z2. Moreover, the throughput can be further enhanced through design optimizations, such as streaming IP integration to minimize DDR memory bottlenecks or deploying the design on a larger FPGA with greater parallelism. These results demonstrate the practical viability of processing high-resolution images on edge devices within a fraction of a second.

While achieving state-of-the-art performance was not the primary objective of our work, it is instructive to compare our results with prior hardware implementations of dehazing algorithms. While previous studies in Table 3 achieved higher frame rates and energy efficiency, they relied on custom ASICs or high-end FPGA platforms with significantly higher resource usage. In contrast, our study achieves Full HD (1920×1080) processing on a low-cost Zynq-7000 FPGA, making it a more practical solution for edge computing scenarios where cost, power, and accessibility are critical. This study demonstrates a balanced use of hardware resources, consuming only 27.6K LUTs and 55 DSP slices—substantially less than the 2013 and 2016 implementations—while still supporting high-resolution video processing. Although the frame rate (3.66 FPS) and 9.15 FPS/Watt are lower, the design operates within the constraints of an edge device and maintains functional performance suitable for non-real-time applications.

This work shows that a classical, physics-based dehazing pipeline can be made practical on a low-cost edge FPGA when restructured into hardware-friendly stages. By mapping the Dark Channel Prior (DCP) flow into five Vitis HLS kernels and adding a diffusion-derived, spatially varying airlight factor, the design achieves Full-HD processing at ~0.27 s per 1920×1080 frame at 100 MHz (≈3.6–3.7 FPS) with sub-watt power and moderate resource usage, offering a balanced quality–throughput–cost point on a Zynq-7020–class platform. The results reflect a pragmatic quality trade-off: while learning-based methods may

achieve higher PSNR/SSIM in benchmarks, the proposed approach produces visually plausible restorations with strong perceptual quality and competitive structural similarity, and the diffusion-based airlight modulation helps avoid over-dehazing in bright/sky regions to improve naturalness. Architecturally, the modular five-kernel chain simplifies development and tuning, but repeated full-frame transfers through external DDR make the system increasingly memory-bound, and the restoration stage remains the main arithmetic/DSP hotspot; consequently, the most effective improvements are to adopt a streaming/dataflow design (AXI-Stream/line buffers) to reduce DDR traffic, increase pixel parallelism specifically in restoration, and systematically calibrate fixed-point scaling and LUT/tone mapping to eliminate brightness shifts between FPGA and software.

Table 3 Performance and Resource Comparison with Prior Hardware Implementations of Dehazing Algorithms

Studies	2013 [23]	2014 [24]	2016 [25]	This Study
Tech.	TSMC 90nm	Zynq-7000	TSMC 65nm	Zynq-7000
Frame Size	1920x 1080	720x 576	1920x 1080	1920x 1080
LUT	92.9K	10.4K	64.8K	27.6K
DSP	-	75	-	55
Freq. (MHz)	100	104	297	100
Power (W)	0.285	-	0.198	0.4
FPS	29.89	72.82	59.79	3.66
FPS/Watt	104.88	-	301.97	9.15
HDL/HLS	Verilog	Verilog	Verilog	Vitis HLS

Future work will target higher throughput and more consistent visual quality while keeping the low-cost, HLS-based workflow. First, we will reduce the dominant DDR overhead by moving the current DDR-buffered multi-kernel chain toward a streaming/dataflow design using AXI-Stream and line buffers (at least for the early stages), minimizing intermediate frame writes and improving sustained FPS and latency. Second, since the restoration stage is the main DSP/arithmetic hotspot, we will increase pixel parallelism specifically there (multi-pixel-per-cycle, tiling/vectorization, selective replication), together with more efficient AXI bursts and aligned buffers. Third, we will address the brightness mismatch between FPGA and software through a systematic fixed-point study (bit-width, scaling, rounding) and a refined LUT/tone-mapping step to stabilize tone reproduction. Fourth, we will explore additional FPGA-friendly refinements (guided/edge-preserving transmission refinement, improved airlight estimation) and, optionally, a tiny post-correction model with bounded compute. Finally, we will strengthen validation by reporting end-to-end video latency and throughput and expanding experiments to standard datasets with ablations and task-level impact (e.g., detection under haze).

4.0 CONCLUSION

This study shows that High-Level Synthesis can bridge the gap between the heavy computation of modern image dehazing and the tight cost, power, and latency budgets of edge hardware. By partitioning the Dark Channel Prior method into five pipelined kernels and enhancing it with an anisotropic-diffusion atmospheric-light map, the proposed Full-HD (1920×1080) implementation achieves a tenfold runtime reduction relative to a software baseline. It also preserves strong reconstruction quality across PSNR, SSIM, and BRISQUE, uses only about half of the Look-Up Table capacity available on a Xilinx Zynq-7020 SoC, maintains stable operation up to 150 MHz, and provides a clear path to higher frame throughput on larger FPGA devices. In addition to offering a competitive cost–performance balance against prior FPGA and ASIC designs, the work delivers a fully synthesizable, resource-efficient DCP accelerator and a hardware-friendly diffusion-based refinement that reduces sky artifacts with minimal added computation. Future work will prioritize a streaming dataflow implementation to reduce off-chip memory traffic, explore quad-pixel and tile-parallel variants to reach 30 FPS video throughput, incorporate learned priors or lightweight CNN-based post-processing, and leverage dynamic partial reconfiguration to adjust quality–performance trade-offs during operation. Together, these directions position the framework as a portable and scalable dehazing solution for resource-limited edge platforms.

Acknowledgement

This research received no external funding. The authors sincerely thank Huy-Hoang Nguyen, Minh-Nhat Phan and Viet-Dung Tien Nguyen for their support and assistance with part of the verification work for this study.

Conflicts of Interest

The author(s) declare(s) that there is no conflict of interest regarding the publication of this paper.

References

[1] Mukhtar, A., Xia, L., & Tang, T. B. 2015. Vehicle detection techniques for collision avoidance systems: A review. IEEE Transactions on Intelligent Transportation Systems, 16(5): 2318–2338. DOI: https://doi.org/10.1109/TITS.2015.2409109

[2] Ge, W., Collins, R. T., & Ruback, R. B. 2012. Vision-based analysis of small groups in pedestrian crowds. IEEE Transactions on Pattern Analysis and Machine Intelligence, 34(5): 1003–1016. DOI: https://doi.org/10.1109/TPAMI.2011.176

[3] Lee, D.-G., Suk, H.-I., Park, S.-K., & Lee, S.-W. 2015. Motion influence map for unusual human activity detection and localization in crowded scenes. *IEEE Transactions on Circuits and Systems for Video Technology,* 25(10): 1612–1623. DOI: https://doi.org/10.1109/TCSVT.2015.2395752

[4] Tetila, E. C., Machado, B. B., Belete, N. A., Guimarães, D. A., & Pistori, H. 2017. Identification of soybean foliar diseases using unmanned aerial vehicle images. *IEEE Geoscience and Remote Sensing Letters*, 14(12): 2190–2194. DOI: https://doi.org/10.1109/LGRS.2017.2743715

[5] He, K., Sun, J., & Tang, X. 2010. Single image haze removal using dark channel prior. *IEEE Transactions on Pattern Analysis and Machine Intelligence,* 33(12): 2341–2353. DOI: https://doi.org/10.1109/TPAMI.2010.168

[6] Qasaimeh, M., Denolf, K., Lo, J., Vissers, K., Zambreno, J., & Jones, P. H. 2019. Comparing energy efficiency of CPU, GPU and FPGA implementations for vision kernels. In *2019 IEEE International Conference on Embedded Software and Systems (ICESS),* 1–8. DOI: https://doi.org/10.1109/ICESS.2019.8782524

[7] Cong, J., Lau, J., Liu, G., Neuendorffer, S., Pan, P., Vissers, K., & Zhang, Z. 2022. FPGA HLS today: Successes, challenges, and opportunities. *ACM Transactions on Reconfigurable Technology and Systems,* 15 (4): 1–42. DOI: https://doi.org/10.1145/3530775

[8] Ngo, D., Son, J., & Kang, B. 2025. VBI-accelerated FPGA implementation of autonomous image dehazing: Leveraging the vertical blanking interval for haze-aware local image blending. *Remote Sensing,* 17(5): 919. DOI: https://doi.org/10.3390/rs17050919

[9] Chen, Z., Du, G., Li, Z., Wang, X., & Yin, Y. 2025. A high-speed hardware accelerator for lightweight dehazing neural network based on SFA-Net. *Journal of Real-Time Image Processing,* 22(2): 99. DOI: https://doi.org/10.1007/s11554-025-01672-4

[10] Shirai, D., & Yamawaki, A. 2022. Development of haze removing hardware using high-level synthesis. In *Proceedings of the 2022 International Conference on Artificial Life and Robotics* (ICAROB 2022), 612–615. DOI: https://doi.org/10.5954/ICAROB.2022.OS15-4

[11] Kumar, R., Kaushik, B. K., & Balasubramanian, R. 2017. FPGA implementation of image dehazing algorithm for real-time applications. In: Tescher AG (ed) Applications of digital image processing XL, vol 10396. International Society for Optics and Photonics, Bellingham, 10396: 639–645. DOI: https://doi.org/10.1117/12.2274682

[12] Tan, Y., Zhu, Y., Huang, Z., Tan, H., & Li, K. 2023. MAPD: An FPGA-based real-time video haze removal accelerator using mixed atmosphere prior. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems,* 42(12): 4777–4790. DOI: https://doi.org/10.1109/TCAD.2023.3291670

[13] Han, Y., Kim, J., Lee, J., Nah, J.-H., Ho, Y.-S., & Park, W.-C. 2024. Efficient haze removal from a single image using a DCP-based lightweight U-Net neural network model. *Sensors*, 24(12): 3746. DOI: https://doi.org/10.3390/s24123746

[14] Mittal, A., Moorthy, A. K., & Bovik, A. C. 2012. No-reference image quality assessment in the spatial domain. IEEE Transactions on Image Processing, 21(12): 4695–4708. DOI: https://doi.org/10.1109/TIP.2012.2214050

[15] Tarel, J.-P., & Hautière, N. 2009. Fast visibility restoration from a single color or gray-level image. In Proceedings of the 2009 IEEE 12th International Conference on Computer Vision (ICCV). 2201–2208. DOI: https://doi.org/10.1109/ICCV.2009.5459251

[16] Meng, G., Wang, Y., Duan, J., Xiang, S., & Pan, C. 2013. Efficient image dehazing with boundary constraint and contextual regularization. In Proceedings of the IEEE International Conference on Computer Vision (ICCV), pp. 617–624. DOI: https://doi.org/10.1109/ICCV.2013.82

[17] Zhu, Q., Mai, J., & Shao, L. 2015. A fast single image haze removal algorithm using color attenuation prior. *IEEE Transactions on Image Processing,* 24(11): 3522–3533. DOI: https://doi.org/10.1109/TIP.2015.2446191

[18] Berman, D., Treibitz, T., & Avidan, S. 2016. Non-local image dehazing. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR),* 1674–1682. DOI: https://doi.org/10.1109/CVPR.2016.185

[19] Cai, B., Xu, X., Jia, K., Qing, C., & Tao, D. 2016. DehazeNet: An end-to-end system for single image haze removal. *IEEE Transactions on Image Processing*, 25(11): 5187–5198. DOI: https://doi.org/10.1109/TIP.2016.2598681

[20] Qin, X., Wang, Z., Bai, Y., Xie, X., & Jia, H. 2020. FFA-Net: Feature fusion attention network for single image dehazing. *Proceedings of the AAAI Conference on Artificial Intelligence,* 34(7): 11908–11915. DOI: https://doi.org/10.1609/aaai.v34i07.6865

[21] Song, Y., He, Z., Qian, H., & Du, X. 2023. Vision transformers for single image dehazing. *IEEE Transactions on Image Processing*, 32: 1927–1941. DOI: https://doi.org/10.1109/TIP.2023.3256763

[22] Forget, L., Harnisch, G., Keryell, R., & de Dinechin, F. 2022. A single-source C++20 HLS flow for function evaluation on FPGA and beyond. In *Proceedings of the 12th International Symposium on Highly Efficient Accelerators and Reconfigurable Technologies* (HEART 2022), 51–58. DOI: https://doi.org/10.1145/3535044.3535051

[23] Kao, C.-C., Lai, J.-H., & Chien, S.-Y. 2014. VLSI architecture design of guided filter for 30 frames/s Full-HD video. *IEEE Transactions on Circuits and Systems for Video Technology*, 24(3): 513–524. DOI: https://doi.org/10.1109/TCSVT.2013.2278145

[24] Liang, Z., Liu, H., Zhang, B., & Wang, B. 2014. Real-time hardware accelerator for single image haze removal using dark channel prior and guided filter. *IEICE Electronics Express,* 11(24): 20141002. DOI: https://doi.org/10.1587/ELEX.11.20141002

[25] Zhang, B., & Zhao, J. 2016. Hardware implementation for real-time haze removal. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems,* 25(3): 1188–1192. DOI: https://doi.org/10.1109/TVLSI.2016.2622404