Image Enhancement and Object Recognition for Night Vision Traffic Surveillance

Monal PATEL1, Arvind YADAV1 and Carlos VALDERRAMA2
1 Department of electronics and Communication, Pand University, VADODARA-391760, INDIA
1 monalnakeshpatel@gmail.com
2 Department of Electronics and Microelectronics (SEMI), University of Mons, 7000 Mons, Belgium
2 CarlosAlberto.VALDERRAMASAKUYAMA@umons.ac.be

Abstract. Image processing is a vital task. Image processing is very useful in analyzing, detecting, surveillance systems, etc. The most challenging task in detection and surveillance systems is capturing and enhancing images during dark light. Night-time low-illumination image enhancement is desired for outdoor computing vision applications. Night time image and video processing are crucial tasks because of the color density, low brightness, low contrast, and in addition to the high amount of noise available in the captured image or video. Hence, to overcome this challenge the Dark Channel Prior (DCP) filter is used in the proposed work. This filter is used to improve the visibility, color contrast, and brightness of the image or video captured during nighttime. Another challenge in real-time applications is the processing speed, which can be improved by implementing the algorithms on a Field-Programmable Gate Array (FPGA). In this study, an Alveo board was used to improve the processing speed. The board was officially introduced by Xilinx.

Keywords: Dark Channel Prior (DCP), Vitis Xilinx, Alveo Board FPGA, CNN Convolutional Neural Network, Image Enhancement, Infrared Imaging, Object Recognition.

1 Introduction

Image and video processing is used to enhance images and videos captured from cameras and sensors. Various techniques and methodologies have been developed and used in image processing during the last few decades, among which many approaches have been developed to enhance captured images for object detection in the surveillance, auto and aviation sectors, space exploration, etc. Object detection in real-time scenarios is one of the most crucial and needful tasks. The quality of images from outdoor scenes is normally degraded by turbid media such
as haze, fog, and smoke. This phenomenon occurs because of atmospheric changes. In this paper, the main focus is on the enhancement of images captured during night, haze, smoke, and dark environments. The images or videos captured during the above-mentioned environments possess low illumination, brightness, contrast, and poor visibility. Various studies have been conducted that contain a number of algorithms to achieve the desired results on poor images. These types of algorithms need to be optimized for object detection in real-time scenarios. Several approaches and methods such as infrared and gamma correction, Discrete Fourier Transform (DFT), and histogram equalization can be implemented to enhance nighttime captures. From many approaches and filters, the Dark Channel Prior (DCP) is used in this work because it is simple and is the fastest algorithm that can be implemented on hardware. To achieve the speed requirements, various hardware resources, such as CPUs, GPUs, FPGAs, and ASICs have been considered in the implementation of the algorithm. ASIC provides the best results in terms of speed and low power consumption for the implementation of the algorithm, but the main limitation in ASIC is to be application-specific. They cannot be remodeled or reconfigured according to the requirements of the user. Hence, these boards result in high costs. An alternative to ASICs is FPGAs. Because of the limitations and cost of ASICs, FPGAs are the most favorable and efficient candidates for hardware implementation. FPGAs possess various advantages over CPU/GPU and ASICs. FPGAs provide a user-friendly environment that allows users to create and customize its architectural structure by implementing any desired function and IP blocks and removing unwanted or unnecessary functionality. These reasons lead to the use of less power and higher speed in comparison with other candidates. The main objective of this paper is to focus on accelerating the processing speed and reducing power consumption while using DCP to images and videos captured during nighttime. The schematic diagram for the enhancement of the images captured during a dark environment or night is presented below.

![Enhancement Diagram](image)

**Fig. 1. Enhancement of the nighttime**

In the above figure, the input frame of the captured image is enhanced by using the DCP algorithm, which is implemented on an FPGA. This hardware module is placed between the camera and monitor. The enhancement algorithm and filters are optimized to be implemented on the Xilinx's ALVEO board, which is targeted by using the Xilinx unified development environment, that is, VITIS unified software tool. VITIS is a new tool introduced by Xilinx, which combines all the aspects from Xilinx hardware as well as software in one platform for the
development of application-specific programs. VITIS allows the user to develop the hardware board architecture according to the requirements by using an embedded software flow, that is, the Xilinx Software Development Kit (SDK). The Xilinx hardware development environment is very beneficial for users who want to explore more and proceed toward new technologies. The software and hardware are interconnected with each other by the help of a driver named as Xilinx runtime. VITIS also provides a new framework supporting deep and machine learning Python libraries such as Caffe and Tensorflow, which were not available in older versions of Xilinx tools such as VIVADO and SDSoC, which were used for the development of applications on FPGA. This unified development framework makes easy for the user to develop a program as per requirement. In VITIS, the application program is divided into two sections: host and kernel. The language used for hardware acceleration is C++, which is a high-level programming language used in both software and hardware for acceleration. This type of high-level programming language uses various sets of libraries to make the desired development easier for the user. Here, the C++ programming language is used for the development of a program that needs to be used in both the host and kernel.

2 State of Art

Image enhancement is the procedure of improving the quality and information content of original data before processing. A few years ago, a variety of different algorithms were used to enhance night time video. Those techniques are described in [3], [4], [5], [6], [7], [8], [9] [10]. Infrared is one of the techniques used to improve the quality of dark input images or video. This technique uses a non-contact device that detects infrared energy and converts it into an electronic signal, it can capture only the objects with higher temperature than their surroundings [3]. In addition to being sensitive to sunlight, this technique is expensive compared to other normal cameras. This disadvantage limits the scope of their applications and future development. We can also use the traditional image processing techniques to improve the quality of night input images or video. Histogram equalization and Gamma correction are mostly used for traditional image processing, proposed methods for removal of noise motion adaptive temporal filtering based on the Kalman structured updating. By adaptive adjustment of RGB histograms causes the increment in dynamic range of denoised video. Ultimately, the remaining unwanted factor which is noise can be removed using Non-local means (NLM) denoising. This method exploits Color Filter Array (CFA) raw data to achieve low memory consumption. The final experimental results indicate that this method is highly promising for various real-time applications to consumer digital cameras, especially CCTV and the surveillance video system [4]. Although they reported pleasing results for low lighting images and videos, they still inevitably introduce undesirable halo effects and excessive enhancement phenomena [5]. Bhagya et al. [6] proposed video enhancement using histogram equalization with the JND model. With this technique, besides being
used to achieve visually pleasant enhancement effects, the over-enhancement and saturation artifacts are avoided. However, this method is complex, and it requires a lot of hardware resources. R. Peng et al. proposed a contrast stretching method which improves the quality of image by stretching the intensity range, while it stretches the low intensity pixels to lower values and high intensity pixels to higher values. As each value in the input image can have several values in the output image, objects in the original image may misplace their relative brightness values [7]. Choi et al. proposed single scale retinex which is one of the new emerging techniques in the enhancement field. The advantage of this method is the speed of execution. But there is also a limitation with Single retinex, as it deals with dynamic range enhancement and color interpretation, it struggles to do both together [8]. Rahman et al. proposed multiple scale retinex to solve the problem of Single scale retinex, but this method fails to produce good quality enhancement of input videos or images. Indeed, it finds input videos or images having high spectral characteristics in the single band [9]. Xuesong Jiang et al. proposed enhancement of dark input video based on the DCP algorithm. This method is simple and improves the quality of night time video or images, but it's quite slower for real-time video processing of ultra-high definition video [10]. Due to the low speed of video processing on CPU, our proposed method is to develop an accelerator for the DCP algorithm, to accelerate the video processing of night time, and obtain a continuous time video processing. In this technique we have split the image into 3 windows, and we provide 3 compute units to process each window. In this manner, in addition to providing an optimal enhancement, we also improve the execution time.

3 Methodology

3.1 Dark Channel Prior

The Dark Channel Prior basically depends on various perceptions of outdoor dimness-free images. In a large portion of the non-sky locations, at least one shading channel possesses few pixels with a low, near zero, intensity. Comparably, the least intensity in such a location is almost zero. To officially portray this perception, we initially characterize the idea of a dark channel. For a self-assertive picture $J$, its dark channel $J^{\text{dark}}$ is given by:

$$J^{\text{dark}}(x) = \min_{y \in \Omega(x)} \min_{c \in \{r, g, b\}} f_c(y)$$

Here, $F$ is a shading channel of $J$, and $\Omega(x)$ is a local patch focused at $x$. A dark channel is the result of two least operators $\min_{c \in \{r, g, b\}}$ performed on every pixel, and $\min_{c \in \{r, g, b\}}$ is a base filter. The least administrators are commutative.

Utilizing the idea of a dark channel, the perception says that if $J$ is an outside dimness free picture, aside from the sky area, the intensity of $J^{\text{dark}}$ is low and will, in general, be zero:
\[ f_{\text{dark}} \to 0 \]

This perception can be considered as a dark channel prior.

The low intensity in this type of dark channel is mostly due to three major factors: the first one is shadows, for example, the shadows of vehicles, structures, and within windows in cityscape pictures, or on the other hand, the shadows of trees, and geological components such as rocks in landscape pictures. The next factor is bright objects or surfaces; for example, any item with low reflectance in any shading channel (for instance, green grass/tree/plant, red or yellow bloom/leaf, and blue water surface) will bring about low qualities in dark channels. The last, but very important factor, is dull articles, or surfaces such as dark tree trunks and stones. As the common open air pictures are typically bright and loaded with shadows, the dark channels of these pictures are truly dim.

### 3.2 Mathematical Model of DCP algorithm

In this paper, we mention the use of DCP algorithms to enhance the input of images from dark environments. The algorithm, which is used to enhance the dark environment image, enhances the input image based on the minimum pixel value of RGB. Here, we use the McCartney equation to recover the foggy image.

![Fig. 2. Atmospheric scattering](image)

The figure above shows that the image captured by the sensor, which is considered as the input, is not as clear as required because of the availability of some components in an aerosol environment (the image was not clearly visible because of the particles in the air). In such a case, the camera absorbs the sunshine while capturing the image. This type of phenomenon is known as an Airlight. Most of the dark channels have very small amounts of intensity, and very few outdoor dines-free images deviate from the prior. Because of the additional Airlight, a hazy image is brighter than its dark version. Because of this, transmission \( t \) is very
low. Hence, it can be considered that the dark channel of a hazy picture possesses a high intensity value in areas with more haze. Initially, it can be considered that the intensity of the darker channel has a more unclear approximation of the haze thickness.

Using the McCartney Equation in order to recover the enhanced resulting picture frame:

\[ I_{inv}(x) = I_{inv}(x) + A(1 - t(x)) \]

Here,

\( X \) is the coordinate of the image \( x = (i, j) \)

\( I(x) \) depicts the input frame from the dark or night environment,

\( J_{inv}(x) \) is the inversion of the input frame.

\( J_{inv}(x) \) is the scene radiance at \( x \).

\( A \) is the Global Atmospheric Light.

\( t(x) \) is the transmission medium of \( x \)

The main goal of the algorithm is to recover \( J, A, \) and \( t \) from the inverted input image.

Using Equation in order to recover the inverted input image \( J_{inv}(x) \):

\[ J_{inv}(x) = \frac{I_{inv}(x) - A(1 - t)}{t(x)} \]

With:

\[ t(x) = 1 - \omega I_{dark}^c(x) \]

\[ I_{dark}(x) = \min_{c \in (r, g, b)} I_{inv}(x) \]

Here,

\( \omega \): a control parameter

\( I_{dark}^c \): indicates a dark channel prior to inverted frames.

### 3.3 The Dark Channel Prior Processing Flow
The design flow of the dark channel prior algorithm is discussed here by considering the input image as a foggy or smoky image. Each function used for the hardware implementation of DCP will be described.

![Diagram of Dark Channel Prior Processing Flow]

\[ I(x) \text{: Input Frame} \]
\[ I_{dark} \text{: Dark Input Frame} \]
\[ Imed \text{: Medium Transmission of } I_{dark} \]
\[ J(x) = \text{Recover Frame} \]

**Fig. 3. The Dark Channel Prior Processing Flow**

As shown in the above chart, the first stage of the processing is to provide an input of the impure image in the system. After the insertion of the image, the image is sent to the dark channel prior (DCP) module, where impurities such as smoke, haze, low or high brightness, low or high saturation, and contrast will be removed or adjusted according to the requirements, and visibility is improved. After processing the image in the DCP module, the resulting image is sent to the medium transmission, which is used to reduce the unwanted noise from an image. The VITIS Vision library plays an important role in determining medium transmission on hardware. Finally, after processing the image in the DCP and Medium Transmission modules, the resulting image is imported to the Dehaze module. The Dehaze function is used to remove the foggy or hazy impurities of the imported image using McCartney's equation.

4 Hardware, Tools and Languages

4.1 VITIS Unified Tool
Xilinx Environment introduced a new development kit known as the VITIS unified platform, which targets various FPGA boards such as Ultrascale, Alveo, and PYNQ. The tool was composed of three parts. The first is the VITIS Library (OpenCV, Ai/ML, Bias Libraries), which is used for the development of programs and kernel programs executing on hardware. It can be useful for various FPGA platforms such as Zynq, Zynq Ultrascale+, and Alveo. The second part is the VITIS Development Kit, which consists of a compiler to convert high-level code into Xilinx Object File linked with the platform for the generation of a binary file (.xclbin), an analyzer that analyzes the work done during the execution of the program, and debugger, which is used to locate the errors, occurs during the compilation of the host and kernel. The last part is Xilinx Run Time (XRT), which is a driver used to maintain communication between the host and kernel. This tool is useful for developing and creating an application program and executing the program on a hardware accelerator. The tool consists of two components: the host and the kernel. The host consists of an application-specified software program, and the kernel consists of a compute unit. In VITIS, APPs can be developed using pure software in which high-level programming languages such as C/C++, OpenCL, and Python are used for the development of AI application-based host code, and C++ languages can be used for the development of hardware. The C++ and OpenCL languages were used because of their simplicity. OpenCL API was used for the development of the host code, and the RTL kernel was used for the development of kernel functions. In VITIS, the host and kernel use a peripheral component express for the interconnection. The host code is executed on the CPU and kernel executes in the FPGA. The global architecture of the host and kernel are represented in the figure below:

Fig. 4. Global Architecture of host and Kernel

The above figure indicates that the host possesses its own memory, which can be accessible only from the host, and the kernel consists of multiple kernels, each having multiple compute units. This compute unit is responsible for the execution of the bit-stream on an FPGA. It has a local memory that can only be accessed by
the kernel. Global memory is also available, which is known as shared memory and can be accessible by both the host and kernel.

4.2 OpenCL and C++ Languages

Open Computing Language (OpenCL) is a framework that is useful for the development of application-specified programs that can be executed on various types of platforms such as CPUs, GPUs, FPGAs, and DSPs. Here, OpenCL is utilized for the development of host code and is also utilized to target the Alveo Board with the help of various commands such as creation of the context, setting the kernel, data writing in shared memory, and the execution of binary files on FPGA.

Because of the simplicity of the C/C++ language, it is very easy to use these languages to develop kernel and RTL kernel functions. In addition, these languages are also fast in comparison with other high-level programming languages, and they also provide several libraries such as OpenCV. C++ was developed in 1979 by a Danish computer scientist named Bjarne Stroustrup in a laboratory in New Jersey. The language was originally an improved version of the C language and named C with classes, but later on, it was modified as C++ in 1983. The language is statistically typed, compiled, general-purpose, case-sensitive, object-oriented, middle-level programming language.

4.3 Alveo Board

The Alveo board is an open-source FPGA board that was introduced by the Xilinx Environment in 2019. The board consists of low-level resources such as gates, flip-flops, SRAMs, and BRAMs. PCIe is used for interconnection of the board with the host. The Alveo board is used for the execution and acceleration of the program developed for enhancement of images captured during shady atmosphere or night time because the usage of the board is very easy, as it supports the VITIS software tool used for the creation of an application program using an advance-level language such as C/C++ and OpenCL. Here, the source code is optimized in order to use it in the creation of the program code in VITIS, which is provided by the Xilinx developing program. Various Alveo boards, such as Alveo u-50, Alveo u-200, Alveo u-250, and Alveo u-280 from Xilinx, and these boards are different on the basis of performance and power consumption. Here, the u-200 board is used for the compilation, implementation, and optimization of the program.

To implement and utilize an Alveo board, there is a requirement for highly compatible hardware resources such as computer systems with high RAM, good graphic card storage, high power supply, multiple modes of connectivity, memory card to write a compatible image, and good Internet connection.
5 Hardware Software Implementation of Dark Channel Prior Algorithm

5.1 System Architecture

In this study, the Alveo Board was accelerated by the program used to enhance the image or video captured during dark or night environments. The desired system is divided into three parts that is, The host, kernel, and global (shared) memory are available in the FPGA. The host and kernel parts are internally connected with the help of PCIe.

![System Architecture Diagram](image)

**Fig. 6. System Architecture**

Here, a Dell Core i7 CPU, which is configured with 2.7 graphic memory and 16 GB of RAM along with a Linux interface, was used as the host. An Alveo u200 FPGA with 182240 LUT, 2364480 FF, and 6840 BRAM was used as the platform, and the VITIS Tool was used to target and work with the Alveo FPGA.
The image below represents the implementation of an Alveo board connected to the host through a peripheral component interconnect. The system was powered with the help of power cables.

5.2 Working of the Global Architecture of proposed method

In this global architecture, the execution of the application-based program is the responsibility of the host and kernel. In this case, the CPU acts as the host, and the FPGA acts as the kernel. Here, the input frame is written in the global memory of the connected device through the PCIe interface, and the host application is responsible for this process. After importing the image in the form of data, the host commands the kernel for execution of the Dark Channel Prior algorithm and medium transmission on the input image. After receiving, the kernel follows the command and executes the data with the help of the VITIS vision library, which leads to the outcome of the image and data. The resulting data or images are then written to the shared memory. The host again enters the process, reads the image from shared memory, executes the Dehaze function on the code, and finally writes the output enhancement in the host memory.

5.3 Design Flow of Proposed method

The host and kernel are two top-level functions that are only and completely accessible by the developer/user. The kernel function utilizes the VITIS to execute the application-based program. The next layer contained the use of tools. These tools are used for the compilation and linking of application-based programs. However, in our scenario, both the host and the kernel were compiled separately. Here, the host uses a GCC compiler. To generate an executable file (.exe) from the high-level (.cpp). On the other hand, the kernel uses the VITIS tools compiling facility to compile the high-level function. The VITIS compiler commands the Vivado HLS tool for conversion of high-level code (.cpp) into the Xilinx object file. The object file from Xilinx is then linked to the platform, which is Alveo u200, using a c++ linker for the generation of byte streams in order to execute it on the FPGA. The final or the lowest layer of the program is the platform, the platform is in the hardware that can be utilized for the execution of the file, and the executable file executes on the CPU and the bit stream executes on the FPGA.
The figure above presents the connection of the layers and the flow between the host and kernel in the proposed method. Along with this, there is also the utilization of the pipeline in the kernel to execute the DCP algorithm on hardware.

The figure above represents the dataflow of the host and kernel. The input image frames are received at the host, segmented, and divided into a small set of images having 9x9 size and these images are sent to the global memory. The frames from the global/shared memory are fetched by the kernel and the Dark Channel Prior filter, and medium transmission is executed on the data and transmitted to the host through global memory; then, the host reads the resulting image data from the global memory and executes the Dehaze function in order to recover the frame.
6 Results and Evaluation

In this study, the frames of images captured during night or dark environments were enhanced and accelerated on an Alveo board, which acts as an FPGA. The captured frame was enhanced or improved by using the dark channel prior algorithm, medium transmission, and haze filter. The speed of image processing using efficient FPGA resources is the main advantage of this approach. The results of the Dark Channel Prior implemented on FPGA indicate that the process is very fast in comparison with the Dark Channel Prior algorithm implemented on the CPU. The processing speed of the proposed methodology implemented on the FPGA is presented in the figure below:

<table>
<thead>
<tr>
<th>Latency (cycles)</th>
<th>Latency (absolute)</th>
<th>Interval</th>
<th>Pipeline</th>
</tr>
</thead>
<tbody>
<tr>
<td>min</td>
<td>max</td>
<td>min</td>
<td>max</td>
</tr>
<tr>
<td>8303056</td>
<td>8321548</td>
<td>27.674 ms</td>
<td>27.736 ms</td>
</tr>
</tbody>
</table>
transmission, and haze function. The table below presents the processing time of the implementation of the DCP algorithm on the CPU with different frame sizes.

<table>
<thead>
<tr>
<th>Frame Size</th>
<th>Dark Channel Function (in sec)</th>
<th>Medium Transmission (in sec)</th>
<th>Dehaze Function (in sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1600 x 900</td>
<td>15.58</td>
<td>15.49</td>
<td>0.14</td>
</tr>
<tr>
<td>1200 x 675</td>
<td>5.29</td>
<td>5.49</td>
<td>0.09</td>
</tr>
<tr>
<td>640 x 360</td>
<td>0.42</td>
<td>0.44</td>
<td>0.02</td>
</tr>
<tr>
<td>320 x 180</td>
<td>0.044</td>
<td>0.046</td>
<td>0.0061</td>
</tr>
</tbody>
</table>

The table below presents a comparison of the processing time of the Dark Channel Prior algorithm implemented on the CPU and FPGA (Alveo U200) along with various image resolutions. The comparison shows that the hardware implementation that utilizes the FPGA is the most efficient and fast. Despite the various complications and difficulties that arise in the implementation, the proposed method is the most suitable.

<table>
<thead>
<tr>
<th>Frame Size</th>
<th>CPU (core i7) (in sec)</th>
<th>Alveo Board (in ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1600 x 900</td>
<td>31.048</td>
<td>4.86</td>
</tr>
<tr>
<td>1200 x 675</td>
<td>10.71</td>
<td>2.763</td>
</tr>
<tr>
<td>640 x 360</td>
<td>0.865</td>
<td>0.81</td>
</tr>
<tr>
<td>320 x 180</td>
<td>0.087</td>
<td>0.23</td>
</tr>
</tbody>
</table>

The targeted clock frequency is 3.33 ns; clock uncertainty is 0.90 ns; and fastest achievable estimated clock frequency is 2.433 ns which belong to the Xilinx Vivado High Level Synthesis tool. The figure below presents the details of the FPGA resources utilized for the proposed methodology:
7 Conclusion and Future scope

In this paper, it is concluded that the methodologies for the improvement and enhancement of the image frames captured in dark, smoky, and hazy environments have been studied and practiced. The practical has been performed by implementing the three functions, that is, dark channel prior, medium transmission, and haze function on the CPU as well as on the FPGA, which is an Alveo board introduced by Xilinx in 2019. The CPU imposes configurability with Core i7 and 16 GB RAM and evaluates the speed of processing on frames with various resolutions. The speed of video processing captured in a dark environment with 240p, 360p, and 480p can be continuous on the CPU. However, as there is an increase in the size and quality of the input image or video, it will affect the processing speed of the CPU. Hence, the best solution is to use an FPGA as a hardware source for the implementation of the DCP function for the improvement and enhancement of the image, and it can be practiced that the improvement of the input image frame is continuous on FPGA with very little usage of resources in hardware. In addition, the benefit of using an FPGA as a hardware source is that it has low latency and fast processing speed. Here, the input frame is executed on the CPU using the Python programming language and assigned to the hardware. The
experimental results show that the proposed methodology and algorithm can attain real-time image and video processing very fast compared to software processing.

Future work of the proposed work is to increase the compatibility and efficiency along with increasing the computing unit in the hardware to improve the design while minimizing the power usage utilized by the FPGA. This is a challenging task; hence, reducing the power consumption can be carried forward in future work.

References

2. Salazar-Colores, Sebastián, Juan-Manuel Ramos-Aregui, Jesús-Carlos Pedraza-Ortega, and Juvenal Rodríguez-Reséndiz, "Efficient single image dehazing by modifying the dark channel prior." EURASIP Journal on Image and Video Processing 2019, 66.
7. Tang, Qunfang, Jie Yang, Xiangian He, Wenjing Jia, Qingnian Zhang, and Haibo Liu, "Nighttime image dehazing based on Retinex and dark channel prior using Taylor series expansion." Computer Vision and Image Understanding 2020, 202, 103086
13. Yong Tang, Congjie Zhang, Renshu Gu, Peng Li, and Bin Yang, Vehicle detection and