CUDA Optimized Real-Time Image and Video Dehazing

Rohin Joshi
Department of Information Science and Engineering
Ramaiah Institute of Technology
Bangalore, India
rrohiinujoshi@gmail.com

Dhruv Dange
Department of Information Science and Engineering
Ramaiah Institute of Technology
Bangalore, India
dhruvdange@gmail.com

Sharath R Maiya
Department of Information Science and Engineering
Ramaiah Institute of Technology
Bangalore, India
sharath.mayya@gmail.com

Kavya KS
Department of Information Science and Engineering
Ramaiah Institute of Technology
Bangalore, India
kavya@msrit.edu

***Abstract*—Real-time image and video dehazing have garnered significant attention for enhancing visibility across various applications. This paper presents the development of a real-time memory-optimized dehazing system utilizing digital image processing techniques and NVIDIA's CUDA architecture for efficient parallel computing.**

Our methodology incorporates a quad tree search algorithm for efficient atmospheric light estimation, significantly improving dehazing accuracy. For transmission estimation, advanced contrast enhancement techniques are employed to ensure clarity and visibility in dehazed images. These methods address common challenges such as non-uniform illumination and varying haze densities. For image restoration, we dynamically clip values for improved brightness and to minimize information loss. Finally, a Gaussian filter is applied to smoothen out the artifacts in the transmission map, enabling more consistent dehazing.

The system's performance was quantitatively evaluated using the Structural Similarity Index Measure (SSIM) and Peak Signal-to-Noise Ratio (PSNR), achieving an SSIM of 0.77 and a PSNR of 27.9.

***Index Terms*—Real-time dehazing, video dehazing, CUDA, parallel computing, atmospheric light estimation, quad-tree search, transmission estimation, contrast enhancement, SSIM, PSNR.**

I. Introduction

The process of dehazing aims to enhance the visual quality of images and videos captured in hazy or foggy conditions, thereby improving visibility and perceptual clarity. In recent years, image and video dehazing techniques have gained significant attention due to their practical applications in various fields such as surveillance, autonomous driving, and outdoor photography. Dehazing techniques aim to cut through this haze, revealing the hidden clarity and enabling these vision systems to function effectively. Various methods for achieving real-time dehazing in images and videos address the trade-off between achieving high-quality results and processing speed limitations.

One seminal work, He et al. [1] introduced the Dark Channel Prior (DCP), a concept inspired by the observation that clear patches in outdoor scenes often have very low intensity in at least one color channel. DCP forms the foundation of a physical model that estimates the amount of haze and the scene's original clarity. This model mathematically describes the relationship between the hazy image, the original haze-free image, the haze itself, and the amount of light that manages to penetrate the haze.

Building on DCP, researchers have proposed various Digital Image Processing (DIP) methods for real-time dehazing. These methods focus on efficiently estimating the amount of haze and atmospheric light using statistical analysis or image filtering techniques.

Ghosh et al. [2] prioritized speed over complex calculations, creating a method specifically designed for portable devices with limited processing power. This makes it ideal for applications like dehazing images captured on smartphones.

Das et al. [3] employed a special filter to achieve real-time dehazing in traffic videos. This is particularly useful for applications where real-time processing is essential, such as autonomous vehicles navigating through foggy roads or intelligent traffic monitoring systems that rely on clear visuals.

Striking a balance between speed and accuracy, Hassan et al. [4] leveraged a technique called superpixel segmentation along with another filter to achieve real-time dehazing for images. This method offers a practical solution for various scenarios where both speed and good dehazing results are important.

For situations where processing images quickly is critical, Hu et al. [5] presented a method that analyzes the saturation and brightness information in an image to achieve fast dehazing. This method prioritizes speed, making it suitable for real-time applications where immediate results are necessary.

Dehazing can also be approached through deep learning techniques, which offer another avenue for achieving impressive dehazing results. But it comes with its own set of challenges. Deep learning models can learn complex patterns from large datasets and achieve impressive dehazing results.

However, these models often require significant computing power to train and run, making them less suitable for real-time applications with limited resources. Additionally, the quality of the training data heavily influences the performance of these models. If the data is insufficient or not well-curated, the dehazing results might not be optimal.

Peng et al. [6] addressed the speed issue by presenting a real-time video dehazing method that combines a simpler deep learning approach with another technique to improve accuracy. However, due to its simpler design, it may not achieve the same level of dehazing quality as some of the more complex deep learning methods.

For instance, Xu et al. [7] proposed a video dehazing approach using a complex deep learning network. While this method achieves excellent dehazing quality, the high computational demands make it less suitable for real-time applications on devices with limited processing power.

While traditional dehazing methods often struggle with real-time processing due to complex calculations, recent advancements offer promising techniques for achieving both speed and quality. One such approach proposed by Kim et al. [8]. This method focuses on enhancing the contrast of hazy images in real-time by considering local variations in haze thickness, which can be particularly beneficial for preserving details in intricate textures. Similarly Van et al. [9], offers another promising technique for real-time dehazing. This method leverages guided filtering at different scales to effectively remove haze while maintaining image edges. By applying this filter at various scales, the approach can efficiently remove haze without compromising the sharpness of edges and fine details in the dehazed image. By combining these techniques with efficient digital image processing methods, we hope to achieve real-time dehazing with exceptional performance. Further details and implementation specifics will be explored in the following sections.

Inspired by the efficiency of DIP methods and the insights from DCP, we propose a real-time image and video dehazing approach that leverages the strengths of traditional image processing techniques. Our method prioritizes computational efficiency while achieving good dehazing performance, making it suitable for resource-constrained real-time applications. This project focuses on the development of a real-time image and video dehazing system using digital image processing techniques, coupled with NVIDIA's CUDA architecture for efficient parallel computing. The motivation behind this project stems from the need for a portable and computationally efficient solution that can operate in real-time on standard hardware configurations, enabling practical deployment in diverse environments.

By leveraging CUDA, this work aims to harness the power of GPU acceleration to expedite the computational demands of the dehazing process, optimizing performance even on modest hardware setups. This project contributes to the advancement of real-time image and video dehazing techniques, with a focus on practical deployment and usability across different hardware platforms.

A. Motivation

The motivation behind this study is driven by the critical importance of enhancing visibility in challenging environmental conditions, such as haze, fog, or smoke, which can significantly degrade the quality of captured images and videos. This degradation presents substantial challenges in various practical applications, including surveillance systems, autonomous vehicles, outdoor photography, and critical scenarios such as indoor fires and search and rescue operations.

An efficient dehazing algorithm serves not only as a standalone solution but also as a crucial preprocessing step for other image processing and computer vision algorithms. Traditional computer vision algorithms often yield suboptimal results when applied to hazy, low-contrast images. By first processing these images through a dehazing algorithm, the enhanced contrast and reduced haze significantly improve the performance and accuracy of subsequent computer vision tasks.

B. Proposed Method

This section provides a concise overview of the implementation of our algorithm. The steps involved in the algorithm are illustrated in Figure 1.

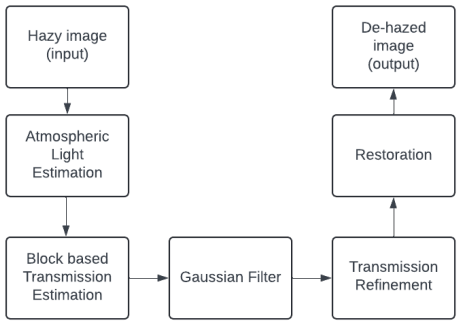


Fig. 1. Algorithm Workflow

Algorithm Overview

The dehazing algorithm was based on modeling the hazy image $I(x)$ as a linear combination of scene radiance $J(x)$, atmospheric light $A(x)$, and medium transmission $t(x)$. To obtain the dehazed image $J(x)$, both $A(x)$ and $t(x)$ were estimated using equations (1) and (2).

1) Atmospheric Light Estimation:

- **Quad-Tree Search:** The input image was partitioned into regions using a quad-tree approach. Atmospheric light was estimated within the region with the highest score iteratively.

2) Transmission Estimation:

- **Contrast-Based Estimation:** Leveraging the inverse relationship between transmission and image contrast, transmission was estimated to balance contrast enhancement and minimize information loss.

- **Gaussian and Guided Filtering:** The estimated transmission was refined using Gaussian filtering to smooth artifacts and guided filtering for further enhancement.

3) Real-Time Processing: CUDA programming was utilized for parallelization across GPU cores. Custom kernels were optimized for efficient computation, including average pixel value calculation, contrast and loss determination, and guided filtering. Image downsampling to 300×400 with bilinear interpolation optimized processing speed.

4) Video Processing: Temporal coherence was leveraged for computational efficiency in video processing. A coherence factor w was computed based on frame differences, allowing for the reuse of atmospheric light and transmission values between similar frames to reduce redundant calculations.

II. FRAMEWORK AND SYSTEM DESIGN

A. Design Overview

1) Implementation:

- **Language and Libraries**: Python with numpy, cupy, numba, opencv, and scipy.
- **Algorithm**: Based on [8], utilizing quad-tree search for atmospheric light estimation and contrast-based techniques for transmission estimation.
- **Real-Time Processing**: Leveraging CUDA programming for parallelization, custom kernels for efficient computation, and image pre-processing with bilinear interpolation.
- **Video Processing**: Incorporating spatial and temporal information, reusing atmospheric light and transmission values for computational optimization.

2) CUDA: The CUDA architecture consists of parallel compute engines built into NVIDIA GPUs, OS kernel-level support for hardware management, a user-mode driver with a device-level API, and the PTX Instruction Set Architecture (ISA) for parallel computing.

3) Numba: Numba is a Python library that uses LLVM to generate optimized machine code, enabling high-performance numerical computing. Through the @cuda.jit decorator, it supports CUDA, removing the requirement for separate CUDA C code and enabling developers to construct CUDA kernels directly in Python.

4) CuPy: CuPy is a GPU-accelerated Python library that provides a recognizable interface for array operations on GPUs. It was inspired by NumPy. It smoothly interacts with current codebases, enabling a wide range of mathematical operations, and enables developers to construct CUDA kernels using Python's array syntax.

5) Definition and Notations: An atmospheric scattering model is a mathematical representation used to simulate how light interacts with atmospheric particles (such as haze, dust, or water droplets) in the atmosphere.

The key equation used to model the observed radiance $I(x)$ at a point x in the image under the influence of haze is given by:

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

Where:

- $I(x)$ is the observed radiance at point x.
- $J(x)$ is the scene radiance (the radiance without haze), which represents the true light intensity emitted or reflected by objects in the scene.
- $t(x)$ is the transmission map, indicating the fraction of light that reaches the camera from the scene. It is related to the haze density and decreases with increasing haze.
- A is the global atmospheric light, representing the light scattered into the camera from the atmosphere.

The transmission map $t(x)$ is typically estimated from the image itself using techniques that leverage the observed degradation of image contrast and brightness due to haze.

The global atmospheric light A is generally assumed to be constant across the image and is estimated based on the brightest pixels in the image, which are likely to correspond to areas unaffected by objects or haze.

B. Architectural Design

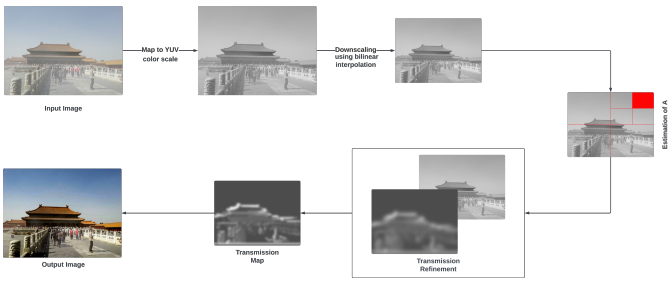


Fig. 2. System Architecture

Based on figure 2, the following steps were performed:

1) **Image/Video Input**: The hazy image or video feed is acquired either through a camera module capturing real-time footage or loaded from memory, depending on the application's requirements. This initial step establishes the source data for the dehazing process, which is essential for subsequent processing.
2) **Downscaling**: The resolution of the input image is reduced to 400 x 300 pixels using bilinear interpolation. Downscaling helps in reducing computational complexity while maintaining essential details for dehazing. Bilinear interpolation is a method used to interpolate values from neighboring pixels to generate a smoother downscaled image.
3) **Color Space Conversion**: The downscaled image is converted from the RGB color space to the YUV color space. In this process, only the luminance component (Y) is extracted. The YUV color space separates the luminance (brightness) information from the chrominance (color) information, which is advantageous for dehazing algorithms that primarily operate on luminance.
4) **Airlight Estimation**: Airlight, representing the global atmospheric light in the scene, is estimated using a quad-tree search algorithm. This algorithm searches for regions within the image with uniform brightness levels, identifying areas likely to be affected by atmospheric

haze. Airlight estimation is crucial for accurately modeling the light attenuation caused by haze in the scene.

5) **Transmission Estimation**: The transmission map, which represents the degree of light attenuation due to haze, is calculated using enhanced contrast techniques. These techniques aim to improve the visibility of scene details by enhancing local image contrast. The computation may leverage parallelization on CUDA cores, utilizing the processing power of GPUs for faster calculations. Key values such as E_{contrast} and E_{loss} are determined during this step, contributing to the accurate estimation of the transmission map.
6) **Filtering**: The initially estimated transmission map (T_{rough}) undergoes filtering processes to refine and smooth artifacts. Firstly, a Gaussian filter is applied to reduce noise and irregularities in the transmission map. Subsequently, a guided filter is employed using the luminance component (Y) of the downscaled input image as a guide. Guided filtering helps preserve edges and details while effectively smoothing the transmission map, leading to better dehazing results.
7) **Image Restoration**: The final dehazing process involves restoring the clarity and visibility of the image using the regular dehazing equation, which incorporates the estimated airlight (A) and transmission map (T). By removing the effects of haze from the image, the restored image provides improved visibility of scene details and enhanced visual quality for further analysis or presentation purposes.

C. Histogram Plot for Image Processing

Histogram plots are fundamental tools in image processing for visualizing the distribution of pixel intensities within an image. A histogram represents the frequency of occurrence of each intensity level (or color channel) in the image. This information is valuable for understanding the overall brightness, contrast, and dynamic range of an image.

1) Construction of Histograms: To construct a histogram, the range of possible intensity values is divided into discrete bins, and the number of pixels with intensity values falling within each bin is counted. For grayscale images, the histogram typically represents the distribution of pixel intensities from 0 (black) to 255 (white). For color images, separate histograms are generated for each color channel (e.g., red, green, and blue).

2) Interpretation of Histograms: Histograms provide insights into various aspects of image characteristics:

- Brightness and Contrast: The overall shape of the histogram indicates the brightness and contrast of the image. A histogram skewed towards higher intensity values suggests a brighter image, while a balanced distribution across the intensity range indicates better contrast.
- The observed difference in the histograms of the hazy image as shown in figure 3 and dehazed image as shown in figure 4 provides valuable insights into the dehazing process. In the hazy image, a notable skew towards higher grayscale values is evident, indicating a prevalence of pixels with higher intensity levels. This skewness is a characteristic feature of hazy images, where atmospheric scattering leads to the accumulation of light, resulting in an overall brighter appearance. Conversely, in the dehazed image, a more uniform distribution of pixels across the grayscale spectrum is observed. This even distribution suggests that the dehazing algorithm has successfully mitigated the effects of haze, effectively balancing the intensity levels throughout the image. The reduction in skewness towards higher grayscale values indicates the restoration of image clarity and contrast, resulting in a more natural and visually pleasing appearance. This observation underscores the efficacy of the dehazing algorithm in enhancing image quality by effectively attenuating the effects of atmospheric haze.

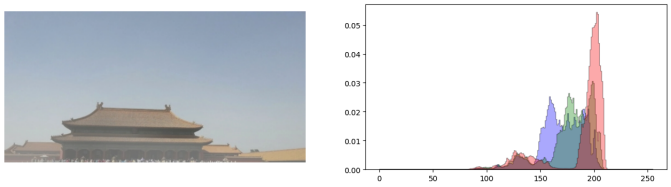


Fig. 3. Hazy Image Histogram

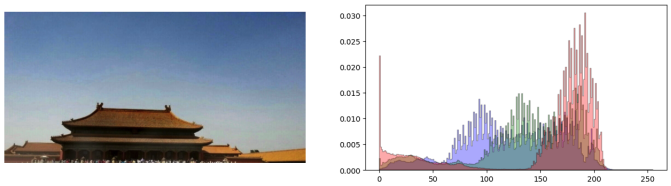


Fig. 4. Dehazed Image Histogram

III. Implementation

In this chapter, the technical details of the implementation of the real-time image/video dehazing algorithm are delved into. Python served as the primary programming language along with several libraries and frameworks such as numpy, cupy, numba, opencv, and scipy to facilitate efficient computation. The core algorithm was based on the model proposed by [8].

A. Algorithm Overview

The dehazing algorithm was rooted in the principle of image formation in hazy conditions, where the hazy image $I(x)$ was modeled as a linear combination of the scene radiance $J(x)$ and atmospheric light $A(x)$, modulated by the medium transmission $t(x)$. Mathematically,

$$I(x) = J(x)t(x) + A(x)(1 - t(x)) \tag{2}$$

To obtain the dehazed image $J(x)$, the atmospheric light $A(x)$ and the transmission $t(x)$ needed to be estimated.

$$J(x) = \frac{1}{t}(I(p) - A) + A \tag{3}$$

1) Atmospheric Light Estimation:

1) **Quad-Tree Search**: The input image was partitioned into regions using a quad-tree approach. Each region's score, based on average pixel value and standard deviation, was computed. The region with the highest score was selected iteratively until reaching a specified threshold. Within the selected region, the atmospheric light was estimated by minimizing the distance from white.

2) Transmission Estimation:

1) **Contrast-Based Estimation**: Transmission estimation leveraged the inverse relationship between transmission and image contrast. Mean squared error contrast was computed, and the transmission was initially estimated. [8]

$$C_{\text{MSE}} = \frac{\sum_{p=1}^{N}(Ic(p) - \overline{Ic})^2}{t^2 N} \tag{4}$$

(3)

As image clarity was improved by reducing haze and increasing contrast, the challenge of potential information loss from pixel underflow and overflow was faced. Thus, it was crucial to balance contrast enhancement with minimizing information loss. To address this, two distinct cost functions were introduced: one to quantify contrast enhancement and another to measure information loss. By simultaneously minimizing both cost functions, optimal image enhancement without sacrificing valuable information was aimed for.
The contrast cost E_{contrast} was calculated as the negative sum of MSE contrast across three colour channels for each block

$$E_{\text{contrast}} = -\sum_{c\in\{r,g,b\}}\sum_{p\in B}\frac{(I_c(p) - \overline{I_c})^2}{t^2 N} \tag{5}$$

where $\overline{\mathbf{I}_c}$ was the average pixel value of $\mathbf{I}_c$ in block B, and $\mathbf{N}$ was the number of pixels in the block. Note that by minimizing E_{contrast}, MSE contrasts could be maximized. Next, the information loss cost E_{loss} was defined as the squared sum of truncated values.

$$E_{\text{loss}} = \sum_{c\in\{r,g,b\}}\sum_{p\in B}(\min\{0, J_c(p)\}^2 + (\max\{0, 255 - J_c(p)\})^2) \tag{6}$$

Combining equation (4) and (5) resulted in:

$$E = E_{\text{contrast}} + \lambda_L E_{\text{loss}} \tag{7}$$

where λ_L was the weight to control the relative importance of the two variables.
Transmission t^* was then estimated iteratively, following the equation:

$$t^* = \max\left\{\min_{c\in\{r,g,b\}}\min_{p\in B}\left\{\frac{I_c(p) - A_c}{-A_c}\right\}, \max_{c\in\{r,g,b\}}\max_{p\in B}\left\{\frac{I_c(p) - A_c}{255 - A_c}\right\}\right. \tag{8}$$

2) **Gaussian Filtering**: The estimated transmission was refined using a Gaussian filter to smooth out artifacts and ensure consistency in transmission values.

$$G(x, y) = \frac{1}{2\pi\sigma^2}e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{9}$$

3) **Guided Filtering**: The transmission map was then passed as the input image, and using the Y channel of the input image converted to YUV colour space as the guiding image, guided filtering was performed on the transmission map. [10]

Algorithm 1 Guided filtering algorithm

Input: Input image I, guidance image P, window size r, regularization parameter ϵ
Output: Filtered output Q

foreach *pixel* i **do**
  $I_i = I * W_i$; `// Compute local mean of` I `using a window centered at` i
  $P_i = P * W_i$; `// Compute local mean of` P `using the same window`
  $I_{ii} = I * W_i^2$; `// Compute local variance of` I
  $\sigma_{ii}^2 = I_{ii} - I_i^2 + \epsilon$; `// Compute variance`
  $\sigma_{ip} = I_i * P_i$; `// Compute covariance`
  $a_i = \frac{\sigma_{ip}}{\sigma_{ii}^2}$; `// Compute slope`
  $b_i = P_i - a_i * I_i$; `// Compute offset`
end
foreach *pixel* i **do**
  $Q_i = a_i * I_i + b_i$; `// Compute filtered output`
end

4) **Tranmission Clipping value estimation**: The clipping values for the transmission map are critical in controlling the amount of information loss in the processed image. Accurate determination of these values is essential for each individual image to ensure optimal balance between noise reduction and preservation of image details. This precision in calculating clipping values helps maintain the integrity of important image features while enhancing overall image quality. 4

Algorithm 2 Fast Transmission Clip Metric Calculation Algorithm

Input: Input image img_input, estimated atmospheric light intensity AtmosphericLight_Y

Output: Calculated fast transmission clip metric metric

Initialization: sample_size $= 50$, sampled_pixels randomly sample 50 pixels from img_input

Calculate average intensity: avg_intensity $=$ mean of sampled_pixels

Compute difference: diff $\leftarrow \left|\frac{\text{avg\_intensity} - \text{AtmosphericLight\_Y}}{255}\right|$

Map difference to metric: metric : $0.3 + \text{diff} \times 0.8$

Return final metric: **return** $\lceil \text{metric} \times 10 \rceil / 10$

5) **Image Recovery**: The final image was recovered using across all colour channels. 5

Algorithm 3 Image Recovery Algorithm

Input: Input image input, atmospheric light A, transmission map $t(x)$

Output: Restored image output

Step 1: Initialize the output image

- Set output to a zero array with the same dimensions as output.

Step 2: Calculate the transmission clip value

- Compute transmission_clip using the fast transmission clip metric based on input and A_Y.

Step 3: Update the transmission map

- Set $t(x)$ to the maximum of $t(x)$ and transmission_clip.

Step 4: Restore the image

- For each color channel, update output using the following steps:
  a) Compute the intermediate value:
  $$\text{inter}(x, y, i) = \frac{\text{input}(x, y, i) - A[i]}{t(x)} + A[i]$$
  b) Clip the intermediate value to the range [0, 255]:
  $$\text{output}(x, y, i) = \text{clip}\left(\text{inter}(x, y, i), 0, 255\right)$$

Step 5: Return the restored image

- Return output.

B. Real-Time Processing

To achieve real-time performance, CUDA programming was leveraged to parallelize the workload across GPU cores. Custom kernels were developed and optimized for calculating the average pixel values within a block, determining E_{contrast}, E_{loss}, and implementing parallelized guided filtering by leveraging various constructs such as shared memory, and downscaling the image to a size of 300x400 using bilinear interpolation.

C. Video Dehazing Optimization via Temporal Coherence

Temporal coherence is leveraged as a critical factor in optimizing computational efficiency for video dehazing. A temporal coherence factor w is utilized to identify similar frames and avoid redundant calculations. This approach helps determine whether to execute the complete dehazing algorithm or to apply a more computationally efficient guided filtering process. The algorithm is detailed as follows:

1. Block-wise Difference Calculation

For each block of size $b \times b$ in the frame:

- **a)** Compute the difference between corresponding pixels in the current frame $f_{current}$ and the previous frame $f_{previous}$:
$$\text{diff} = \sum_{i=0}^{b-1}\sum_{j=0}^{b-1} |f_{current}(x+i, y+j) - f_{previous}(x+i, y+j) \quad (10)$$
- **b)** Compute the mean difference for each block:
$$\text{mean\_diff} = \frac{\text{diff}}{b^2} \quad (11)$$
- **c)** Calculate the block coherence using an exponential decay function:
$$\text{block\_coherence} = \exp\left(-\frac{\text{mean\_diff}}{100}\right) \quad (12)$$

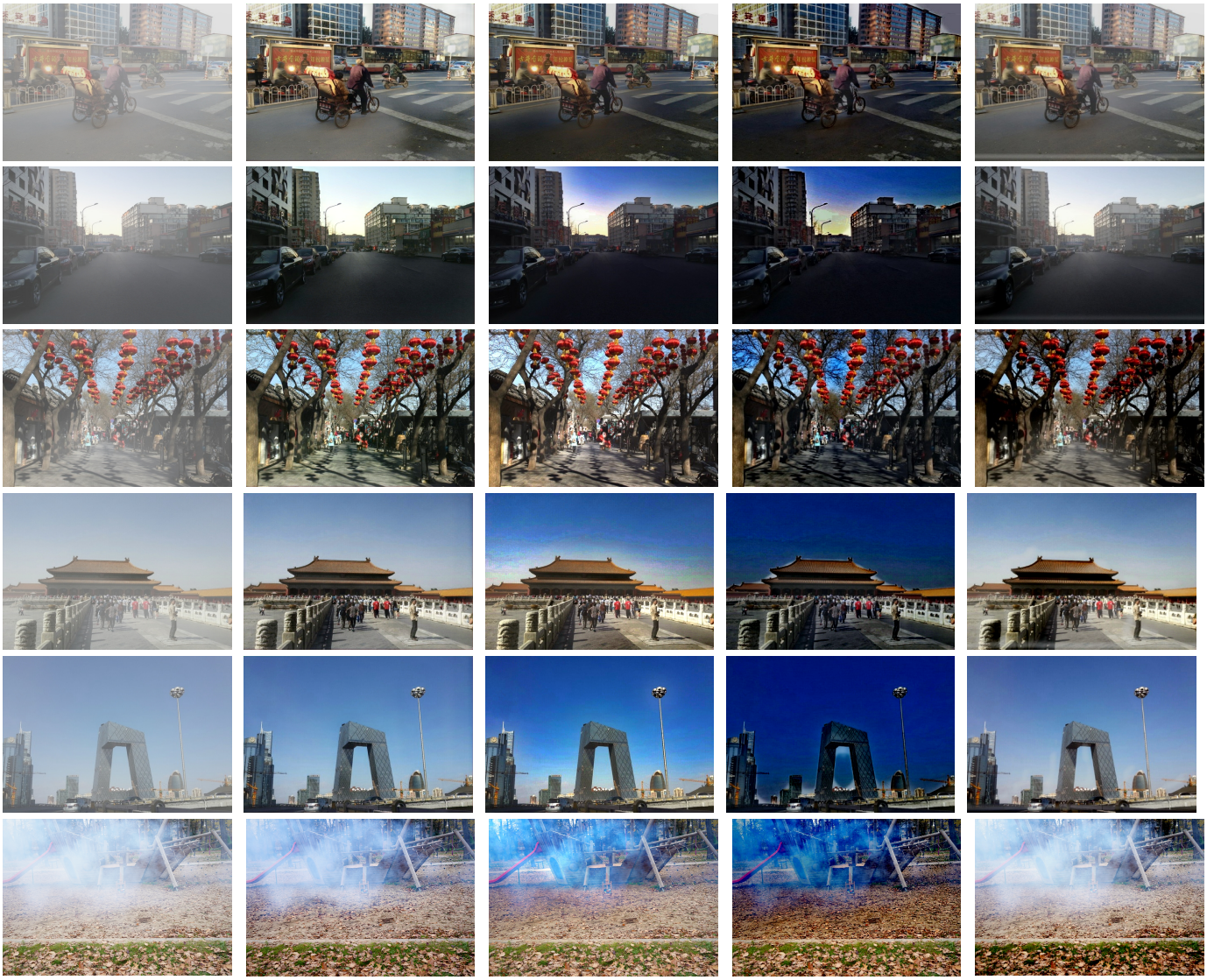
2. Temporal Coherence Calculation

- **a)** Compute the temporal coherence for the entire frame by averaging the coherence values across all blocks:
$$w = \frac{1}{N}\sum_{i=1}^{N} \text{block\_coherence}_i \quad (13)$$
where N is the total number of blocks.
- **b)** Determine the computational path based on the calculated temporal coherence:
  - **If** $w < 8.8$**:** Execute the full dehazing algorithm.
  - **If** $w \geq 8.8$**:** Use the current frame as the guiding image and the previous transmission map for a guided filter to achieve dehazing.

This algorithm effectively balances computational efficiency with dehazing quality by dynamically adjusting the processing method based on the similarity of consecutive frames.

IV. Experiments and Results

This chapter evaluates the performance of our real-time image/video dehazing system across diverse hazy image datasets. Using an Nvidia Tesla T4 GPU, processing times were measured, showcasing the system's efficiency. Key metrics such as **PSNR**, **SSIM** [13], and **processing time** were analyzed to assess dehazing performance.



(a) Original Image (b) DADFNET (c) Dark Channel Prior (d) Tarel et al (e) Proposed Method

Fig. 5. Comparative results of the proposed algorithm and the conventional algorithms on the "RESIDE-" Dataset Samples. (a) The input hazy image. The dehazed images obtained by (b) DADFNET [11], (c) dark channel prior method [1], (d) Tarel et al's algorithm [12], (e) the proposed algorithm.

Algorithm 4 PSNR Calculation

Input: Original image I_{original}, Compressed image $I_{\text{compressed}}$

Output: PSNR (Peak Signal-to-Noise Ratio)

Compute the Mean Squared Error (MSE) using: $\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(I_{\text{original}}[i] - I_{\text{compressed}}[i])^2$

if $MSE = 0$ **then**

  return 100 ; `// PSNR is maximum if MSE is zero`

end

Set the maximum pixel value: MAX_PIXEL $= 255.0$

Calculate PSNR using: $\text{PSNR} = 20 \times \log_{10}\left(\frac{\text{MAX\_PIXEL}}{\sqrt{\text{MSE}}}\right)$

return PSNR

Notably, the proposed system demonstrated significantly faster processing times compared to existing benchmarks, underscoring its suitability for real-time applications. Through quantitative analysis and visual comparisons, the superior performance and efficiency of the approach in enhancing visibility under hazy conditions are validated.

The Structural Similarity Index (SSIM) is a metric used to measure the similarity between two images. It takes into account the luminance, contrast, and structure of the images. The SSIM index is computed as follows:

$$\text{SSIM}(x, y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)} \tag{14}$$

where μ_x and μ_y are respectively the average values of x and y. Similarly, σ_x^2 and σ_y^2 represent the variances of x and

y respectively. σ_{xy} denotes the covariance between x and y. Additionally, c_1 and c_2 are constants employed to stabilize the division with a weak denominator.

TABLE I
PERFORMANCE METRICS (TESTED ON NVIDIA TESLA T4)

Model	PSNR	SSIM	Time(s)
He et al.	27.81	0.69	0.06
Tarel et al.	28.15	0.42	0.74
DADFNET	27.83	0.79	0.0085
Our approach	27.90	0.77	0.035

As shown in Table Tarel et al. [12] achieved the highest PSNR among the models tested. However, this came at the cost of a significantly longer runtime of 740 ms, as well as a comparatively lower SSIM score. The other three models demonstrated similar performance in terms of PSNR and SSIM, with DADFNET being the fastest, completing in just 8.5 ms. Our approach followed at 35 ms, and He et al.'s method at 60 ms. Notably, despite its speed, DADFNET exhibited poor generalization across the testing data, necessitating the use of two distinct models to maintain performance consistency. Its also important to note DADFNET is a deep-learning based approach requiring high training times for the models.I,

V. CONCLUSION AND FUTURE SCOPE

A. Conclusion

The objective of this study was to develop a real-time dehazing solution for images and videos. By leveraging image processing techniques such as contrast enhancement and guided filtering, an exact algorithmic approach was pursued, prioritizing performance in a generalized setting.

The experiments yielded promising results, achieving a dehazing time of 35 milliseconds per image, which meets the real-time requirement. Additionally, the approach attained a respectable PSNR (Peak Signal-to-Noise Ratio) of 28 and an SSIM (Structural Similarity Index) of 0.84, comparable to current state-of-the-art vision transformer models while demonstrating superior efficiency.

These findings underscore the effectiveness of exact algorithms in real-time dehazing applications. By enabling the utilization of lightweight hardware, such as edge devices, this approach opens up new avenues for deployment across various domains.

B. Future Scope

1) Advancements in Real-Time Dehazing: This study represents a significant advancement in the field of real-time dehazing. By leveraging innovative techniques and algorithms, notable progress has been made in mitigating the effects of haze and fog in images and videos.

2) Optimization of Hardware Requirements: Broadening Applicability through Hardware Optimization

While the proposed solution demonstrates effectiveness on specialized hardware, there is potential for improvement in hardware optimization. Reducing the dependency on GPUs would enhance the accessibility and applicability of this solution across a broader range of computing platforms.

3) Integration of Mechanisms for Haze Detection: Reducing Computation with Haze Detection Mechanisms Incorporating mechanisms to detect the degree of haze in a frame prior to initiating the dehazing process can offer significant computational savings. By analyzing image features or utilizing metrics such as saturation and value ratio, the level of haze can be assessed, enabling informed decisions about the necessity of dehazing. This proactive approach minimizes unnecessary computation and enhances overall efficiency, particularly in dynamic environments where conditions may change rapidly.

REFERENCES

[1] Kaiming He, Jian Sun, and Xiaoou Tang. Single image haze removal using dark channel prior. *IEEE transactions on pattern analysis and machine intelligence*, 33(12):2341–2353, 2010.

[2] Avra Ghosh, Asfak Ali, Sangita Roy, and Sheli Sinha Chaudhuri. Novel parametric based time efficient portable real-time dehazing system. *Journal of Real-Time Image Processing*, 20(2):23, 2023.

[3] Apurba Das, Shashidhar Pai, Vinayak S Shenoy, Tanush Vinay, and SS Shylaja. : Real-time dehazing in traffic video analytics by fast dynamic bilateral filtering. In *Proceedings of 3rd International Conference on Computer Vision and Image Processing: CVIP 2018, Volume 2*, pages 127–137. Springer, 2019.

[4] Haseeb Hassan, Ali Kashif Bashir, Muhammad Ahmad, Varun G Menon, Imran Uddin Afridi, Raheel Nawaz, and Bin Luo. Real-time image dehazing by superpixels segmentation and guidance filter. *Journal of Real-Time Image Processing*, 18:1555–1575, 2021.

[5] Daosong Hu, Yang Yang, Bo Li, Huiming Tang, and Yu Xu. Fast outdoor hazy image dehazing based on saturation and brightness. *IET Image Processing*, 16(3):900–912, 2022.

[6] Shu-Juan Peng, He Zhang, Xin Liu, Wentao Fan, Bineng Zhong, and Ji-Xiang Du. Real-time video dehazing via incremental transmission learning and spatial-temporally coherent regularization. *Neurocomputing*, 458:602–614, 2021.

[7] Jiaqi Xu, Xiaowei Hu, Lei Zhu, Qi Dou, Jifeng Dai, Yu Qiao, and Pheng-Ann Heng. Video dehazing via a multi-range temporal alignment network with physical prior. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18053–18062, 2023.

[8] Jin-Hwan Kim, Won-Dong Jang, Jae-Young Sim, and Chang-Su Kim. Optimized contrast enhancement for real-time image and video dehazing. *Journal of Visual Communication and Image Representation*, 24(3):410–425, 2013.

[9] Thuong Van Nguyen, An Gia Vien, and Chul Lee. Real-time image and video dehazing based on multiscale guided filtering. *Multimedia Tools and Applications*, 81(25):36567–36584, 2022.

[10] Kaiming He, Jian Sun, and Xiaoou Tang. Guided image filtering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(6):1397–1409, 2013.

[11] Yu Guo, Ryan Wen Liu, Jiangtian Nie, Lingjuan Lyu, Zehui Xiong, Jiawen Kang, Han Yu, and Dusit Niyato. Dadfnet: Dual attention and dual frequency-guided dehazing network for video-empowered intelligent transportation. *arXiv preprint arXiv:2304.09588*, 2023.

[12] Jean-Philippe Tarel and Nicolas Hautiere. Fast visibility restoration from a single color or gray level image. In *2009 IEEE 12th international conference on computer vision*, pages 2201–2208. IEEE, 2009.

[13] Alain Horé and Djemel Ziou. Image quality metrics: Psnr vs. ssim. In *2010 20th International Conference on Pattern Recognition*, pages 2366–2369, 2010.