Image Haze Removal of Optimized Contrast Enhancement Based on GPU

Che-Lun Hung, Zhaohui Ma, Chun-Yuan Lin and Hsiao-Hsi Wang

Abstract In the domains of computer vision and graphical computation, image haze removal has been a significant issue. By the use of haze removal process, it can significantly improve the visibility of the scene in the image. However, most of the haze removal algorithms bring high computational cost and make algorithms failed in processing huge amount of images. In this paper, we propose a parallel image haze remove algorithm, adopting optimized contrast enhancement approach, to optimize the performance based on GPU platform. The optimization from the proposed algorithm obtains performance acceleration with about 5 times as compared the original version while the haze removal effect is the same. Some haze free images and its original hazy images are shown in the later chapter during this paper. Our work after improvement can process a single picture in a much higher speed after optimization and make it more sufficiently fast for large-scale application which needs image haze removal in computer vision area.

Keywords Image haze removal · Parallel computing · GPU · CUDA

C.-L. Hung (✉)
Department of Computer Science and Communication Engineering,
Providence University, Taichung, Taiwan
e-mail: clhung@pu.edu.tw

Z. Ma
Department of Computer Science and Information Engineering,
Providence University, Taichung, Taiwan
e-mail: g1020435@pu.edu.tw

C.-Y. Lin
Department of Computer Science and Information Engineering,
Chang Gung University, Taoyuan, Taiwan
e-mail: cyulin@mail.cgu.edu.tw

H.-H. Wang
Department of Computer Science and Information Management,
Providence University, Taichung, Taiwan
e-mail: hhwang@pu.edu.tw

J.C. Hung et al. (eds.), *Frontier Computing*, Lecture Notes
in Electrical Engineering 375, DOI 10.1007/978-981-10-0539-8_7

1 Introduction

In almost every practical scenario, the images of outdoor scenes often are blurry caused by fog or other types of atmospheric degradation. Due to the atmospheric absorption and the atmospheric scattering, the light of scenery which camera captures is attenuation in different level. Besides, the photometeor in air is a part of ingredient of the light of scenery. Image dehazing plays an important role in the domain of image processing. Image dehazing can improve the clarity of the image significantly and revert the color cast of atmospheric scattering of light. Similarity, it can be used to improve the interpretability of images for computer vision and preprocessing tasks.

Generally, the image dehazing algorithms can be classified into two groups: image enhancement and image restoration [1]. The image enhancement algorithms [2–5] are used to improve the contrast in images directly. These algorithms are simple and fast, but they are difficultly to be used to adjust the image characteristics as color changes. The image restoration algorithms [6–8] aim to emphasize features of the image to be suitable for human visual perception. The algorithms have been used to improve the image problem caused by the atmospheric scattering based on strong prior or assumption atmospheric transmission and environmental luminance model. Tan [6] discovers that in general, non-fog image contains higher contrast than the images with fog. The original image can be reverted through maximizing the local contrast of the image. The result is very convincing in theory, but the practical results are not particularly good. Fattal [7] estimates the reflectance and transmittance of the image to infer by assuming that the transmittance and the surface of the projection in the local scene are irrelevant. This approach achieves more accurate and very good dehazing results. However, this method cannot copy with the image with high concentrations of fog.

He et al. [8] proposed an algorithm based on dark channel prior. It presents a simple but effective image prior law—dark channel prior to a single input image dehazing. Dark channel prior is a statistical law for outdoor non-fog image. It is based on a key idea; mostly outdoors image without fog, the each local area exists some pixels in a color channel with low intensity values. Using this effective image prior law and defogging model the algorithm estimates the concentration of fog and revert high-quality images. The experimental results show that this algorithm can achieve the better defogging effect based on a single image, and also obtain more accurate depth image information. However, the processing speed of this algorithm is very slow. The computation time for processing a 1024×768 image is about 60–80 s.

Some of the recent algorithms based on He's algorithm are proposed on improving contrast and luminace of degraded image. Matlin and Milanfar [9] proposed a method based on BM3D [10] and He's algorithm to remove haze and noise from a single image. They also proposed an iterative regression method. Both of these two algorithms can achieve good processed image when the noise level is precisely known. The drawback of these algorithms is that the latent errors by denoising can be amplified when noise level is unknown. Nan et al. [1] proposed a

Bayesian framework to avoid dynamic range compression in He's algorithm. The haze and noise in the input image are removed simultaneously. They adopted an iterative strategy with feedback to achieve the more accurate results than Matlin's approach. For these algorithms above, the computation cost is still high for processing a single image. Kim et al. [11] proposed an algorithm to improve the contrast of the given image and design a cost function in order not to lose too much information while recovering the contrast.

Actually, these image-dehazing algorithms are time-consuming leading to be used as real-time applications difficultly. To enhance the computational performance, Graphic Processing Units (GPUs) has been adopted in many image-dehazing algorithms. Xue et al. [12] proposed haze removal algorithm using dark channel prior implemented on GPU. Valderrama et al. [13] proposed a GPU-based local adaptive algorithm, which uses local statistics of the hazed image, for single image. Ok et al. [14] proposed a GPU-based dehazing algorithm, which executes CPU-based MSR algorithm for each RGB channel and CUDA Gaussian Blur algorithm, for video surveillance.

In this paper, we propose a GPU based image-dehazing algorithm based on by Kim's algorithm [11] through optimized contrast enhancement approach. From the experimental results, the execution time to process 1024×768 image is about more than 2 s by Kim's algorithm on CPU and it can be reduced to 0.4 s by the proposed algorithm. The optimization from the proposed algorithm obtains performance acceleration with about more than 5 times when the image size growing while the dehazing result is as excellent as the original algorithm.

2 Background

2.1 Haze Formation Model

In computer graphics and computer vision, there is a model describing the observed color of an image in the presence of haze or fog. The model widely used to describe the formation of a haze image is the equation as below [8, 11]:

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

Note that in the Eq. 1, $I(x)$ is the observed intensity, $J(x)$ is the scene radiance, $\boldsymbol{A}$ is the atmospheric light, and $t(x)$ is the transmission of the reflected light. The $I(x) = (I_R(x), I_G(x), I_B(x))$ and the $J(x) = (J_R(x), J_G(x), J_B(x))$ represent the original and the observed R, G, B color channels at the specific pixel position x, respectively [15]. Image haze removal is actually to compute J, A, and t from I [16, 17]. Usually, most of the image haze removal algorithms need to calculate these parameters in order to get the haze-free image finally. When the atmosphere is homogenous, the transmission $t(x)$ can be presented as following [18]:

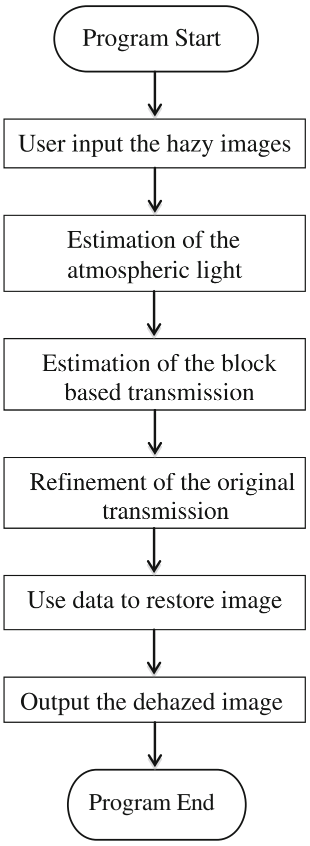
Program Start
User input the hazy images
Estimation of the atmospheric light
Estimation of the block based transmission
Refinement of the original transmission
Use data to restore image
Output the dehazed image
Program End

Fig. 1 Modules of image haze removal algorithm

$$t(x) = e^{-p*d(x)} \tag{2}$$

As the Eq. (2), $d(x)$ is the scene depth from the captured camera at pixel position x while p is the attenuation coefficient which is decided by the weather condition.

2.2 Image Haze Removal

The process flow diagram of image dehazing is shown in Fig. 1. Most of haze removal algorithms, either about images or videos, have the similar process that contains these steps to obtain the haze free image. The common steps of dehazing are listed as followings: User input the hazy image to process; Rough estimation of the atmospheric light which is A that represents the ambient light in the atmospheric [6]; Estimation of the block based transmission and the block can be customizing by the user; Refinement of the rough transmission to obtain $\boldsymbol{t}(x)$ using the previous transmission result. Finally, we can restore the image to become haze-freed according to the hazy image $\boldsymbol{J}(x)$, the computed result $\boldsymbol{A}$ and $\boldsymbol{t}(\mathrm{x})$.

2.3 CUDA Programming Model

Compute Unified Device Architecture (CUDA) is a parallel computing platform and programming model invented by NVIDIA. It enables dramatic increases in computing performance by harnessing the power of GPU. CUDA is an extension of C/C++ which enables users to write scalable multi-threaded programs for CUDA-enabled GPUs [19].

CUDA programs usually contain a special part, called kernel, which will be parallel executed on GPU. The kernel represents the operations or computation to be performed by a single thread and is invoked as a set of concurrently executing threads. These threads are organized in a hierarchy consisting of so-called thread blocks and grids. A grid is a set of independent thread blocks and a thread block is a set of concurrent threads. The total size of a grid (dimGrid) and a thread block (dimBlock) is explicitly specified in the kernel function-call:

```
  kernel<<<dimGrid, dimBlock, … >>> (parameters);
```

Thread communication and synchronization are implemented in the thread blocks so that threads within a thread block can communicate each other through a per-block shared memory and are synchronized using barriers. However, threads located in different blocks cannot communicate or synchronize directly. Besides the shared memory, there are four other types of memory: per-thread private local memory, global memory for data shared by all threads, texture memory and constant memory. Texture memory and constant memory can be regarded as fast read-only caches. One of the optimizing the performance of GPU is to enable threads to access shared memory rather than global memory.

The CUDA architecture consists of a number of streaming multiprocessors (SMs). Each SM contains 8 streaming processors (SPs), which share a per-block shared memory of size 16 KB. All threads of a thread block are executed concurrently on a single SM. The SM executes threads in small groups of 32, called warps. Thus, parallel performance is generally penalized by data-dependent conditional branches and improves if all threads in a warp follow the same execution path.

3 Method

In this section, we will introduce the important works to implement the parallel version of GPU based on the optimized contrast enhancement to improve the computational performance. Each module of the image haze removal shown in

Table 1 Parallel degree of each module

Module	Computing process	Computational complexity by CPU	Parallel degree
Estimation of atmospheric light	Sequencing, comparison	O(m * n * log(m * n))	O(m * n)
Estimation of block based transmission	Image partition and matrix operation	O(m * n)	O(m * n)
Refinement of original transmission	Matrix operation and inversion	O(m * n)	O(m)
Use data to restore image	Matrix operation	O(m * n)	O(m)

Fig. 1 has been analyzed to identify the steps that can be parallelized. Then these modules have been implemented on GPU.

3.1 Parallelism Analysis

Based on the computing process of the optimized contrast enhancement algorithm, the computational complexity of the algorithm can be analyzed by the data dependency during the computing process. The parallel degree represents time complexity of the module implemented on GPU. Table 1 shows the result of the parallel degree of each module for $m \times n$ image.

3.2 Dynamic Parallelism

The new feature of dynamic parallelism released in CUDA 5.5 enables the Kepler GK110 GPU to dynamically invoke new threads by adapting to the data by kernel directly. Kernel on GPU has the ability to independently launch additional workloads as needed. The global atmospheric light which is represented the ambient light in the atmospheric is often considered the brightest color in the image. To estimate the atmospheric light more reliably, we should exploit the fact that the variance of pixel values is generally low in hazy regions. As shown in the Fig. 2, a hazy image can be split into four rectangular regions and the score of each region as the average pixel values within the region can be computed. Then, the region with highest score is selected to be split into other four smaller regions until the area of selected region is smaller than the pre-defined threshold. The threshold is set 200 as default.

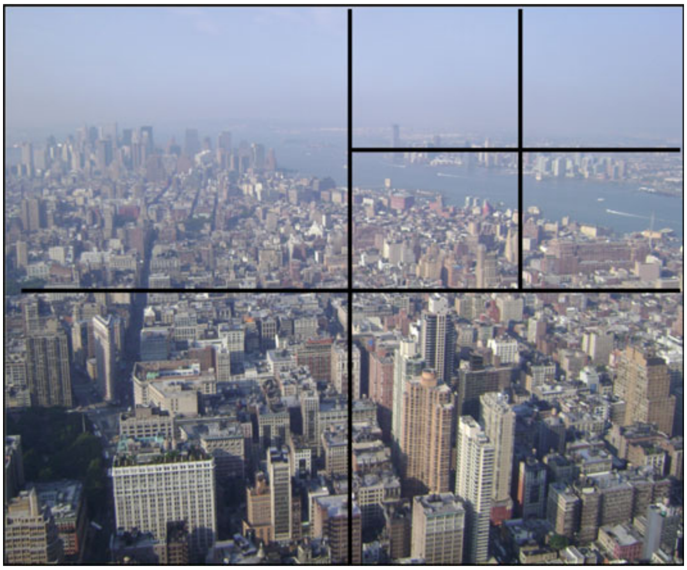

Fig. 2 Atmospheric estimation division

The CUDA feature called dynamic parallelism can be use on the quad-tree subdivision. As shown in the pseudo code, a threshold is defined in the kernel specified to perform the recursive division step.

```
__global__ void Atmospheric_Estimation(…){
 //define threshold to make the recursion stop;
 if(the size is larger than threshold){
      Atmospheric_Estimation <<<dimGrid,dimBlock, …>>>(…);
 }
}
```

3.3 Shared Memory Optimization

The shared memory is on-chip memory of GPU and it has considerable speed as compared to global memory. On CUDA, we launch several hundred of threads in a single block. All the threads in a block which are using the same the shared memory can cooperate with the other in the same block. The build-in function __syncthreads () is used to specific the synchronization points of all threads in the kernel; It is the barrier function, and all threads in the block must wait until all the threads in a block arriving the point. From the time of each module of CPU version, the module guided filter cost the biggest part of the total time of program. In the guided filter, box filtering algorithm is executed to get cumulative function for calculating the integral image, either in one dimension or three dimensions. The box filtering produces the cumulative results of each array which is in X axis or in Y axis. In a factor, the computation between two pixels is not totally independent. If the array data onto the global memory is used for box filter computation, it will decrease the speed apparently because of accessing the global memory too many times leading the performance reduction.

4 Experimental Result

4.1 Haze Removal Effect

We evaluate the haze removal effect of the proposed algorithm on images used in [11]. As shown in Fig. 3, we can obtain very nice images after removing haze by the proposed algorithm. We also apply the proposed method to recover several hazy images and compare the proposed method with Kim's algorithm.

4.2 Performance Optimization

Our platform mainly included CUDA and OpenCV is built on a computing environment of Ubuntu 13.04 with Intel® Core(TM)2 Duo @2.00 GHz 2.00 GHz, Nvidia Tesla K40 with 8G memory. For an image of 1024×768, the comparison of computational performance is shown in the Table 2. In comparison of the performance of each module, on pure CPU version, atmospheric light estimation costs only 0.04 s, color transmission estimation costs 0.45 s, guided filter costs more than 1.38 s, and restore image cost 0.29 s. 5. Because the refinement of transmission of guided filter is the most time-consuming, we focus our attention on it and make great effort to improve the performance. From the Table 2, the GPU program can earn a speedup which is more than 5 times of the CPU version.

4.3 Performance Comparison

Figure 4 is the comparison of computational performance based on the different hazy image sizes. From the figure, the running time of the CPU dehazing and the proposed algorithms have a linear relationship with the sizes of images. We can get an increasing the trend from the figure, the operating time of the GPU increase in a stable and gentle trend when the image size becomes large, and basically stable at less than 0.4 s. However, the CPU program has bigger and unstable amplitude with the image enlarging.

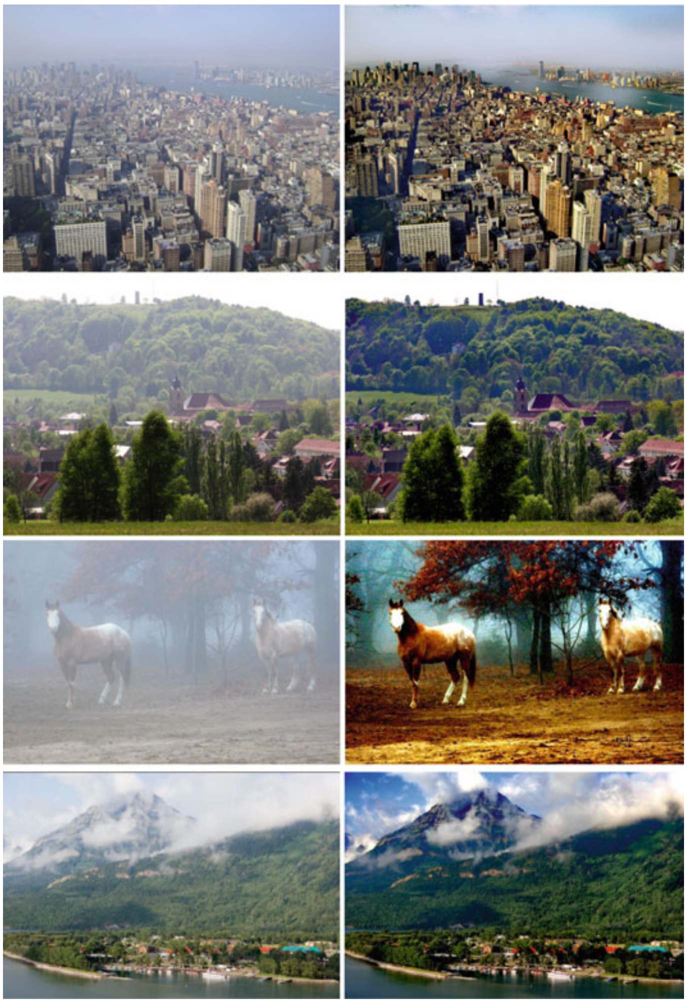

Fig. 3 Images before and after optimization

Table 2 Runtime performance on CPU and GPU

Component name	CPU Runtime(s)	GPU Runtime(s)	Speedup
Estimation of the atmospheric light	0.0483	0.0481	1
Estimation of the transmission	0.4537	0.0727	6.24
Refinement of the guided filter	1.3875	0.2451	5.66
Using data to restore dehazed image	0.2932	0.0573	5.12
Image haze removal	2.1827	0.4232	5.16

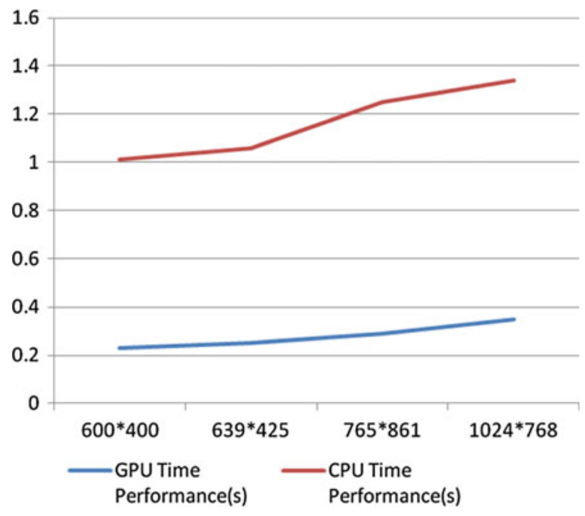

Fig. 4 Performance comparison on different image size

5 Conclusion

In this paper, we have proposed a GPU-based parallel image-dehazing algorithm. The proposed algorithm adopts the same strategies as Kim's aglorithm, called optimized contrast enhancement, for single image haze removal. The proposed algorithm can achieve very good haze removal effect and reduce the computational cost. Obviously, the proposed algorithm can be used as real time application for removing haze.

Acknowledgment This research was partially supported by the Ministry of Science and Technology under Grants MOST104-2221-E-126 -004 and MOST103-2221-E-126-013.

References

1. Nan D, Bi D, Liu C, Ma S, He L (2014) A Bayesian framework for single image dehazing considering noise. Sci World J, Article ID 651986
2. Nan D, Bi D, Xu Y, He Y, Wang Y (2011) Retinex color image enhancement based on adaptive bidimensional empirical mode decomposition. J Comput Appl 31:1552–1555
3. Pizer SM, Amburn EP, Austinetal JD (1987) Adaptivehistogram equalization and its variations. Comput Vis Gr Image Proces 39:355–368
4. Guo F, Cai Z, Xie B (2011) Videodefoggingalgorithmbasedon fog theory. Acta Electronica Sinica 39:2019–2025
5. Li C, Gao S, Bi D (2009) A modified image enhancement algorithm based on color constancy. Chin Opt Lett 7:784–787
6. Tan R (2008) Visibility in bad weather from a single image. In: Proceedings of the 26th IEEE conference on computer vision and pattern recognition, pp 1–8

7. Fattal R (2008) Single image dehazing. In: Proceedings of the international conference on computer graphics and interactive techniques, pp 1–9
8. He K, Sun J, Tang X (2009) Single image haze removal using dark channel prior. In: Proceedings of the IEEE computer society conference on computer vision and pattern recognition workshop, pp 956–1963
9. Matlin E, Milanfar P (2012) Removal of haze and noise from a single image. In: Proceedings of SPIE, pp 82–96
10. Dabov K, Foi A, Katkovnik V, Egiazarian K (2007) Image denoising by sparse 3-D transform-domain collaborative filtering. IEEE Trans Image Process 16:2080–2095
11. Kim JH, Jang WD, Sim JY, Kim CS (2013) Optimized contrast enhancement for real-time image and video dehazing. J Vis Commun Image R 24:410–425
12. Xue Y, Ren J, Su H, Wen M, Zhang C (2013) Parallel Implementation and optimization of haze removal using dark channel prior based on CUDA. Commun Comput Inf Sci 207:99–109
13. Valderrama JA, Diaz-Ramirez VH, Kober V (2014) Single image dehazing using local adaptive signal processing. In: Proceedings of SPIE 9217, applications of digital image processing, p XXXVII
14. Ok S, Kim M, Cho H (2014) GPU-accelerated dehazing algorithm for video surveillance. GPC
15. Herk M (1992) A fast algorithm for local minimum and maximum filters on rectangular and octagonal kernels. Pattern Recogn Lett 13:517–521
16. Schechner YY, Narasimhan SG, Nayar SK (2001) Instant dehazing of images using polarization. CVPR 1:325
17. Nayar SK, Narasimhan SG (1999) Vision in bad weather. ICCV 820
18. He K, Sun J, Tang X (2010) Guided image filtering. In: Proceedings of ECCV, pp 1–14
19. NVIDIA: CUDA Programming Guide 2.0. http://www.nvidia.cn/docs/IO/57399/NVIDIA_CUDA_Programming_Guide_2.0Final.pdf