GPU accelerated Foreign Object Debris Detection on Airfield Pavement with visual saliency algorithm

Jun Qi, Guoping Gong and Xiaoguang Cao

Image Processing Center, Beihang University, Beijing, China

E-mail: xgcao@buaa.edu.cn

Abstract. We present a GPU-based implementation of visual saliency algorithm to detect foreign object debris(FOD) on airfield pavement with effectiveness and efficiency. Visual saliency algorithm is introduced in FOD detection for the first time. We improve the image signature algorithm to target at FOD detection in complex background of pavement. First, we make pooling operations in obtaining saliency map to improve recall rate. Then, connected component analysis is applied to filter candidate regions in saliency map to get the final targets in original image. Besides, we map the algorithm to GPU-based kernels and data structures. The parallel version of the algorithm is able to get the results with 23.5 times speedup. Experimental results elucidate that the proposed method is effective to detect FOD real-time.

1. Introduction

A clean and flat airport pavement provides a safe environment for airplane's take-off and landing. It is vital to detect foreign object debris(FOD) timely, such as little stones and screws in pavement, which could make damages to airplane tires and even lead to heavy accident.

Li [1] designed a multi-sensor system and applied combined methods to make FOD detection. Liu [2] proposed a traversal method cutting images into small pieces and extracting features, such as Harris corner, GLCM of each piece for classifying. However, these two methods are complicated or time consuming. The detection of FOD with effectiveness and efficiency becomes a challenging task.

Recently, visual saliency has drawn much attention in many areas, including neurobiology and computer vision [3]. It is suggested that human's attention is directed to visually distinctive regions in an image rapidly without much guidance. This capability inspires us to make saliency detection, aiming at highlighting visually salient regions or objects in an image. Considering the background characteristics of airport pavement, where FOD would be visually conspicuous, we apply visual saliency algorithm to make FOD detection in airfield pavement.

At the same time, in order to meet the need of real-time detection, parallelization is required to reduce the computation time. Nowadays, GPU has been widely used in various fields, especially in computer vision and image processing [4]. GPU contains thousands of cores to process parallel workloads efficiently. Using GPU can make full use of hardware performance and improve the speed of detection algorithm obviously.

Our main contributions are summarized as follows:

First, an effective visual saliency algorithm for FOD detection is proposed.

Second, we accelerate the algorithm using GPU with highly parallel structure, and achieve real-time detection, which will be of great significance in real scene.

Content from this work may be used under the terms of the Creative Commons Attribution 3.0 licence. Any further distribution of this work must maintain attribution to the author(s) and the title of the work, journal citation and DOI.
Published under licence by IOP Publishing Ltd

The paper is organized as follows. Section 2 introduces our visual saliency algorithm. The GPU implementation details are presented in Section 3. In Section 4, experiments and analysis are made to show the performance and speed of proposed algorithm. Conclusions are drawn in Section 5.

2. Visual saliency algorithm

In existing works, visual saliency algorithms can be generally divided into two parts, spatial domain and frequency domain. Considering the speed of FOD detection in airfield pavement, we take frequency domain algorithms into account to compute saliency, which can be efficiently mined through simple spectrum modulation.

Our visual saliency algorithm is shown in Fig 1. Firstly, preprocessing methods including frequency and spatial filter are employed to eliminate interference of complex background. Secondly, we use an improved algorithm based on image signature to obtain saliency map. Finally, saliency blocks are segmented with connected component analysis and the locations are mapped to original image.

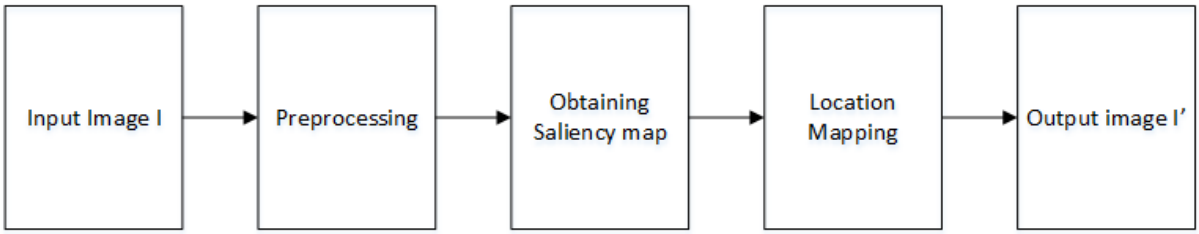


Figure 1.visual saliency algorithm.

2.1. preprocessing

Actually, airfield pavement image has complex background with road groove, maker line, underground lamp, which can make obstruction to FOD detection. Firstly, we make Discrete Fourier Transform(DFT) to the original image, and do band pass in frequency domain. Road groove was filter without losing FOD detection object. Then, Inverse Discrete Fourier Transform(IDFT) and normalization are made to get image in spatial domain back. At last, average filter is used to eliminate noisy points.

2.2. obtaining saliency map

Image signature [5] is defined as follows:

A grey image combines with foreground and background.

$$\mathbf{x} = \mathbf{f} + \mathbf{b} \quad \mathbf{x}, \mathbf{f}, \mathbf{b} \in \mathbb{R}^N \tag{2.1}$$

$\mathbf{f}$ represents the foreground and is assumed to be sparsely supported in the standard spatial basis. $\mathbf{b}$ represents the background, and is assumed to be sparsely supported in the basis of the Discrete Cosine Transform (DCT). In airfield pavement image, our detection target FOD can be treated as foreground. The image signature is defined as:

$$\mathbf{x}=\text{ImageSignature}(\mathbf{x}) = sign(DCT(\mathbf{x})) \tag{2.2}$$

We can approximately isolate the support of $\mathbf{f}$ by taking the sign of the mixture signal $\mathbf{x}$ in the transformed domain and then inversely-transform it back into the spatial domain.

$$\bar{\mathbf{x}} = IDCT(sign(\mathbf{x})) \tag{2.3}$$

Pooling is widely used in deep neural network, which can maintain spatial invariance. The pooling window size can be arbitrary, and windows can be overlapping. We improve Hou's saliency model by applying pooling methods.

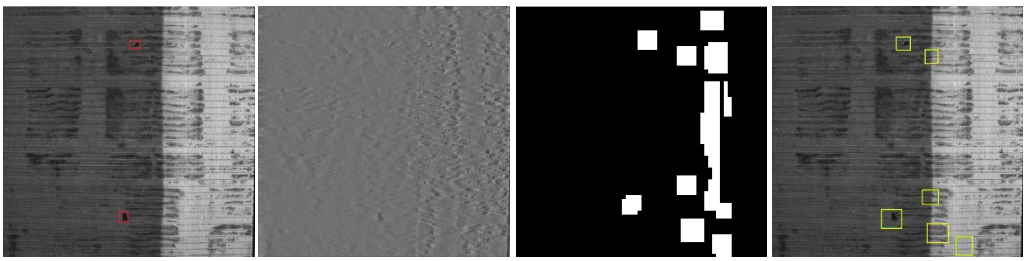
In order to improve speed, we first make subsampling and overlapping pooling to the image to reduce the dimension to from (2048, 2048) to (202, 202). Then, image signature is calculated to the image.

At last, we apply non-overlapping mean pooling and non-overlapping max pooling to improve recall. Max pooling is a way of taking the most responsive node of the given interest region. The saliency map size we obtained finally is (64, 64), where 1 pixel corresponds to 32 pixels in original image. Max pooling helps find the most likely node to avoid missing true targets in location mapping stage.

2.3. location mapping

To get final saliency regions, we make connected component analysis of the saliency map. And rules are made to map location in original image according to the area of connected component. For example, large area of connected component is excluded, which is not our detection target, such as small stones and screws. Small area is judged as effective saliency regions and its location will be mapped to a box in original image.

The results of our algorithm are shown in Fig 2. Fig 2(a) is the original image. Fig 2(b) is image after preprocessing. Fig 2(c) is the saliency map and Fig 2(d) is the location mapping results.



(a) (b) (c) (d)

Figure 2. Results of visual saliency algorithm.

3. Implementing with GPU

Compute Unified Device Architecture(CUDA) is a parallel computing platform and a software environment for parallel computing created by Nvidia [6]. It allows software developers to use a CUDA-enabled graphics processing unit(GPU) which can launch thousands of threads in parallel for general purpose processing. In order to achieve higher performance, we make parallel optimizations to some parts of the algorithm in consideration of memory model and threads scheduling of CUDA.

3.1. convolution

Convolutions are used widely by many applications in image processing task, including blur filters, mean pooling and so on. For instance, each pixel has to sum up its own value and others values in neighbourhood in an average filter. Then, an average application will be made to get the output pixel value.

It is noticed that the calculation of adjacent output pixel value shares most input pixel value. Considering shared memory's high bandwidth and low latency, we can load pixels to be filtered and the corresponding apron pixels [7] into shared memory first. Then, in next step, data to be calculated would be read from shared memory other than global memory, which could reduce repeated and time-consuming communications between global memory and processing units.

3.2. reduction

To do normalization of an image, we have to get minimum and maximum pixel value of an image first. Sequential computation complexity of the operation reduction is over the number of input data $O(N)$, whereas parallel computation complexity can be $O(\log N)$. Following is the parallel reduction algorithm implemented in CUDA.

For a (2048, 2048) size image, we set gridDim (64, 64), and blockDim (32, 32). We first make reduction in every block, which has 1024 threads. Threads allocation details are shown in Fig 3.

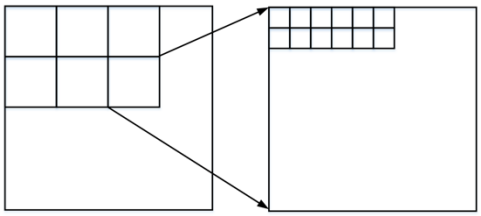


Figure 3. Threads allocation.

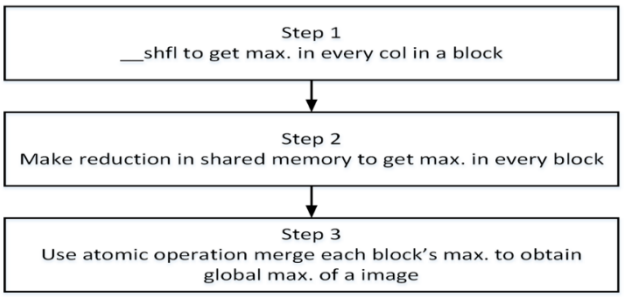


Figure 4. Reduction steps.

SM3.0 introduced the "shuffle" instruction, which can be used to perform a reduction across the 32 threads in a wrap [8]. In a block which has 32*32 threads, we get maximum pixel value in every row using __shfl instruction and copy the value to an array with size 32, which is allocated in shared memory. Then, we only have to make reduction in the shared memory to get the maximum of each block. In order to eliminate the need to invoke multiple kernels, we use atomic operations atomicMax to merge maximum value in every block. The reduction details are show in Fig 4.

3.3. image signature

The main operations in image signature are making DCT and IDCT transformation. Take DCT for example, a two dimension DCT defines as follows:

$$F(u,v) = c(u)c(v)\sum_{i=0}^{N-1}\sum_{j=0}^{N-1} f(i,j)\cos\left[\frac{(i+0.5\pi)}{N}u\right]\cos\left[\frac{(j+0.5\pi)}{N}v\right] \tag{3.1}$$

$$c(u) = \begin{cases} \sqrt{1/N}, & u = 0 \\ \sqrt{2/N}, & u \neq 0 \end{cases} \quad u = 0,1,...,N-1 \tag{3.2}$$

The two dimension approach performs DCT on input sample X by subsequently applying DCT to rows and columns of the input signal. In matrix notation this can be expressed using the following formula:

$$F = AXA^T \quad A(i,j) = c(i)\cos(\frac{j+0.5\pi}{N}i) \tag{3.3}$$

As DCT 8×8 algorithm [9] will loss accuracy of the result, we make DCT transform using matrix multiply instead. CUDA provides cuBLAS library to make matrix multiplying operations with high efficiency. Coefficient matrix are calculated only once and load to device memory in initialization stage. The image signature of each image can be computed very fast with CUDA in later stage.

3.4. multi-streams and multi-GPUs

Stream is a sequence of operations that execute in issue-order on the GPU [10]. SM 2.x-class and later GPUs are capable of concurrently running multiple kernels. To improve the utilization of hardware, we can process a batch of images with starting multiple streams, which would be concurrently running. Comparing to processing one by one, batch operation will reduce average processing time of a single image.

At the same time, multiple GPUs can operate in parallel for enough computational density task. CUDA has supported multiple GPUs since the beginning, and each GPU can be controlled by a

separate CPU thread. An amount of images is assigned to multiple GPUs, which highly improve the processing speed.

4. Experiments and analysis

Experiments were performed using two Nvidia TITAN X GPU and one Intel i7-4790K CPU. In order to evaluate the detection performance of our algorithm, we set up a pavement image dataset containing 1000 images. The FOD on the airfield pavement is mainly stones and screws.

The recall and precision are defined as follows in Fig 5.

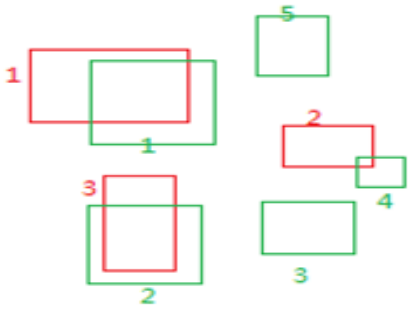


Figure 5. The definition of recall and precision.

Red boxes are real target and green boxes are generated by saliency algorithm. It is regarded as correct detection that the generated box covers the real target mostly. The recall of Fig 5 is 2:3, and the precision is 2:5.

Table 1 gives the detection precision and recall of the saliency detection algorithm.

Table 1. Recall and precision of Hou's model and ours.

	recall	*precision*
Image Signature	87.83%	64.76%
Ours	96.46%	62.93%

The result shows that our algorithm gets higher recall in FOD detection, which guarantees that almost all FOD will be detected in the original image.

As saliency map segmentation and connected domain analysis need complicated logical operation, which are not suitable for transplanting to GPU, we make parallel accelerations on the remaining parts of the detection framework.

We experimented with algorithms by varying number of concurrent streams. Fig. 6 shows the decrement in average executional time as stream size gets varied from 1 to 50. It is often better to set more streams to make full utilization of hardware.

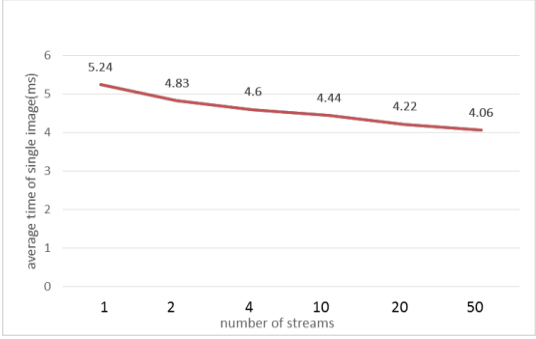


Figure 6. Average processing time of single image with multi-streams

Table 2 shows the average execution time of single image in CPU, one GPU and multiple GPUs.

Table 2. Speed up with GPU.

	Preprocess and image signature time(ms)	Speed up	Saliency map segmentation and connected component analysis time(ms)	Total time	Total Speedup
CPU	287.94	--	8.54	296.48	--
GPU	4.06	70.9	--	12.60	23.5
Two GPUs	2.15	133.9	--	10.69	27.7

Saliency map segmentation and connected component are executed in CPU only. The experiment results show that the GPU implementation parts obtained 70.9 times speedup and get approximately linear speedup in multiple GPUs with 133.9 times speedup. In conclusion, 23.5 times speedup is achieved using single GPU in total algorithm.

5. Conclusion

The paper presents a GPU accelerated visual saliency algorithm, which performs effectiveness and efficiency in FOD detection on airfield pavement. We improve image signature algorithm with pooling operations in generating saliency map and connected component analysis in locating saliency blocks. We make parallel optimizations to some parts of the algorithm with GPU. Experiments result shows that recall rate gets increased and the GPU implementation achieves 23.5 times speedup. The processing time of single image reduce to 12.60ms, which is significant in real-time detection.

References

[1] Y Li and G Xiao 2011 A new FOD recognition algorithm based on multi-source information fusion and experiment analysis *International Symposium on Photoelectronic Detection and Imaging Advances* vol 34 (Infrared Imaging and Applications) pp 112-112
[2] D Liu, X Cao, B Xue and H Li 2013 Feature analysis and detection of FOD under complex background of airport pavement vol 21 (Electronic Design Engineering) pp 12-15
[3] Itti Laurent and Christof Koch 2000 A saliency-based search mechanism for overt and covert shifts of visual attention vol 40(Vision research) pp 1489-1506.
[4] Vinith B, Akhila M K, Naik N, and Rathna G N 2015 GPU Accelerated Face Recognition System with Enhanced Local Ternary Patterns Using OpenCL *Digital Image Computing: Techniques and Applications (DICTA), 2015 International Conference on IEEE* pp 1-7
[5] Hou X, Harel J and Koch C 2012 Image signature: Highlighting sparse salient regions *IEEE transactions on pattern analysis and machine intelligence* pp 194-201
[6] *Cuda C. Programming guide* 2012 pp 1-175
[7] Podlozhnyuk V 2007 *Image convolution with CUDA* (NVIDIA Corporation white paper) p 6
[8] Harris M 2014 *CUDA Pro Tip: Do The Kepler Shuffle* Parallel Forall. Np
[9] Obukhov A and Kharlamov A 2008 *Discrete cosine transform for 8x8 blocks with CUDA* NVIDIA white paper
[10] Wilt Nicholas 2013 *The cuda handbook: A comprehensive guide to gpu programming* Pearson Education