GPU-enabled pavement distress image classification in real time

Kristina Doycheva (Corresponding Author)
Ruhr-University Bochum
Universitätsstr. 150, Gebäude IC 6-69, 44801, Bochum, Germany
kristina.doycheva@rub.de
+49 (2) 234 32 24640

Christian Koch
University of Nottingham
Room B27 Coates Building, University Park, NG7 2RD, Nottingham, UK
christian.koch@nottingham.ac.uk
+44 (0) 115 84 68933

Markus König
Ruhr-University Bochum
Universitätsstr. 150, Gebäude IC 6-59, 44801, Bochum, Germany
koenig@inf.bi.rub.de
+49 (2) 234 32 23047

ABSTRACT

Pavement assessment is a crucial process for the maintenance of municipal roads. However, the detection of pavement distress is usually performed either manually or offline, which is not only time-consuming and subjective, but also results in an enormous amount of data being stored persistently before processing. State-of-the-art pavement image processing methods executed on a CPU are not able to analyze pavement images in real time. To compensate this limitation of the methods, we propose an automated approach for pavement distress detection. In particular, GPU implementations of a noise removal, a background correction and a pavement distress detection method were developed. The median filter and the top-hat transform are used to remove noise and shadows in the images. The wavelet transform is applied in order to calculate a descriptor value for classification purposes. The approach was tested on 1549 images. The results show that real-time pre-processing and analysis are possible.

INTRODUCTION

In recent years, the condition of municipal roads has deteriorated rapidly, leading to increased fuel consumption, thus increased emissions and environmental pollution, and even greater number of vehicle damages and traffic accidents [Spielman 2014]. To reduce the negative impact of deteriorated roads on the driving quality, roads need to be maintained in good condition, for example by repairing parts of the road surface where pavement distress, visible as cracks or potholes, is present. For this purpose, knowledge about the exact location of pavement distress is required and pavement assessment is an essential task [Orr 2015].

Several techniques for distress detection in asphalt pavement have been proposed in the last few years. The most intuitive approach is manual observation, during which an expert makes notes about the condition of the road by hand while walking over the road shoulder. The evaluation is performed with the help of manuals specifying criteria for pavement assessment and rating [NCHRP 2004]. There also exist methods which are based on the various types of pavement data being collected, such as sensor data or images of the pavement surface. Sensor devices are often utilized to measure parameters of the pavement surface. This approach is referred to as *sensor-based pavement assessment*. On the other hand, visual data obtained by images or videos of the pavement surface is also used for pavement assessment. The so-called *visual-based pavement assessment techniques* analyze features of the images or video frames with respect to criteria identifying the presence of distress. Visual-based pavement assessment techniques have been widely applied recently, because they are less subjective and hazardous compared to manual observations [Koch et al. 2015].

Furthermore, these techniques can be classified as purely manual, semi-automated or automated based on the manner of processing the data. The observation by experts is an example of a purely manual technique, while semi-automated and automated methods require only little or no human intervention. Despite of the advances in automated pavement assessment in recent years, there is still room for improvement. For example, video data is usually stored before it is actually processed. Considering the length of the municipal road network in Germany, which is approximately 610,000 km according to the German Association of Towns and Municipalities [DStGB 2014], the amount of stored data is large (approx. 5 gigabytes per kilometer). To reduce this amount of data, methods capable of analyzing the pavement surface in real time are required. Such methods could be employed in order to store only those images on which distress had been identified and discard all other images without distress, resulting in less memory requirements and less subsequent processing time needed compared to the state-of-the-art case.

However, although the central processing unit (CPU) technology has evolved during the last decade, modern CPUs are still not able to cope with the requirement of real-time execution of related analysis methods, mainly due to the fact that image pre-processing is also needed. For instance, noise removal as well as correction of non-uniform background illumination needs to be applied to the images to enhance their quality in order to produce more accurate analysis results.

Yet, the real-time processing requirement can be fulfilled by utilizing Graphics Processing Units (GPUs). Applied not only for graphic operations, but also for computational tasks, GPUs have proven their efficiency in diverse scientific fields in recent years [Owens et al. 2005].

In this work, GPUs were used to accelerate the pre-processing and the analysis of pavement surface images for the purpose of real-time pavement defect detection. In particular, a noise removal method, a shadow removal method and an approach towards pavement analysis based on the wavelet transform were implemented and validated.

The next two sections provide information on state of practice and research concerning pavement distress detection. Afterwards, GPUs are introduced. The approach is presented in thereafter, and then the implementation is described. Performance tests were carried out to evaluate the capability of the proposed implementation to process the images in real time. A case study was performed to validate the approach and is described in the “Case Study” section. The paper concludes with a summary of the main contributions and an outlook on future developments.

STATE OF PRACTICE

In the United States, the annual assessment and reporting of pavement conditions is currently performed by transportation departments. For example, the New York State Department of Transportation collects a variety of information about the pavement condition in cooperation with the Federal Highway Administration (FHWA) [NYSDOT 2010]. A pavement surface rating survey is conducted by a team consisting of a driver and a rater. The rater assesses the condition of the pavement based on what is seen on the pavement and photographs of the pavement at each rating point. As stated in New Yorks’s Pavement Condition Assessment Document [NYSDOT 2010], the rater should be experienced in condition survey procedures and possess knowledge of road construction.

In Germany, the state of practice is similar. For example, in Bochum in 2013 seven teams with 15 employees have manually been assessing the pavement condition using portable computers [Buske 2013]. The current data is entered in a database by extending very detailed road maps. According to Carlos dos Santos [Buske 2013], this procedure is very laborious and one team consisting of two employees can only assess two kilometers of road per day.

Obviously, the surveys are mostly conducted manually, but as technology improves, automated assessment should become possible in the near future. For instance, a rule requiring rear visibility technology in all new vehicles by May 2018 has been issued by the U.S. Department of Transportation's National Highway Traffic Safety Administration (NHTSA) [2014]. This rule has been issued in order to expand the required field of view for all passenger cars, trucks, multipurpose passenger vehicles, buses, and low-speed vehicles with a gross vehicle weight of less than 10,000 lbs. According to this rule, an area behind the vehicle which encompasses 5 feet laterally from the longitudinal centerline of the vehicle and extends 20 feet rearward of the vehicle's rear bumper must be visible to the driver.

STATE-OF-RESEARCH METHODS FOR VISION-BASED PAVEMENT DISTRESS DETECTION

Pre-processing

In order to guarantee accurate analysis results, pre-processing operations are applied to the pavement images. An issue related to distress in pavement images is the existence of noise. Varadharajan et al. [2014] calculated the blur magnitude of the images and selected only images for which the blur magnitude was below a certain threshold value. Gaussian smoothing was applied by Li et al. [2014] for denoising.

Median filter

The most commonly applied method for noise removal is median filtering [Lokeshwor et al. 2013, Radopoulou and Brilakis 2014]. The median filter is an order-statistics filter used very often for noise reduction [Gonzalez and Woods 2006]. It introduces less blurring to the image than linear filters of the same size and it is particularly effective in the presence of salt-and-pepper noise. Experimental results have shown that the median filter has a good performance in gray and RGB images [Ahmed et al. 2015]. The median filter replaces the value of the pixel on which the kernel is centered by the median value of the gray levels in the neighborhood of that pixel. To apply the median filter, the gray level values of the pixels in the neighborhood including the value of the pixel itself are sorted in an ascending or descending order. Then, the value in the middle of the sorted sequence is taken and assigned to the pixel in the center of the kernel. Yet, the median filter is characterized by a high computational cost. The computational complexity for sorting n values, a basic step within median filtering, with efficient sorting algorithms is $O(n*\log n)$.

Another problem related to pavement images is the non-uniform background illumination. Commonly, the images are taken under various lighting conditions because of different weather conditions or varying times of day. This results in a non-uniform background illumination and lets shadows exist in the images. Since most of the analysis methods are based on the assumption that distress pixels, such as crack pixels, have a darker intensity than pixels belonging to the undamaged background, non-uniform background illumination could induce misleading results.

Several methods to handle this problem have been proposed. Varadharajan et al. [2014] selected for the analysis only images taken under good weather conditions (i.e., when the weather was overcast or mostly cloudy). However, the selection of the images is also a manual and time-consuming process and all images have to be stored before the analysis can begin. Zou et al. [2012] presented a geodesic shadow-removal algorithm which is able to preserve the cracks in the images while removing shadows in the background. Cheng and Miyojim [1998] proposed an image enhancement algorithm which corrects non-uniform background illumination by dividing the image into rectangular windows. For each window, the average light intensity is calculated and multipliers are generated for all pixels based on the window average intensity and a common base intensity.

Top-hat transform

The top-hat transform [Gonzalez and Woods 2006] with a larger structuring element can be used to estimate the background and subtract it from the image. It has been shown [Jähne and Haussecker 2000; Solomon and Breckon 2010; Wu et al. 2008] that the top-hat transform can be used for mitigating illumination gradients and producing evenly illuminated images without shading variations. It is useful for enhancing details in the presence of shading. Opening the image with a structuring element large enough so that it does not entirely fit within the details, here within the distress area, produces an estimate of the background across the image. By subtracting the background (i.e. the opening) from the original image, an image with more uniform background can be obtained.

The opening $f \circ b$ of an image f by a structuring element b is denoted as

$$f \circ b = (f \ominus b) \oplus \mathrm{b} \tag{1}$$

where $\ominus$ and $\oplus$ denote erosion and dilation, respectively. Erosion and dilation are morphological operations that consist in convoluting an image with a kernel called structuring element [Gonzalez and Woods 2006]. In case of dilation, the maximal gray level value overlapped by the structuring element anchored at a certain pixel in the image is used to replace the value of this pixel. As a result of the dilation, bright regions within the image become larger. Hence, the operation is called dilation. In case of erosion, the minimal value is used, resulting in bright valued areas getting thinner in a manner similar to erosion in geomorphology and geology.

As in the case with the median filter, the main drawback of the top-hat transform is its computational complexity. The size of the structuring element required to preserve the edges or details in the images leads to a vast number of pixels being considered for each anchor point.

Image analysis

A range of methods for distress detection in pavement images has been proposed in recent years. Most of them have been specifically developed for particular types of distress, such as cracks, potholes or patches. The role of digital image processing as a tool for pavement distress evaluation was described by Georgopoulos et al. [1995]. A critical assessment of available distress segmentation methods for crack detection and classification was presented by Tsai et al. [2010].

Cracks are the most common distress type and, consequently, the majority of the methods presented recently consider cracks. An automatic crack detection system was proposed by Oliveira & Correia [2013]. The system is capable of crack type characterization and a methodology for the assignment of crack severity levels was introduced. Subirats et al. [2006] used wavelet transforms for crack detection, while Vivekanandreddy et al. [2014] utilized Hough transforms for this purpose. Morphology-based methods have also been applied. For example, Tanaka and Uematsu [1998] suggested black pixel extraction, saddle point detection, linear feature extraction and connecting processing for crack detection in road surface images. Fang et al. [2014] presented a crack detection technology based on an improved K-means algorithm.

Zou et al. [2012] built a crack probability map using tensor voting to enhance the connection of crack fragments. After sampling a set of crack seeds from the crack probability map, minimum spanning trees are defined from a graph model of these seeds and recursive tree-edge pruning is applied to identify cracks. Li et al. [2014] classified image pixels into two categories: pixels that belong to cracks and pixels that do not belong to cracks. Then, they applied Otsu's segmentation method to separate the foreground from the background. The images containing cracks are afterwards classified to distinguish between linear and alligator cracks using binary trees and back propagation neural networks. Varadharajan et al. [2014] also adopted machine learning approaches. Considering images, which can contain cars, traffic signs and buildings, they segmented the ground plane out from the rest of the image and calculated feature descriptors based on the color and texture of the pixels. Using data annotated by humans, they trained a support vector machine capable of classifying the images. Moussa and Hussain [2011] used machine learning, namely support vector machines, and applied graph cut segmentation to segment an image into crack and background pixels. They extracted seven features from a binary vector created after segmentation. The features were used to classify the crack type as transverse cracking, longitudinal cracking, block cracking, or alligator cracking. In addition, they also proposed an approach to calculate the extent and severity of the crack. An algorithm based on the Gabor filter was proposed by Salman et al. [2013]. After convolution with the filter, the real component of the resulting image was thresholded and a binary image was obtained. Huang and Xu [2006] divided the image into cells for classification purposes. Each cell was classified as a crack or non-crack cell depending on its contrast.

Compared to cracks, approaches towards patch detection in pavement images are fewer in number. Radopoulou and Brilakis [2014] applied morphological operations to segment out patch regions. Texture information was also used to generate feature vectors of both intact and patch regions. Cafiso et al. [2006] applied a clustering method to analyze pavement images with respect to patches.

Koch and Brilakis [2011] proposed a method for pothole detection in asphalt pavement images. They first used histogram shape-based thresholding to segment an image into defect and non-defect regions. The potential pothole shape was approximated based on morphological thinning and elliptic regression. An improved method capable of tracking potholes in subsequent frames is presented in [Koch et al. 2013]. Buza et al. [2013] also employed image processing and spectral clustering for identification and rough estimation of potholes. In addition, they estimated the surface of the potholes. Yu and Salari [2011] introduced an approach for pothole detection and severity management based on laser imaging. The proposed algorithm also analyses the severity of the pothole.

Methods exist capable of identifying pavement distress in general. Some of them, namely multi-resolution texture analysis techniques using wavelet, ridgelet, and curvelet-based texture descriptors, were compared in [Nejad and Zakeri 2001]. The curvelet-based method outperformed all other multi-resolution techniques for pothole distress, while the ridgelet-based yielded the most accurate results for cracks.

Most of the presented methods were developed solely for a specific type of distress. Since the idea of this work is to roughly assess the condition of the pavement surface, methods capable of detecting all types of distress need to be investigated. Thereby, it is not important whether the methods distinguish between the different distress types, but rather if they are suitable for parallel implementation. In order to enable real-time distress detection, we considered only methods which achieved good results for all types of distress and do not require many computational steps that depend on each other.

Wavelet transform for pavement distress detection

In this work, we chose a method based on the wavelet transform for pavement distress detection and evaluation as it fulfills the requirements mentioned above. The method was proposed by Zhou et al. [2006] and tested on 81 images. According to the developers of the method, it achieved 100% reliability for these 81 images. Initially applied for signal processing, the wavelet transform is used to decompose an image into a set of different-frequency components. Based on the frequency, the components are arranged in groups called subbands. The subband components are calculated by applying low pass (L) and high pass (H) digital filters to the image. (The original image can be reconstructed from the wavelet components.) After one pass of the filters, the image is decomposed into four subbands: three detail subbands (HL, LH, HH), and one approximation subband (LL), whereby each subband has a width of ½ of the original image width and a height of ½ of the original image height. The detail subbands contain detail components with different orientation. HL contains the horizontal, LH the vertical, and HH the diagonal components. An example of an image before application of the wavelet-transform is presented in Figure 1. The horizontal details of the crack image are represented in the horizontal subband HL. The approximation subband is further decomposed into four subbands. In this way, different levels of decomposition can be achieved. In Figure 2, the 3-level wavelet transform is presented. The LL_3 subband contains approximation coefficients and is most similar to the original image before applying the wavelet transform.

Several wavelet families, i.e. sequences of functions that are performed to transform an image into the wavelet domain, exist. The most commonly used are the Haar wavelet [Haar 1910] and the Daubechies wavelet [Daubechies 1990]. The Haar wavelet is highly suitable for parallel (or GPU) implementation. Hence, it was chosen for the real-time detection of pavement distress in this work. The Haar transform is based on a technique called *averaging and differencing*

[Mulcahy] which only makes use of the simple mathematical operations addition, subtraction and division by two. First, the average sum and the average difference of each pair of neighbor elements in a row of the image are calculated. The sum is stored as a coefficient in the L subband, while the difference is stored in the H subband. This step is performed for all rows of the image. Afterwards, the same step is performed column-wise for all vertical neighbors in the image. The horizontal and vertical step can be combined and executed at once, as shown in Figure 3, where A, B, C, and D denote pixels and the corresponding wavelet coefficients are highlighted in the transformed "image" on the right.

When applying the wavelet transform on pavement images, Zhou et al. observed that a homogeneous background is transformed into the approximation subband, while distress is represented in the detail subbands. Considering the latter observation, Zhou et al also developed three statistical criteria for distress detection: standard deviation of wavelet coefficients (STD), high-frequency energy percentage (HFEP), and high-amplitude wavelet coefficient percentage (HAWCP). STD and HAWCP correctly detected all the distresses in the images. However, 2.6% of the images which actually do not contain distress were incorrectly isolated by STD as distress images, while HAWCP did not isolate any image wrongly. Hence, HAWCP is used in the work presented in this paper.

HAWCP is calculated only at the first level of the wavelet transform, which results in a reduced number of required wavelet transform operations. HAWCP represents a measure of the number of wavelet coefficients in the detail subbands that are larger than a threshold used as an index for pavement distress. To calculate HAWCP, first the wavelet modulus *M* is obtained as

$$\mathrm{M}(\mathrm{p},\mathrm{q}) = [\mathrm{HL}^2(\mathrm{p},\mathrm{q}) + \mathrm{LH}^2(\mathrm{p},\mathrm{q}) + \mathrm{HH}^2(\mathrm{p},\mathrm{q})]^{\frac{1}{2}} \quad (2)$$

where (p,q) is the position of the coefficient in the corresponding subbands.

Then, the modulus is binarized according to Equation (3):

$$D(p,q) = \begin{cases} 1 \text{ if } M(p,q) \geq C_{th} \\ 0 \text{ if } M(p,q) < C_{th} \end{cases} \quad (3)$$

where D is the binarized modulus and C_{th} is a threshold value estimated by wavelet thresholding.

Finally, HAWCP is calculated as

$$HAWCP = \sum_{p=0}^{W/2} \sum_{q=0}^{H/2} D(p,q) \Big/ \left(\frac{W}{2}\frac{H}{2}\right) \quad (4)$$

where W and H represent the width and height of the image, respectively.

The HAWCP value ranges between 0 and 1 (or 0% and 100 %), where a value near 0 indicates a good pavement surface, and high HAWCP values represent pavement distress.

GRAPHICS PROCESSING UNITS

During the last few years, GPUs have emerged as powerful computational hardware available at low prices [Owens et al. 2005]. The utilization of GPUs for general-purpose computing (GPGPU) has gained interest among developers of non-graphical applications. Often combined with a CPU, GPUs are used to accelerate scientific, analytics, engineering, consumer or enterprise applications [Nvidia Corporation 2015]. While CPUs are remarkably suitable for control-intensive applications, such as searching or sorting, due to branch predictions, data-intensive applications like image processing are appropriate for GPUs [Gaster et al. 2013].

The most common GPU programming frameworks are the Compute Unified Device Architecture (CUDA) and the Open Computing Language (OpenCL). CUDA was developed by Nvidia and supports only Nvidia devices, while OpenCL can be executed on diverse platforms produced by different vendors, such as AMD, Intel, Nvidia, and others. OpenCL was developed by the

Khronos Consortium in 2008 and is often referred to as the *industry standard for heterogeneous computing* [Khronos OpenCL Working Group 2013].

In OpenCL, a single host is defined that is responsible for the coordination of code execution on one or more devices [Gaster et al. 2013]. The host also interacts with the environment external to the OpenCL program, for example with the user. The device can be a CPU, a GPU, a digital signal processor (DSP), or another processor supported by OpenCL. Streams of instructions called *kernels* (not to be confused with convolution kernels) are executed on the device. A portion of the code, called *host program,* runs on the host and defines kernels or collections of kernels that are submitted to the devices by issuing a command for execution. An instance of the kernel is executed for each point of an index space in parallel.

The kernels operate on the values of memory objects. Five distinct memory regions are defined in OpenCL, namely host memory, global memory, constant memory, local memory and private memory. They are used for different purposes. For example, global memory can be accessed by all kernel instances in contrast to local and private memory.

Stürmer et al. [2012] and Sharma and Vydyanathan [2010] proposed GPU implementations of the wavelet transform. However, in both cases the wavelet coefficients of the wavelet transform are calculated at all decomposition levels. The method proposed by Zhou requires only the values of the first wavelet decomposition level. Therefore, the computational overhead due to unnecessary further decomposition should be eliminated for the purpose of real-time pavement distress detection. Moreover, the computation of the HAWCP criterion could also be carried out on GPU, as shown in this paper.

PROBLEM STATEMENT AND OBJECTIVES

Despite of the advances in vision-based pavement distress detection, gaps still exist in research which we try to address in this paper. First, pavement assessment is usually carried out either manually or by using special dedicated vehicles. Second, the data acquired for pavement distress detection is mostly processed offline, which results in a huge amount of data being stored persistently.

To address the aforementioned problems, the following two research questions have to be answered:

1. How can we automate the pavement distress detection process, while using inexpensive vehicles?
2. How can we reduce the amount of data saved for offline processing?

APPROACH

This paper addresses the issues described previously by presenting an approach which is founded on common vehicles. Instead of using dedicated vehicles, the idea pursued hereby is to use vehicles which drive daily on the roads, such as buses and taxis. Nowadays, such vehicles are equipped with built-in cameras, for example backup cameras, which can be used not only to support the driver while parking, but also for other tasks, particularly in this case for road distress detection.

In order to address the second research question, we propose online processing of pavement images in real-time. With the aim of reducing storage consumption, only images which contain distress will be stored, while images of good pavement surface will be discarded directly after they have been taken and processed. However, to enable real-time pavement distress detection while driving, either methods which do not require a long execution time need to be developed or existing methods should be enhanced or implemented for faster architectures. In this work, GPUs are utilized to enhance the performance of existing pavement image pre-processing and analysis methods. As a result, real-time pavement distress detection is possible.

The approach proposed here is presented in Figure 4. To remove the noise, the images are first convolved with a median filter. Second, the top-hat transform is applied to produce a more uniform background. The third step in the pipeline is transforming the image into the wavelet domain. Then, the high-amplitude wavelet coefficient percentage is calculated. HAWCP is used as a descriptor for classification. Based on a previously generated classification model, the image is classified as a *good pavement image* or an *image containing distress*. This classification model is created in advance using existing machine learning algorithms. To this end, training images are acquired and manually labeled and a data mining tool is used to induce general rules that map pavement images to the two aforementioned categories. Currently, all steps, except classification, are implemented on GPU. An example of a processed image is presented in Figure 5.

IMPLEMENTATION

An overview of the implementation is depicted in Figure 6. First, the input image data that is initially located only on the host (CPU) needs to be transferred to the device (GPU). For this purpose, the image data is copied into a global memory buffer on the device. A kernel performs median filtering on this data and the result (denoised image) is saved in another memory buffer on the device. Then, a top-hat transform kernel is executed. The latter is used to correct the background of the image and the result is also saved in a buffer on the device. The wavelet transform and the calculation of the HAWCP descriptor are combined in one pavement analysis kernel. The wavelet coefficients are stored in local memory to achieve better performance. The HAWCP descriptor value is saved in global memory and, at the end, transferred to the host. In the current implementation, this value is submitted to a third-party learning machine called WEKA [Witten et al. 2011] and the image is classified based on a classification model generated by the learning machine with the help of the HAWCP values of training images.

Median Filter

There exist several implementations of the median filter on GPUs [Banger and Bhattacharyya 2013, Intel Corporation 2012]. Both implementations provide very good results in terms of performance enhancement. Since an Intel GPU is used for testing in this work, we adopted the implementation proposed by Intel. It uses partial bitonic sorting to perform median filtering.

Top-hat transform

Naïve implementation

The top-hat transform is performed by subtracting the opening of an image from the input. The opening is obtained by dilating the eroded image. Since there are no global synchronization barriers among different workgroups in OpenCL, at least two kernels are required for the GPU implementation of the top-hat transform. To guarantee that the erosion is completed for all pixels in the image, it is defined in its own kernel. After the kernel had been executed, a dilation kernel can be started. The last operation in the top-hat transform (i.e. the subtraction of the opening from the original image) can also be performed in the dilation kernel. The erosion and dilation kernels are implemented in a manner similar to the median filter. However, instead of computing the median value of the neighborhood, the minimal and maximal value are taken. This implementation is presented in Figure 6.

Separable filter implementation

Two-dimensional convolution operations can, in some cases, be separated into two one-dimensional filters, namely a horizontal and a vertical filter. The horizontal filter is first applied to the image row by row. Then, the vertical filter is applied column-wise to the result of the horizontal convolution. The separable convolution is associative, so the one-dimensional filters can be applied in reverse order. Separating the single 2D convolution into two 1D convolutions usually results in reduced execution time even on the CPU when the convolution is executed sequentially. This performance improvement can be explained if we look at Equations 5 and 6. For example, for a rectangular image convolution kernel, the 2D convolution requires a total of

$$(K*L)*(M*N) \tag{5}$$

pixel accesses, where K and L denote the width and height of the convolutional kernel, respectively, and M and N represent the width and height of the image, respectively.

When the 1D horizontal convolution is performed, the number of pixel accesses is only

$$K*(M*N) \tag{6}$$

for the 1D vertical convolution it is

$$L*(M*N) \tag{7}$$

If we execute these convolutions consecutively, we obtain

$$(K + L)*(M*N) \tag{8}$$

pixel accesses.

Theoretically, this leads to an improvement factor of

$$K*L/(K+L) \tag{9}$$

Since the top-hat transform is based on erosion and dilation, it can be implemented as a combination of consecutive horizontal and vertical filters. An overview of the improved implementation is presented in Figure 7, in analogy to Figure 6.

Still, the number of sorting/search operations required to find the minimum or maximum element in the one-dimensional filters is also lower than in case of the two-dimensional convolution. This allows for improvement factors even greater than expressed in Equation 9.

Wavelet transform and HAWCP

The wavelet kernel is executed for each group of four adjacent pixels in the image. For example, if we consider Figure 3, the same computations would be performed in parallel for the groups (A, B, E, F), (C, D, G, H), (I, J, M, N), and (K, L, O, P). The detail coefficients (i.e. LH, HH, and HH) are calculated using addition and subtraction. Then, the modulus at the certain position is calculated according to Equation 2. The value of the modulus is compared to the threshold value and if it exceeds it, the HAWCP value is incremented. Atomic operations are used to increment the HAWCP value. A schematic of the implementation is presented in Figure 8.

PERFORMANCE EVALUATION

To evaluate the computational speed-up achieved by implementing the median filter, the top-hat transform and the wavelet transform on GPU, performance tests were carried out. The objective pursued was to measure the time required to execute the different pavement distress detection steps on different architectures and to compare them. In particular, a sequential version of the methods executed on a CPU, an OpenCL parallel version executed on the same CPU, the OpenCL version executed on an integrated GPU, and the OpenCL implementation executed on a discrete GPU were compared. In case of the OpenCL implementations of the median filter and the top-hat transform, both the times for the 2D and for the separable convolution were measured. As recommended in [Intel Corporation 2013], the same set of operations was wrapped in the sequential and OpenCL implementations in order to make sure that the observed code patterns are as similar as possible. Moreover, to guarantee accurate results, the methods were invoked on 1000 images and the average value of all the 1000 executions was taken for performance evaluation.

Profiling events were used to measure the OpenCL execution time. The data transfer time (i.e. the time required to write data to the device or read data from the device) and the kernel execution time were tracked separately due to the following two reasons. First, both the data transfer time and the kernel execution time are highly dependent on the hardware. The time needed to transfer data between a host and an integrated GPU is usually much lower than the time required to transfer the same data between the host and a discrete GPU. Second, if we consider Figure 4, it is obvious that only the input image data and the HAWCP results need to be transferred between the host and the device. All other intermediate results are saved in memory buffers on the device. Thus, only the kernel execution times are relevant for the overall performance evaluation of the real-time pavement assessment approach.

The OpenCL initialization time, i.e. the time required to create a program, a context, command queues, the kernels, and set the kernel arguments, is also not considered, because these initialization steps are executed only once at application startup and are not repeated for each frame or image that has to be processed.

The following hardware was used for the performance evaluation tests: a 2.10 GHz Intel Core i7-4600 CPU, an integrated Intel HD Graphics 4400 GPU, and a dedicated Nvidia Tesla C2070 GPU. In addition, the approach was tested on images of different sizes, namely 256x256, 512x512, 1024x1024, and 2048x2048 pixels, because universal rear view cameras have different resolutions. Resolutions of 500x500 pixels are common nowadays, but vehicle manufacturers have already developed rear view cameras with 1,300,000 pixels [Nissan Motor Corporation

2014]. The speed-up achieved by implementing the approach on GPUs was computed, . This speed-up is defined as shown in Equation 10.

$$\text{Speed-up} = \text{Sequential C++ time} / \text{Best OpenCL time} \tag{10}$$

Data transfer

The data transfer time differs depending on what kind of device is used. The time required to transfer the image data to the integrated Intel GPU and the dedicated Nvidia GPU are illustrated in Figure 9. The transfer to the discrete GPU is significantly slower than the transfer to the integrated GPU for large images.

The difference between the times required to transfer the HAWCP value of a single image is not so considerable, because only one value needs to be transferred.

Median Filter

In our work, we used a median filter with a square structuring element of a size 3x3. The execution times in milliseconds are shown in Table 1.

Top-hat transform

The top-hat transform was tested with a structuring element of a size 10x10. The performance evaluation results are presented in Table 2 in milliseconds. For all image sizes, the separable implementation executed on the dedicated Nvidia GPU was the fastest one. In contrast to the median filter, a considerable performance improvement was achieved by using separate horizontal and vertical filters.

Wavelet transform and HAWCP

The wavelet transform execution time, including the time required to calculate the HAWCP descriptor, is presented in Figure 10. The operations were executed approximately 109 times faster on the Nvidia GPU compared to the sequential CPU. As shown in Figure 10, the calculation takes more than 8 milliseconds when executed sequentially, which makes it unsuitable for real-time processing of videos taken at high speeds. In contrast, all GPU implementations require less than one millisecond, so there is sufficient time for pre-processing operations.

Overall enhancement

To compare the execution of the different implementations on the CPU and the two GPUs, the total execution times were calculated. As can be seen in Figure 11, in case of an image size of 2048x2048, the data transfer time is approximately 0.72 milliseconds, which is about 50% of the total execution time. However, the Nvidia execution still significantly outperforms all other implementations.

The total execution times for all image sizes are shown in Table 3. The speed-up calculated according to Equation 10 is also presented. In case of the Nvidia GPU, the total execution time is below 1.5 milliseconds. Theoretically, this allows processing more than 650 images per second.

CASE STUDY

To validate the approach, a case study was conducted. A road segment located in Bochum, Germany, was chosen for validation due to the presence of parts of the road with a good pavement surface and parts with pavement distress. The length of the road segment is approximately 24 kilometers. The road segment includes different types of pavement. An example of two different road surface textures is presented in Figure 14. To collect video data, a Basler acA2040-180kc camera was mounted on a rear door back carrier. As a variety of rear view cameras and vehicles exist, there are different ways and positions to mount the cameras. While license mounted cameras are easy to install on the existing license plate, surface mounted cameras are commonly mounted higher and would be a better choice for larger vehicles [Rearview Camera Reviews]. The setup of the camera in this case study tries to imitate state-of-the-art rear view camera setups as far as possible. The position and orientation of the camera are presented in Figure 12. The camera is capable of acquisition with a frame rate of up to 180 frames per second, which are currently not achievable by rear view cameras. However, we anticipate that in the near term vehicle manufacturers will use rear view cameras with even higher frame rates. The pitch angle of the camera is approximately -70 degrees, which is almost perpendicular to the road surface. The camera is placed at a height of 1.16 m above the road surface.

In order to enable the validation of the applied methods, all images were saved. Under real conditions, the images on which no distress was identified would be discarded and only images on which pavement defects were detected would be saved. To test the classification, 1549 images were selected. Both images of a good pavement surface as well as images containing cracks, potholes and patches were considered (Figure 13).

The images were manually labeled and ten-fold cross validation was performed in order to get a reliable error estimate. For this purpose, the data was split into ten approximately equal partitions. Each of these partitions was used for testing once, while the remaining data was used for training. Three algorithms were used for classification, namely the C4.5 [Quinlan 1993] algorithm, Multilayer Perceptron [Witten 2011], and Rotation Forest [Rodriguez 2006]. The results of the classification are presented in Table 4. The confusion matrix for the test images classified with the Rotation Forest algorithm is presented in Table 5. The time required to test the tree models on the training split was 0.02 seconds for C4.5, 0.66 seconds for Multilayer Perceptron, and 0.14 seconds for Rotation Forest.

The 5% of the images that were classified incorrectly are 77 images in total. Out of them, 15 images without distress were classified as images containing distress (false positives). In Figure 14, an example of a correctly classified intact pavement image (left) and an intact pavement image that was incorrectly classified as image containing distress (right) is presented. Nevertheless, this is still a promising classification result, because the objective of the rough distress detection stage described in this paper is to identify potential distress locations. In a further step, these potential locations will be assessed in detail by more comprehensive algorithms.

Vice versa, the other 62 images which actually contain distress were classified as distress free images (false negatives), mainly because of the different types of road surfaces considered in the case study. Consequently, the locations these images were acquired at would not be taken into account within the fine analysis. In order to counteract such errors, the methodology presented here will be extended by incorporating textural features.

CONCLUSION

Pavement condition assessment is a key component of pavement maintenance programs. Currently, pavement distress is detected during observations by trained personnel and reported manually. State-of-the-art automated methods for pavement distress detection utilize special vehicles equipped with sensors and cameras and try to compensate the limitations of the manual distress detection process. However, the need to reduce the amount of required memory to capture all pavement related data is still present.

With the aim of enabling real-time pavement image processing and, thus, reducing the amount of stored data, this paper proposed an approach based on graphics processing units (GPUs).

Specifically, GPU implementations of a noise removal, a background correction and a pavement distress detection method were developed. In order to remove noise in the images and correct their non-uniform background, the median filter and the top-hat were used. The wavelet transform was applied in order to calculate a descriptor value for classification purposes. Based on this value, the images were classified as good pavement images or images containing distress. To compare the performance of the GPU implementations against sequential applications and to validate the classification methodology, the approach was tested on 1549 images. The results show that by exploiting the computational power of the GPU it is possible to do pre-processing and analyze pavement images with a resolution of 2040 x 2048 pixels in real time. In addition, it has been demonstrated that the wavelet transform can successfully be applied on pavement images for the purpose of distress detection. Based on the high-amplitude wavelet coefficient percentage descriptor, 95% of the images used for testing were classified correctly by the Rotation Forest algorithm.

Yet, some images containing small cracks were incorrectly classified as good pavement images. The approach presented in this paper can be improved by combining multiple descriptors to obtain a more accurate representation of the distress. Future steps include the implementation of other pavement distress detection techniques on the GPU, as well as the employment of Graphics Processing Units for further pre-processing steps, such as the Bayer pattern de-mosaicing.

ACKNOWLEDGMENT

The authors gratefully acknowledge the support of this ongoing project by the German Research Foundation (DFG) under grant KO4311/2-1.

REFERENCES

Ahmed, E. S. A., Elatif, R. E. A., and Alser, Z. T. (2015). Median filter performance based on different window sizes for salt and pepper noise removal in gray and RGB images. *International Journal of Signal Processing, Image Processing and Pattern Recognition*, 8 (10), pp. 343-352

Banger, R. and Bhattacharyya, K. (2013). *OpenCL Programming by Example*. Packt Publishing, Birmingham, UK

Buske, K. (2013). Schlagloch-Kataster wird auf Vordermann gebracht. *DerWesten*, Online, available at: http://www.derwesten.de/staedte/bochum/schlagloch-kataster-wird-auf-vordermann-gebracht-id8177431.html, Accessed on 11.01.2016

Buza, E., Omanovic, S., and Huseinovic, A. (2013). Pothole detection with image processing and spectral clustering. *2nd International Conference on Information Technology and Computer Networks*, Antalya, Turkey

Cafiso, S., Di Graziano, A., and Battiato, S. (2006). Evaluation of pavement surface distress using digital image collection and analysis. *Seventh International Congress on Advances in Civil Engineering*

Cheng, H.D., and Miyojim, M. (1998). Novel system for automatic pavement distress detection. *Journal of Computing in Civil Engineering*, vol. 12, pp. 145-152

Daubechies, I. (1990). The Wavelet Transform, Time-Frequency Localization and Signal Analysis. *IEEE Transactions on Information Theory*, vol. 36, no. 5, pp. 961 – 1005

DStGB (Deutscher Städte- und Gemeindebund). (2014). PKW-Maut für alle Straßen richtiger Ansatz – Beteiligung der Kommunen an den Einnahmen unverzichtbar. Online, available at: http://www.dstgb.de/dstgb/Home/Pressemeldungen, accessed on 03.12.2014

Fang, C., Zhe, L., and Li, Y. (2014). Images crack detection technology based on improved K-means algorithm. *Journal of Multimedia*, 9 (6), pp. 822-828

Gaster, B. R., Howes, L., Kaeli, D.R., Mistry, P., and Schaa, D. (2013). *Heterogeneous Computing with OpenCL*: Revised OpenCL 1.2 Edition. Morgan Kaufmann Publishers Inc. San Francisco, CA, USA

Georgopoulos, A., Loizos, A., and Flouda, A. (1995). Digital image processing as a tool for pavement distress evaluation. *ISPRS Journal of Photogrammetry and Remote Sensing*, 50 (1), pp. 23-33

Gonzalez, R. C., Woods, R. E. (2006). *Digital Image Processing*. Prentice-Hall, Inc. Upper Saddle River, NJ, USA

Haar, A. (1910). Zur Theorie der Orthogonalen Funktionensysteme. *Mathematische Annalen*, vol. 69, pp. 948 – 956

Huang, Y, and Xu, B. (2006). Automatic inspection of pavement cracking distress, *J. Electron. Imag. 15 (1)*

Intel Corporation (2012). Intel® SDK for OpenCL* - Median Filter Sample. Document Number: 325264-003US, Revision: 1.3

Intel Corporation (2013). Intel® SDK for OpenCL* Applications 2013 R2 Optimization Guide. Document Number: 326542-003US

Jähne, B. and Haussecker, H. (2000). *Computer Vision and Applications: A Guide for Students and Practitioners*. Academic Press

Khronos OpenCL Working Group (2013). The OpenCL Specification, Version: 2.0. Document Revision 19

Koch, C. and Brilakis, I. K. (2011). Pothole detection in asphalt pavement images. *Advanced Engineering Informatics*, vol. 25, no. 3, pp. 507–515.

Koch, C., Georgieva, K., Kasireddy, V., Akinci, B., and Fieguth, P. (2015). A review on computer vision based defect detection and condition assessment of concrete and asphalt civil infrastructure. *Advanced Engineering Informatics* 29, pp. 196–210

Koch, C., Jog, G. M., and Brilakis, I. (2013). Automated pothole distress assessment using asphalt pavement video data. *Journal of Computing in Civil Engineering* 27(4), pp. 370-378

Li, L., Sun, L., Ning, G., and tan, S. (2014). Automatic Pavement Crack Recognition Based on BP Neural Network. *Promet – Traffic&Transportation*, Vol. 26, No. 1, pp. 11-22

Lokeshwor, H., Das, L.K., and Sud, S.K. (2013). Method for automated assessment of potholes cracks and patches from road surface video clips. *Procedia - Social and Behavioral Sciences* 104, pp. 312–321.

Moussa, G., and Hussain, K. (2011), A new technique for automatic detection and parameters estimation of pavement crack. *4th International Multi-Conference on Engineering Technology Innovation (IMETI 2011)*

Mulcahy, C. Image compression using the Haar wavelet transform. *Spielman Science and Math Journal*

NCHRP (National Cooperative Highway Research Program). (2004), Automated Pavement Distress Collection Techniques: A Synthesis of Highway Practice

Nejad F. M. and Zakeri, H. (2001). A comparison of multi-resolution methods for detection and isolation of pavement distress. Expert Systems with Applications, vol. 38, pp. 2857 – 2872

NHTSA (National Highway Traffic Safety Administration) (2014). Federal Motor Vehicle Safety Standards; Rear Visibility. Federal Register, the Daily Journal of the United States

Government, Online, available at: https://www.federalregister.gov/articles/2014/04/07/2014-07469/federal-motor-vehicle-safety-standards-rear-visibility, Accessed on 11.01.2016

Nissan Motor Corporation (2014). Nissan Motor develops the “Smart rearview mirror”, which helps provide clear rearward visibility in various conditions. Online, available at: http://www.nissan-global.com/EN/NEWS/2014/_STORY/140228-01-e.html, Accessed on 11.01.2016

Nvidia Corporation (2015). CUDA ZONE. Online available at: https://developer.nvidia.com/cuda-zone Accessed 11.06.2015

NYSDOT (New York State Department of Transportation) (2010). Pavement condition assessment. V2.0w

Oliveira, H. and Correia, P.L. (2013). Automatic road crack detection and characterization. *IEEE Transactions on Intelligent Transportation Systems*, 14 (1)

Orr, S. (2015). Officials want the public to report pothole locations. *Evansville Courier&Press*, Online, available at: http://www.courierpress.com/news/local-news/officials-want-the-public-to-report-pothole-locations_36253717, accessed on 09.06.2015

Owens, J. D., Luebke, D., Govindaraju, N., Harris, M., Krüger, J., Lefohn, A. E., and Purcell, T. J. (2005). A survey of general-purpose computation on graphics hardware. *Eurographics, State of the Art Reports*, August 2005, pp. 21-51

Quinlan, J. R. (1993). C4.5: programs for machine learning. Morgan Kaufmann Publishers Inc. San Francisco, CA, USA

Radopoulou, S.C., and Brilakis, I. (2014). Improving patch distress detection using vision tracking on video data. *Proceedings of the 21st International Workshop on Intelligent Computing in Engineering*

Rearview Camera Reviews. The complete buyers guide for Rear View Cameras. Online, available at: http://rearviewcamerareviews.com, Accessed on 11.01.2016

Rodriguez, J. J. & Kuncheva, L. I. (2006) Rotation Forest: A New Classifier Ensemble Method. IEEE Transactions on Pattern Analysis and Machine Intelligence 28(10), pp. 1619-1630

Salman, M., Mathavan, S., Kamal, K., and Rahman, M. (2013). Pavement crack detection using the Gabor filter. *16th International IEEE Conference on Intelligence Transportation System (ITSC 2013)*, pp. 2039–2044

Sharma, B., and Vydyanathan, N. (2010). Parallel Discrete Wavelet Transform using the Open Computing Language: a performance and portability study. *2010 IEEE Int. Symp. Parallel and Distributed Processing, Workshops and Ph.D. Forum (IPDPSW)*, pp.1 – 8

Solomon, C. and Breckon, T. (2010). *Fundamentals of Digital Image Processing: A Practical Approach with Examples in Matlab*, Wiley

Spielman, F. (2014). Chicago potholes trigger record number of damage claims. *Chicago Sun-Times*, Online, available at: http://chicago.suntimes.com/?p=167606, Accessed on 09.06.2015

Stürmer, M., Köstler, H., and Rüde, U. (2012). Fast wavelet transform utilizing a multicore-aware framework. *PARA'10 Proceedings of the 10th international conference on Applied Parallel and Scientific Computing*, vol. 2, pp. 313-323, Springer-Verlag Berlin, Heidelberg

Subirats, P., Dumoulin, J., Legeay, V., and Barba, D. (2006). Automation of pavement surface crack detection using the continuous wavelet transform. International Conference on Image Processing, Atlanta, USA

Tanaka, N. and Uematsu, K. (1998). A crack detection method in road surface images using morphology. *IAPR Workshop on Machine Vision Applications*, Makuhari, Japan

Tsai, Y-C., Kaul, V., and Mersereau, R.M., (2010). Critical Assessment of Pavement Distress Segmentation Methods,.*J. Transp. Eng*., 136(1), pp. 11–19.

Varadharajan, S., Jose, S., Sharma, K., Wander, L., and Mertz, C. (2014). Vision for Road Inspection. *Proceedings of WACV 2014: IEEE Winter Conference on Applications of Computer Vision*

Vivekanandreddy Navaneetha, D., Kammar, A. & Sowmyashree.B (2014).Hough transforms to detect and classify road cracks. *International Journal of Engineering Research & Technology*, 3(6), pp. 1500 – 1505

Witten, I. H., Frank, E., and Hall, M. A. (2011) Data mining: practical machine learning tools and techniques, Elsevier

Wu, Q., Merchant, F.A., and Castleman, K. R. (2008). *Microscope Image Processing*, Elsevier

Yu, X. and Salari, E. (2011). Pavement pothole detection and severity measurement using laser imaging. *IEEE International Conference on Electro/Information Technology (EIT)*, Mankato, USA

Zhou, J., Huang, P.S., and Chiang, F.-P. (2006). Wavelet-based pavement distress detection and evaluation, *Opt. Eng. 45* (2).

Zou, Q., Cao, Y., Li, Q., Mao, Q., Wang, S. (2012). CrackTree: automatic crack detection from pavement images, *Pattern Recog. Lett.* 33 (3), pp. 227–238.

List of Figures

Table 1: Median filter execution times in milliseconds

	256x256	512x512	1024x1024	2048x2048
Sequential	14.3	57.936	230.758	889.876
OpenCL Intel CPU	0.108943	0.327316	1.22963	4.77966
OpenCL Intel GPU	0.013582	0.049675	0.193708	0.769058
Nvidia GPU	0.002747	0.010321	0.0399	0.156663

Table 2: Top-hat transform execution times in milliseconds

	256x256	512x512	1024x1024	2048x2048
Sequential	203.007	765.11	2980.79	11611
OpenCL Intel CPU Naïve	1.13034	5.03241	18.0581	76.2757
OpenCL Intel CPU Separable	0.431628	4.80577	15.9215	58.1489
OpenCL Intel GPU Naïve	0.584406	2.31147	8.23475	25.4106
OpenCL Intel GPU Separable	0.0851977	0.314928	1.25112	5.08258
Nvidia GPU Naïve	0.025724	0.0961443	0.370388	1.4927
Nvidia GPU Separable	0.00853265	0.0301136	0.11383	0.43868

Table 3: Total execution times of all implementations

	256x256	512x512	1024x1024	2048x2048
Sequential	217.407	823.436	3213.158	12509.4564
OpenCL Intel CPU	1.29138047	5.51308102	19.8314412	83.2016187
OpenCL Intel CPU Separable	0.57764077	5.22357002	17.2738532	63.4995187
OpenCL Intel GPU	0.6230764	2.44574345	8.64068146	26.7954523

OpenCL Intel GPU Separable	0.1264993	0.45310435	1.66696046	6.49687731
Nvidia GPU	0.03927566	0.14796738	0.58636053	2.4431657
Nvidia GPU Separable	0.02226623	0.08221098	0.33134483	1.3884667
Speed-up	9763.97715	10016.1317	9697.32345	9009.54728

Table 4: Results of the classification of the pavement images

Algorithm	Correctly classified in %	Precision	Recall
C4.5	95	0.949	0.950
Multilayer Perceptron	87	0.880	0.872
Rotation Forest	95	0.950	0.950

Table 5: Confusion matrix for the test images classified with the Rotation Forest algorithm

Image containing distress	**Good pavement image**	**Classification outcome / Actual condition**
306	62	**Image containing distress**
15	1166	**Good pavement image**

Figure1

Click here to download Figure Fig1.tiff

Figure2

Click here to download Figure Fig2.tiff

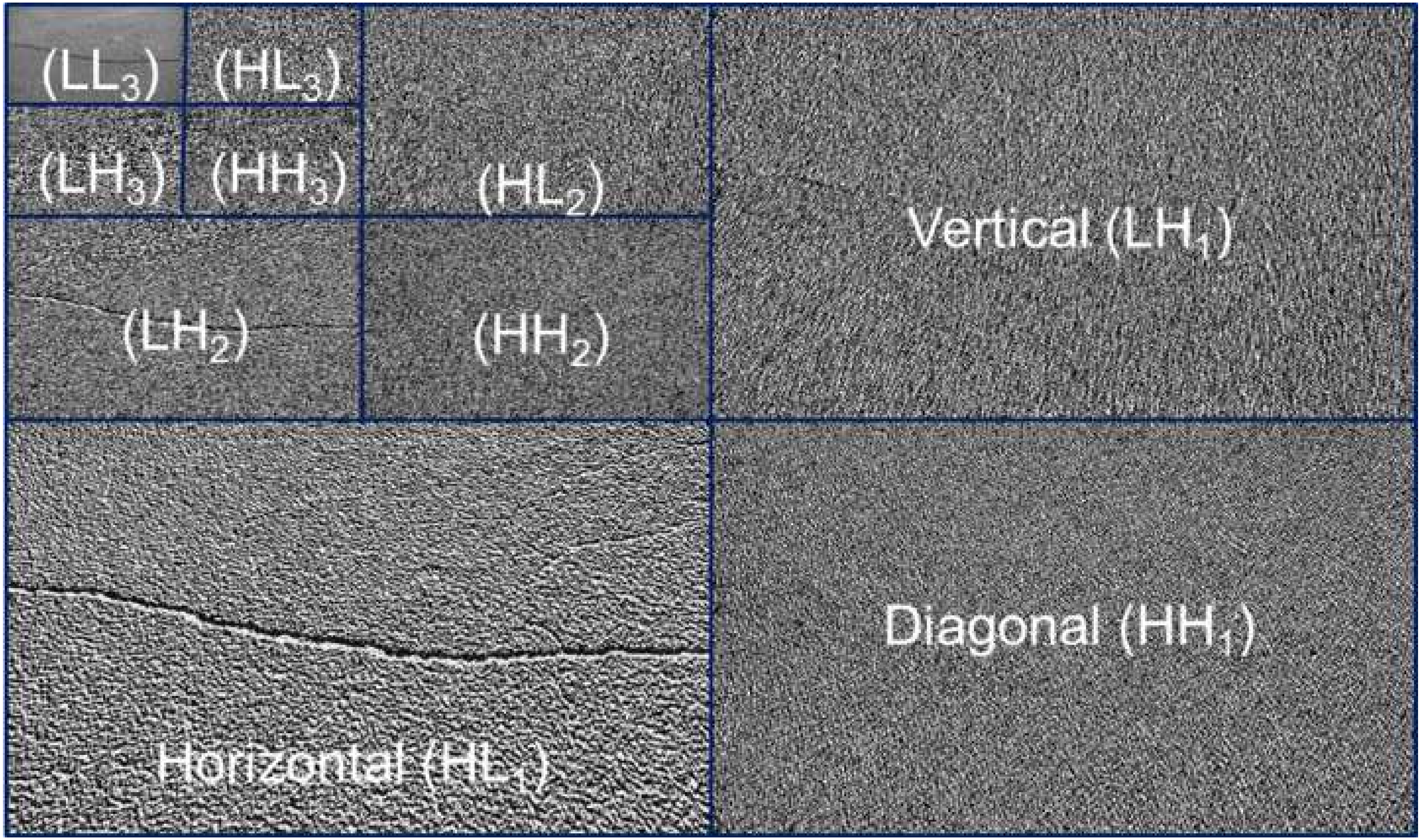


Figure3

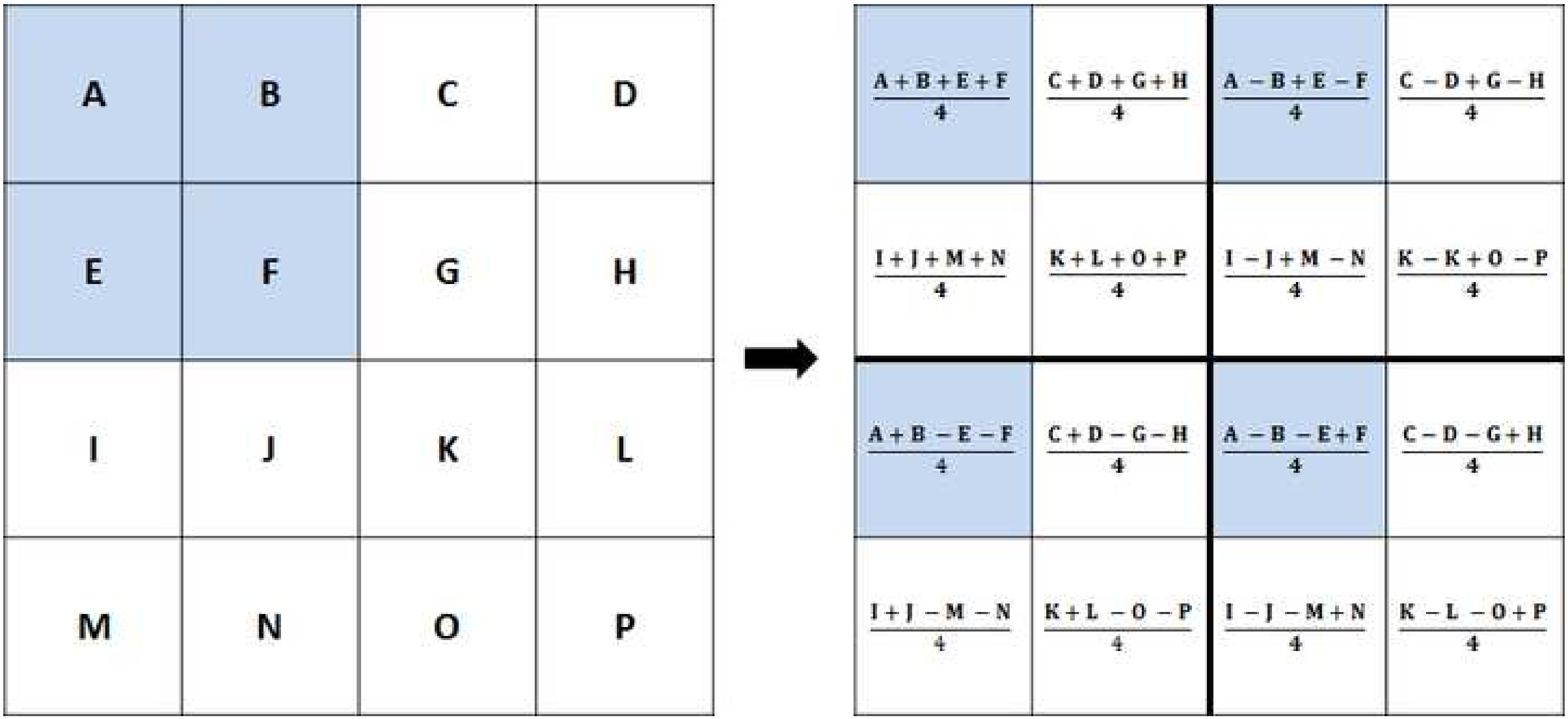

A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
$\frac{A+B+E+F}{4}$
$\frac{C+D+G+H}{4}$
$\frac{A-B+E-F}{4}$
$\frac{C-D+G-H}{4}$
$\frac{I+J+M+N}{4}$
$\frac{K+L+O+P}{4}$
$\frac{I-J+M-N}{4}$
$\frac{K-K+O-P}{4}$
$\frac{A+B-E-F}{4}$
$\frac{C+D-G-H}{4}$
$\frac{A-B-E+F}{4}$
$\frac{C-D-G+H}{4}$
$\frac{I+J-M-N}{4}$
$\frac{K+L-O-P}{4}$
$\frac{I-J-M+N}{4}$
$\frac{K-L-O+P}{4}$

Figure4

Click here to download Figure Fig4.tiff

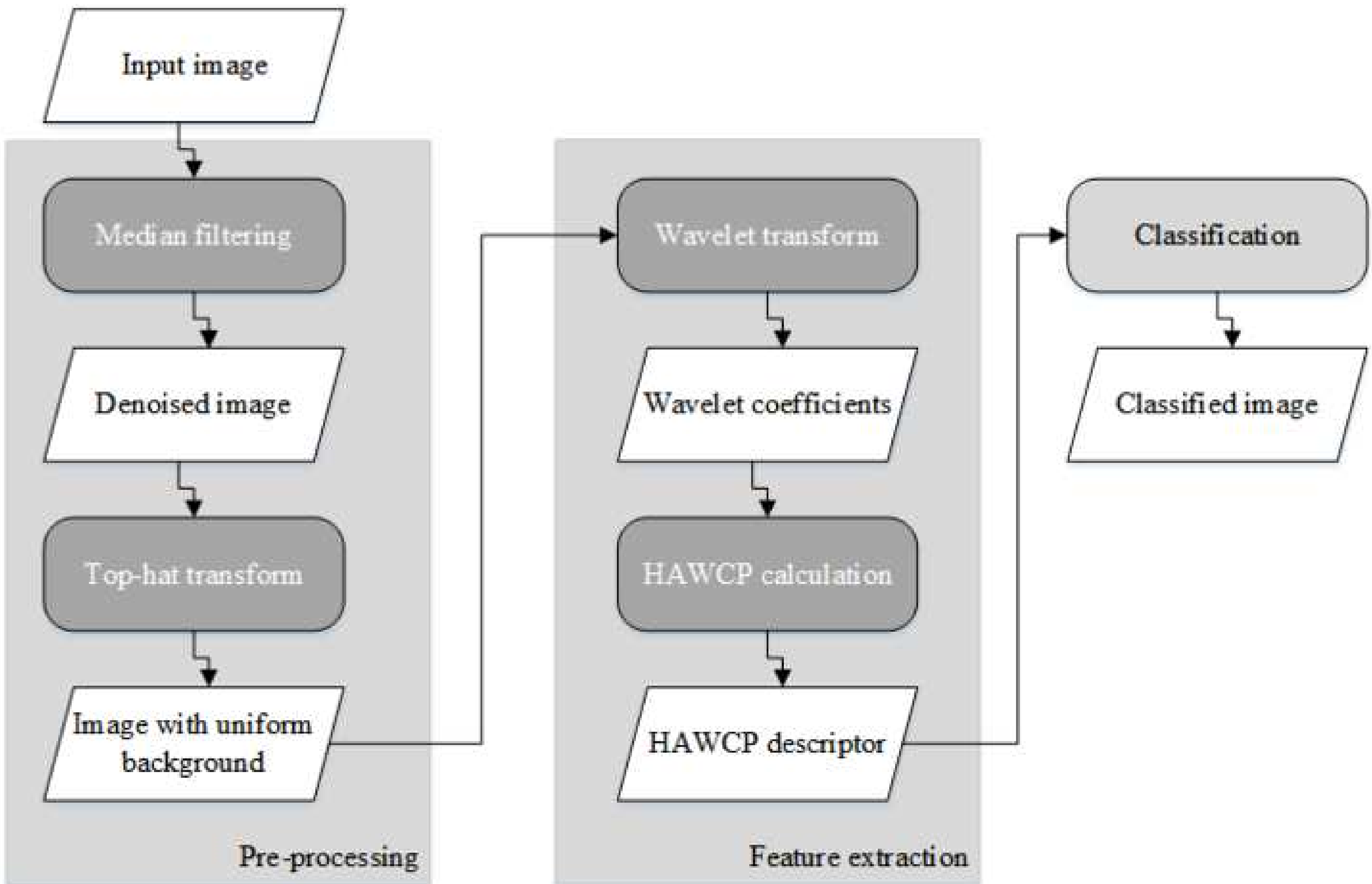


Figure5

Click here to download Figure Fig5.tiff

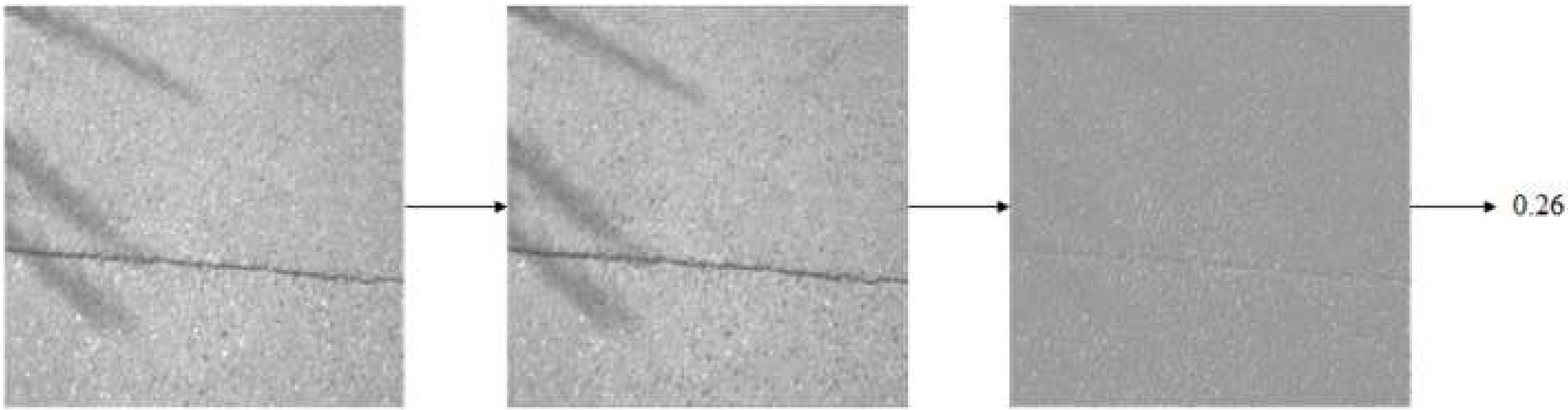


Figure6

Click here to download Figure Fig6.tiff

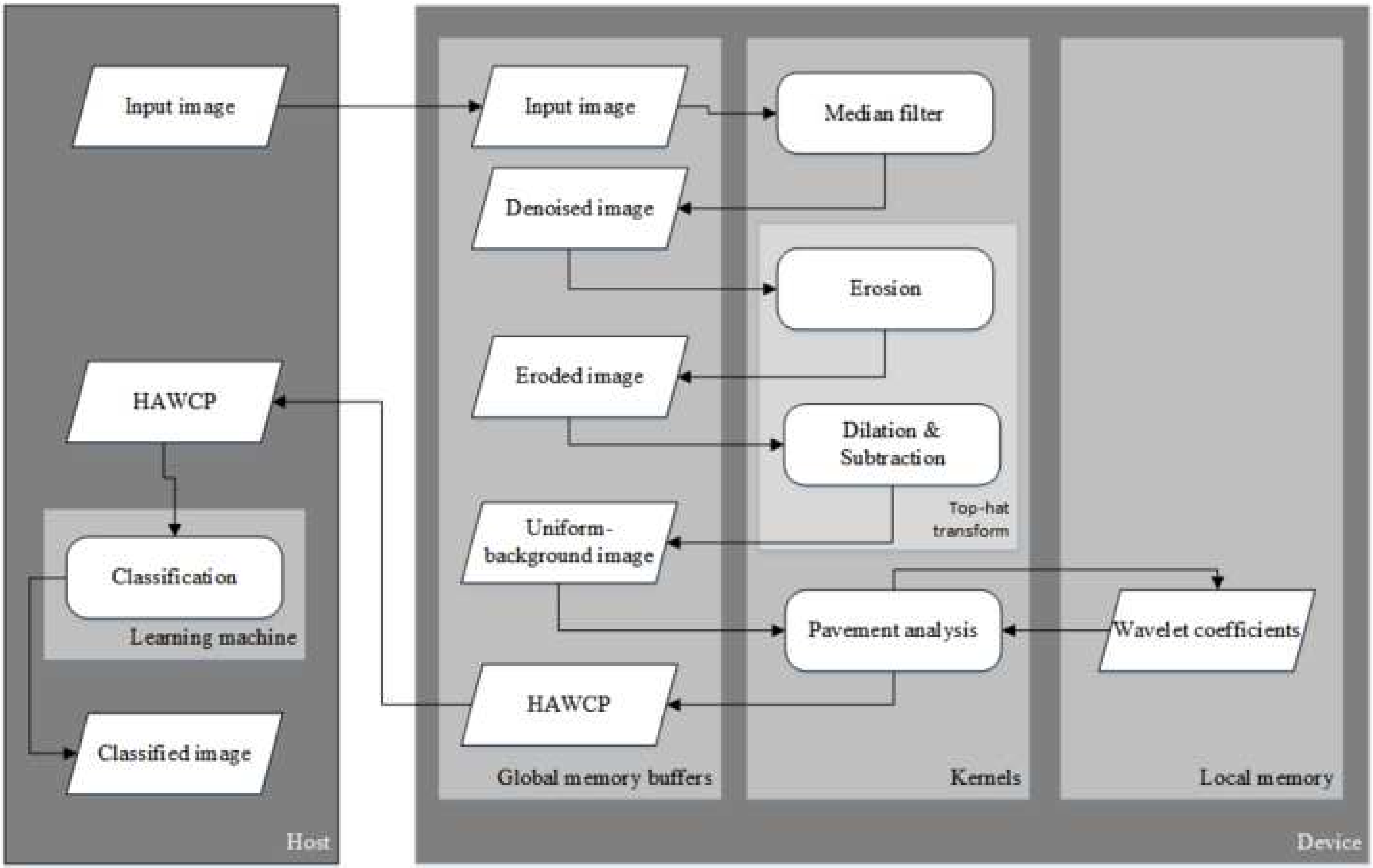
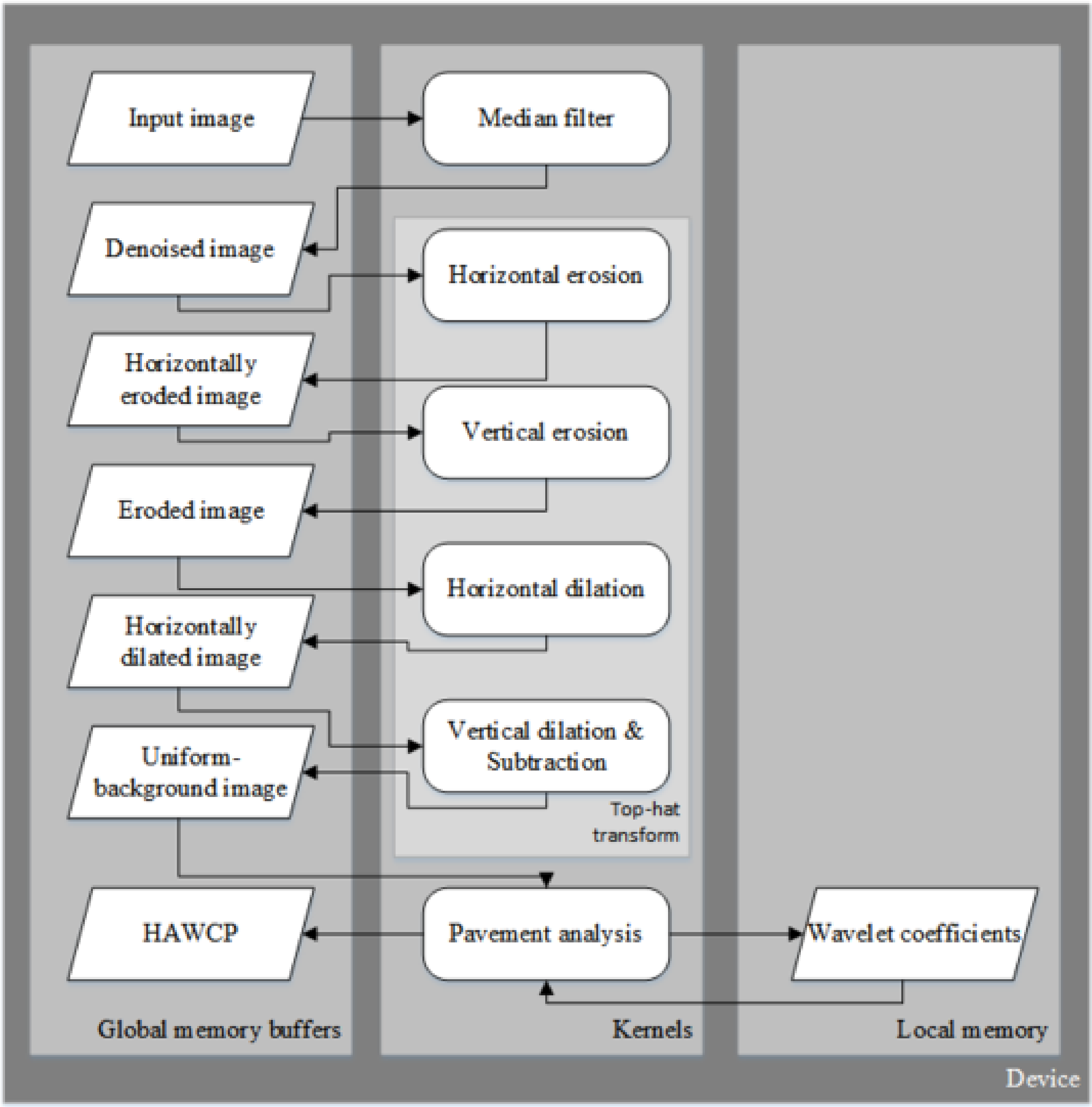
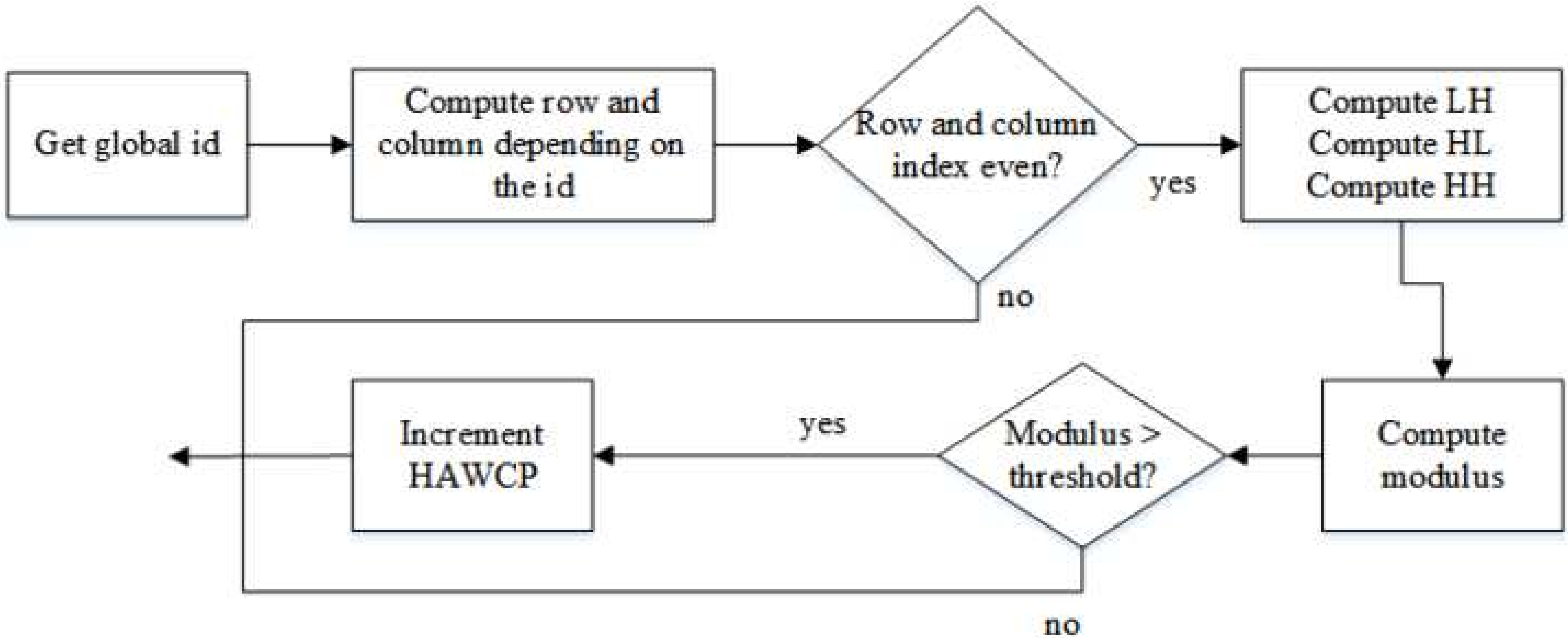


Figure7

Click here to download Figure Fig7.tiff




Get global id
Compute row and column depending on the id
Row and column index even?
yes
no
Compute LH
Compute HL
Compute HH
Compute modulus
Modulus > threshold?
yes
no
Increment HAWCP

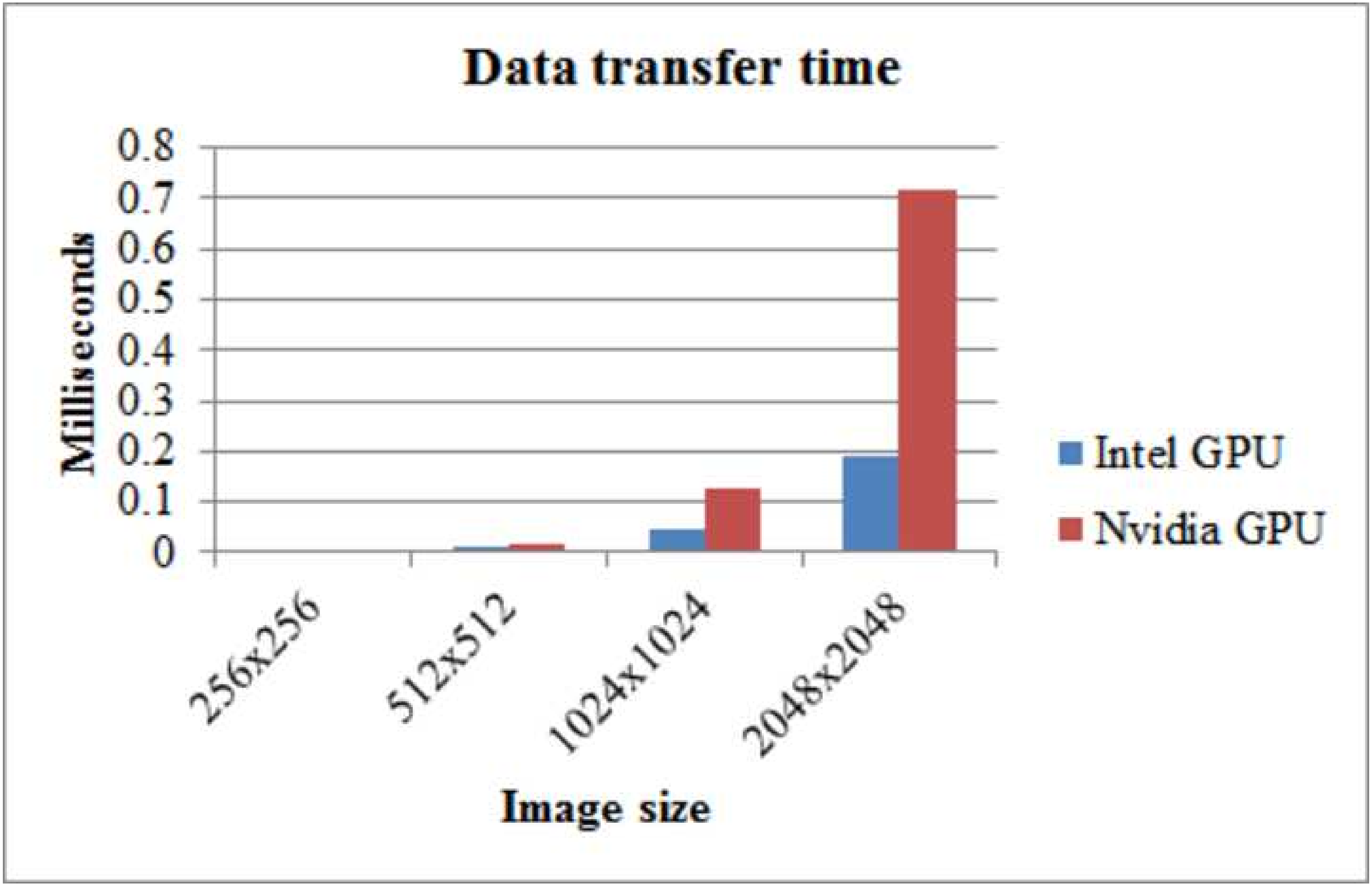

Data transfer time
Milliseconds
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
256x256
512x512
1024x1024
2048x2048
Image size
Intel GPU
Nvidia GPU

Figure10

Click here to download Figure Fig10.tiff

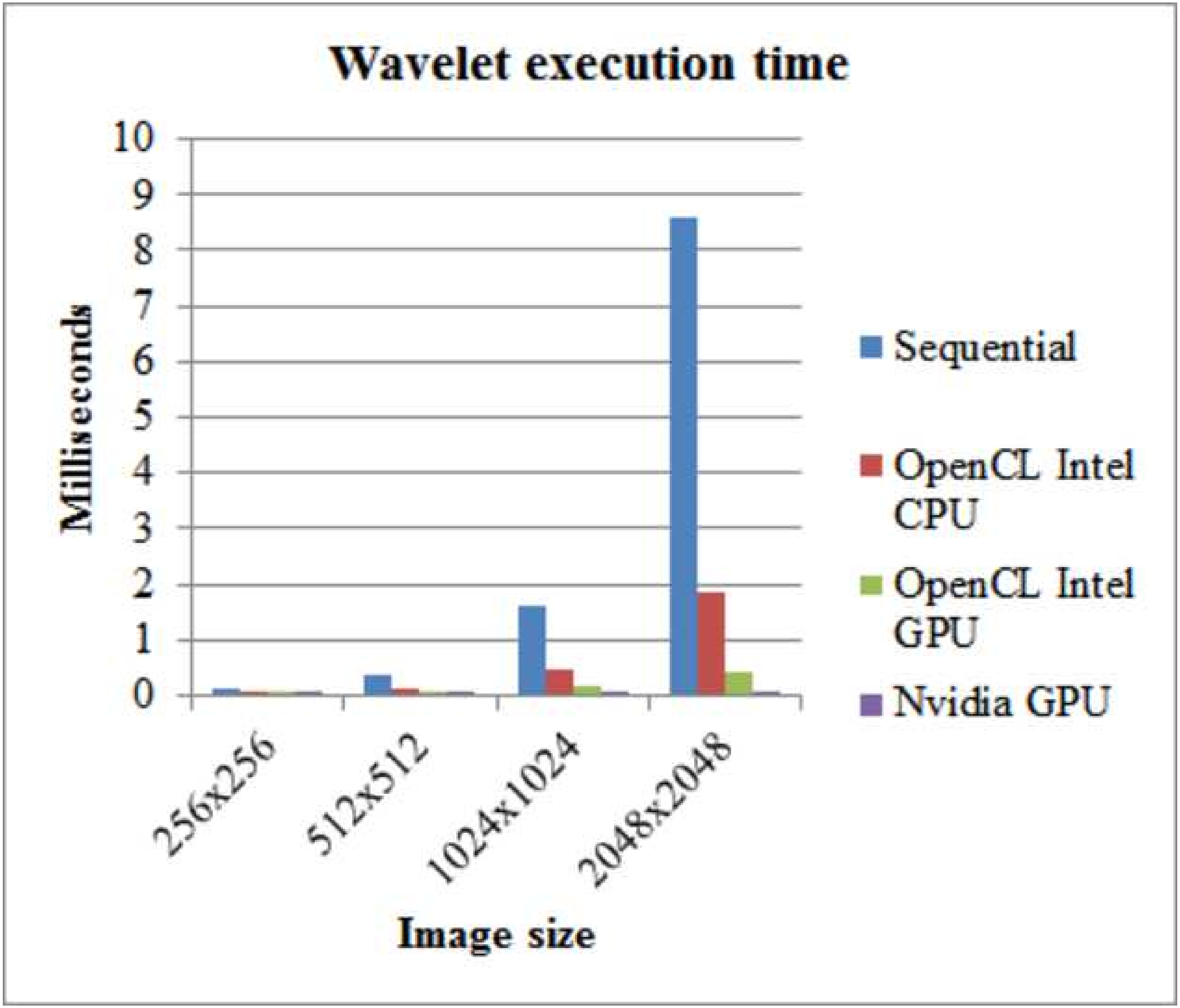


Figure11

Click here to download Figure Fig11.tiff

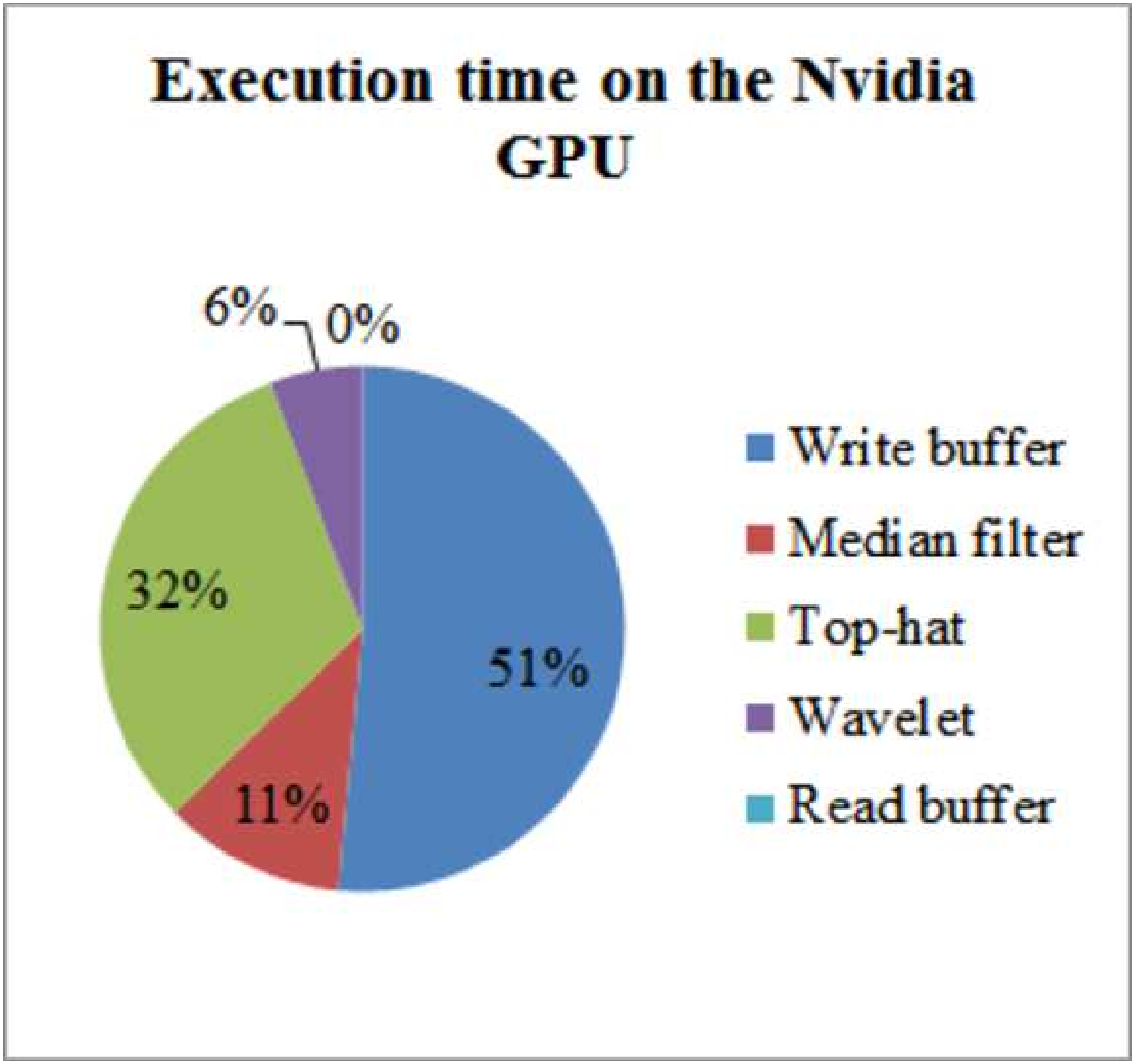


Figure12

Click here to download Figure Fig12.tiff

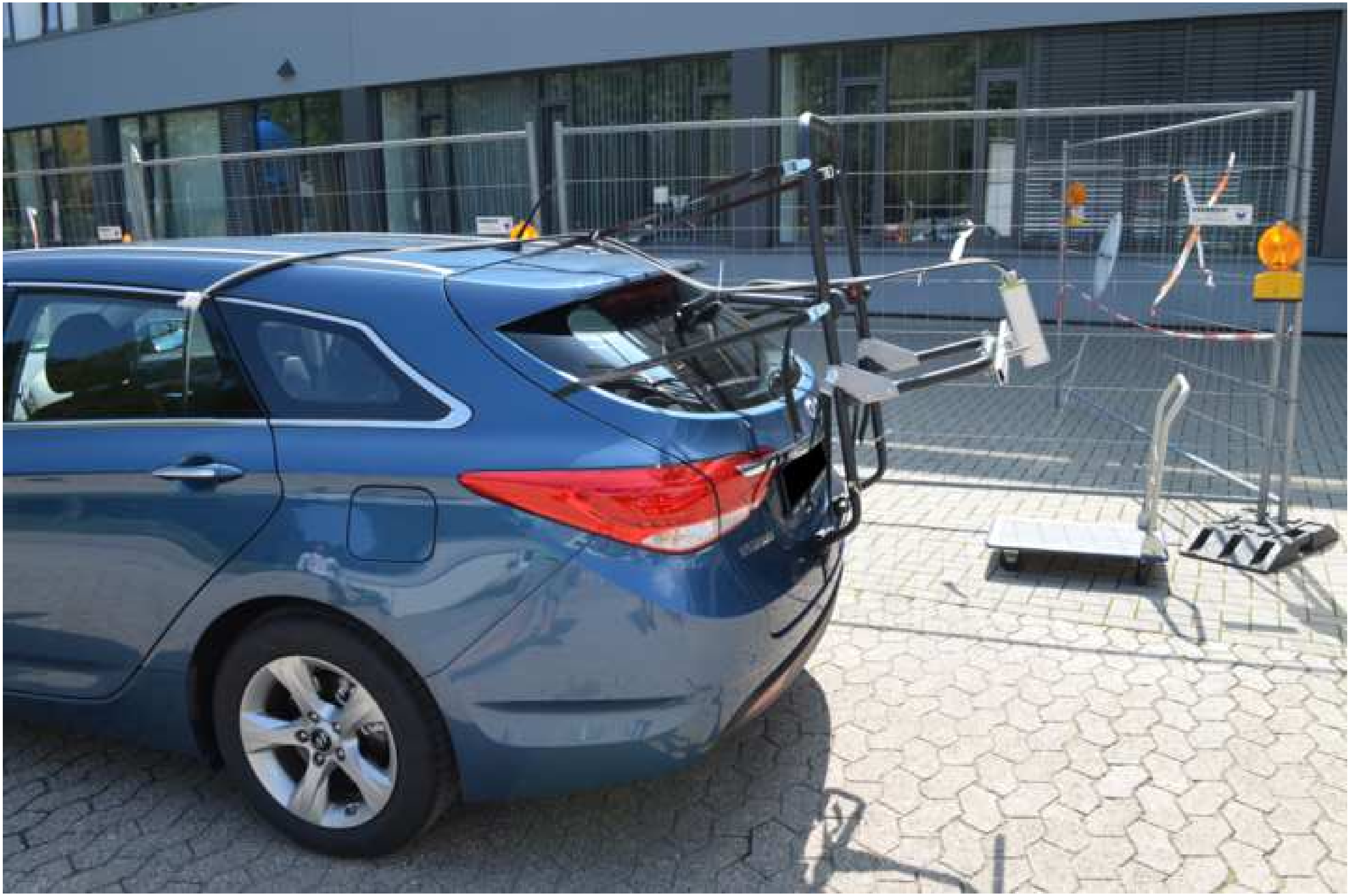


Figure13 Click here to download Figure Fig13.tiff

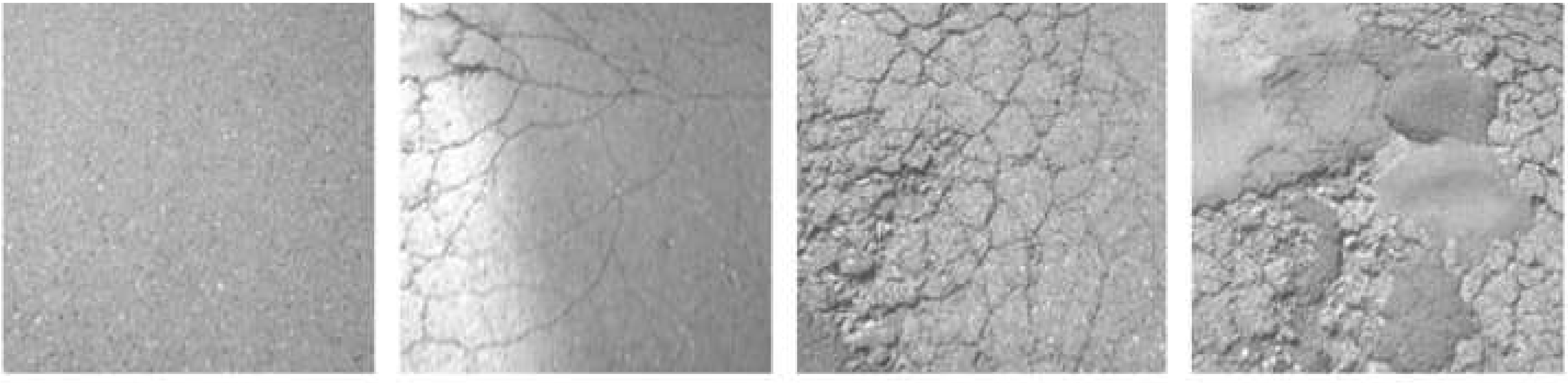
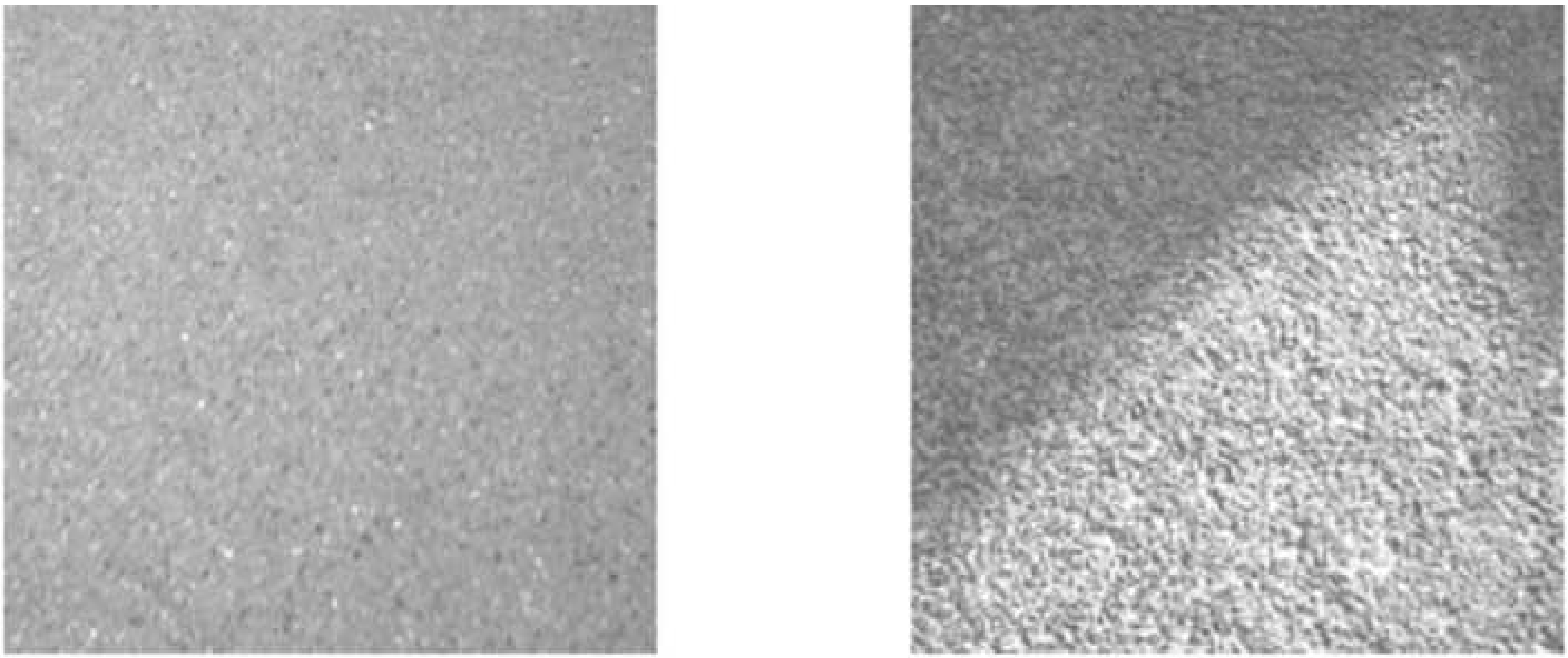


Figure14

Click here to download Figure Fig14.tiff



List of Figures