Ongoing Tests and Improvements of the MPS Algorithm for the Automatic Crack Detection Within Grey Level Pavement Images

V. Baltazart, Ph. Nicolle, L. Yang *
Ifsttar, UBL University, Nantes, France

***Abstract*—The MPS approach (Minimal Path Selection) has shown in [1] to provide robust and accurate segmentation of cracks within pavement images compared to other algorithms. As a counterpart, MPS suffers from a large computing time. In this paper, we present three different ongoing improvements to reduce the computing time and to improve the overall segmentation performance. Most of the work focuses on the first three steps of the algorithm which achieve the segmentation of the crack skeleton. This is at first the improvement of the MPS methodology under Matlab coding, then, the C language MPS version and finally, the first attempt to parallelize MPS under the GPU platform. The results on pavement images illustrate the achieved improvements in terms of better segmentation and faster computational time.**

***Keywords*— road surface monitoring, crack segmentation, performance assessment, optimization, parallelization, GPU.**

I. Introduction

Monitoring road surface conditions is an important issue in many countries. The conventional survey consists in collecting 4 meters wide pavement images at traffic speed by devoted imaging devices [2]. The objective is then to detect surface distresses, like raveling, potholes and cracking in order to plan the effective road maintenance of the pavement structure. Among the surface distresses, cracking can serve as a condition indicator of the pavement structure and may reveal different structural pavement pathologies throughout the different crack patterns. This paper then focuses on the detection of open cracks on the pavement surface.

Today, image processing techniques have been developed to computerize the survey of cracking as the support of human visual control [3]. The pavement images can provide information on the presence of cracks through the pixel intensities and the shape of the darker image features. Many image processing techniques exist for the detection of cracking on grey level images, e.g., [4-14].

The benchmarking of some existing techniques is provided in [1,4-7]. One of the latest technique, namely the Minimal Path Selection (MPS) technique has shown to outperform the other methods at the pixel scale on simulated and field pavement images. As a counterpart, it suffers from a large computing time.

Within this scope, the aim of this paper has been to improve the original MPS coding. Then, after a brief recall of the general synoptic of the MPS approach in Section II, Sections III and IV present the ongoing improvements which have been tested so far. We especially focus on the first 3 steps which allow the segmentation of the crack skeleton. Section III proposes a more efficient strategy for the selection of minimal paths and a faster Matlab coding. Then, in Section IV, a big step forward in reducing the computational time has been obtained by converting the latest MPS algorithm into the C programming language [16]. Finally, a further speed increase has been achieved by parallelizing the MPS code under the Graphics Processing Units (GPU) platform.

(*) actually MS student at Xi'an Jiaotong University, Shaanxi, China.

The performance assessment of the three new versions of the MPS approach is illustrated on an image sample. This illustration allows evaluating the achievable speedup factor of the MPS algorithm thanks to both the C-programming and the parallelization under GPU. The conclusion summarizes the results and draws some working perspectives.

II. General Synoptic of MPS

This section briefly recalls the five steps of the MPS method in [1]. Assuming cracks match to darker pixels within pavement image, MPS is based on the selection of minimum paths and then, emphasizes the connectivity between crack pixels. This method has been originally coded under the Matlab platform.

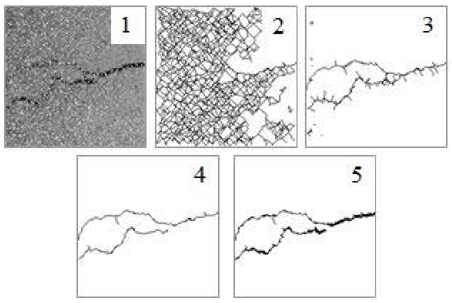


Fig. 1. The five steps of the MPS method; (top) steps 1-3 to obtain the segmentation of the crack skeleton (1: endpoints selection within P×P images subsets ; 2: shortest path computation between selected endpoints within 3P×3P neighbourhood; 3: selection of minimum path costs); (bottom) the two post-processing steps to achieve the segmentation of the final crack pattern (4: refinement of the crack skeleton; 5: thickness estimation of the cracking).

MPS consists in the two following phases as shown on Fig. 1. The first phase (steps 1-3 in Fig. 1) which is improved in Section III, allows the segmentation of the one-pixel wide crack skeleton in the image. The second phase (steps 4-5 in

Fig. 1) consists in refining the crack skeleton and estimating the crack width along the crack skeleton (see [1,27] for details).

The overall good performance of MPS mainly relies on the segmentation of the crack skeleton. It has been shown in [1] that the crack skeleton reach a DICE similarity coefficient of about 50% and the post-processing stage further improves the segmentation performance by 30% to 40%. The latter rates may greatly vary from one image to one another depending on the complexity of the crack pattern.

Nonetheless, the proposed improvements focus on the first three steps of MPS to obtain the crack skeleton. The improved methodology of MPS is detailed in Section III, and then, the Matlab coding of MPS is converted in Sections IV to the C programming language and to GPU platform.

III. IMPROVED MPS METHODOLOGY (MPS-V1)

A. *Selection of crack endpoints at step 1*

Endpoints serve as seed points for initializing the search for shortest paths at step 2. According to [1], the choice of endpoints obeys two conditions: endpoints are the darkest pixels within P×P image subsets (with P = 8) and the grey level of each is lower than the following threshold:

$$T_e = \mu_e - K_e \times \sigma_e \tag{1}$$

where μ_e and σ_e are the mean and the standard deviation values of the grey level pavement image.

The proposed improvement at step 1 acts as a post-processing to further reduce the amount of endpoints. It is assumed that the endpoints within the crack pattern show a larger local anisotropic index. Among the three tested algorithms in [15], namely, FFA [14], CTA [14], the Hough transform and LBP [17], the FFA index has shown to provide better results compared to others. As proved later in Section IV, FFA achieves a large reduction of the amount of endpoints within the image texture.

B. *Path finding strategy at step 2*

This step aims at computing the shortest paths between the crack endpoints selected at step 1. It requires defining the strategy to apply the shortest path algorithm between P×P image subsets (with P = 8) and the strategy to browse the whole image.

The shortest paths are originally computed within the 3P×3P image subsets. Then, 8 paths computation at the most has to be performed from the central endpoint. This results in the star-like strategy as shown on Fig. 2 (left).To browse the whole image, the path search is then moved to the next 3P×3P image subset. To insure the full connectivity between the computed paths, adjacent image subsets are overlapping to each other by either one row or one column.

The star-like strategy implies some redundant computed paths that slow down the running of the algorithm. The new strategy in [15] consists in using 3P×2P image blocks. Therefore, there are only 4 paths to compute at the most for each endpoint without repetition, as shown on Fig. 2(right), while maintaining the connection between all the pieces of crack.

Within each image subset, the single source shortest path (SSSP) algorithms have been shown in [15] to be the fastest strategy to compute the shortest path between neighboring endpoints. Dijkstra's algorithm is thus performed to this aim [18].

A large time saving was achieved thanks to the improvement on the Matlab coding of the Dijkstra's algorithm. Using the Matlab profiler, two Matlab sub-functions in the original coding, namely '*ismember*' and '*sub2index*', were found too much time consuming. They have been replaced by specific coding with lower complexity. Finally, a simpler way has been used to update the set of pixels for the shortest path computation. As a result, the new Matlab coding of step 2 was found about 10 times faster than the original coding in [1].

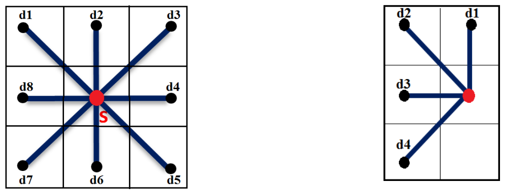


Fig. 2. Strategy to browse the pavement image in the MPS approach: (left) the original strategy in [1] ; (right) the improved strategy in [15].

C. *Selection of minimum cost paths at step 3*

This step allows selecting among the computed shortest paths at step 2 those which have a higher probability of belonging to the crack pattern. To this aim, the following path cost is defined as the average of the pixel values along the computed shortest paths [1]:

$$C_{P_{ij}} = \frac{1}{\text{card}(P_{ij})} \sum_{m=1}^{j} I(x_m) \tag{2}$$

where P_{ij} is the path between the endpoints (i, j) and x_m are the pixels of the path.

As opposed to the grey level distribution, the path costs distribution in Fig. 3 (left) exhibits roughly a bimodal distribution. According to photometric consideration, the paths within the crack pattern correspond to the path costs below the following threshold [1]:

$$T_c = \mu_c - K_c \times \sigma_c \tag{3}$$

where μ_c and σ_c are the mean and the standard deviation of the path costs computed at step 2, and K_c is a constant which is adjusted to optimize the segmentation performance.

As shown later in Section V, the proposed modifications result in a large reduction of the amount of computed shortest paths within the image background. This strongly modifies the path costs distribution in Fig. 3 (right), as well as the corresponding mean and standard deviation. Then, a new sensitivity analysis is required to match the threshold T_c to the new path costs distribution.

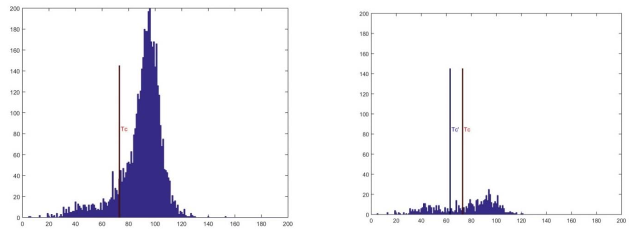


Fig. 3. Example of path costs distribution in (2) for the original MPS version (left) and for the proposed MPS-V1 version (right).

To this aim, the parameter K_c in (3) is varying in order to scan a large interval in grey level that insures a global optimum to be determined. For each K_c value, the segmented crack is compared to the pseudo-ground truth (PGT) and the amount of false positive (FP), false negative (FN) and true positive (TP) pixels are used to compute the DICE similarity coefficient, (DSC or F1-score, equivalently) [19]. In [15], the improved MPS version is shown to provide a more sensitive parameter setting, because the DSC goes through a maximum. In practice, the optimum thresholds for both MPS versions (shown by vertical lines on Fig. 3 (right)) are found very close to each other.

D. *Post-Processing Steps*

Steps 4 and 5 in Fig. 1 have benefited of the proposed improvements on steps 1-3 at some extent. As shown in section IV, the segmented result at step 3 includes a fewer artefacts, compared to the original version. As a result, the refining step of the crack skeleton at step 4 is simpler to perform. The coding improvements in subsection III-B may also speed up the running time at that step. The pixel aggregation at step 5 is also expected to be more reliable to perform the thickness estimation.

IV. Parallelizing MPS algorithm

In order to achieve a larger speedup factor on the computational time, further work focused on parallelizing the MPS algorithm. The first step has been to convert the Matlab code into a compiled programming language. Within the scope of the application, it is worth to recall at first the real time processing limit to achieve.

A. *Real-time processing limit*

The real-time processing limit is defined in relation to both the speed of the data collection over the pavement surface and the size of the pavement images to process. For convenience, the latter limit is converted into millisecond per Mega pixel (MPx).

Assuming the imaging system collects 4 meters wide by 1 meter long pavement images at traffic speed, i.e., 90 km/h, the processing speed to achieve corresponds to 25 images / second, namely 40 ms processing time per image. As the size of the pavement images may vary from 1 to 4 MPx (from the oldest to the latest imaging device), the real-time processing limit then ranges from 40 to 10 ms / MPx.

Within this context, it is worth noticing the data processing speed of the few existing imaging devices which provide mm^2 cell resolution at traffic speed. According to [20], the latter speed was 22 km/h for the worldwide laser-based 3D LCMS system in 2012. It was ranging from 10 to 40 km/h for the stereo photometric Roadscout system. In contrast, the Roadcrack system is the only operating system which has been affording real-time crack measurements on 2D pavement images since the end of 1990s [20].

B. *C coding (MPS-CPU version)*

C language is known as the most efficient programming language in terms of computational speed. Then, the latest Matlab MPS version, namely MPS-V1 in section III, has been converted into C. As for section III, the work has focused on the steps 1-3 of the algorithm which provide the segmentation of the crack skeleton.

At steps 1-2, CPU sequentially performs the same processing in each image block. This is P×P image blocks at step 1, and then 2P×3P image blocks at step 2. At step 2, the key point is to code the Dijkstra's algorithm with an efficient priority queue. When Dijkstra wrote his paper, priority queues did not yet exist. According to the literature, e.g., [21], Fibonacci heap is known as one of the most efficient priority queue. In practice, the latter was found difficult to implement and a conventional heap sorting was used instead.

C. *Parallelizing MPS using GPU (MPS-GPU version)*

In this section, the first attempt to parallelize the MPS algorithm is briefly presented. It is worth to recall that MPS performs the same processing on each image blocks at steps 1-2. Within this scope, steps 1-2 may be parallelized instead of sequentially performed as for the MPS-V1 and the MPS-CPU versions.

Compare to multicore programming, the use of GPU (Graphics Processing Unit) programming technique has been more appealing because of the larger amount of specialized CPU on the graphic card [22]. Nowadays, the latest GPU technology enables restoring interest in the use of computationally expensive algorithms such as MPS. It implies, in return, to adapt the principle of the algorithm at some extent. To address larger hardware capabilities, the GPU version of MPS has been implemented using the Open CL API [23].

For MPS, one of the keypoint is once again the parallelization of the shortest path computation at step 2. Different solutions have been proposed in the literature to parallelize Dijkstra's algorithm with OpenCL or CUDA, e.g., see the basic work in [24] and the extensive literature review in [25]. Two methods have been tested in [16]. The first one is the straightforward application of the C program by managing a priority queue for Dijkstra. The second one uses an alternative scanning of the 2P×3P image blocks in both the horizontal and the vertical directions. The scanning starts on either the row or the column of the crack seed point, and a GPU is associated with selected pixel along the scanned row or column. Both solutions have shown to provide similar segmentation result, but the second one was found faster and is selected in the next section for test.

The hardware characteristics and the computer configuration may influence the processing speed on GPU. This is first of all the amount of chips on the GPU card. To achieve a larger speed up factor, it must be adapted to the image size and the size of the image blocks to be processed. The specialized GPU card which has been used in section V for test has shown to outperform the performance of the standard GPU card. Among other things, it is better to limit as much as possible the background applications on the computer and the data transfer between CPU and GPU cards. The latest generation of the Peripheral Component Interconnect (PCI) data bus affords faster data transfer. The latest GPU cards include larger RAM capacity and then also may limit the data transfer.

V. Illustrative Example

The four versions of MPS have been tested on a 960×456 pavement image, i.e. 0.43 MPx, which has been also used for test in [1, Fig. 13]. In this section, only a 256×256 subset of the initial image is shown for illustration. The pseudo-ground truth (PGT) is displayed as the reference to visually assess the quality of the segmentation.

The column wise results in Fig. 4 allow a gradual valuation of the four MPS versions at the two key steps of the algorithm. This is first, the shortest paths computation at step 2 (section III-B), and second, the raw crack skeleton segmentation at step 3 (section III-C). Step 1 is not displayed owing to the lack of visibility. Fig. 4 shows the constant improvement which has been obtained so far to reduce the computational time and to improve the segmentation performance.

We can observe that the modified step 2 has significantly reduced the amount of the computed shortest paths within the image background, thus reducing the computational time and improving the visual rendering of the segmentation at the same time. At step 3, the artefacts along the crack skeleton and the small scattered pieces of cracks which are readable for the original MPS version partly disappear for MPS-V1. Then, MPS-V1 displays a crack skeleton which is closer to the PGT. The performances assessment which is conducted over the full size pavement images in [27] has shown that MPS –V1 reach a larger DSC rate than the original MPS version in [1].

C and GPU versions, namely, MPS-CPU and MPS-GPU respectively, show similar segmentation results at steps 2 and 3. The segmentation result at step 2 includes less false positive pixels compared to the result by MPS-V1. In contrast, the result at step 3 for MPS-CPU and MPS-GPU visually seems to be intermediate between the original MPS and MPS-V1. This discrepancy is believed to be due to the imperfect manual thresholding at step 3.

In Fig. 3, the total computational time to segment the crack skeleton was converted into millisecond per megapixels (MPx) by a simple proportionality rule. The latter time is approximate because it may greatly vary from one image to one another. Compared to the original MPS version [1], the improved Matlab coding, namely MPS-V1, has lowered the computational time by a factor 23. MPS-CPU allows faster processing, namely, 70 times faster than MPS-V1. MPS-GPU reduces the execution time by an additional factor, ranging from 6 to 70 according to [16]. The use of the specific GPU card with a large amount of processing chips (as shown in the last row in Fig. 2) reaches the real time processing limit of 10 ms / MPx, as defined in section IV-A. Further speed up may be achieved by using a more recent personal computer.

VI. Conclusion and Perspectives

This paper has presented the ongoing improvement of the MPS algorithm which was found to provide robust and accurate crack segmentation within grey level pavement images. The work has focused on the first three steps of MPS, to afford a faster and more reliable crack skeleton.

In section II, the improved algorithm has greatly reduced the amount of computed paths within the image texture along the computational time. A larger speedup factor has been achieved by converting the latest Matlab version of MPS into the C programming language. The use of GPU has introduced a first attempt to parallelize the MPS approach. Thanks to this latter version, the real time processing of MPS is within reach.

Further illustration will be provided by the time of the conference. Since the implementation under GPU required a modification of the initial coding, the parameters setting of MPS-GPU must be reconsidered, as well as the statistical validation on the image data base in [26]. Faster computation may be achieved by combining more recent computer with the latest GPU technology. Ongoing work in [27] is still hoping for improved MPS algorithm.

Acknowledgments

We wish to thank Dr Rabih Amhaz for valuable discussion and the reviewers for comments and suggestions. This work may contribute to the RILEM Technical Committee 241-MCD.

References

[1] Amhaz R., S. Chambon, J. Idier, V. Baltazart, Automatic crack detection on 2D pavement images : An algorithm based on minimal path selection, IEEE Trans. Intel. Transport Systems, Vol 17(10), pp 2718-2729, 2016

[2] PIARC, Evaluating the performance of automated pavement cracking measurement equipment, Technical Committee CT 4.2 on the Interactions Vehicle/Road, 2012

[3] Vavrik W., L. Evans, J. Stefanski and S. Sargand, PCR Evaluation – Considering Transition from Manual to Semi-Automated Pavement Distress Collection and Analysis, Ohio Department of Transportation, USA, 2013

[4] Special session 19 in IEEE Int Conf. on Image Processing (ICIP) on "Image Processing for the Detection of Road-Surface Degradations", chaired by Correia P.L. and Oliveira H, Paris, France, 2014.

[5] Chambon S. and J-M. Moliard, Automatic Road Pavement Assessment with Image Processing: Review and Comparaison, International Journal of Geophysics, pp. 1-20, 2011

[6] Shi Y., L. Cui, Z. Qi, F. Meng and Z. Chen, Automatic Road Crack Detection Using Random Structured Forests, IEEE Trans. on Intelligent Transportation Systems, 17, pp. 3434-3445, 2016

[7] Baltazart V., J-M. Moliard,R. Amhaz, L-M. Cottineau, A. Wright, M. Jethwa, Automatic crack detection on pavement images for monitoring road surface conditions – Some results from the collaborative FP7 TRIMM project, Rilem Conference, Nantes, France, 2016

[8] Oliveira H. and P. L. Correia, "Automatic road crack detection and characterization," IEEE Trans. Intell. Transp. Syst., vol. 14, no. 1, pp. 155–168, 2013

[9] Zou Q., Y. Cao, Q. Li, Q. Mao and S. Wang. CrackTree : Automatic Crack Detection from Pavement Images. Pattern Recognition Letters, 33(3), 2012

[10] Tang J. and Y. Gu, "Automatic crack detection and segmentation using a hybrid algorithm for road distress analysis," in Proc. IEEE Int. Conf. Syst., Man, Cybern., pp. 3026–3030, 2013

[11] Gavilán M., D. Balcones, O. Marcos, D. F. Llorca, M. A. Sotelo , I. Parra, M. Ocaña, P. Aliseda, P. Yarza and A. Amírola, Adaptive road crack detection system by pavement classification, Sensors, vol. 11, pp. 9628–9657, 2011

[12] Medina R., J. Gomez-Garcia-Bermejo and E. Zalama, Automated Visual Inspection of Road Surface Cracks., in Int. Symp. on Automation and Robotics in Construction, 2010

[13] Tsai Y., V. Kaul and R. Mersereau, Critical Assessment of Pavement Distress Segmentation Methods, Journal of Transportation Engineering, vol. 136, pp. 11-19, 2010

[14] Nguyen TS, Begot S, Duculty F, Avila M (2011) Free-form anisotropy: A new method for crack detection on pavement surface images, in Proc. IEEE Int. Conf. Image Process., pp. 1069–1072, 2011

[15] L. Yang, Contributions to improve and speed the existing MPS-based algorithm for both the semi-automated and the automated crack segmentation on 2D pavement images, internship report, LUNAM University, France, 2015

[16] Ph. Nicolle, V. Baltazart, Vers l'exécution temps réel de la méthode MPS pour la détection automatique de fissures dans les images 2D de chaussées, RFIA workshop, Clermont-Fd, France, 2016

[17] Hu Y. and Zhao C., A local binary pattern based methods for pavement crack detection, Journal of Pattern Recognition Research, 2010.

[18] Dijkstra E.W., A note on Two Problems in Connection with Graphs, Numerische Mathematik, pp. 269-271, 1959

[19] Sampat M.P., Z. Wang, S. Gupta, A.C. Bovik and M.K. Markey. Complex Wavelet Structural Similarity : A New Image Similarity Index. IEEE Transaction on Image Processing, 18(11):2385–2401, 2009

[20] Wix R., R. Leschinski, Cracking : A tale of four systems, 25th ARRB Conference – Shaping the future: Linking policy, research and outcomes, Perth, Australia, 2012

[21] Chen M., R. A. Chowdhury, V. Ramachandran, D. L. Roche, L. Tong, Priority Queues and Dijkstra's Algorithm, UTCS Technical Report TR-07-54, October 12, 2007

[22] General-purpose processing on graphics processing units, Wikipedia

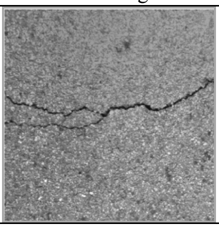
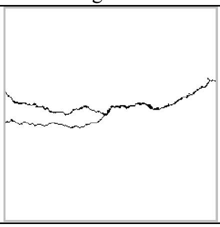
[23] OpenCL reference pages: Kronos, OpenCL 1.2 reference page, 2012

[24] Crauser A., K. Mehlhorn, U. Meyer, and P. Sanders, A Parallelization of Dijkstra's Shortest Path Algorithm, MFCS'98, pp. 722-73, Berlin Heidelberg, 1998

[25] Sommer C., Approximate Shortest Path and Distance Queries in Networks, PhD thesis, Univ., Department of Computer Science, The University of Tokyo, 2010

[26] Chambon S., Image data base. [Online.] Available: http://www.irit.fr/~Sylvie.Chambon/Crack_Detection_Database.html

[27] W. Kaddah, M. Elbouz,Y. Ouerhani, V. Baltazart, A. Alfalou, Optimized MPS for the automatic and unsupervised crack segmentation within two-dimensional pavement images, unpublished, 2017.

	Raw image	Pseudo-ground truth
256×256 subset of the 960×456 pavement image in [1, Fig. 13]		

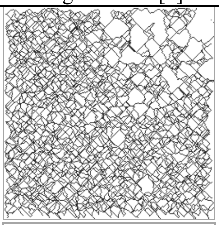
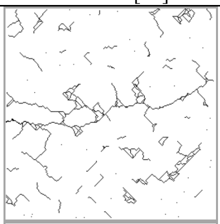
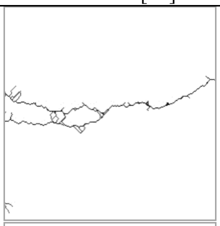
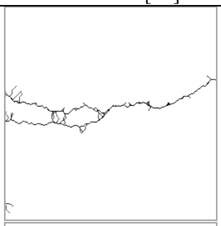
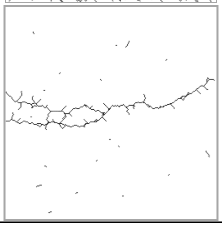
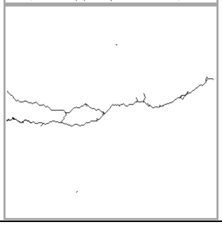
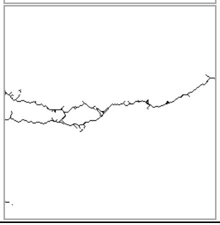
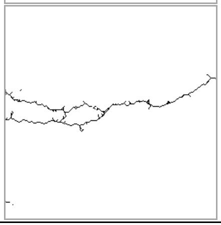
Skeleton segmentation	Original MPS [1]	MPS-V1 [15]	MPS-CPU [16]	MPS-GPU [16]
Step 2: Shortest path computation between selected endpoints at step 1				
Step 3: raw skeleton after selecting the minimum path costs.				
Computational time	1380 sec./MPx	46 sec./MPx	630×10^{-3} sec /MPx	9×10^{-3} sec/MPx
Computing language	Matlab R14	Matlab R14	C	C + API Open CL
Computer	HP Zbook14 Intel Core i7 at 2.7 GHz	HP Zbook14 Intel Core i7 at 2.7 GHz	Dell T3500 Intel Xeon dual-core at 2 GHz	Dell T3500 Intel Xeon dual-core at 2 GHz;
Operating System	Windows	Windows	Ubuntu	Ubuntu
GPU card				2816 cores Nvidia GTX 980-Ti

Fig. 4. Segmentation of the crack skeleton by the four MPS versions discussed in the paper.