Comparison of Crack Detection Methods for Analyzing Damage Processes in Concrete with Computed Tomography

Karsten EHRIG[*], Jürgen GOEBBELS[*], Dietmar MEINEL[*],
Olaf PAETSCH[**], Steffen PROHASKA[**], Valentin ZOBEL[**]
[*] BAM Federal Institute for Materials Research and Testing
(Unter den Eichen 87, 12205 Berlin, Germany,
karsten.ehrig@bam.de, juergen.goebbels@bam.de, dietmar.meinel@bam.de)
[**] Zuse Institute Berlin (ZIB) (Takustr. 7, 14195 Berlin, Germany,
paetsch@zib.de, prohaska@zib.de, zobel@zib.de)

Abstract. Analyzing damages at concrete structures due to physical, chemical, and mechanical exposures need the application of innovative non-destructive testing methods that are able to trace spatial changes of microstructures. Here, the utility of three different crack detection methods for the analysis of computed tomograms of various cementitious building materials is evaluated. Due to the lack of reference samples and standardized image quality evaluation procedures, the results are compared with manually segmented reference data sets. A specific question is how automatic crack detection can be used for the quantitative characterization of damage processes, such as crack length and volume. The crack detection methods have been integrated into a scientific visualization system that allows displaying the tomography images as well as presenting the results.

1. Introduction

3D micro computed tomography is a novel nondestructive method to analyze spatial changes of microstructures in different cementitious building materials. CT measurements have been performed at computed tomography laboratories at BAM with x-ray energy of 210 kV, 1 mm Cu prefilter and a sample diameter of 70 mm, resulting in a voxel size of 40 µm. After data preprocessing (median filter on projection data, beam hardening correction) standard Feldkamp reconstruction has been applied resulting in a 3D voxel data set as starting point for the crack detection process in ZIBAmira [1].

Three different crack detection methods (template matching, a sheet filter based on Hessian eigenvalues, and percolation) have been implemented and modified according to their application on 3D CT data sets. The percolation algorithm works originally only for 2D images and has been extended to 3D images. To evaluate the quality of the crack detection methods qualitatively and quantitatively, the results have been compared with manual segmented reference data sets.

Licence: http://creativecommons.org/licenses/by-nd/3.0

2. Related Work

Several techniques for crack detection have been developed recently. The most obvious solution is the use of standard image processing methods or combinations of it. In [2], for example, thresholding is applied after a beam hardening correction to detect cracks in coal samples. In a second step, noise and artifacts are removed from the result. The suitability for crack detection of the fast Haar transform, the fast Fourier transform, the Sobel filter, and the Canny filter are compared in [3]. For the detection of cracks in concrete bridges Lee et al. [4] first subtract the smoothed image to obtain uniform brightness, followed by removing isolated points to remove noise and morphological operations.

To the best of our knowledge, there is no work on the use of template matching for crack detection. But the problem of finding cell membranes is similar. In both cases, surface-like structures have to be found. In [5], for example, template matching is applied to this problem.

For the extraction of surface-like structures in the context of volume rendering, a so-called sheet filter is proposed in [6]. This filter is computed from the eigenvalues of the Hessian matrix. In [7] this filter is also used for crack detection in 2D images, followed by probalistic relaxation and locally adaptive thresholding.

An approach employing the percolation model, which is based on the phenomenon of liquid permeation, is proposed in [8]. However, this algorithm works only for 2D images and has no obvious generalization to the 3D case.

3. Methods

In this section, we briefly introduce the three methods for crack detection that we compare in this work. The first method is template matching, where the image is searched for a pattern modeling a crack. For details on template matching see [9]. The second method is the sheet filter proposed in [6] which detects sheet structures in volume data. It is computed from the eigenvalues of the Hessian matrix. The third method is the percolation algorithm described in [8], which is based on the physical model of liquid permeation. The latter works only for 2D images but we propose a generalization in Section 4.

3.1 Template Matching

The idea behind template matching is to find a certain pattern, given as an image T (the template), in the image I. The template T is moved over the image I, and at each position, the difference between T and (the corresponding subimage of) I is measured. A popular choice to measure is the correlation coefficient

$$C_L(r,s,t) = \frac{1}{N} \sum_{(i,j,k)\in R} \frac{\left(I(r+i,s+j,t+k)-\bar{I}\right)\left(T(i,j,k)-\bar{T}\right)}{\sigma_I \sigma_T}$$

where $\bar{I}$, $\bar{T}$ are the means and σ_I, σ_T are the standard deviations of T and I, respectively, and N is the number of voxels in T. Note that the correlation coefficient compares $\sigma_I^{-1}(I-\bar{I})$ and $\sigma_T^{-1}(T-\bar{T})$ rather than T and I. This normalization ensures that the correlation coefficient does not vary with intensity changes in T or I.

To model a crack, we use a plate of zeros, which represent the crack, surrounded by ones, which represent the material. That is the crack is assumed to be dark in contrast to the surrounding material. To be more precise, the template $T(i,j,k)$ with dimension $(b+c+b,n,n)$ has the value 1 for $i=1,\dots,b$ or $i=b+c+1,\dots,b+c+b$, and the value 0 for $i = b+1, \ldots, b+c$. Thus c controls the width of the crack and b the width of the border. Consequently, for $b = 2$ and $c = 2$ the template is given by

$$T = \begin{array}{c} \iddots \begin{bmatrix} 1 & 1 & \cdots & 1 & \cdots & 1 & 1 \\ 1 & 1 & \cdots & 1 & \cdots & 1 & 1 \\ & & & & & & 0 \\ & & & & & & 0 \\ & & & & & & 1 \\ & & & & & & 1 \end{bmatrix} \\ \begin{bmatrix} 1 & 1 & \cdots & 1 & \cdots & 1 & 1 \\ 1 & 1 & \cdots & 1 & \cdots & 1 & 1 \\ 0 & 0 & \cdots & 0 & \cdots & 0 & 0 \\ 0 & 0 & \cdots & 0 & \cdots & 0 & 0 \\ 1 & 1 & \cdots & 1 & \cdots & 1 & 1 \\ 1 & 1 & \cdots & 1 & \cdots & 1 & 1 \end{bmatrix} \iddots \end{array}$$

To ensure that cracks of arbitrary orientation are detected, we rotate T by small steps in both directions (azimuth, elevation) and search I for each resulting template, see Figure 1.

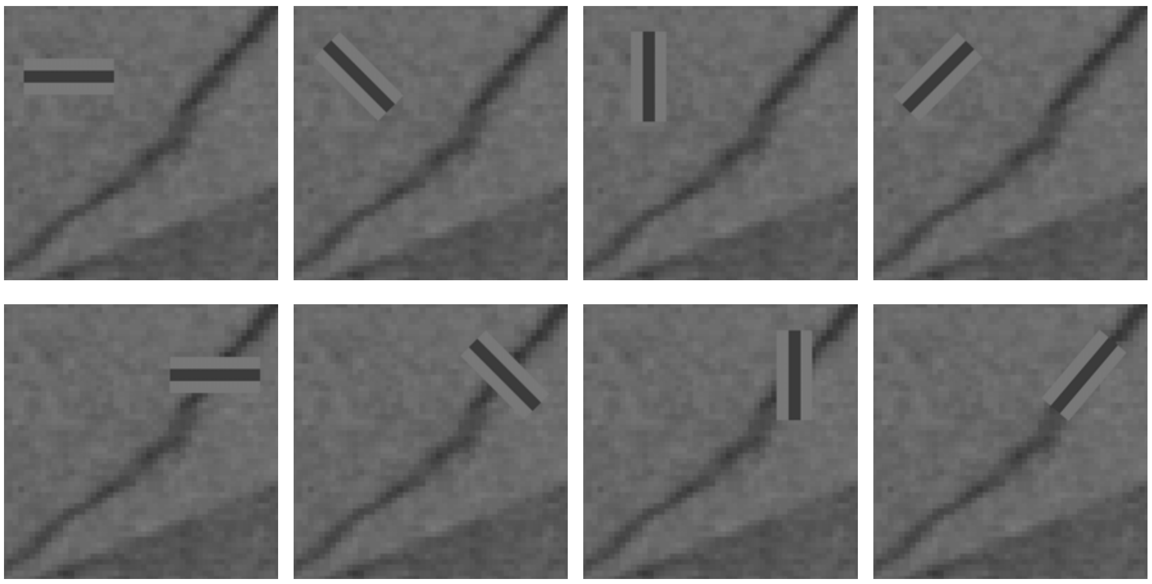

Figure 1. The process of template matching is illustrated here in 2D. The template of a crack in different orientations is shown at a background pixel (upper row) and a crack pixel (lower row).

3.2 Sheet Filter based on Hessian Eigenvalues

A sheet filter for detecting surface-like structures in volume data, what cracks obviously are, is proposed in [6]. This filter is computed from the eigenvalues of the Hessian Matrix Δ, which is defined by

$$\Delta I = \begin{bmatrix} I_{xx} & I_{xy} & I_{xz} \\ I_{yx} & I_{yy} & I_{yz} \\ I_{zx} & I_{zy} & I_{zz} \end{bmatrix} \; ,$$

where $I_{xx} = \frac{\partial^2}{\partial^2 x} I$, $I_{xy} = \frac{\partial^2}{\partial x \partial y} I$, and so on. Let $\lambda_1 \leq \lambda_2 \leq \lambda_3$ be the eigenvalues of Δ at a point p and v_1, v_2, v_3 the corresponding eigenvectors at p. Now v_3 is the direction along which the second derivative of I is largest. Its value is λ_3. Similarly, the second derivative is smallest along v_1 with value λ_1. In the plane orthogonal to v_3, the direction of the maximum second derivative has value λ_2 in direction v_2.

The second derivative at points of a sheet structure has a high value in the direction orthogonal to the sheet and small values in the directions tangential to the sheet (under the assumption that sheets are darker than the surrounding region, like cracks in our datasets). Thus the points of a sheet satisfy the conditions

$$\lambda_3 \gg 0 \quad \text{and} \quad \lambda_3 \gg \lambda_1 \simeq 0 \quad \text{and} \quad \lambda_3 \gg \lambda_2 \simeq 0 \ ,$$

i.e. Δ has one large eigenvalue and two small eigenvalues, see Figure 2. A sheet filter $\mathcal{S}$ is now defined by

$$\mathcal{S}\{I\} = \begin{cases} \lambda_3 \left(1 - \frac{\lambda_3}{|\lambda_1|}\right)^\gamma \left(1 - \frac{\lambda_3}{|\lambda_2|}\right)^\gamma & , \ \lambda_3 \geq 0 \\ 0 & , \ \text{else} \ , \end{cases}$$

where the terms $(1 - \lambda_3/|\lambda_1|)^\gamma$ and $(1 - \lambda_3/|\lambda_2|)^\gamma$ represent the conditions $\lambda_3 \gg \lambda_1 \simeq 0$ and $\lambda_3 \gg \lambda_2 \simeq 0$, respectively, and λ_3 represents the condition $\lambda_3 \gg 0$. The parameter γ controls the sharpness of the selectivity for the conditions $\lambda_3 \gg \lambda_1 \simeq 0$ and $\lambda_3 \gg \lambda_2 \simeq 0$.

In order to make the filter $\mathcal{S}$ tunable to a certain crack width, Gaussian convolution is used to smooth the image data I. The filter response can be adjusted to a certain crack width by the standard deviation σ of the Gaussian function, i.e. for larger σ the filter responds to cracks of larger width.

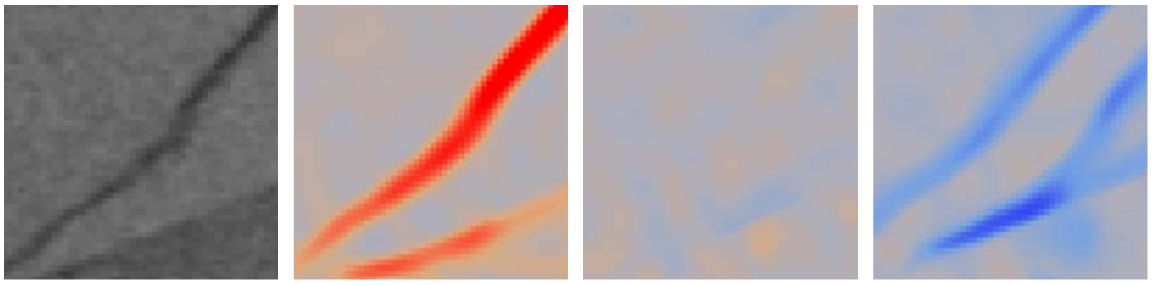

Figure 2. A 2D slice of an image of a crack and the three eigenvalues λ_3, λ_2, λ_1 (from left to right).

3.3 Percolation

The percolation algorithm described in [8] works only for 2D images. The percolation process, which is based on the physical model of liquid permeation, is started from each pixel. Depending on the shape of the percolated region, the pixel is considered as a crack pixel or not. Omitting some technical details, the percolation process consists basically of the following steps:

1. The starting pixel p_s is added to the set of percolated pixels P.
2. The threshold t is set to the value

$$t = \max\left(\max_{p \in D_p} I(p), t\right) + w \ ,$$

   where w is a parameter to accelerate the percolation.
3. Each pixel p neighboring P is added to P if $I(p) \leq t$.
4. Steps 2 and 3 are repeated until P reaches the boundary of a $M \times M$ window with center p.

5. If the circularity F_c of P is close to 0, p_s is considered as a crack pixel. The circularity F_c is defined by

$$F_c = \frac{4|P|}{\pi(\operatorname{diam} P)^2}$$

where $|P|$ is the cardinality and $\operatorname{diam} P$ the diameter of P.

The percolation process is illustrated in Figure 3.

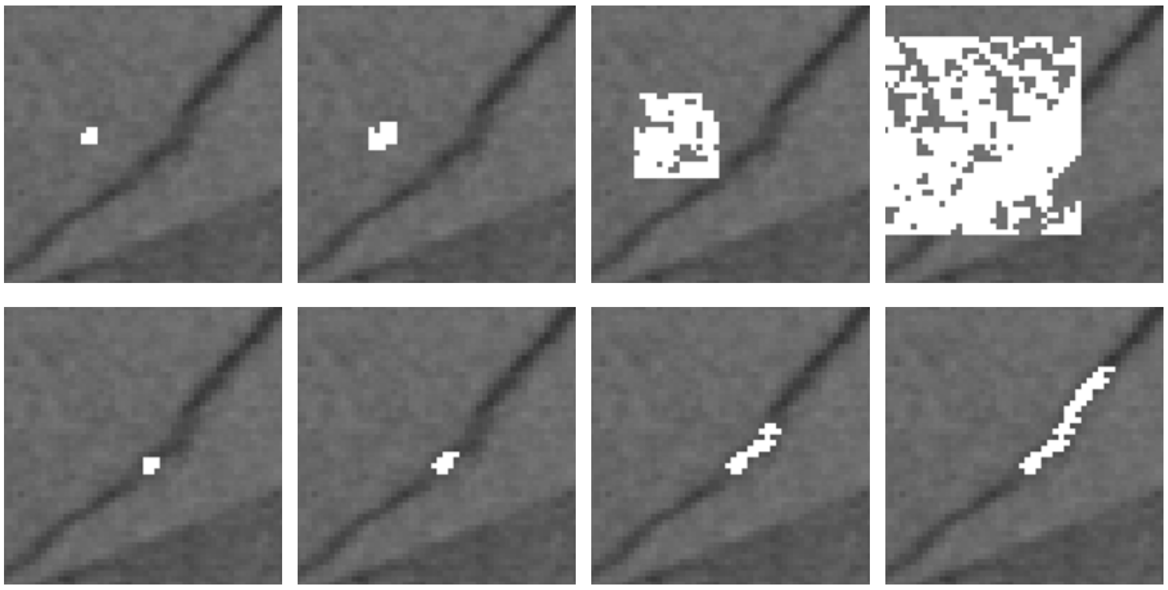

Figure 3. The percolation process in 2D starting from a background pixel (upper row) and a crack pixel (lower row).

4. Hessian-Driven Percolation

The percolation algorithm described in the previous section cannot be used immediately for the detection of cracks in 3D images. While steps 1–4 can be directly applied to 3D images, the circularity F_c computed in step 5 has no obvious generalization to 3D. Moreover the computation time would take very long if the percolation process was started from each voxel (approximately 87 hours for a small dataset of 64 MB on a 2.66 GHz quad-core Xeon using multiple threads). An extension of the percolation algorithm to 3D with improved computation time is proposed in this section.

To overcome the problems mentioned above, we employ the sheet filter $\mathcal{S}$. Let H be the result obtained from $\mathcal{S}$ by choosing an appropriate threshold. The computation of the circularity of the percolated region P is replaced by the computation of

$$\frac{|P \cap H|}{|P|} \;,$$

i.e. we check if the percentage of voxels in P which is contained in H is close to 1. Furthermore we use H to improve the speed of the calculation. This is achieved by starting the percolation process only at voxels contained in H. Note that if we considered the starting voxel of the percolation process as a crack voxel only if $(|P \cap H|)/|P|$ is close to 1, then the result of the percolation algorithm would be a subset of H. Thus it would be impossible to correct false negatives in H. To avoid this limitation, we consider the whole set P as crack voxels if $(|P \cap H|)/|P|$ is close to 1. Moreover, we count how often a voxel is considered as crack voxel in this way. So we obtain a field of integers and can choose a threshold to exclude voxels which are accidently detected for only a few times.

At first glance, it might seem unnecessarily complicated to compute the filter $\mathcal{S}$ to extend the percolation to 3D. However, the filter $\mathcal{S}$ can be computed quite fast. Moreover, as we will see in Section 5, $\mathcal{S}$ is not suitable as a crack filter on its own, but well suited for our purpose.

5. Results and Discussion

The results of template matching, the sheet filter $\mathcal{S}$ and the Hessian-driven Percolation are discussed in this section. We use a sub-volume of 128^3 voxels of a concrete dataset containing bending and branching cracks. A manually created segmentation serves as a reference to compare the results qualitatively and quantitatively. Timings of the methods are also provided.

The following parameters were used: For template matching, a template with crack width $c = 2$, border width $b = 2$ and side length $n = 20$ was chosen. The template was rotated by 5 degree steps. The sheet filter $\mathcal{S}$ was used with sharpness $\gamma = 2$ and a standard deviation of $\sigma = 3$ for Gaussian smoothing. The computation of the threshold t in the percolation algorithm was modified; we use a fixed threshold of $t = I(p_s) + 5$, where p_s is the starting voxel. With the variable threshold described in Section 3.3, we achieved no improvements, rather the opposite.

Figure 4 demonstrates the qualitative differences between the methods. Template matching does not detect bending and branching of cracks properly. Lowering the threshold does not solve the problem; edges between stones and cement matrix are falsely detected in this case. Similar problems arise with the sheet filter $\mathcal{S}$ while the problem of discriminating cracks from edges is even bigger. Moreover cracks are detected much thicker than they actually are. The Hessian-driven percolation algorithm detects thick cracks reliably; bending and branching is no problem in contrast with template matching or the sheet filter $\mathcal{S}$. However, thin cracks might be missed, in this regard template matching is superior.

To evaluate the quality of the methods quantitatively, we use a manually created reference segmentation, which we consider correct, and compute the sensitivity and the false discovery rate. The sensitivity is the percentage of crack voxels in the reference segmentation that are detected by the automatic method. Note that a voxel is considered detected if there is a voxel in the computed result within a radius of 2 voxels. Consequently, small differences concerning position and thickness of cracks are tolerated. The false discovery rate is the percentage of voxels in the computed result that do not belong to cracks according to the reference segmentation. Here, a voxel is considered not belonging to a crack if the distance to the reference segmentation is at least 2 voxels, i.e. small deviations from the reference segmentation are, again, tolerated.

Table 1 shows the sensitivity and the false discovery rate for the different methods; for each method three meaningful thresholds were chosen. It is obvious that a higher threshold causes a lower sensitivity, but also a lower false discovery rate. Since we are interested in reliable result, we demand a low false discovery rate. With the sheet filter $\mathcal{S}$, it is not possible to obtain a low false discovery rate unless the sensitivity is also very poor. With template matching, the results are much better. The Hessian-driven percolation, though, gives the best results.

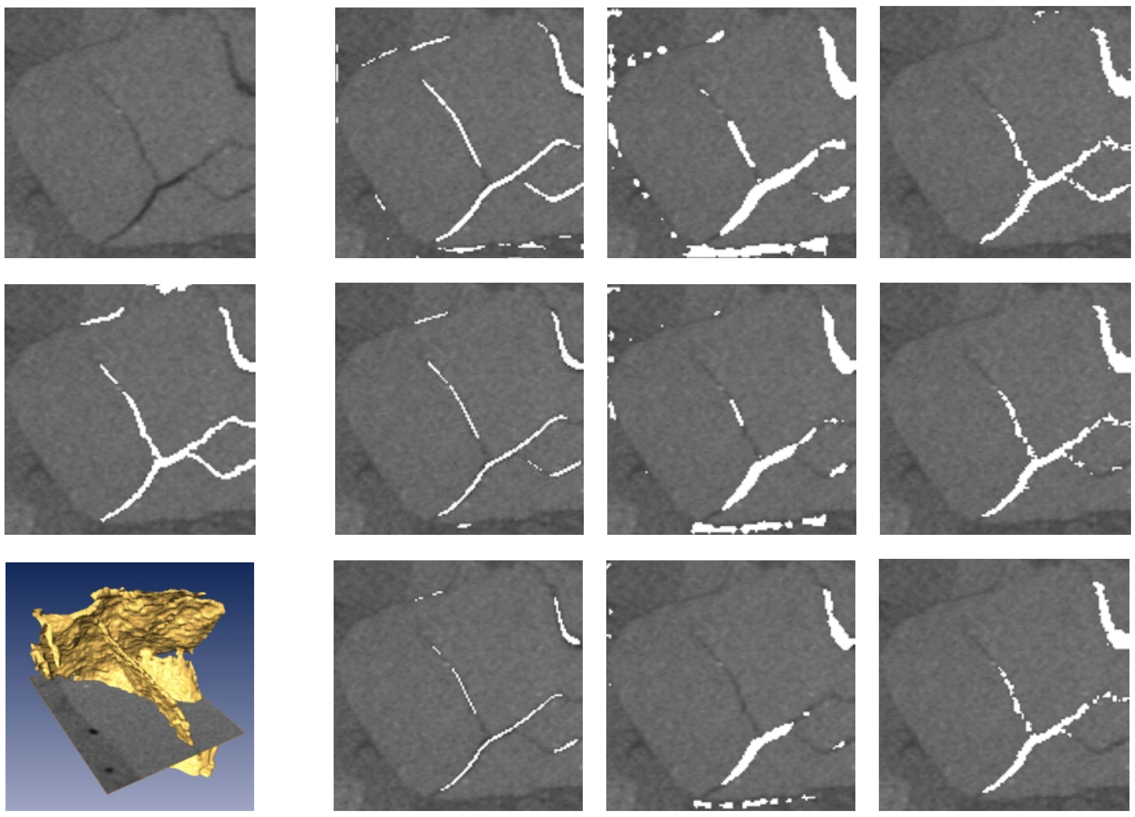

Figure 4. First column: dataset (top), manually segmented dataset (middle) and 3D visualization of Hessian-driven percolation with threshold 1 (bottom). First column: results of template matching with thresholds 0.4, 0.5 and 0.6. Second column: results of the sheet filter $\mathcal{S}$ with thresholds 0.2, 0.3 and 0.4. Third column: results of Hessian-driven percolation with thresholds 1, 10 and 20.

Table 1. Comparison of computed results with reference segmentation

Method	**Threshold**	**Sensitivity %**	**False Discovery Rate %**
Template matching	0.4	89.39	24.02
	0.5	76.75	5.71
	0.6	57.19	0.93
Sheet filter $\mathcal{S}$	0.2	89.42	43.31
	0.3	72.23	29.94
	0.4	55.93	16.97
Hessian-driven percolation	1	81.14	2.49
	10	72.32	0.46
	20	69.19	0.26

The timings for a volume of 64 MB are given in Table 2. Template matching and the sheet filter $\mathcal{S}$ were executed on the GPU, the Hessian-driven percolation algorithm on the CPU using multiple threads. Note that the size of complete datasets is about 6 GB.

Table 2. Timings for a dataset of 64 MB on a Quad-Core Xeon X5355 with GeForce 8800 GTX

Method	Time
Template matching	2h 41m
Sheet filter $\mathcal{S}$	8m 52s
Hessian-driven percolation	46m 23s

The long computation time of template matching explains why we combined the percolation algorithm with the sheet filter $\mathcal{S}$, although a combination with template matching might give better results. Moreover the limitations of the sheet filter $\mathcal{S}$ are not crucial in combination with the percolation algorithm. Bending or branching parts of a crack that the filter $\mathcal{S}$ missed are corrected: If the percolation process starts close to such a missed part, the percolated region P covers the bending or branching. The coincidence of P with the result H of the sheet filter $\mathcal{S}$ is still high since the missed parts are small. Thus P is considered as a crack. The inability of $\mathcal{S}$ to discriminate cracks from edges is also compensated; if the percolation process is started at an edge, the resulting percolated region P consists either of the region on one or both sides of the edge, thus the coincidence with H is low.

6. Conclusion

In this work, we analyzed the suitability of template matching, a sheet filter $\mathcal{S}$, and a percolation algorithm for crack detection. We extended the latter to 3D images by employing the result of the sheet filter $\mathcal{S}$. This method, which we called Hessian-driven percolation, seems to be the most suitable when reliable results are the goal. The false discovery rate ist low; thick cracks are reliably detected, including bending and branching of cracks. However, thin cracks are in many cases missed by this method, whereas they are often found with template matching. Thus, it might be relevant to consider template matching when thin cracks are important in the analysis.

References

[1] D. Stalling, M. Westerhoff, H.-C. Hege. Amira: A Highly Interactive System for Visual Data Analysis. In C. D. Hansen, C. R. Johnson (eds.), The Visualization Handbook, Chapter 38, pp. 749-767, Elsevier, 2005.
[2] S. Mazumder, K.-H.A.A. Wolf, K. Elewaut, R. Ephraim. Application of X-ray computed tomography for analyzing cleat spacing and cleat aperture in coal samples. International Journal of Coal Geology, 68:205-222, 2006.
[3] I. Abdel-Qader, O. Abudayyeh, M. E. Kelly. Analysis of Edge-Detection Techniques for Crack Identification in Bridges. Journal of Computing and Civil Engineering, 17(4):255-263, 2003.
[4] J.H. Lee, J.M. Lee, J.W. Park, Y.S. Moon. Efficient Algorithms for Automatic Detection of Cracks on a Concrete Bridge. 23rd International Technical Conference on Circuits/Systems, Computers and Communication, 2008.
[5] M.N. Lebbink, W.J.C. Geerts, T.P. van der Krift, M. Bouwhuis, L.O. Hertzberger, A.J. Verkleij, A.J. Koster. Template matching as a tool for annotation of tomograms of stained biological structures. Journal of Structural Biology, 158:327-335, 2007.
[6] Y. Sato, C.-F. Westin, A. Bhalerao, S. Nakajima, N. Shiraga, S. Tamura, R. Kikinis. Tissue Classification Based on 3D Local Intensity Structures for Volume Rendering. IEEE Transactions on Visualization and Computer Graphics, 6(2):160-180, 2000.
[7] Y. Fujita, Y. Hamamoto. A robust automatic crack detection method from noisy concrete surfaces. Machine Vision and Applications, 22:245-254, 2011.
[8] T. Yamaguchi, S. Hashimoto. Fast crack detection method for large-size concrete surface images using percolation-based image processing. Machine Vision and Applications, 21:797-809, 2010.
[9] A.M. Roseman. Particle finding in electron micrographs using a fast local correlation algorithm. Ultramicroscopy, 94:225-236, 2003.