

# Minimal Paths for 3D Crack Detection in Concrete

Franziska Müsebeck<sup>1</sup>, Ali Moghiseh<sup>2</sup>, Claudia Redenbach<sup>1</sup>  
and Katja Schladitz<sup>2</sup>

<sup>1</sup> Department of Mathematics, University of Kaiserslautern  
Gottlieb-Daimler-Straße 47, 67663 Kaiserslautern

<sup>2</sup> Fraunhofer-Institut für Techno- und Wirtschaftsmathematik,  
Fraunhofer-Platz 1, 67663 Kaiserslautern

**Abstract** Concrete is a commonly used construction material for buildings, bridges and roads. As safety is very important in such constructions, the investigation of damage processes in concrete is a highly relevant topic. For instance, the early detection of cracks can prevent the collapse of a bridge. Thus, the necessity of automated crack detection and segmentation arises. We generalize and modify a 2D method for crack detection in road pavement images to 3D. It is based on modeling the image as a graph and searching for minimal paths therein. The proposed 3D method is evaluated on synthetic crack images and applied to a 3D computed tomography image of real concrete.

**Keywords** Crack detection, image segmentation, three-dimensional, computed tomography

## 1 Introduction

The investigation of damage processes in concrete has become an increasingly popular topic. For design, monitoring and maintenance of buildings or other constructions, it is essential to understand damage of building materials. In order to gain a deeper understanding of the mechanical properties of concrete, crack initiation and development are studied on concrete specimens during loading tests using

computed tomography (CT). Visual inspection of such a 3D CT image and manual segmentation of cracks is a time intensive process and can therefore only be executed on a few slices as these images are usually very large [1]. This makes automated detection of cracks in 3D images desirable. While crack segmentation in 2D images has already been widely researched, so far, there is no satisfying solution available in 3D [1].

A typical motivation to look for cracks in 2D images is monitoring of the road surface conditions by identifying cracks in pavement images of roadways. There is a variety of crack detection methods based on two general assumptions. The first one concerns the brightness, i. e., the crack is darker than the background which means that the gray values of the crack pixels are smaller compared to the neighboring background intensities. The second one is related to geometry, namely that the crack is continuous which means that the crack pixels are connected.

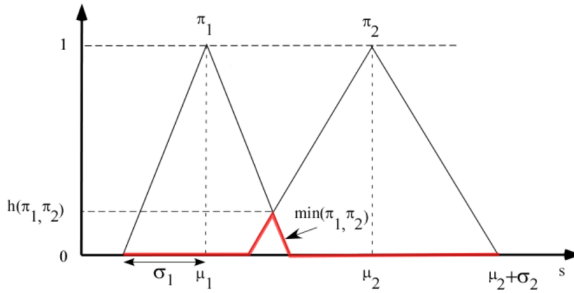
Here, we present a generalization of the 2D approach in [2] which uses exactly these characteristics. The image is modeled as a graph and minimal paths are computed. If the nodes are weighted by the pixels' gray values, a minimal path possesses the above mentioned crack properties: it is connected and consists of vertices with small weights, i. e. small intensity values.

We first introduce the 2D method proposed in [2] which we will refer to as *MinPath2D* and then present our approach of a generalization to 3D images. Adjustments are made on the one hand to obtain an appropriate image model and on the other hand to save computation time.

## 2 2D cracks by minimal paths

### Image model

The *MinPath2D* algorithm uses a representation of the gray value image as a directed, vertex-weighted graph. In fact, several graphs can be defined for one image depending on the directions represented in the 8-neighbourhood on the pixel grid. In all these graphs, the set of vertices corresponds to the pixels in the image and the weights are the pixels' intensities. The graphs differ by the set of



**Figure 2.1:** Sup-min function to evaluate the degree of coherence of two information sources [2]

edges which is chosen depending on direction. For each direction, e. g. *up*, a set of three discrete directions is selected. Edges are constructed by connecting all vertices to their three neighbors towards these. For instance, in the graph for the direction *up*, a vertex is connected to its neighbor directly above as well as to the two vertices left and right from the vertex above. This way, we can construct directed, vertex-weighted graphs for the eight directions in 2D: *up*, *down*, *left*, *right*, *upper left*, *lower left*, *upper right*, *lower right*.

### Searching minimal paths

Based on this image model, the MinPath2D algorithm computes minimal paths in the above described graphs. The algorithm gets a graph  $G = (V, E)$  and a predefined path length  $\ell$  as inputs and computes for each vertex  $x \in V$  a path with minimal weight of length  $\ell$  starting in the considered vertex  $x$ . Applying this algorithm to all graphs yields eight paths for each vertex. The two paths for opposite directions are merged into one path of length  $2\ell + 1$ , whereby the start pixel of the individual paths becomes the center pixel of the merged path. This procedure results in four minimal paths for each pixel.

### Pixelwise classification

In the next step, it has to be decided whether or not a pixel belongs to a crack. The basic idea is to introduce a characteristic which takes a small value in an orientation along the crack and higher values along other orientations. We use the Free Form Anisotropy (FFA) measure which was derived from possibility theory [3]. This theory based on fuzzy sets is used to model uncertainty by introducing a degree of membership for the elements of a set. Possibility distributions are modelled by fuzzy set membership functions. For possibility theory, they play the role of probability densities for probability theory. The relationship between probability theory and possibility theory has been discussed as both theories seem to be similar in the sense that both deal with some type of uncertainty and both use the  $[0, 1]$  interval as range of their respective functions [4]. It is however difficult to compare them as it is not clear on which level – e. g. mathematically, semantically, or linguistically. See [4] for details.

Here, we use the conversion of each minimal path found in the first step into a so-called information source  $\pi_i$  [5] which is represented by a possibility distribution. To this end, the mean value and the standard deviation of gray levels of each path are computed. Figure 2.1 shows two partially overlapping possibility distributions. This means, that certain gray values are considered possible by both information sources. The degree  $h$  of coherence between the two information sources is measured by the sup-min function. The greater the intersection, the higher the degree of coherence. This concept is applied to the classification problem.

If a pixel belongs to a crack, then among the minimal paths found in the first step, at least the path with smallest mean intensity follows the crack. Comparing the information source of this path with another one corresponding to a path running in the background, we observe a small coherence  $h$ . This is due to the fact that the number of gray levels occurring in both paths is rather small. In contrast, the coherence  $h$  of two different background sources is comparatively large since the gray level distributions of the two corresponding paths are more similar. The Free Form Anisotropy of a pixel  $x$  is then defined as the degree of conflict between information sources  $\pi_{\min} = (\mu_{\min}, \sigma_{\min})$  and  $\pi_{\max} = (\mu_{\max}, \sigma_{\max})$  of the two paths with

center  $x$  and minimal and maximal mean gray value, respectively, i. e.

$$\text{FFA}(x) = 1 - h(\pi_{\min}, \pi_{\max}) = 1 - \sup \min(\pi_{\min}, \pi_{\max}).$$

The FFA measure takes values between 0 and 1 and is close to 1 if  $x$  is a crack pixel. Thus, the final algorithm has two parameters, the length parameter  $\ell$  and a threshold  $t \in [0, 1]$  for which all pixels  $x$  with  $\text{FFA}(x) < t$  are classified as background and pixels with  $\text{FFA}(x) \geq t$  are identified as crack pixels.

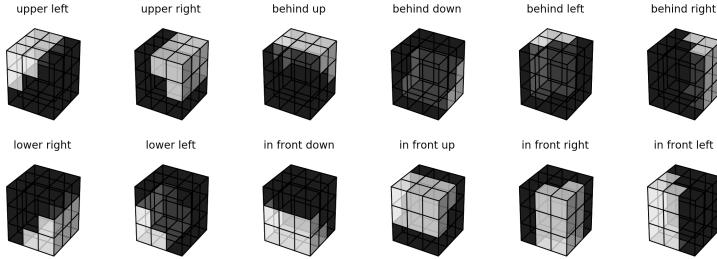
In practice, we calculate the coherence  $h$  according to Figure 2.1 as intersection of two lines  $L_1$  and  $L_2$ , where  $L_1$  is passing trough the two points  $(\mu_1, 1)$  and  $(\mu_1 + \sigma_1, 0)$  and  $L_2$  is passing trough  $(\mu_2, 1)$  and  $(\mu_2 - \sigma_2, 0)$ .

### 3 3D cracks by minimal paths

#### Image model

The image model is constructed in a similar way as in 2D. In particular, we use a 3D graph, where the set of vertices corresponds to the voxels in the image which are weighted by their gray values. The definition of the set of edges leaves more room for discussion. The number of direct neighbors of one voxel increases with dimension. An inner voxel has 26 neighbors in 3D while in 2D there are only eight neighbors. As a consequence, there are 26 directions implying 13 different orientations. The question arises, how many and which orientations we want to consider in the algorithm and how to define the discretization of one direction.

Directions can be categorized into three groups, where the first consists of the three main orientations: *up* and *down*, *left* and *right*, *in front* and *behind*. Secondly, there are the diagonals of a coordinate plane: *upper left* and *lower right*, *upper right* and *lower left*, *in front left* and *behind right*, *in front right* and *behind left*, *in front up* and *behind down*, *in front down* and *behind up*. The last group is formed by space diagonals (connecting two vertices that are not in the same coordinate plane): *upper left behind* and *lower right in front*, *upper right behind* and *lower left in front*, *upper left in front* and *lower right behind*, *upper*



**Figure 3.1:** Diagonal directions in 3D illustrated for the center voxel

*right in front* and *lower left behind*. We will only consider directions from the first and second category. This restriction is justified by saving computational time and moreover makes sense since the FFA measure only uses two paths and thus only two orientations to perform the classification. The decision to neglect the third category is based on the fact that this is the category which has the most overlap of voxels with the other categories.

The natural extension of the discretization is to take nine discrete directions into account for each direction. For instance, in 2D there are three pixels above an inner pixel in the same way as there are nine voxels above an inner voxel in 3D. Thus, we define the set of discrete directions for one direction by nine discrete directions as illustrated for the diagonals in Fig 3.1. Summarizing, a neighborhood for one voxel and for one direction is determined by the nine neighbors towards that direction. Moreover, we choose to take nine different orientations into account, namely the three main orientations and the six plane diagonals.

### Searching local minimal paths

Due to simplicity and the increasing computational effort in 3D, we deviate from the 2D method when calculating minimal paths by using a greedy propagation algorithm that returns only a locally minimal path. Beginning from the start node, the neighbor with the smallest weight is successively added to the path. This local propa-

gation does not necessarily yield a path with total minimal weight, but in the case of a crack voxel as start voxel, the resulting path still follows the crack path approximately and thus the classification by assessing the degree of coherence still works. Ultimately, it is more important that the different orientations are represented than that the path actually has minimal weight.

### **Voxelwise classification**

The classification per voxel is performed as in the MinPath2D algorithm. However, we use the coherence  $h$  (rather than the FFA measure) to compare two information sources. Then, a threshold  $t$  is chosen in  $[0, 1]$  such that all values less than or equal to  $t$  are classified as crack voxels and all values above are classified as background.

### **Post-processing**

The output of the classification algorithm shows some discontinuities where single voxels are missing in the crack surface, see Section 4, Figure 4.1. Hence, we apply a morphological closing followed by an erosion [6] to the binary classification output to connect the crack pixels and to remove single falsely identified background pixels.

### **Parameter selection**

As described above, our algorithm has two parameters, the length parameter  $\ell$  and the threshold  $t$  which must be selected appropriately. The influence of the parameters was investigated experimentally by keeping one fixed while varying the other one. We observed that the length parameter  $\ell$  should be chosen in the interval  $[24, 48]$  depending on the resolution of the image. The choice of the second parameter, the threshold  $t$ , turns out to be less important than the choice of  $\ell$  as long as it is sufficiently small. A good choice seems to be  $t \in [0.001, 0.1]$ .

## 4 Application

### Evaluation metrics

For the evaluation on the synthetic crack images, we use the F1-measure as an overall accuracy measure which is calculated from precision and recall.

The Precision is defined as

$$\text{Precision} = \frac{\text{true positives}}{\text{true positives} + \text{false positives}}$$

and measures how exact a positive result is, i. e. what proportion of voxels classified as positive are indeed positive.

The Recall is defined as

$$\text{Recall} = \frac{\text{true positives}}{\text{true positives} + \text{false negatives}}$$

and measures the proportion of positives which were identified correctly.

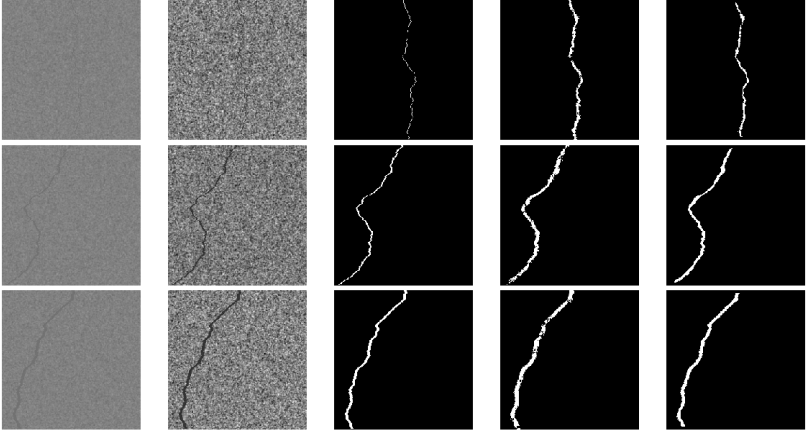
The F1-measure can be interpreted as a weighted average of Precision and Recall taking both aspects into account. It is given by

$$F1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}.$$

All three measures take values between 0 and 1 and reach their best value at 1. In practice, it is impossible to exactly define the crack boundary in real images. Hence, it is not uncommon to introduce a certain pixel (voxel) tolerance in the evaluation of a segmentation algorithm. In the following, we evaluate the results allowing for a tolerance of falsely classified voxels within a certain distance of the true crack. In our case, using a tolerance of *tol* voxels has the following meaning:

- A false negative voxel is counted as true positive, if in the predicted image there is a voxel classified as positive within a distance of *tol*.
- A false positive voxel is counted as true positive, if in the ground truth there is a voxel classified as positive within a distance of *tol*.

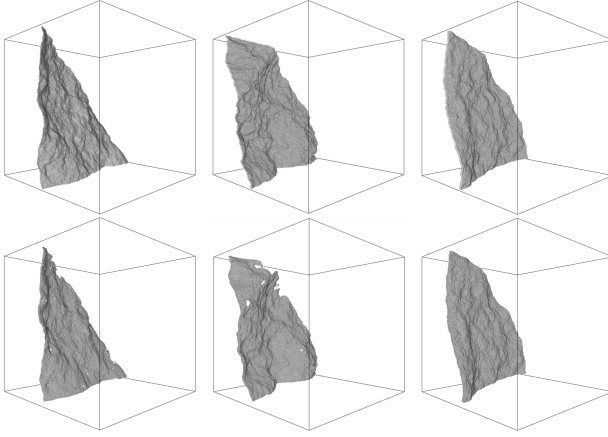




**Figure 4.1:** Z-slices of the results of the application on  $256^3$  synthetic 3D images with  $l = 24$  and  $t = 0.01$ . From left to right: input image, normalized input image, ground truth (generated by a realization of the fBS), output before post-processing, output after post-processing

### Results on synthetic images

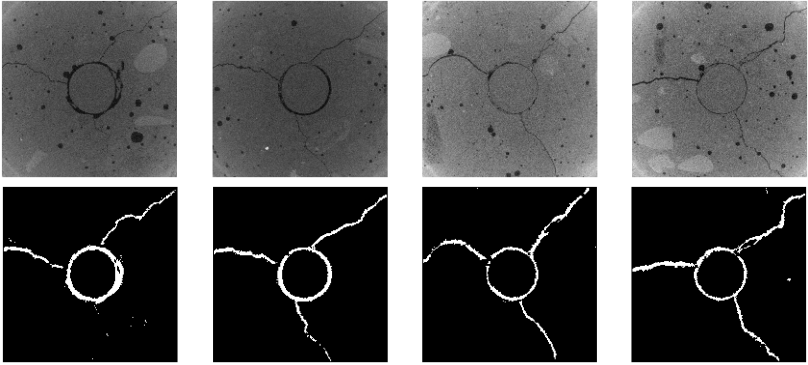
Synthetic cracks are simulated by a realization of a fractional Brownian Surface (fBS) [7] which provides a ground truth by which we can evaluate the output of the algorithm. Given the ground truth as binary crack image (generated by [8]), a corresponding realistic gray value crack image is obtained by generating noisy gray levels for background and crack. Varying crack widths can be generated by dilating the crack. We evaluate the algorithm on synthetic crack images with a crack width of  $w = 1$ ,  $w = 3$ , and  $w = 5$  voxels. Z-slices of the resulting segmentations are shown in Figure 4.1. Figure 4.2 shows the corresponding 3D renderings. The crack is properly segmented in all three cases. However, the detected crack is thicker than the true one. This is especially noticeable when the original crack is very thin as in the case  $w = 1$ . Evaluating the output by the F1-measure with a tolerance of  $tol = 1$ , we obtain a value of 0.9609 for the image with  $w = 1$ , 0.9599 for  $w = 3$  and 0.9744 for  $w = 5$ .



**Figure 4.2:** 3D renderings of the results on  $256^3$  synthetic images with  $\ell = 24$  and  $t = 0.01$ . Top row: ground truth. Bottom row: output after post-processing. From left to right:  $w = 1$ ,  $w = 3$ ,  $w = 5$ .

## Results on real images

In the next step, we apply our algorithm to 3D images of real concrete. In particular, we use a 3D CT image of a concrete specimen with a glass fiber reinforced polymer bar used as reinforcement. Pulling out the bar generates cracks. The sample has a diameter of 4.8 cm and the original image has a size of  $1986 \times 1986 \times 1576$  voxels. After shrinking the CT image by a factor of two and cutting out a cubic patch, we obtain an image of size  $620^3$  which we pass as input to the algorithm. Depending on the mixing conditions and the coarseness of the aggregates, the concrete matrix may appear rather heterogeneous in CT images. In our case, for instance, we observe pores as well as highly absorbing grains (see Figure 4.3). The real concrete is clearly more complex than synthetic data. Nevertheless, our algorithm can handle the background structure and is able to segment the crack except for a few really thin and fine structures. Z-slices of the output of our algorithm are shown in Figure 4.3.



**Figure 4.3:** Z-slices of the results for a 3D CT image of real concrete using the parameters  $\ell = 48$  and  $t = 0.001$ . Top row: normalized input. Bottom row: output after post-processing. Sample: C. Caspari, University of Kaiserslautern, CT imaging: F. Schreiber, Fraunhofer ITWM

### Implementation and runtime

The algorithm was implemented in C++. It can be integrated as a plug-in into the image processing software *ToolIP* (*Fraunhofer-Institut für Techno- und Wirtschaftsmathematik*, [9]) which is used to segment the crack images. The runtime increases with the image size as well as with increasing parameter  $\ell$ . It is roughly 3 minutes for a  $256^3$  image with  $\ell = 24$  and about 30 minutes for a  $620^3$  image with  $\ell = 48$ . All runtimes are measured on a standard computer of the image processing department of the ITWM with eight processors of the type Intel(R) Core(TM) i7-4790 CPU @ 3.60GHz distributed on four cores, with a working memory of 16 GB.

## 5 Summary

We propose a new algorithm for crack detection in 3D images. The method is based on the 2D algorithm from [2] which we generalized to the 3D case. Our generalization includes the adaption of the image modeling as 3D graph, whereby the definition of the neighborhood relation of the vertices by directions as well as the definition and

choice of directions (and orientations) were adapted. Further adjustments were made as we propose the usage of approximate minimal paths obtained by a local propagation algorithm. We demonstrated that the algorithm is able to segment cracks in 3D images. In particular, we applied the algorithm to synthetic images whereby we were able to achieve F1 values between 0.9599 and 0.9744 for test images with different crack width. Moreover, the algorithm is able to deal with images of real concrete and to correctly segment the cracks. Even the more complex background structure containing dark pores can be handled. The main weakness of the algorithm is that the segmented crack is too thick compared to the original one and that it misses some very thin and fine crack structures.

## Acknowledgments

This research was supported by the German Federal Ministry of Education and Research under project *Detektion von Anomalien in großen räumlichen Bilddaten* (DAnoBi) and by the Rhineland-Palatinate research initiative through project “Mathematics applied to real world challenges” (MathApp).

We thank Christian Caspari (Department of Civil Engineering, University of Kaiserslautern) for the concrete sample and Franz Schreiber (Fraunhofer ITWM) for the CT imaging.

## References

1. O. Paetsch, “Possibilities and limitations of automatic feature extraction shown by the example of crack detection in 3d-ct images of concrete specimen,” in *9th Conference on Industrial Computed Tomography (iCT) 2019*, iCT 2019 Conference Proceedings, 2019.
2. M. Avila, S. Begot, F. Duculty, and T. S. Nguyen, “2d image based road pavement crack detection by calculating minimal paths and dynamic programming,” in *2014 IEEE International Conference on Image Processing (ICIP)*, 2014, pp. 783–787.
3. L.A. Zadeh, “Fuzzy sets as a basis for a theory of possibility,” *Fuzzy Sets and Systems*, vol. 100, pp. 9 – 34, 1999.

4. J.-H. Zimmermann, *Fuzzy Set Theory—and Its Applications*. Springer, Dordrecht, 2008.
5. I. Bloch, *Information Fusion in Signal and Image Processing: Major Probabilistic and Non-Probabilistic Numerical Approaches*, ser. ISTE. Wiley, 2008.
6. R. C. Gonzalez and R. E. Woods, *Digital Image Processing*. Pearson Education, 2009.
7. P. S. Addison, L. T. Dougan, A. S. Ndumu, and W. M. Mackenzie, “A fractional Brownian motion model of cracking,” in *Paradigms Of Complexity: Fractals And Structures In The Sciences*, 2000, pp. 117–123.
8. Z. Botev, “Fractional Brownian field or surface generator,” <https://de.mathworks.com/matlabcentral/fileexchange/38945-fractional-brownian-field-or-surface-generator>, retrieved June 10, 2020.
9. Fraunhofer ITWM, “Toolip,” <https://www.itwm.fraunhofer.de/toolip>, retrieved June 10, 2020.