Micro Crack Detection with Dijkstra's Shortest Path Algorithm

Christina Gunkel[1] Alexander Stepper[1] Arne C. Müller[2]
Christine H. Müller[3]*

December 8, 2009

[1]University of Kassel, Department of Mathematics, D-34109 Kassel, Germany
[2]Free University of Berlin, Department of Mathematics and Computer Science, Arnimallee 14, D-14195 Berlin, Germany
[3] University of Technology Dortmund, Faculty of Statistics, Vogelpothsweg 87, D-44221 Dortmund, Germany

Abstract

A package based on the free software R is presented which allows the automatic detection of micro cracks and corresponding statistical analysis of crack quantities. It uses a shortest path algorithm for detecting micro cracks in situations where the cracks are surrounded by plastic deformations and where a discrimination between cracks and plastic deformations is difficult. In a first step, crack clusters are detected as connected components of pixels with values below a given threshold value. Then the crack paths are determined by

*Research supported by the SFB/TR TRR 30 Project D6

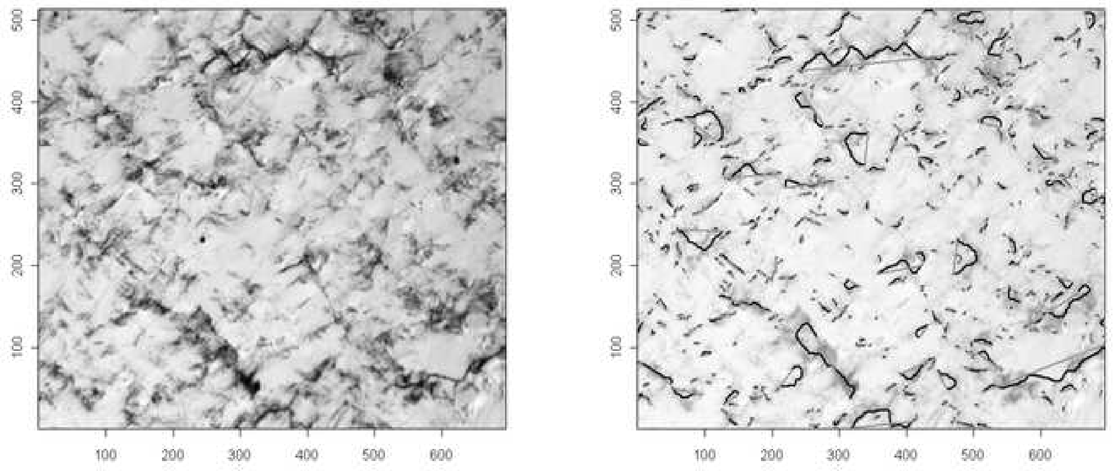
Dijkstra's algorithm as longest shortest paths through the darkest parts of the crack clusters. Linear parts of kinked paths can be identified with this. The new method was applied to over 2000 images. Some statistical applications and a comparison with another free image tool are given.

Keywords: Image analysis; Crack detection; Crack cluster; Crack path; Dijkstra's algorithm; Linear parts of a path

1 Introduction

The understanding of crack initiation and crack growth is very important for predicting the life time of products as wheels of trains or hip replacement. Macro cracks origin from micro cracks which are only visible with a microscope. The initiation and growth of micro cracks is a stochastic process. More and more approaches exist for describing the initiation and growth of micro cracks by stochastic models. See e.g. [1], [2], [3], [4], [5], [6]. These models must be validated by statistical methods. Since these models are rather complicated models, a large amount of data on cracks must be used. These data can be obtained from microscopic images from the surface of assays under strain. Often hundreds of micro cracks are visible in such images, at least at a later stage of the fatigue process.

To detect this lot of micro cracks, we developed a method based on a shortest path algorithm. Edge detection methods (see e.g. [7], [8], [9]) cannot be used since the cracks are surrounded by dark areas. The reason is that micro cracks origin from plastic deformations of the material visible in darker areas. Indeed it is often not easy to distinguish between a plastic deformation and a micro crack since micro cracks are always surrounded by more or less large areas of plastic deformations (see Figure 1 a)). Therefore we call the areas of micro cracks and surrounding plastic deformations ”crack clusters”.



a) b)

Figure 1: a) original image at Time 18, b) detected crack paths

Since micro cracks origin from plastic deformations, it is no problem for a quantitative analysis if the discrimination between micro cracks and plastic deformations is not perfect. But real micro cracks differ from plastic deformations by darker colors. Hence the aim is to follow mainly the darkest parts of a crack cluster.

According to the knowledge of the authors, existing crack analysis tools are only able to highlight cracks and crack clusters (see e.g. [10] and [11]). Or they have the ability to find crack clusters but cannot find the crack paths. Others are only able to detect few cracks. See e.g. [12], [13], [14] [15] [16]. Like other free and commercial tools, for example the free software `UTHSCSA Image Tool`[1] developed

[1]`http://ddsdx.uthscsa.edu/dig/itdesc.html`, Department of Dental Diagnostic Science at The University of Texas Health Science Center, San Antonio, Texas

by C. D. Wilcox, S. B. Dove, W. D. McDavid, and D. B. Greer describes the crack clusters only by ellipses and rectangles which are easily obtained by calculating the variances and covariances of the positions of the pixels inside the cluster. The lengths of the main axes of the ellipses are used as the lengths of the cracks (see also [15]). This is of course only a bad approximation of true crack paths, i.e. the darkest parts of the crack cluster. Moreover cracks are kinked and curved so that the true length of a crack is much longer then the distance between its start and end point. It is also important for a detailed crack analysis to know what are the kinks and curves of a crack.

Better approximations of the crack paths can be given by skeletonization and thinning. See e.g. [17], [8], [9]. But simple skeletonization methods do not take into account the gray levels in the clusters. Hence they do not result in the darkest parts of the cluster. Certainly, it is possible to modify skeletonization so that it is resulting in the darkest parts. But this modification would not be easy. Moreover, every skeletonization method is very sensitive to small changes of the cluster and provides usually tree-like paths for one cluster. For micro cracks however, it is very important to obtain only one path for each cluster since the lengths of such main paths are most relevant for the stability of the material. This is different to the approach of Iyer and Sinha [10] who used a tree-like geometry for cracks.

The method proposed here bases on a shortest path algorithm. Shortest path algorithms have many applications in image analysis as for object segmentation and disparity estimation (see [16]). They were also applied for crack detection, but only in the case of one crack as a circular crack of a borehole core [16] or one crack from top to bottom as in [14]. Moreover, the crack paths in [14] and [16] follow structures which are given already by lines. These conditions are not satisfied for micro cracks.

Micro crack paths must be determined inside the crack clusters and there is a large amount of crack clusters. Our method uses Dijkstra's shortest path algorithm to determine the longest shortest path in the crack cluster which follows the darkest parts of the crack cluster. Thereby it uses also gray areas so that detected cracks are not interrupted by less dark parts. Since our method provides the whole crack path, kinks and curves of the cracks can be analyzed. It is applicable in the presence of hundreds of crack clusters.

The method is implemented in `C` and included in a `R` package. `R` provided by R Development Core Team [18] is a free software with a huge amount of statistical methods. It is meanwhile widespread in the statistical community. Hence cracks detected inside `R` can be easily analyzed with many statistical methods provided by `R`. Since `R` is also a programming language, additional tools for analyzing the cracks can be easily implemented as well. Hence crack detection and statistical analysis can be carried out with the same software.

Section 2 describes the new method. In Section 3, the method is applied to two series of images each consisting of over 800 images. It is compared with the free software UTHSCSA Image Tool in Section 4. In Section 5, some statistical analysis of crack behavior is shown. In particular, there is shown how linear parts of a crack path can be found.

2 Package `crackrec`

The `R` package `crackrec`, which can be downloaded from our homepage, contains six functions, namely `rbmp`, `shadow.remove`, `median.filter`, `threshold.msi`, `crackrec`, and `crackplot`, where `rbmp` and `crackrec` base on `C` and `C++`, respectively. The new crack detection method is included in `crackrec`. The other functions are auxiliary functions as the function `rbmp` which converts a 24-bit bmp file of an image into a `R` matrix of pixel values ranging from 0 to 255.

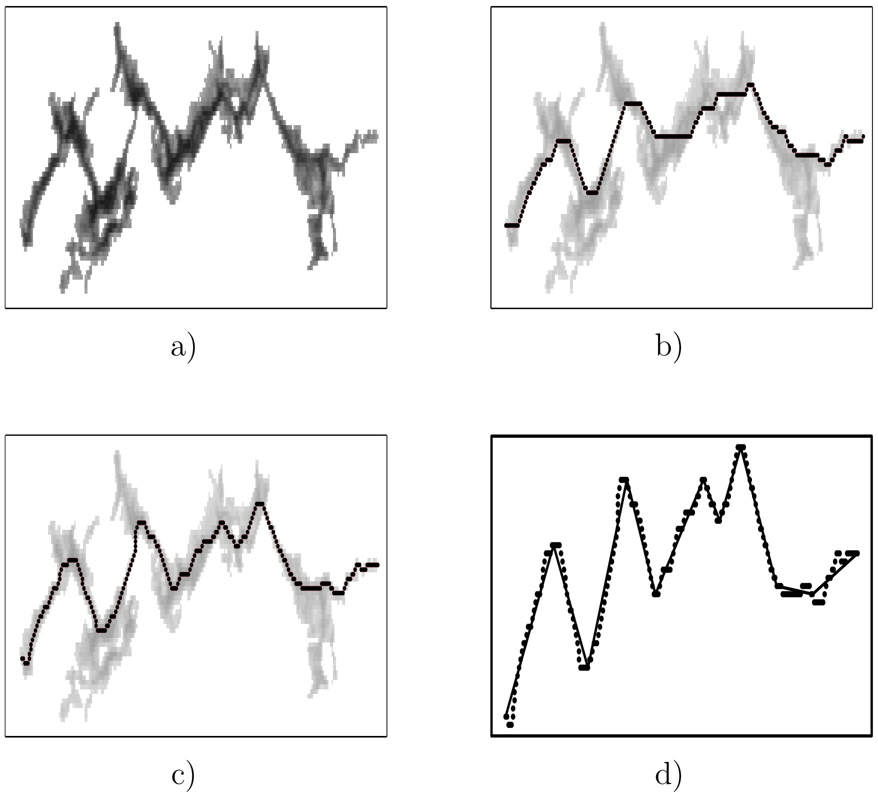


Figure 2: a) crack cluster for a zigzag crack, b) detected crack path with $\alpha = 0 = \beta$, c) detected crack path with $\alpha = 1 = \beta$, d) linear parts of the zigzag crack

The function `crackrec` provides the cracks from a gray level image matrix. It proceeds in two steps: In a first step, crack clusters are identified as connected components of pixels with pixel values below a given threshold value. Such a crack cluster is shown in Figure 2 a) for a zigzag crack visible in the upper part of Figure 1 a). To calculate the crack path, the pixels of a crack cluster are understood as vertices. Vertices that are directly adjacent are connected with edges, so a graph arises. Edges connecting vertices which coincide in one component have length $L = 1$. All other edges have length $L = \sqrt{2}$ since they connect vertices in diagonal direction. The longest shortest path through this graph is the identified crack path for this cluster. It can be determined with Dijkstra's shortest path algorithm in $O(n \cdot n \cdot log\, n)$ steps when the cluster has n vertices (see e.g. Cormen et al. [19], Chapter 24). The output of `crackrec` consists of three components. One component called `crackclusters` is a list containing K matrices with the pixel positions of the pixels of the K crack clusters which were found. The matrices are of size $2 \times n_k$, $k = 1, \ldots, K$, where n_k is the number of pixels in the k'th crack cluster. The second component called `crackpaths` is a list containing K $2 \times m_k$ matrices with pixel positions of the pixels of each crack path. The third component called `cracks` is a $6 \times K$ matrix where the six rows contain the lengths of the crack paths, the sizes of the crack clusters, and the x and y coordinates of the start and end points of the crack paths.

Simply using the length L of neighbor pixels leads to the curve given by triangles in Figure 3 a) or to the detected crack path of Figure 2 b). These crack paths are not following the darkest parts of the crack cluster and angles of kinked cracks are cut. Note that several longest shortest paths exist in Figure 3 for this situation since the cluster is a disk and every diameter of the disk and additional paths like that given by triangles in Figure 3 a) are longest shortest paths through the disk. To force the path to the darkest parts of the cluster, the pixel values $greyValue_u$ and $greyValue_v$ of the two neighbor pixels u and v were used with some weights additionally to the length L. Thereby the weights α and β are applied to the sum of these pixel values and to the absolute value of their difference, respectively. I.e.

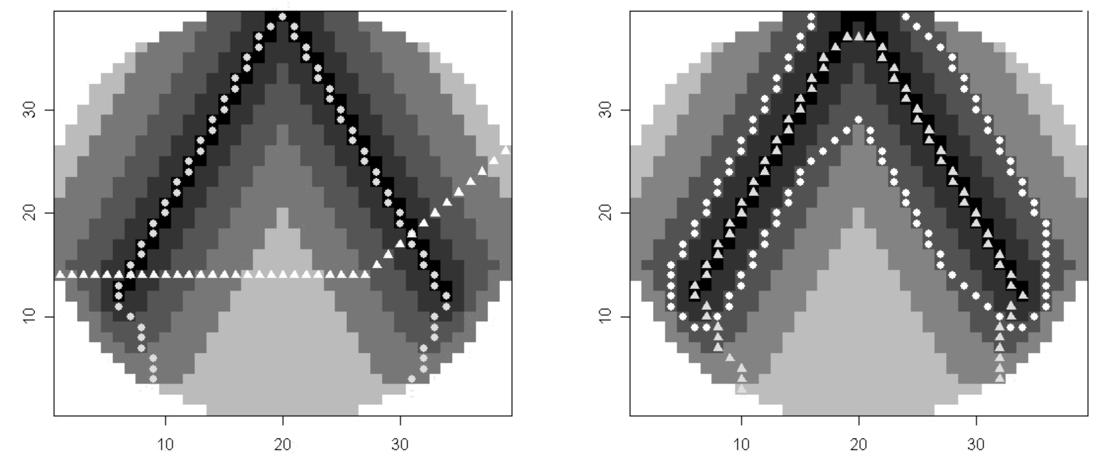


Figure 3: test image for the weights: a) the path given by $\alpha = 0$, $\beta = 0$ is marked with triangles, that given by $\alpha = 1$, $\beta = 1$ is marked by circles, b) the path given by $\alpha = 1$, $\beta = 0$ is marked by triangles, that given by $\alpha = 0$, $\beta = 1$ is marked with circles

the following distance was used

$$L * (1 + \alpha * (greyValue_u + greyValue_v) + \beta * |greyValue_u - greyValue_v|)$$

This distance was used to determine the shortest paths between points inside the crack cluster. However, the longest shortest path was calculated by using simply the optical length, i.e. with $\alpha = 0 = \beta$, to obtain the optical longest paths. The test image in Figure 3 shows the effect of the choices $\alpha = 0 = \beta$, $\alpha = 0, \beta = 1$, $\alpha = 1, \beta = 0$, $\alpha = 1 = \beta$ for the determination of shortest paths. The best result was obtained with $\alpha = 1 = \beta$ so we worked with this. But the choice $\alpha = 1, \beta = 0$ is also quite good so that it is most important to have high weight on the sum of the pixel values. Figure 2 c) shows the result for $\alpha = 1 = \beta$ for the zigzag crack.

The detected crack path now follows all angles and is not cutting them.

The recognized crack paths can be plotted with the function `crackplot`, either in the original image or separately. If they are plotted in the original image, the crack paths are plotted in red. For an improved identification of the start and end points of the cracks, they are connected with a yellow line. These are the default colors, but other colors can be chosen. Figure 1 b) shows the detected crack paths for the image in Figure 1 a), where 448 cracks with a size greater than 4 pixels were found.

3 Application of the method

We applied the new method to two series of images providing the surface of two small steel assays under strain. Although the surface of interest has only a size of 7×10 mm^2, it was too large to cover it with one microscopic photo. Hence only microscopic photos of segments of the surface could be obtained. The photos were taken at different points of time t, where t means that the photos were made after $t \cdot 1000$ load cycles. For one assay, Assay 31, 15 different points of time were considered, where photos of 54 segments were available for each point of time. For the other assay, Assay 10, 29 different points of time were used and photos of 45 segments were obtained at each point of time. Each image segment has 696×512 pixels. The segments have different quality. Some have shadows at the border and some are blurred. Since the segments overlap, they could be joined leading to images of the whole surface with 3337×4165 pixels for Assay 31 and images with 2659×4221 pixels for Assay 10. These images were compressed for a faster analysis to 1669×2083 and 1330×2111 images.

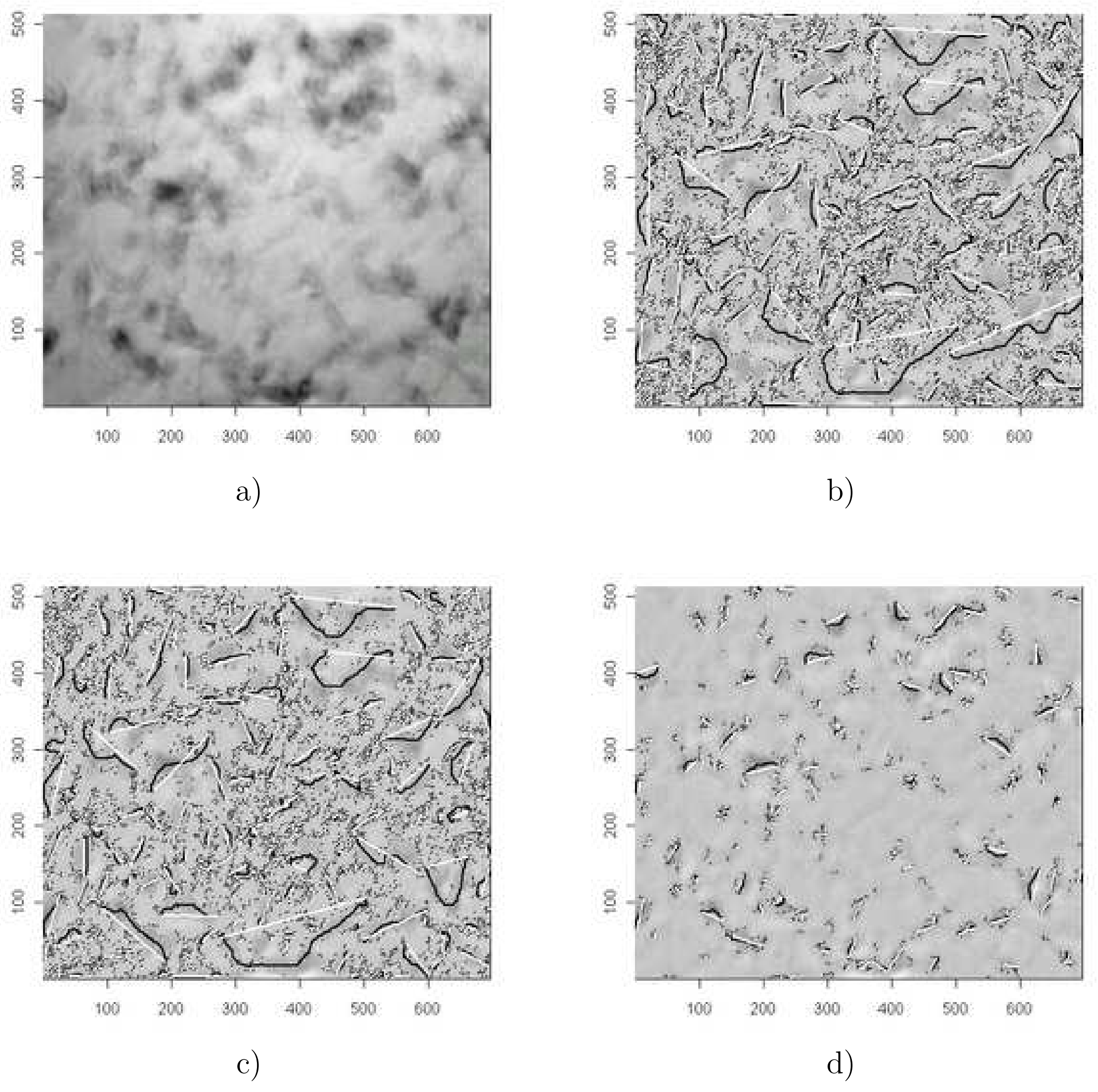
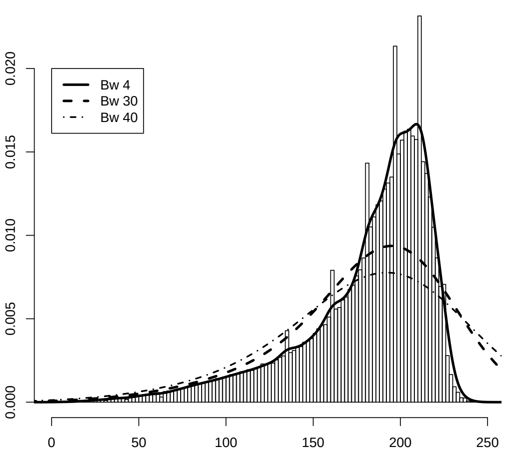


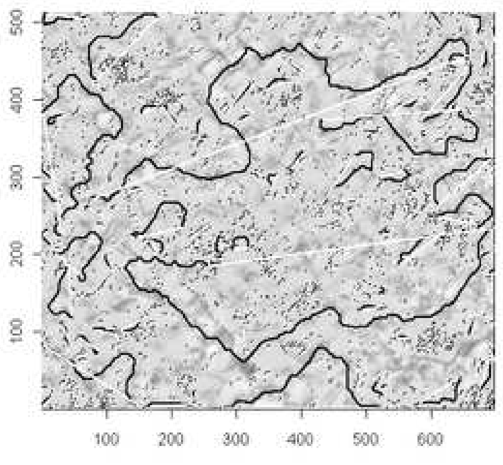
Figure 4: different threshold methods applied on the blurred image at $t = 6$ given in a): b) method of Otsu (threshold=130), c) method of Ridler/Calvard (threshold=128), d) method of steepest increase with bandwith 30 (threshold=105)

Crucial for the recognition method is a good threshold value. This should also be determined automatically. We tried several methods and found that a method proposed by Tsai [20] is most appropriate. Thereby the threshold is determined as the steepest increase of a smoothed histogram of the pixel values where the histogram is smoothed with the Gaussian kernel density estimator. The problem is that the histogram of the pixel values and thus also the density estimator is almost always unimodal so that classical methods like the method of Ridler and Calvard [21] and Otsu [22] are overestimating the threshold. The same holds if simply the mean gray level or more specialized threshold methods for unimodal and multimodal distributions as described in [23], [24], [25] are used. The method of steepest increase of the kernel density estimator is rather flexible since different bandwidths of the kernel density estimator can be used. The higher the bandwidth is the smoother the estimated density is. A smooth density has steepest increase at a point much smaller than the mode of the distribution. Hence by using a rather high bandwidth, we obtained the best result. This method is given by the function `threshold.msi`. Figure 4 provides a comparison of different threshold methods. A blurred image was used there, but similar results hold for the other images.

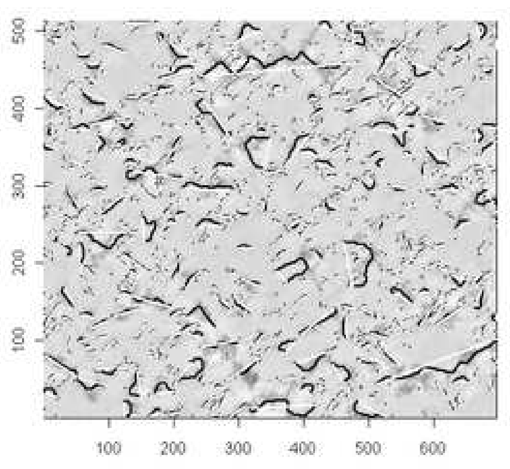
The best bandwidth depends on the method which is used for removing shadow. For removing the shadow we used the method based on the median filter (see e.g. [8]). We investigated a 51×51 and 201×201 window for the median filter. Figure 5 shows the results of three bandwidths for an image where the 51×51 median filter was applied. The best bandwidth is here 30. Note thereby that a small bandwidth, which provides the best approximation to the histogram of gray levels, leads to a much too high threshold value as seen in Figure 5 a) and b). A too large bandwidth like 40 is also bad since then the threshold is so small that cracks are separated as the zigzag crack in the upper part of Figure 5 d). Other images behaves similar so that we used for the 51×51 median filter always the bandwidth 30. For the 201×201 median filter, the bandwidth 50 appeared as appropriate bandwidth. This was applied in Figure 1 b).



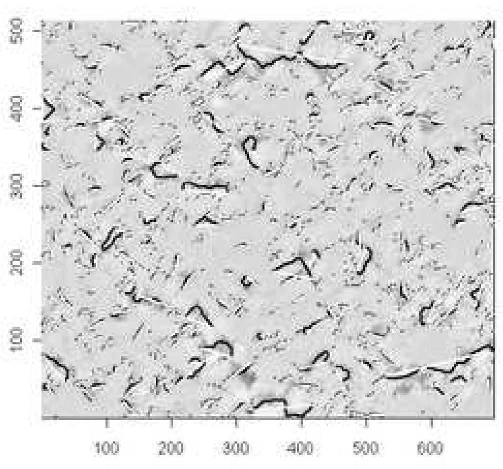
a)



b)



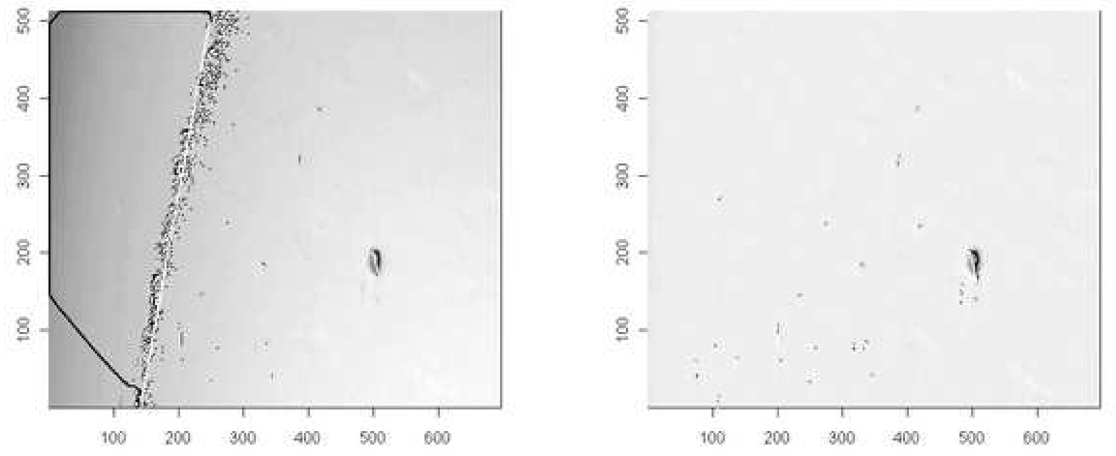
c)



d)

Figure 5: effect of different bandwidths on the image in Figure 1 a): a) histogram with different kernel density estimators, b) bandwidth = 4 (threshold=173), c) bandwidth = 30 (threshold=143), d) bandwidth = 40 (threshold=130)

That a method for removing shadow is necessary shows Figure 6 a). There, the shadow at the left border provides a very large crack cluster in which a very large crack is found. Figure 6 b) shows the result of `crackrec` after applying the 51×51 median filter. For both images, `threshold.msi` was used with bandwidth 30 leading to a threshold of 146 for the unfiltered image and a threshold of 188 for the filtered image. We also tested other bandwidths for the unfiltered image but the result was always unsatisfying.



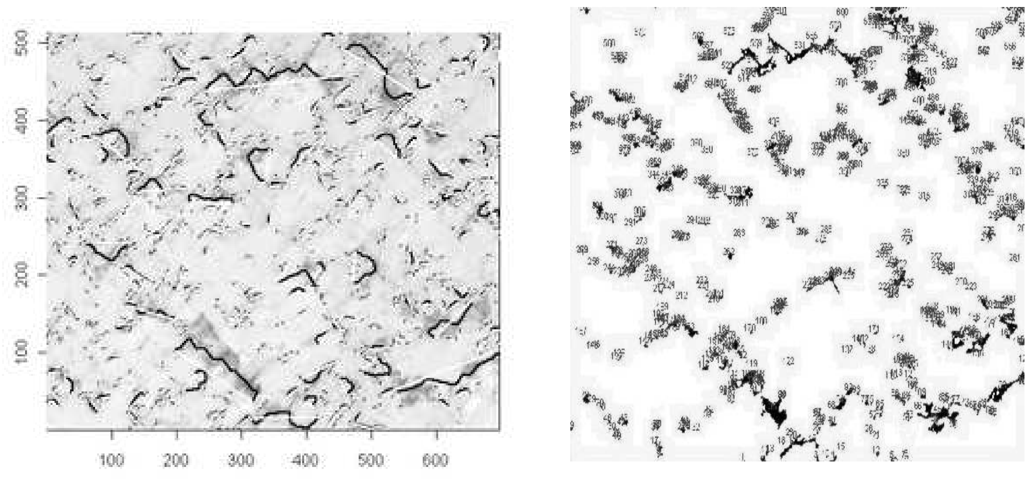
a) b)

Figure 6: effect of filtering: a) without median filter, b) with median filter

Since the shadow only appears at the border of the image segments, we also developed a special method `shadow.remove` for removing the shadow at the border. This method is based on the mean gray level $\overline{G}_B$ of a border strip with a width of 40 pixels and the mean $\overline{G}_I$ of all gray levels in the inner part. If the difference between $\overline{G}_B$ and $\overline{G}_I$ is greater than 10, then `shadow.remove` adds an exponential decreasing function f with maximum at 0 of the form

$$f(x) = a \exp\left(-\frac{x^2}{30W}\right) + \frac{a}{4}\exp\left(-\frac{x^2}{W}\right)$$

to the pixel values of the border. The quantity W is the width of the image in horizontal or vertical direction, respectively. The value a is given by $a = 40(\overline{G}_I - \overline{G}_B)/\beta$, where β is chosen such that the mean of the gray levels at the border strip becomes $\overline{G}_I$. A result using this method, where the threshold is the minimum of 140 and 4/5 of the mean of the gray levels, is shown in Figure 7 a). In this case, it is very similar to that in Figure 1 b) using the 201×201 median filter. But in Section 4 it is shown that there are some differences with respect to crack characteristics. However, this method can only be used for the image segments but not for the whole image consisting of the union of the image segments since the shadows are not only appearing at the border in this case. For the whole image, only the median filter is applicable. Here a median filter with a large window as a 201×201 window is preferable since otherwise cracks passing over several image segments are separated.



a) b)

Figure 7: detected cracks for the image given in Figure 1 a): a) using `shadow.remove` with min(140,4/5*mean(graylevel)), b) using UTHSCSA Image Tool

The computing time which is needed to calculate the detected cracks depends on the number of crack clusters and the size of the crack clusters. Hence the larger the threshold is, the more time is needed. The computing time is exceptionally high for Figure 6 a), where the shadow provides a very large crack cluster. However, the

Image	Size	Band-width	Threshold	Number of crack clusters	Maximum crack length in pixel	Computing time in sec
Fig. 1 a), $t = 0$	696×512	30	203	25	28,49	<0,1
Fig. 1 a), $t = 0$	696×512	40	193	22	24,07	0,1
Fig. 1 a), $t = 5$	696×512	30	179	748	89,18	1,4
Fig. 1 a), $t = 5$	696×512	40	169	543	70,70	0,8
Fig. 1 a), $t = 18$	696×512	30	143	1091	299,29	14,8
Fig. 1 a), $t = 18$	696×512	40	130	970	211,89	6,1
Fig. 6 a), $t = 0$	696×512	30	146	732	842,47	4060,9
Fig. 6 a), $t = 0$	696×512	40	131	549	727,49	1966,5
Fig. 6 b), $t = 0$	696×512	30	188	29	53,63	0,3
Fig. 6 b), $t = 0$	696×512	40	178	20	48,38	0,3
Assay 31, $t = 0$	1669×2083	50	144	388	113,43	0,9
Assay 31, $t = 5$	1669×2083	50	139	4842	94,25	1,5
Assay 31, $t = 18$	1669×2083	50	115	12449	262,55	16,6
Assay 10, $t = 0$	1330×2111	50	144	432	72,57	0,6
Assay 10, $t = 10$	1330×2111	50	139	3798	84,01	0,7
Assay 10, $t = 44$	1330×2111	50	114	7217	325,84	3,7

Table 1: computing times of user and system for different images

computation is rather fast in all other cases as Table 1 shows. Even the 1669×2083 image given as the union of 54 image segments and the 1330×2111 image given as the union of 45 image segments can be calculated in a rather short time at late points of time t ($t \mathrel{\widehat{=}} t \times 1000$ load cycles) where many cracks exists. The computing times in Table 1 were obtained using a 51×51 median filter for the image segments and a 101×101 median filter for the compressed union of image segments.

4 Comparison with UTHSCSA Image Tool

To compare the new package with another free package, namely the `UTHSCSA Image Tool`, the microscopic photos at 15 time points of Assay 31 described in Section 3 were used. Although some of the 54 image segments are blurred, all of them were used. The comparison was based on the numbers of cracks, the maximum crack lengths, the mean crack lengths, and the added crack lengths. Hence 4×54 quantities for each of the 15 points of time were calculated. While our package with standard `R` commands provides these quantities automatically, a lot of manual labor was necessary to obtain these quantities for `UTHSCSA Image Tool`. `UTHSCSA Image Tool` produces for each image only a text file with some quantities as number of cracks and lengths of the main axes of the ellipses circumscribing the cracks. These results for the 54 image sections had to be combined by hand for each point of time. There was no easy possibility of an automatic transfer of the results to a statistical package.

Moreover, there was no simple possibility to add the recognized cracks to the original image. The recognized cracks can only be visualized by a plot as shown in Figure 7 b). Another disadvantage of `UTHSCSA Image Tool` is that different computers provide different results. The results given below were produced by a Fujitsu Siemens Computer running Micrsoft Windows XP with NVIDIA GeForce FX 5700 Ultra.

Figure 8 a) shows the medians and the interquartile ranges of the numbers of detected cracks in the 54 image segments at the 15 time points. Only crack clusters with size greater than 4 were used. Figure 8 a) shows that the new package detects much more cracks than `UTHSCSA Image Tool` although the accuracy given by interquartile ranges is often similar. One explanation for this different behavior is that `UTHSCSA Image Tool` does not recognize pixel positions as connected which are only connected diagonally. It recognizes two pixel positions only as connected if either the x or the y component of the position differs by one. In `crackrec` both components can differ by one to provide a connection. This would lead to more crack clusters found by `UTHSCSA Image Tool`. But since only crack clusters were

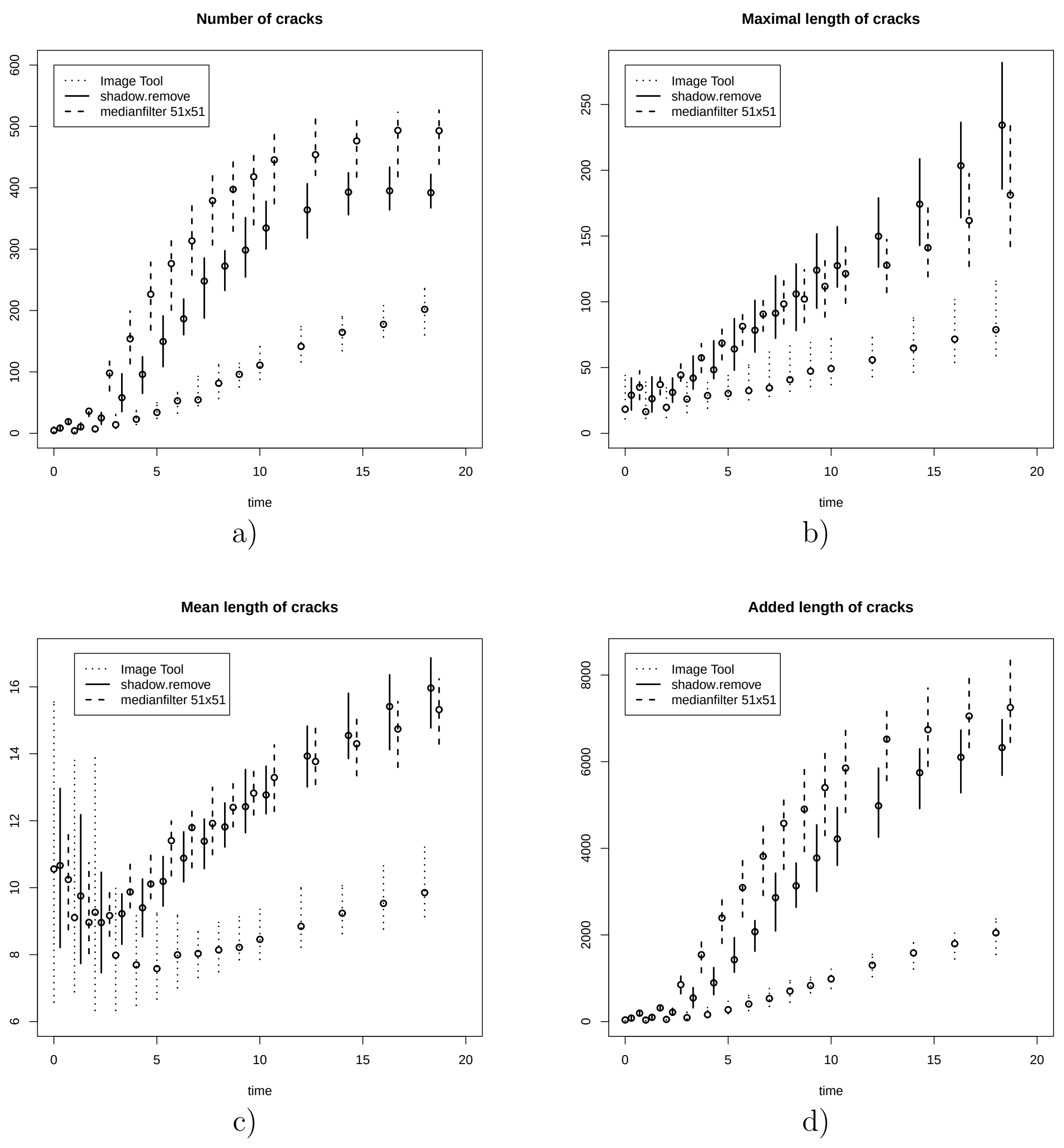


Figure 8: medians and interquartile ranges for crack numbers a), maximum crack lengths b), mean crack length c), and added crack lengths d) found by `crackrec` and `UTHSCSA Image Tool`

used which have more than 4 pixels and the majority of crack clusters is small (see also Figure 1), this leads to a significant smaller number of cracks found by `UTHSCSA Image Tool`. It is debatable which approach is better. But also `crackrec` has the problem that visually connected crack clusters are not connected. Hence it is better to have a less restrictive definition of connection.

Another reason for smaller numbers detected by `UTHSCSA Image Tool` is certainly a smaller threshold value which is automatically determined. Therefore, the zigzag crack in the upper part of Figure 1 a) is separated as seen in Figure 7 b).

Since it is more likely that `crackrec` recognizes two crack clusters as connected, it is not surprising that the recognized maximum crack lengths found by `crackrec` are larger than those found by `UTHSCSA Image Tool`. See Figure 8 b). Another reason for the larger lengths found by `crackrec` is that it uses the length of the crack path and not the difference between the start and end point while `UTHSCSA Image Tool` uses the straight line given by the main axis of the circumscribing ellipsis. The variability given by the interquartile ranges is similar at early points of time.

The variation produced by `crackrec` is much smaller, in particular for early points of time, if the mean crack length is used, see Figure 8 c). This Figure also shows that the mean lengths do not differ very much between the two packages. This holds although crack clusters which are connected with `crackrec` are not connected with `UTHSCSA Image Tool`. Note that the scale ranges here only between 6 and 16. However, it is not surprising that the added lengths given in Figure 8 d) are much larger for `crackrec` since much more cracks are found. This is also the reason that here the accuracy of `crackrec` is worse.

Figure 8 show that the difference between the two methods for removing the shadow is much smaller than the difference between the two packages. The variability between the image segments is very similar. Sometimes the method `shadow.remove` shows less variation. However, it finds slightly less crack clusters so that also the added crack lengths are a little bit smaller. Maybe this is the reason for slightly smaller variation. Using the 201×201 median filter with bandwidth 50, leads to results which are similar to those with the 51×51 median filter. However, the variation is sometimes larger due to a less effective shadow elimination at borders.

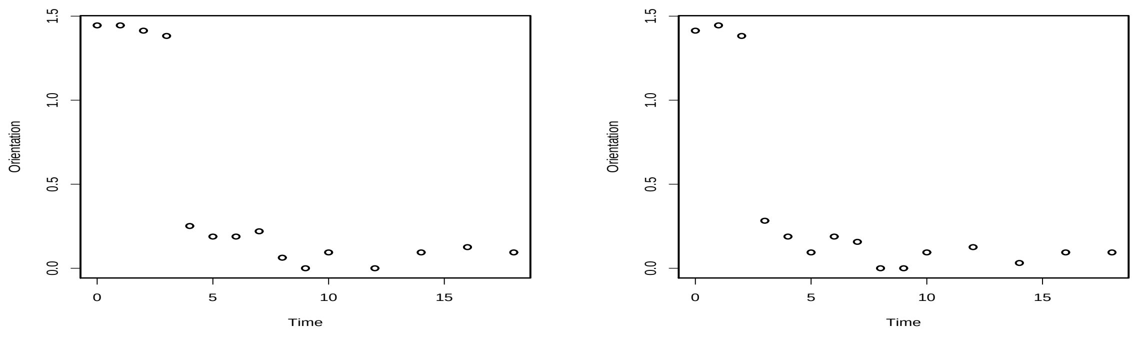


Figure 9: orientations of the cracks calculated by least squares method (left hand side) and by l_1 method (right hand side)

5 Statistical analysis of crack behavior

It is not surprising that with increasing time the number of cracks and the mean and maximum length of cracks increase. More specific hypotheses state that the number of initiated cracks follows a Poisson distribution, that initiated cracks have no specific orientation and that by-and-by the orientation becomes perpendicular to the load. The last hypothesis is supported by Figure 9: It shows clearly that the orientation of the cracks is close to zero, i.e. perpendicular to the load, from Time 3 or 4 on, depending on the method used for calculating the mean orientation. Here the least squares method and the l_1 method for calculating simple orthogonal regression were used. The regression analysis was based on the differences of the start and end points of the crack paths given by `cracks` in the function `crackrec`.

A more specific question is how often the cracks kink. This can only be analyzed using the crack paths. In particular the linear parts of the crack path can be determined. In Figure 2 d), the linear parts of the crack path shown in Figure 2 c) were plotted. They were determined as in Alt and Guibas [26] in Section 4 of Chapter 3 on shape simplifications and approximations. The mean of the orthogonal distances between the path points and the lines were used as distance measure. If this distance is less than a given precision value, then the path points can be approximated by one line segment. The minimum number of line segments with distance less than the precision value is determined again with a shortest path algorithm. If there are several solutions then the solution with minimum sum of distances is used. Figure 2 d) was obtained by setting the precision value equal to 2. This lead to 10 linear parts of the zigzag crack.

6 Discussions

Our package originally was aimed to provide an automatic method to determine covariates based on cracks which can be used for predicting the life time of the material. This purpose is satisfied quite well although no special preprocessing besides a shadow removement via the median filter or `shadow.remove` was used. In particular, we were able to formulate the hypothesis that the time where the crack orientation changes significantly to perpendicular direction to the load (see Figure 9) is a predictor of the life time. However, further experiments must verify this hypothesis. But analyzing more experiments can now be done easily with our package.

But our package is not only able to analyze the orientation of whole cracks. It can also analyze the linear parts of the cracks. Here it will be interesting to check whether the linear parts are the result of the growth of one crack or of the union of different cracks. This can be investigated by studying the temporal history of a crack. This will be the next extension of the package.

Our package bases heavily on the assumption that only one path describes the crack appropriately. For tree-like cracks as considered in Iyer and Sinha [10], the package must be extended to find all branches of the tree.

Acknowledgment. We thank Prof. Dr. Angelika Brückner-Foit, Michael Besel, and Frank Zeismann from the Institute for Materials Engineering of the University of Kassel for providing us the series of images and for the discussions about crack behavior. We are also grateful that Janine Keppler did all the manual labor for calculating the confidence intervals for UTHSCSA Image Tool and that she supported us in finding good threshold values.

References

[1] Nicholson, D. W., Ni, P., Ahn, Y.: Probabilistic theory for mixed mode fatigue crack growth in brittle plazes with random cracks. Engineering Fracture Mechanics 66, 305–320 (2000)

[2] Ihara, C., Tanaka, T.: A stochastic damage accumulation model for crack initiation in high-cycle fatigue. Fatigue Fract. Engng Mater. Struct. 23, 375–280 (2000)

[3] Meyer, S., Brückner-Foit, A., Möslang, A., Diegele, E.: Stochastic simulation of fatigue damage accumulation in a martensitic steel. Mater. wiss. Werkstofftech. 33, 275–279 (2002)

[4] Brückner-Foit, A., Meyer, S., Möslang, A.: A stochastic simulation model for microcracks in a martensitic steel. Comp. Mater. Sci. 26, 102–110 (2003)

[5] Heron, E. A., Walsh, C. D.: A continuous latent spatial model for crack initiation in bone cement. Appl. Statist. 57, 25–42 (2008)

[6] Chiquet, J., Limnios, N., Eid, M.: Piecewise deterministic Markov processes applied to fatigue crack growth modelling. J. Statist. Plann. Inference 139, 1657–1667 (2009)

[7] Jain, A. K.: *Fundamentals of Digital Image Processing*. Prentice-Hall, Upper Saddle River, NJ (1989)

[8] Burger, W., Burge, M.J.: *Digital Image Processing: An Algorithmic Introduction Using Java*. Springer, New York (2007)

[9] O'Gorman, L., Sammon, M. J., Seul, M.: *Practical Algorithms for Image Analysis with CD-ROM*. Cambridge University Press, Cambridge (2008)

[10] Iyer, S., Sinha, S. K.: A robust approach for automatic detection and segmentation of cracks in underground pipeline images. Image and Vision Computing 23, 921–933 (2005)

[11] Fujita, Y., Mitani, Y., Hamamoto, Y.: A method for crack detection on a concrete structure. Proceedings - International Conference on Pattern Recognition 3, 901–904 (2006)

[12] Purcell, D.: Automatic crack detection. Sensor Review 3, 130–131 (1983)

[13] Cheu, Y. F.: Automatic crack detection with computer vision and pattern recognition of magnetic particle indications. Materials Evaluation 42, 1506–1510, 1514 (1984)

[14] Buckley, M., Yang, J.: Regularised shortest-path extraction. Pattern Recogn. Lett. 18, 621–629 (1997)

[15] Fletcher, D.I., Franklin, F.J., Kapoor, A.: Image analysis to reveal crack development using a computer simulation of wear and rolling contact fatigue. Fatigue Fract. Engng Mater. Struct. 26, 957–967 (2003)

[16] Appleton, B., Sun, C.: Circular shortest paths by branch and bound. Pattern Recognition 36, 2513–2520 (2003)

[17] Russ, J. C.: *The Image Processing Handbook*. Taylor & Francis Ltd, London (2006)

[18] R Development Core Team: *R: A language and environment for statistical computing*. R Foundation for Statistical Computing, Vienna, Austria, (2009), `http://cran.r-project.org/`

[19] Cormen, T.H.,Leiserson, C.E., Rivest, R.L., Stein, C.: *Introduction to Algorithms*. MIT Press, Cambridge, (2001)

[20] Tsai, D. M.: A fast thresholding selection procedure for multimodal and unimodal histograms. Pattern Recogn. Lett. 16, 653–666 (1995)

[21] Ridler, T. W., Calvard. S.: Picture thresholding using an iterative selection method. IEEE Transaction on Systems, Man, and Cybernetics SMC-8, 630–632 (1978)

[22] Otsu, N.: A threshold selection method from gray-level histograms. IEEE Transaction on Systems, Man, and Cybernetics 9, 62–66 (1979)

[23] Rosin, P. L.: Unimodal thresholding. Pattern Recognition 34, 2083–2096 (2001)

[24] Sezgin, M., Sankur, B.: Survey over image thresholding techniques and quantitative performance evaluation. Journal of Electronic Imaging 13, 146–165 (2004)

[25] Medina-Carnicer, R., Madrid-Cuevas, F. J.: Unimodal thresholding for edge detection. Pattern Recognition 41, 2337–2346 (2008)

[26] Alt, H., Guibas, L. J.: Discrete geometric shapes: matching, interpolation, and approximation. In: Handbook of Computational Geometry, eds. J.-R. Sack, J. Urrutia, Elsevier Science, 121–153 (1999)

Corresponding Author:

Christine Müller

University of Technology Dortmund, Faculty of Statistics

Vogelpothsweg 87

D-44221 Dortmund, Germany

E-mail: cmueller@statistik.tu-dortmund.de

Phone: +49 (0) 231 755 - 4238

Fax: +49 (0) 231 755 - 3454

Homepage: `http://www.statistik.tu-dortmund.de/mueller1.html`