Methods for scale and orientation invariant analysis of lower dimensional structures in 3d images

Tin Barisin

Vom Fachbereich Mathematik
der Rheinland-Pfälzischen Technischen Universität Kaiserslautern-Landau
zur Verleihung des akademischen Grades
Doktor der Naturwissenschaften
(Doctor rerum naturalium, Dr. rer. nat.)
genehmigte Dissertation

D 386

1. Gutacherin:	Prof. Dr. Claudia Redenbach
	Rheinland-Pfälzische Technische Universität Kaiserslautern-Landau
2. Gutacherin:	Prof. Dr. Gabriele Steidl
	Technische Universität Berlin

Datum der Disputation: 07.06.2023

Acknowledgements

This thesis was supported by the German Federal Ministry of Education and Research (BMBF) [grant number 05M2020 (DAnoBi)].

First, I wish to thank my colleagues from the Image Processing Department of Fraunhofer ITWM, where I spent these three wonderful years doing research for this thesis. Special thanks go to my ITWM advisor Katja Schladitz who shared her research ideas and resources with me, always helped me resolve all of my doubts and technical problems, and was always extremely kind to me. Here, I also acknowledge Ali Moghiseh and Michael Godehardt for all the technical support related to MAVI and ToolIP.

Next, I wish to thank the current and past members of the AG Statistics Group from the Department of Mathematics at RPTU. Here, special thanks go to my academic advisor Claudia Redenbach who helped me shape my research ideas and manuscripts, always offered great pieces of advice, and gave me continuous support and mentorship during the past three years. Thank you for all the patience and kindness.

I thank and acknowledge Jesus Angulo from CMM, Paris Mines for the research stay in Fontainebleau, France. Thank you for all the interesting and inspiring discussions.

Furthermore, I thank all people who actively cooperated with me during this period: Christian Jung, Anna Nowacka, Franziska Müsebeck, and Szymon Grzesiak.

During this time I met many great people who have become my dear friends. My thanks go to you as well: Lovro Bosnar, Alexander Geng, Alex Keilmann, Tessa Nogatz, and so on.

I wish to thank my family for their continuous love, patience, support, and kindness. Thank you: Anita, Jurica, and Marijan!

Final thanks go to my girlfriend Lakshmi for making my life easier and happier, and for all the love, patience, and perseverance during the whole period, especially at the end.

Contents

Introduction

This thesis is based on motivation from two groups of scientific fields: engineering and mathematics. It consists of theoretical results which were highly motivated and supported from the applied perspective.

On the one hand, engineering sciences such as civil engineering and materials science want to design sustainable and cost-effective materials with desirable mechanical properties. For that purpose, computational models which estimate mechanical properties from microstructural properties and production parameters are made. Hence, microstructural properties are characterized from real samples. In our case, computed tomography (CT) is used to non-destructively gain insight into the material microstructure. This results in large 3d images which are usually processed by means of algorithms or methods from the fields of image processing and computer vision.

On the other hand, mathematics is focused on algorithm and method design and its properties. Here, the image is interpreted as a composition of different objects or structures. In many cases, objects are anisotropic and occur at different scales. Hence, scale and orientation combined with object position in the image convey important information related to the understanding of the image. For automatic (non-manual) analysis of the object, object features are extracted and used for the characterization of the class to which the object belongs. The natural assumption of human vision is that we can extract these features regardless of object position, orientation, or scale. This assumption is formalized through the concepts of equivariance and invariance. In the literature, translation invariance, i.e. invariance to the position of the object, in the image is well-studied in contrast to the orientation and scale invariance. For these reasons, Part I deals with the directional filtering for various types of materials, while Part II focuses on the scale in the context of crack segmentation.

The goal of this thesis is to bring these engineering and mathematics closer and to design methods which satisfy mathematical formalism and rigour, and help engineers practically process the large amounts of data that they nowadays produce. Next, we give a more detailed overview of the thesis content and highlight its main contributions.

In Part I, we deal with lower dimensional and directed structures in engineering materials such as cracks, fibers, or closed facets in foams. The characterization of such structures in 3d is of particular interest in various applications in materials science. In image processing, knowledge of the local structure orientation can be used for structure enhancement, directional filtering, segmentation, or separation of interacting structures. The idea of using banks of directed structuring elements or filters parameterized by a discrete subset of the orientation space is proven to be effective for these tasks in 2d. However, this class of methods is prohibitive in 3d due to the high computational burden of filtering on a sufficiently fine discretization of the unit sphere. Our first contribution of Part I is a method for 3d voxelwise orientation estimation and directional filtering inspired by the idea of adaptive refinement in discretized settings (see [1] and Chapter 2). Furthermore, an operator for the distinction between isotropic and anisotropic

structures is defined based on our method. This operator utilizes orientation information to successfully preserve structures with one or two dominant dimensions. Finally, the feasibility and effectiveness of the method are demonstrated on 3d micro-computed tomography images in three use cases: detection of a misaligned region in a fiber-reinforced polymer, segmentation of cracks in concrete, and separation of facets and strut system in partially closed foams. As a second contribution of Part I, by varying the length of the line structuring element, our method is used to construct granulometry in the sense of mathematical morphology and characterize fiber length and orientation distributions in fiber reinforced polymers produced by either 3d printing or by injection moulding (see [2] and Chapter 3).

In Part II, we investigate possible applications of the Riesz transform as a feature extractor motivated by the need to achieve scale invariance of neural networks. In convolutional neural networks, scale invariance is typically achieved by data augmentation. However, when presented with a scale far outside the range covered by the training set, neural networks may fail to generalize. As a first contribution of Part II, we introduce the Riesz network, a novel scale invariant neural network (see [3] and Chapter 6). Instead of standard 2d or 3d convolutions for combining spatial information, the Riesz network is based on the Riesz transform which is a scale equivariant operation. As a consequence, this network naturally generalizes to unseen or even arbitrary scales in a single forward pass. As an application example, we consider detecting and segmenting cracks in tomographic images of concrete. In this context, 'scale' refers to the crack thickness which may vary strongly even within the same sample. To prove its scale invariance, the Riesz network is trained on one fixed crack width. We then validate its performance in segmenting simulated and real tomographic images featuring a wide range of crack widths.
Furthermore, scattering networks have shown how to design powerful and robust hierarchical image descriptors which do not require a lengthy training procedure and which work well with very few training examples. The scattering network also represents a simple mathematical model proxy of neural network. However, scattering networks rely on sampling of scale dimension and hence become sensitive to scale variations and are unable to generalize to unseen scales. As a second contribution of Part II, we define an alternative feature representation based on the Riesz transform (see [4] and Chapter 8). It inherits the property of scale equivariance from the Riesz transform and completely avoids sampling of the scale dimension. Mathematical foundations behind this representation are laid out and analyzed. This representation is compared to the scattering network with 4 times more features and performs comparably well on texture classification with an interesting addition: scale equivariance. Our method gives superior performance when dealing with scales out of the training set distribution. The usefulness of this property is further shown on the digit classification task, where accuracy remained stable even for scales that are 4 times larger than the scale in the training set.

Part I

3d adaptive morphology and applications

Chapter 1

Introduction

In the past few decades, micro-computed tomography (μCT) has gained momentum in materials science and engineering for the purpose of detailed and fully 3d investigation of materials' micro-structures. Materials of interest include concrete, fiber-reinforced materials, foams, woven, and non-woven materials. Characterization of the geometric micro-structure enables understanding and modelling of the material structure and gives a unique insight into the connection of parameters of the production process and the resulting material's properties.

Many materials have a highly anisotropic structure and contain lower dimensional or directed features. In these cases, classical image processing methods such as noise reduction filters should be used carefully to preserve directional information as well as thin lower dimensional features. Moreover, nowadays image sizes well above $2\,000^3$ pixels have become common. Therefore, efficient image processing techniques that can handle this massive amount of 3d data are needed.

Many methods in image processing have been developed for the 2d case and extended to 3d. Hence, 2d image processing methods often serve as an inspiration for methods in 3d. In 2d, robust smoothing, segmentation, and analysis of fibrous structures is achieved by rotated anisotropic Gaussian filters whose main axis orientation is evenly sampled on the unit semi-circle [5, 6]. The optimal orientation is then chosen pixelwise. However, Wirjadi [7] shows that in 3d, this approach becomes computationally too intensive as the number of orientations to be checked to sample as densely as in 2d increases enormously. The idea of filtering using a fine sampling of the orientation space can nevertheless be utilized in 3d. To this end, we reduce the computational burden by checking only a subset of all orientations (Figure 1.1). This subset is adapted to the current pixel based on orientation information derived from its local neighbourhood. This idea of adaptive filtering around local orientation serves as a central building block of two methods developed in Part I.

Chapter 2 has a focus on directional filtering and orientation analysis. The goal here is to design a general method based on the idea of adaptive filtering which can efficiently remove noise and distinguish structures in 3d CT images based on their dimensionality. This method is proven to be useful for analysing various types of materials: fiber-reinforced composites, concrete, and partially closed foams. This chapter is based on [1].

Chapter 3 shows an application only for fiber-reinforced composites. Here, we extend the method from Chapter 2 to measure fiber length voxelwise using the concept of granulometry from the field of mathematical morphology. Using voxelwise length information and local orientation, the number-weighted fiber length and orientation distributions can be estimated. We apply this method for the comparison of microstructural properties of 3d printed and injection moulded fiber-reinforced composites. This chapter is based on [2].

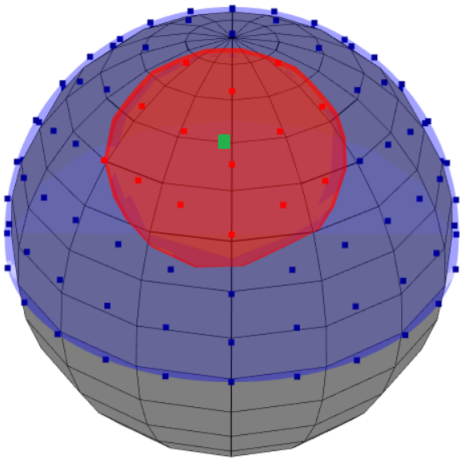


Figure 1.1: Example of a search cone (red) around $(\theta, \varphi) = (2, 2)$ (green, in spherical coordinates, units: radians) on the unit half-sphere (blue) and their discretized counterparts. This depicts the idea behind adaptive filtering: filter only on the orientations (red points) around the estimated orientation (green point) and avoid filtering on the remainder of the orientation space (blue points). The parameters are $\delta_{max} = 0.5$ and $n = 24$, see Section 2.3 for their interpretation.

Chapter 2

3d adaptive morphology

In this chapter, we introduce an adaptive framework for directional filtering and orientation analysis in 3d. Our method requires an initial orientation estimation in each voxel to start with. Filtering is then performed only on orientations close to the given initial orientation (Figure 1.1). This drastically reduces the run-time compared to classical directional filter banks which search the full evenly sampled orientation space.

In 3d, lower dimensional structures may be either linear (1d) or planar (2d). To be able to treat both cases, we use two classes of structuring elements (SEs): line segments and squares. We estimate the initial orientation by the established method based on the eigenvectors of the Hessian matrix or more roughly by probing the main directions (main axes, plane, and space diagonals) induced by the voxel grid and choosing one of them. Furthermore, following an idea of [8], we derive a local structure shape operator by comparing filter responses with varying SE in orthogonal spaces. Finally, we apply our method to three tasks in the analysis of materials structures based on computed tomography (CT): segmenting misaligned regions in a polymer reinforced with long glass fibers, segmentation of cracks in concrete, and facet identification in ceramic foams.

To summarize, we devise a method for fast and yet effective directional filtering in 3d and an operator for characterizing local structure shape. Both prove to be useful for analyzing materials structures highly relevant in practice.

2.1 Connections to adaptive mathematical morphology and filter banks

Our approach combines ideas from two fields of classical image processing: filter steerability or filter banks, and adaptive mathematical morphology. So far, these methods were mostly developed for and applied to 2d images. In 2d, directional filter banks are based on the principle of filtering in a set of orientations evenly-sampled on the unit semi-circle and selecting an orientation based on the maximal/minimal filter response [6, 9, 10, 11, 12, 13]. This class of methods has been applied to the tasks of filtering, segmentation, and orientation analysis.

In 2d, an even sampling of the unit semi-circle is easily obtained by choosing equidistant points on the interval $[0, \pi)$. In 3d, sampling of the unit semi-circle is replaced by sampling on the unit half-sphere [8, 14, 15]. Achieving an even sampling is not a trivial task [5, 14]. Sets of exactly equidistant points on the unit sphere exist only for certain numbers of points [16], with the simplest examples being $N = 2$ and $N = 6$. For other numbers of points, sets of

approximately equidistant points can be computed using optimization methods. See [5, 17] for overviews on this topic. Wirjadi [5, 7] adapted the optimization method from [18] to the upper half-sphere.

Classical mathematical morphology uses fixed structuring elements (SEs) for filtering (e.g. dilation, erosion, opening, or closure). Adaptive morphology expands these concepts by varying shape and/or size of the SE in each pixel location depending on local image information. Most common features used for selecting adaptive SEs are local gray value differences [19, 20], local orientation [21, 22], local structure tensor [23], salience [24], or local path alignment with the image structure [25, 26, 27].

Our approach is closely related to [23] who construct elliptical SEs whose axis orientations and sizes are derived from the eigenvectors and eigenvalues of the local structure tensor. Alternatively, directional information can be determined by the analysis of the Hessian matrix in each pixel. The extracted directions have been used for directional line filtering [21, 22], adaptive morphological filtering (erosion or dilation with linear SE) or anisotropic Gaussian filtering for enhancement of linear structures [28].

Many approaches in adaptive mathematical morphology rely on constructing complex SEs which in 3d result in enormous run-times and require significant computational resources [29]. Hence, 3d applications of adaptive morphology are still rare, with a few exceptions [19, 22, 26].

2.2 Preliminaries

Let $D \subset \mathbb{Z}^3$ be a discrete grid. Then, $\mathscr{L}_D = \{I | I : D \to \mathbb{R}^+,\ x \mapsto I(x) \in \mathbb{R}^+\}$ is the family of all mappings from the grid D to the real non-negative numbers. An element $I \in \mathscr{L}_D$ is called an image.

Most morphological operations and image filters require the choice of a SE (or filter mask). A SE B is defined as a subset of $\mathbb{Z}^3$. Here, we only consider reflection invariant SEs ($x \in B \iff -x \in B$) with $0 \in B$, namely line segments and squares. For any pixel (lattice point) $p \in D$, $B(p) = \{m \in D | m - p \in B\}$ refers to a copy of B translated to the pixel p. Then, some filter $\gamma_B : \mathscr{L}_D \to \mathscr{L}_D$ is defined as

$$\gamma_B(I)(p) = \bigotimes \{I(m) \,|\, m \in B(p)\},$$

where $\bigotimes$ represents the filtering operation. For instance, computing the mean or median of the gray values yields a mean or median filter while computing the minimum or maximum yields an erosion or dilation, respectively.

Line SEs of a given length can uniquely be parametrized by their orientation. Square SEs will be parametrized by their normal orientation. However, squares are not invariant with respect to rotations around the normal orientation. To obtain a unique parametrization, we choose one of the square's edges to be contained in the intersection of the xy plane and the plane with the desired normal orientation. The center of the square is then shifted to the origin to ensure reflection invariance. In the special case that the normal orientation is the z axis, the square is aligned to the x and y axes with center in the origin. Line SEs are discretized using the Bresenham line algorithm [30]. Starting from the center pixel, the line is discretized only in one direction and reflected to ensure symmetry of the SE. In this way, the line is sampled in a translation invariant way but efficient implementation[1] is not possible. For detailed discussion on line discretization, we refer to [31]. Square SEs are constructed by discretizing one of their edges and shifting it along the orthogonal edge.

[1] We refer to the efficient implementation from [10] whose computation time is constant and independent of line length. It is based on exploiting periodicity in the line discretization.

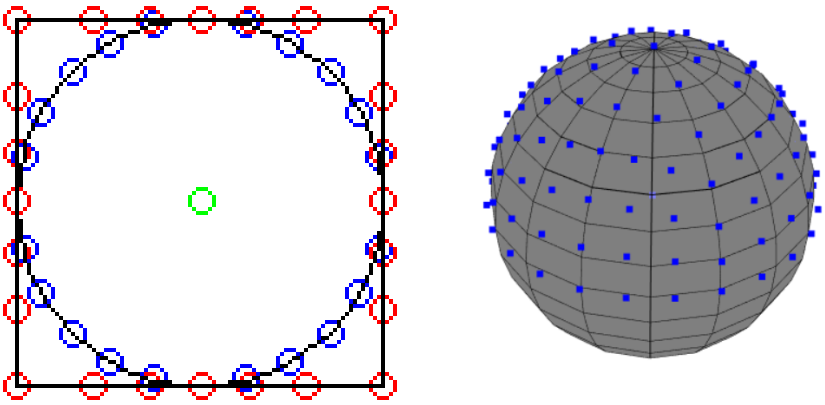


Figure 2.1: Discretization of S^2_+ (right) by extrapolation from the unit circle in 2D (left) for $n = 24$. Blue dots represent the uniform discretizations of the circle (left) or the sphere (right), while red dots (left) are projections of points from the unit circle to the unit square.

The unit sphere in $\mathbb{R}^3$ will be denoted by S^2. Due to the symmetry of the SEs, orientations on the unit half-sphere $S^2_+ := \{u = (u_1, u_2, u_3) \in \mathbb{R}^3 : ||u|| = 1, u_3 \geq 0\}$ are sufficient for the parametrization. In practice, only finitely many orientations can be considered which requires an even sampling of the orientations on S^2_+ to avoid introducing systematic errors. We follow the approach of [17] which exploits analogies from the 2D case to compute fast and easily an approximately even sampling on S^2_+ without complicated optimization. The idea is to evenly discretize the unit circle with n points which are then projected onto the unit square. Applying this procedure in the three coordinate planes (xy, yz, and xz) yields a sampling on the unit cube. The sampling points are then projected on the unit sphere, see Figure 2.1, and finally restricted to S^2_+.

The parameter n should be a multiple of 8 to ensure that the coordinate axes and diagonals are contained in the sampling. The total number of sampled points on S^2_+ is $N = n^2/8 + (n/4+1)^2$.

2.2.1 Estimating input orientations

Our method for directional filtering and orientation estimation from Section 2.3 requires roughly estimated local orientations as input. We suggest two ways to obtain this input.

Local orientation from Hessian matrix

An established approach for estimating orientation in 3d images is by analysis of the eigenvalues and eigenvectors of the Hessian matrix [32]

$$H_\sigma = \left(\nabla^2\right)(I * g_\sigma), \tag{2.1}$$

see e.g. [7]. Here, $g_\sigma : D \to \mathbb{R}^+$, $g_\sigma(x) = \frac{1}{(2\pi\sigma^2)^{3/2}} e^{-\frac{||x||^2}{2\sigma^2}}$ is a Gaussian kernel with standard deviation parameter σ. The parameter σ affects the scale of the observation and should be chosen depending on local structure thickness.

Throughout this chapter, the objects of interest will be bright structures on dark background. In this case, the eigenvalues of high absolute value have a negative sign [33]. Thus the largest eigenvalue is in fact the smallest in absolute value and vice versa. Hence, the eigenvector of the Hessian matrix H_σ with the largest eigenvalue represents the direction of the smallest change of

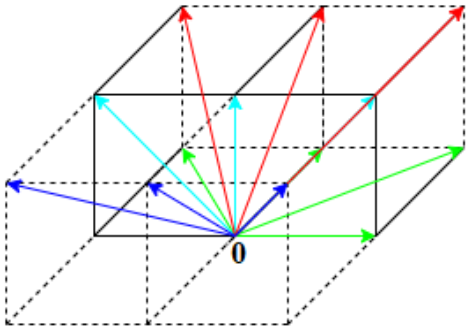


Figure 2.2: Base axis search. The 13 orientations from pixel 0 on the upper half-sphere: 4 green in the xy plane, 3 red in the first diagonal plane, 3 cyan in the xz plane, 3 blue in the second diagonal plane.

gray values and is used to estimate the local structure orientation in the image when using a line segment as SE. When using the square SE, we use the eigenvector corresponding to the smallest eigenvalue to estimate the normal direction, as this vector represents the orientation yielding the strongest change of gray values.

Another established approach for this task is to use the structure tensor [7, 34], which is based on the first order derivatives:

$$S_{\sigma,\rho} = g_\rho * ((\nabla f * g_\sigma)(\nabla f * g_\sigma))^T,$$

where ρ is an additional spatial smoothing parameter on the tensor space. The orientation is then estimated from the eigenvectors as done for the Hessian matrix.

Base axis search

In some cases, local orientation cannot be estimated well via the Hessian matrix, for instance due to poor contrast, noise, imaging artifacts or interaction between structures (as for example in fiber-reinforced materials with high fiber volume fraction). In this case, we consider a small subset of orientations and use the orientation of maximal response as input. For example, the subset consisting of the orientations of the edges, face and space diagonals of the lattice's unit cell yields a test set consisting of 13 orientations (Figure 2.2).

This way of initial orientation estimation may appear rather rough. However, it will be refined and adapted to the local image structure by the subsequent adaptive filtering.

2.3 3d adaptive line morphology

We aim at efficient and robust directional filtering and orientation estimation in large 3D images. Our main strategy is fine filtering adapted to a given local input orientation. This way, computationally expensive filtering on all sampled orientations can be avoided.

In practice, we assume that each image pixel is assigned a rough estimate of the local orientation of the image content obtained by one of the methods described above. Input images for the adaptive filter are thus an original image $I \in \mathscr{L}_D$, and an image of the same size as I containing input orientations $v(p) \in S^2_+$ for each pixel $p \in I$.

We then consider filter banks with line segment or square SEs that are uniquely parametrized by the line orientation or square normal, respectively. For selecting a subset of filter orientations from the sphere discretization, we define a proximity measure on the parameter space and filter

on the much smaller subset of orientations that are close to the given input orientation. Due to the restriction to a subset, we can afford to sample the subspace of the parameter space very finely. This allows to align the SEs precisely to the local image content. We measure proximity by the angular distance between the input orientation in a pixel and the orientations of the SE. That is, for two points $u, w \in S^2_+$, we set

$$d(u, w) = \arccos(|u \cdot w|), \tag{2.2}$$

where "$\cdot$" denotes the standard scalar product.
Let $S = \{u_1, ..., u_N\}$ be an even sampling of the unit half-sphere. Then, for every pixel $p \in I$ with input orientation $v(p) \in S^2_+$ we define the search cone for $\delta_{max} > 0$ as

$$C(p) = \{u \in S : d(u, v(p)) \leq \delta_{max}\}, \tag{2.3}$$

see Figure 1.1 for an illustration.

For line or square SEs of half-length L parametrized by orientation $u \in S^2_+$, the filtering operation is denoted as $\gamma_{u,SE} : \mathscr{L}_D \to \mathscr{L}_D$. Here, $SE \in \{\ell_L, s_L\}$ represents the type of structuring element (ℓ line segment or s square) used for filtering. For every pixel p we then report the maximal filter response on its search cone $C(p)$ and the corresponding orientation by setting

$$\Gamma^{SE}_{max}(I)(p) = \max_{u \in C(p)} [\gamma_{u,SE}(I)(p)], \tag{2.4}$$

and

$$\Gamma^{SE}_{arg}(I)(p) = \arg\max_{u \in C(p)} [\gamma_{u,SE}(I)(p)]. \tag{2.5}$$

Note that the correct full notation would be $\Gamma^{SE}_{max}(I, v)$ for input orientation image $v : D \to S^2_+$. We restrict to $\Gamma^{SE}_{max}(I)$ for the sake of easier readability. In summary, parameters of the method are the sampling size N of the unit (half-)sphere, the bound δ_{max} on the angular distance defining the size of the search cone, and the half-length of the SE L (half edge length in case of a square) measured in the maximum (ℓ^∞) metric[2].

2.3.1 Operator for local shape characterization

Identification and characterization of lower dimensional structures is needed in many applications. For example, fibers are locally one-dimensional substructures of the image while cracks are often locally planar. To formalize this, we differentiate three types of structures in 3d images based on their dimension following [33]: blob-like (3d), plate-like (2d), and tubular (1d).

Information on the local dimension or shape can be obtained by comparing filter responses with line and square as SE in orthogonal spaces. To be more precise, in a pixel p, filter responses of a square filter with normal u and a line segment filter oriented in u are compared via

$$R_{aniso}(p) = |\,[\gamma_{u,\ell_L}(I)](p) \; - [\gamma_{u,s_L}(I)](p)\,|. \tag{2.6}$$

Note that the operator is actually a function of the input image I and the orientation image, but again we opt for the more compact notation omitting the orientation dependency.
If the structure is isotropic, filter responses in orthogonal directions should be similar and hence $R_{aniso}(p)$ tends to be low. On the other side, if the structure is anisotropic, filter responses tend

[2]Maximum or Chebyshev distance metric here refers to the following distance function $d_\infty : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}_+$ defined as $d_\infty(a, b) = \max_{i=1,\cdots,d} |a_i - b_i|$. The reason why we prefer maximum distance over Eucledian distance is that we want to filter on lines that have approximately the same number of voxels.

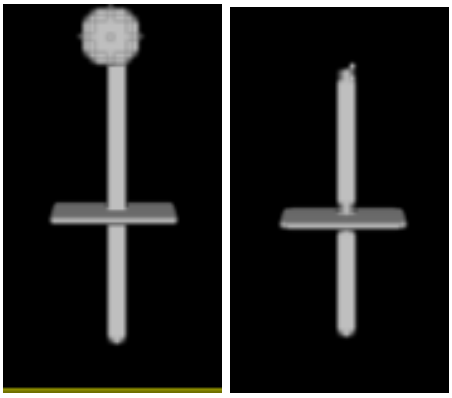


Figure 2.3: Example: Removing a 3D ball from a 2D plate and a 1D fiber from a binary image by using the shape operator R_{aniso} as defined in (2.6) with $SE = \ell$ with γ_{u,ℓ_L} being a median filter.

to differ because line and plate cannot be fitted to the structure at the same time. Hence, high values of $R_{aniso}(p)$ indicate local anisotropy.

The orientation $u = \Gamma^{SE}_{arg}(I)(p)$ is determined by equation (2.5) with SE chosen depending on the goal of the analysis: If $SE = s$, then R_{aniso} distinguishes 2d from 3d structures. If $SE = \ell$, then R_{aniso} differentiates 3D structures on the one hand and 1d/2d structures jointly on the other hand, see Figure 2.3 for a toy example.

Junctions, where two or more oriented structures meet as in Figure 2.3 and Figure 2.7, are hard to classify based on local orientation or anisotropy as the superposition may no longer feature a clearly preferred orientation. There are several works in the literature [8, 17, 28] that propose ways to deal with junctions when segmenting vessels. In 2d, Su et al [28] detect junction points in a post-processing step and handle them by a tailor-suited filter. That results in increased computational burden and run-time. Altendorf [17] detects junctions, removes them, and finally reconnects fibers based on local orientation. Sazak et al [8] achieve correct junction handling in 3D, mostly thanks to covering the full sampling space on several scales.

We adapt the operator R_{aniso} from equation (2.6) for improved junction handling as

$$R^*_{aniso}(p) = |\Gamma^{s_L}_{max}(I)(p) - (\Gamma^{\ell_L}_{max}(I^c))^c(p)|, \tag{2.7}$$

where $\Gamma^{\ell_L}_{max}(I^c)$ represents filtering with $SE = \ell$ on I^c i.e. on the inverted or negative image of I with $\Gamma^{s}_{arg}(I)$ as an input orientation and $\delta_{max} = 0.5$. This operator R^*_{aniso} is well suited for detecting junctions of locally plate-like structures as in Figure 2.7, comes however also at the cost of increased computational effort.

We give a short intuition behind this operator on crack junction example (Figure 2.7). The operator $\Gamma^{s_L}_{max}(I)(p)$ adjusts the plane normal using our adaptive approach to one of the cracks in the junction. Hence, the filter response here should be high. When filtering with a line on the inverse image with $\Gamma^{\ell_L}_{max}(I^c)$, adaptive line filtering has the highest filter response when the line is contained within the concrete. The same would apply to single cracks. After inverting back the filter response we get the lowest filter response and hence the difference between the two operators is high in the junctions. When the same operations are performed on a 3d (spherical) object, R^*_{aniso} will still remain low because of the isotropic local neighbourhood. Hence, this operator is able to distinguish between intrinsically 3d objects and intersections between two overlapping 2d objects.

Method	Time (s)
Our	45.3
Base	166.7
Base (efficient)	75.5

Table 2.1: Run-time comparison of our method with brute-force approach and efficient algorithm of [10] for parameters $L = 5$, $n = 16$, $\delta_{max} = 0.5$, and $SE = \ell$ applied to simulated crack image of size 256^3.

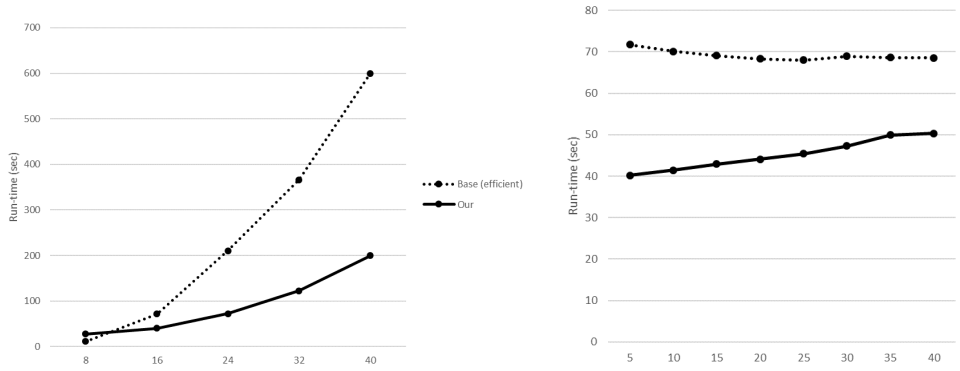


Figure 2.4: Run-time analysis for varying parameter configurations on the image from Table 2.1. Left: varying discretization parameter n for $\delta_{max} = 0.5$ and $L = 5$, right: varying half-length parameter L for $\delta_{max} = 0.5$ and $n = 16$. Here, the parameter n refers to the number of sampled points on the intersection of unit sphere and xy plane in contrast to the total number of sampled points on the sphere N, as described in Preliminaries.

2.3.2 Improved efficiency

Here, we argue why our approach is more efficient than filter banks using all evenly sampled points on S^2_+. For sampling N orientations, the filter bank would run N filters per pixel. For our adaptive filtering, the number of orientations considered per pixel can be approximated by $\frac{A_{cone}}{A_{sphere}}N$, where $A_{cone} = 2\pi(1 - \cos(\delta_{max}))$ is the surface area of the spherical cap representing the search cone (red region in Figure 1.1) and $A_{sphere} = 2\pi$ is the surface area of S^2_+. This yields $(1 - \cos(\delta_{max}))N$ orientations to be considered in each pixel.

For $\delta_{max} = 0.5$, that we use throughout the chapter, this equals $0.1224\,N$. For the adaptive filtering with base axis search input, 13 additional scans are required for the computation of the initial orientation estimate. Hence, the total number of filtered orientations per pixel equals $13 + (1 - \cos(\delta_{max}))N$. This is smaller than N for all $N \geq 16$. Note that according to [7] a sampling with $N = 98$ is not sufficiently fine for local fiber orientation estimation. Thus, efficiency is significantly improved.

For the case of adaptive filtering with input orientations derived from the Hessian, an explicit formula is harder to find as the complexity of the input orientation estimation is the sum of complexities of Gaussian filtering ($O(m_W m_H + m_W m_D + m_H m_D)$ complexity[3]), finite difference filters ($O(m_W \cdot m_H \cdot m_D)$ complexity), and the eigenvalue analysis. In practice, it turned out to be even faster than the base axis search.

[3]For an image of size $m_W \cdot m_H \cdot m_D$ voxels. The Gaussian kernel is separable and there are O(1) per pixel in recursive implementations of a 1D Gaussian filter.

Our method is compared with methods for directional line filtering ($SE = \ell$) in all sampled orientations and run-times are given in Table 2.1 and Figure 2.4. Our method could be extended brute force to line filtering in every orientation by setting $\delta_{max} = \frac{\pi}{2}$, i.e. by selecting the largest possible search cone. We denote this approach by "Base" in Table 2.1. However, more efficient implementations of directional filters have been suggested [10, 35, 36, 37, 38]. These implementations exploit periodicity in the SE discretization to achieve linear complexity in image size and constant complexity in the length of the SE. We compare our method to the one of [10], denoted by "Base (efficient)" in Table 2.1. As expected, our method is more than three times faster than "Base". Furthermore, our method also proves to be faster than "Base (efficient)" for $L = 5$. However, for increasing half-length L, "Base (efficient)" will eventually[4] become faster than our method due to constant complexity in the half-length L (Figure 2.4 right). Unfortunately, our method cannot be implemented in this way, since each pixel has its own search cone i.e. its own subset of orientations for filtering.

2.4 Applications

We now apply the proposed methodology to three types of structures: fibers, cracks, and partially closed foams. Fibers are long, locally cylindrical objects that can be detected using 1d line SEs. Cracks can be seen as 2d surfaces which can locally be captured by a 2d square SEs. Since plate-like cracks have to be distinguished from blob-like pores, our local shape operator is needed. Finally, in ceramic foams, essentially one-dimensional struts and essentially two-dimensional closed walls are intertwined, but shall be analyzed separately.

All run times reported in the following are observed using a machine equipped with an Intel i7-8665U processor running at 1.90 MHz and 16 GB of RAM, running on Linux OS.

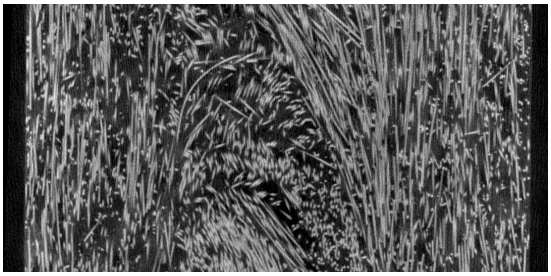
2.4.1 Use case 1: Misaligned region segmentation in a GFRP

Glass fiber reinforced polymers (GFRP) are widely used in light-weight construction. Mechanical properties of the material like strength and stiffness are strongly anisotropic depending on the local orientation of the fibers in the material. Therefore, fiber orientation analysis plays an important role in developing fiber reinforced composites and dimensioning parts made of them. A common way to produce this type of materials is injection molding. During this process, fibers essentially follow the flow. However, it is well known that fiber orientations deviate in a central layer whose exact characteristics depend on the production parameters. Local fiber orientations can be predicted by numerical simulations, but prediction gets harder with increasing fiber length and fiber volume fraction. Quantifying the misalignment helps to understand how production parameters influence it and to check the quality of flow simulations [39].

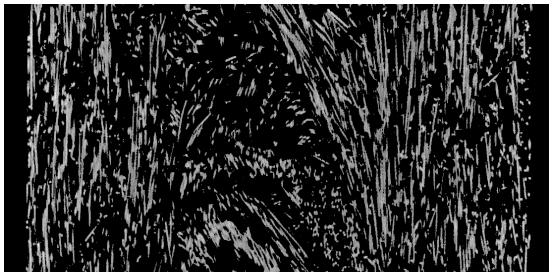
Procedures for detecting anomaly regions are based either on orientation analysis using the Hessian matrix, see [40], or on fiber separation and subsequent analysis of the orientations of these individual fibers. In many cases, single fiber separation is not possible due to sample properties or insufficient resolution. This is a particular problem in materials with high fiber volume fraction, where fibers frequently touch even if the fiber cross-section is sufficiently well resolved. In these cases, Hessian based orientation analysis may be locally unstable. We thus use the base axis search for estimating input orientations.

We reconsider one of the 3d images of GFRP discussed by [39]. Samples and CT images are provided by the Leibniz Institute for Composite Materials (IVW) in Kaiserslautern, Germany.

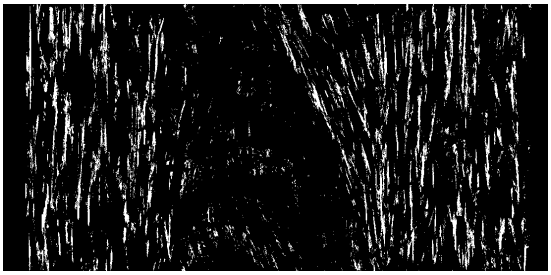
[4] By using linear interpolation one can estimate that this should happen when the half-length L is close to 100. However, for the image size of 256^3 used here, many voxels will be affected by edge effects.



(a) Original reconstructed CT image

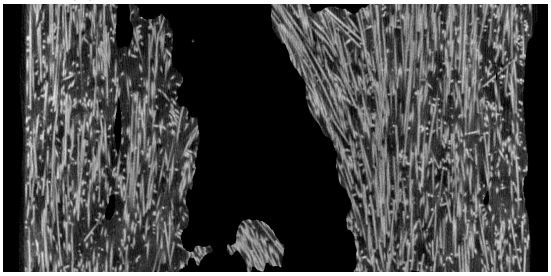


(b) Image (a) masked with segmented fiber component

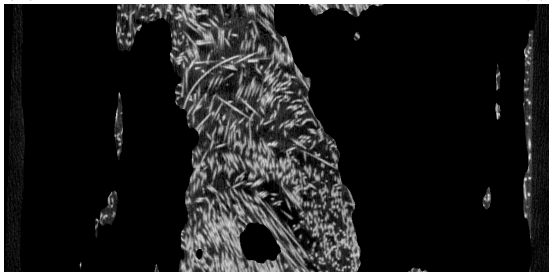


(c) Fibers aligned with injection direction

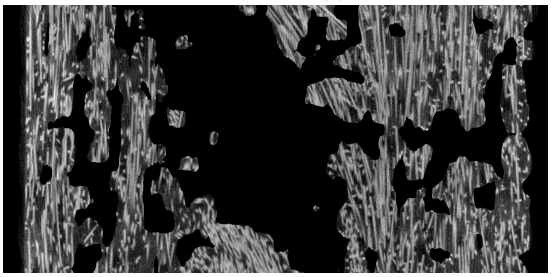
(d) Mean filter in $20 \times 20 \times 20$ pixel cube applied to (c)



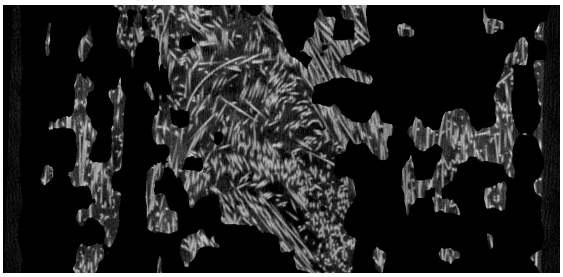
(e) Aligned region by thresholding (d) overlaid on (a)



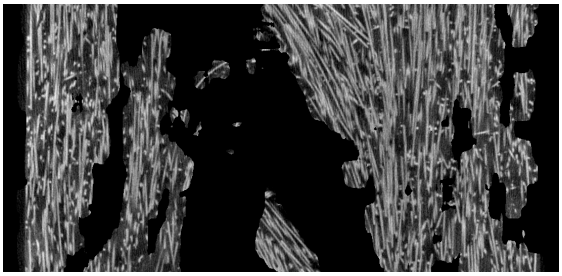
(f) Misoriented region (complement of (e)) overlaid on (a)



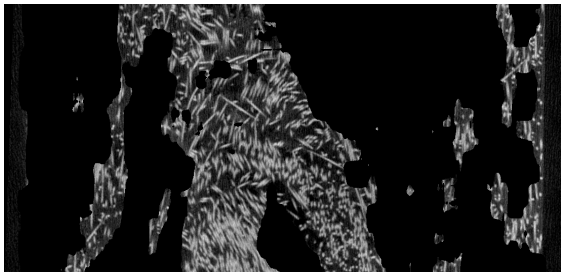
(g) Comparison 1: aligned region overlaid on (a) using orientation from Hessian matrix ($\sigma = 1$)



(h) Comparison 1: misoriented region overlaid on (a) using orientation from Hessian matrix ($\sigma = 1$)



(i) Comparison 2: aligned region overlaid on (a) using orientation from structure tensor ($\sigma = 1$, $\rho = 3$)



(j) Comparison 2: misoriented region overlaid on (a) using orientation from structure tensor ($\sigma = 1$, $\rho = 3$)

Figure 2.5: Use case 1. Misaligned region detection in a long-glass-fiber-reinforced polymer: from left to right, from top to bottom - 2d slice views of the input image, fiber component, fibers following the injection direction, smoothed system of those fibers, region where fibers are aligned, and misaligned region. Slices consist of $1\,100 \times 500$ pixels cropped from $1\,100 \times 500 \times 200$ pixels with spacing $4\,\mu$m.

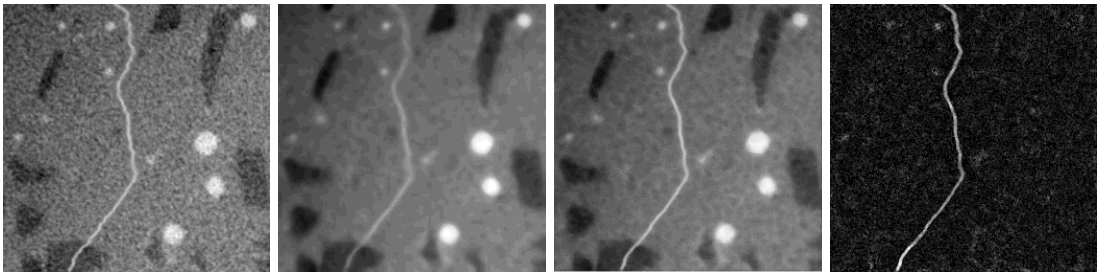


Figure 2.6: Use case 2. 2D slice views of crack detection by adaptive square filtering: input image I (inverted version of the original image), standard median filter with $7 \times 7 \times 7$ mask (for comparison), $\Gamma^{s_L}_{max}(I)$, and R_{aniso} (from left to right). The image consists of $256 \times 256 \times 256$ pixels with spacing 37.83μm.

We crop a sub-volume of $1\,100 \times 500 \times 200$ pixels, see Figure 2.5a).

Due to the high fiber volume fraction, fibers cannot be separated completely. Hence, we aim at segmenting the misaligned region based on the estimated orientation $\Gamma^{\ell_L}_{arg}(I)$. This is achieved through a four step procedure which includes:

1. orientation estimation,
2. separation of the fiber system based on orientation,
3. region detection using the dominant orientation,
4. post-processing.

It demonstrates the abilities of our improved adaptive orientation estimation. Step by step results of the procedure are shown in Figure 2.5. Next we describe each step in detail.

In the first step, we apply our method using $\Gamma^{\ell_L}_{arg}(I)$ from equation (2.5) with a line SE with parameters $n = 40$, $L = 20$, $\delta_{max} = 0.5$ using the median filter. This takes 98 minutes. In the second step, the fiber component is segmented. Fiber bundles should provide more stable orientation information than single fibers which may also be outliers. To extract the bundles, we apply the approach of [41] to the image $\Gamma^{\ell_L}_{arg}(I)$. The method is based on the 2nd order orientation tensor, roughly the 2nd moment of the local orientation, averaged over a small neighborhood, see [7] for details. In fiber pixels, the orientation tensor has one dominant eigenvalue. Hence, the ratio of the largest and the second eigenvalue can be used to select fiber bundles. The result is shown in Figure 2.5b.

In the third step, the orientation image $\Gamma^{\ell_L}_{arg}(I)$ is masked with the detected fiber bundles. A dominant orientation and its range are clearly visible from a 2d histogram of the remaining parts of $\Gamma^{\ell_L}_{arg}(I)$ in spherical coordinates which enables separation of the aligned region. That is, pixels whose orientation in spherical coordinates is in the range $[1.3, 1.7] \times [1.1, 1.9]$ are considered aligned with the dominant orientation (y, thus vertical in the xy slices shown in Figure 2.5c). In a final fourth step, a mean filter with a $30 \times 30 \times 30$ filter mask (Figure 2.5d) and final thresholding yield the aligned and misaligned regions (Figure 2.5e and 2.5f, respectively).

Note that the refined orientation estimation is crucial in this process. To show effectiveness of our method, we use the same four-step procedure with one change: in the first step we plug in the orientation estimation from the established methods. Applying the same framework to the orientation data obtained from the Hessian matrix did not yield convincing results (Figure 2.5g and 2.5h). Using the structure tensor [7, 34] instead yields slightly better results

(Figure 2.5i and 2.5j). However, single misaligned fibers in the aligned region still cause artifacts, and boundary and shape of the misaligned region remain less smooth and compact compared to the one derived by our approach.

2.4.2 Use case 2: Crack segmentation

Concrete is the most used construction material. 3d imaging by μCT enables non-destructive investigation of its internal structure in high resolution. In particular, damage processes and crack formation can be analyzed which is vital for better understanding of the properties of various concrete types and mixtures.

Crack segmentation in 3d images of concrete is challenging due to: (i) cracks being thin structures of varying shape and thickness, and (ii) concrete being a highly heterogeneous material with a variety of sub-types and components (e.g. pores, cement matrix, larger gravel, reinforcement structures). In CT images, cracks can be distinguished from most other structure components by means of their low gray value. However, cracks and pores, both being filled with air, appear typically similarly dark. Hence, distinguishing both components requires additional local shape characterization. We perceive cracks to be 2d structures being thin compared to other concrete components. For their segmentation, we apply our operator R_{aniso} from equation (2.6) with input orientation $v = \Gamma^{s_L}_{arg}(I)$ computed on the inverted image. Crack structures will have high values of R_{aniso} such that they can successfully be distinguished from ball-shaped pores by simple thresholding.

We test our method on a $256\times256\times256$ pixel simulated crack image. The concrete background is derived from a sample provided by the Department of Civil Engineering, Rheinland-Pfalz Technical University Kaiserslautern-Landau, and scanned at Fraunhofer ITWM with a pixel edge length of $37.83\,\mu$m. For details on the crack simulation and image synthesis, we refer to [42].

Results are shown in Figure 2.6 for the parameters $n = 16$, $L = 3$, and $\delta_{max} = 0.5$ using the median filter on a square. The initial orientations are estimated from the Hessian matrix. The adaptive filtering using the input orientation from the Hessian matrix takes 57 seconds. For comparison, we also show the original image filtered by a $7 \times 7 \times 7$ median filter, i. e. the edge length of the filter mask is equal to the SE edge length. Both filters reduce noise. The $7 \times 7 \times 7$ median filter blurs the crack and reduces the contrast. Our filter $\Gamma^{s_L}_{max}(I)$ is able to enhance the crack structure while reducing background noise at the same time. Additionally, R^*_{aniso} allows for a better handling of crack junctions than R_{aniso}, see Figure 2.7.

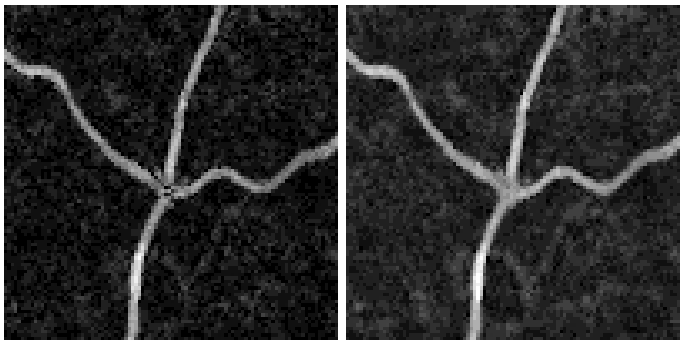


Figure 2.7: Use case 2. Junction analysis on 3d crack image: R_{aniso} (left) and R^*_{aniso}(right).

For the final segmentation, we threshold the output of R^*_{aniso} with lower and higher thresholds

	Single crack			Junctions		
	P	R	F1	P	R	F1
R_{aniso}	0.78	0.83	**0.79**	**0.84**	0.80	0.82
R^*_{aniso}	0.73	**0.84**	0.77	0.83	0.84	**0.83**
FF	**0.80**	0.79	0.76	0.78	**0.87**	0.82
TM	0.78	0.76	0.76	0.80	0.74	0.77

Table 2.2: Quantitative comparison of operators R_{aniso} and R^*_{aniso} with Frangi filter (FF) and template matching (TM).

and use the results as mask and marker image, respectively, in a morphological reconstruction by dilation according to [43]. The higher threshold extracts just the crack centerline, while the lower threshold extracts the full crack structure. The morphological reconstruction reduces noise and ensures extraction of the crack as a connected component.

Since the crack was simulated, there exists an unambiguous ground truth to compare our thresholded and post-processed result with (Figure 2.8). Overall, the crack structure is well captured in the segmentation, albeit being slightly smoothed. The boundary regions of pores being erroneously segmented as crack can be removed by post-processing. Note that the morphological reconstruction improves crack coverage compared to the simple thresholding applied earlier [42].

Table 2.2 provides further quantitative analysis and comparison with related methods. The methods are tested on two sets of simulated cracks: a single crack (Figure 2.6) and cracks with junctions appearing in the simulated images with two intersecting cracks. Crack intersections or junctions are challenging because there is no single dominant crack direction (Figure 2.7). Each set has five samples. Average values of Precision (P), Recall (R, true positive rate), and F1 score are reported for a fixed parameter configuration. More details on these metrics can found in Appendix D. For all methods, the final segmentation is derived by morphological reconstruction as described earlier. The Frangi filter [33] and template matching as devised by [44] are compared with R_{aniso} and R^*_{aniso}. Our methods give slightly higher average F1 scores for both the single crack and the crack junctions samples. Our operator R^*_{aniso}, designed to improve junction handling, gives higher recall values than R_{aniso} on the crack junctions samples (Figure 2.7).

2.4.3 Use case 3: Partially closed foams

Ceramic foams are routinely used to filter metal melts. They are produced by covering an open cell polymer foam template by a ceramic slurry. The resulting structure can be decomposed into struts and two-dimensional walls. The latter are formed when facets of the open cells of the polymer foam are closed by the slurry. Closed windows in foams affect the permeability. The walls are preferably parallel to the direction in which the polymer foam is squeezed when wetting it with the slurry. Detection and orientation analysis of closed facets enables realistic modelling of foam structures [45, 46, 47] and the impact of closed windows on permeability [48].

Our operator R_{aniso} from equation (2.6) with input orientation $v = \Gamma^{s_L}_{arg}(I)$ is applied to simulated 3D ceramic foams with partially closed facets generated by [46] (referred to as Example 1). The foam is simulated by the strut system of a Laguerre tessellation and some closed facets of the tessellation to create the walls. This results in a foam system whose struts and closed facets have thickness 1. Locally adaptable dilation then yields a realistic foam structure with variable thickness of struts and facets (Figure 2.10). This simulated data provides a precise ground truth for validation of our approach. Both, the varying structure thickness and the smooth transition between struts and walls make the separation challenging, see Figure 2.9. We expect our framework to be most effective in the central part of the facet since this is where the

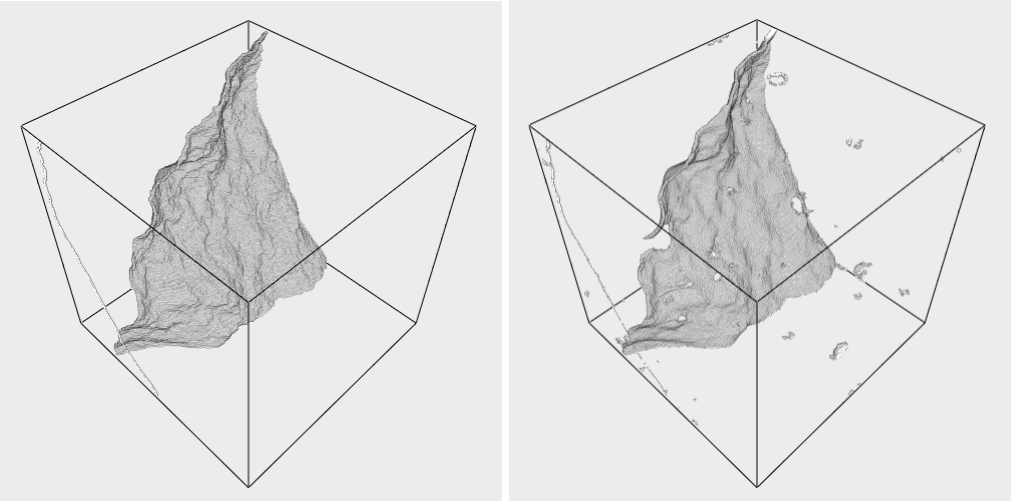


Figure 2.8: Use case 2. 3D renderings of the crack segmentation: ground truth and crack obtained by thresholding the post-processed local R_{aniso}. The image consists of $256 \times 256 \times 256$ pixels with spacing $37.83\,\mu$m.

planar structure is most pronounced.

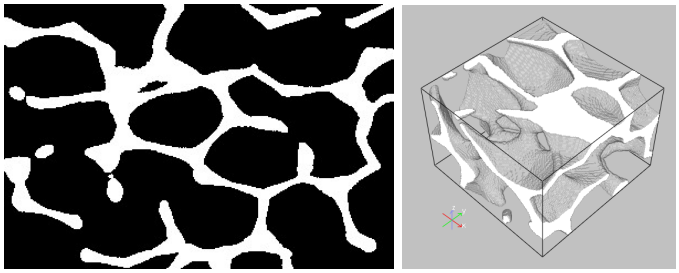


Figure 2.9: Use case 3, Example 1. Zoomed in 2d slice view and 3d rendering of the simulated ceramic foam structure: no clearly perceivable boundary between facets and struts.

Our framework is applied with square SE and parameters $n = 24$, $L = 10$, and $\delta_{max} = 0.5$ using the mean filter on the SE. The run time is ~ 30 minutes on an image of size $670 \times 670 \times 270$ pixels. Furthermore, the operator R_{aniso} is used to distinguish between closed facets and struts. Afterwards, the segmented facet system is post-processed by applying $\Gamma^{s_L}_{max}(I)$ using the median filter to refine detection and remove artifacts.

The segmentation results together with the ground truth are shown in Figure 2.11. 3D renderings of the results can be found in Figure 2.12. Visually, no obviously misclassified facets can be observed. The segmented strut system covers 75% of the pixels in the strut ground truth. If we introduce an error tolerance by dilating the segmented strut system by 1 pixel, coverage further improves to 93.9%. Since each facet's medial surface is known, we can calculate the percentage of the facets that was at least partially detected. Approximately 89% of the facets are recognized by our framework. Small and thick facets are the hardest to detect and are often confused with struts even in the eye of the observer. Additionally, the location of the precise boundary between the facets and the struts is highly subjective. Nevertheless, both the coverage

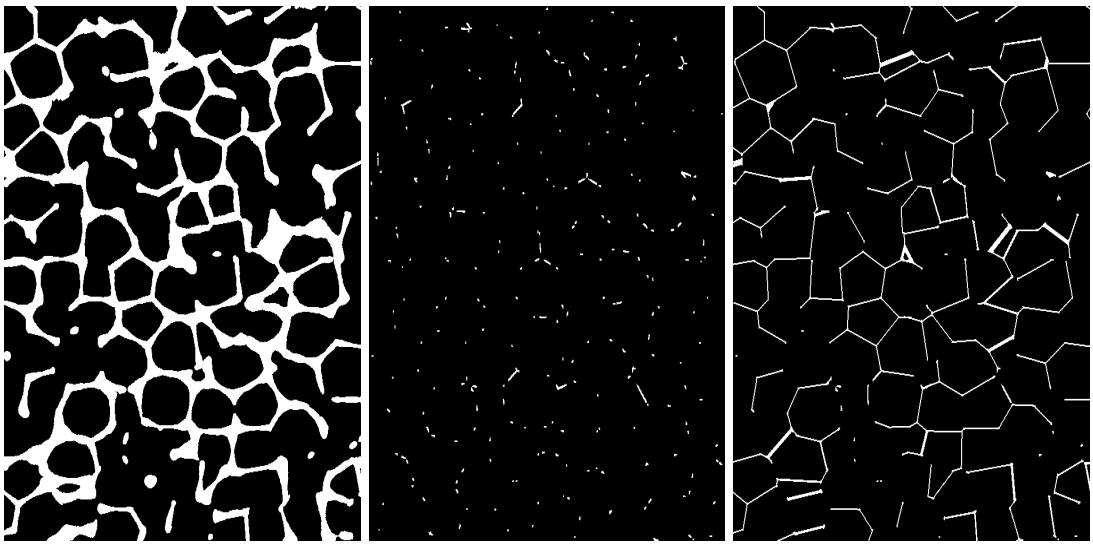


Figure 2.10: Use case 3, Example 1. 2d slice views of the ground truth from left to right: simulated foam, strut system, facet system.

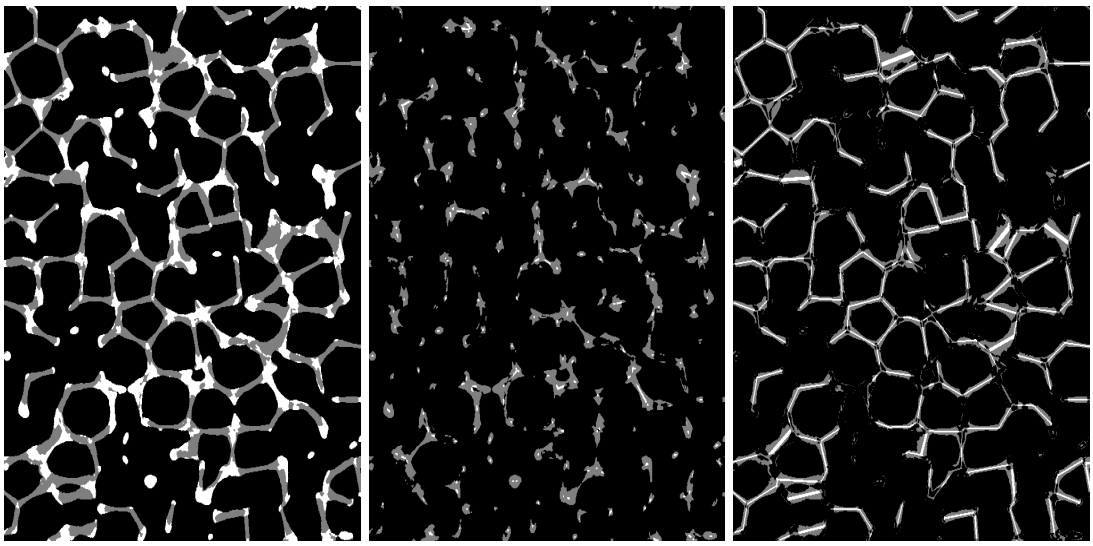


Figure 2.11: Use case 3, Example 1. 2d slice views of results. From left to right: separated facets and struts (white - struts, gray - facets), struts overlap with ground truth, and facets overlap with ground truth (for the last two images: white - overlap, gray - difference).

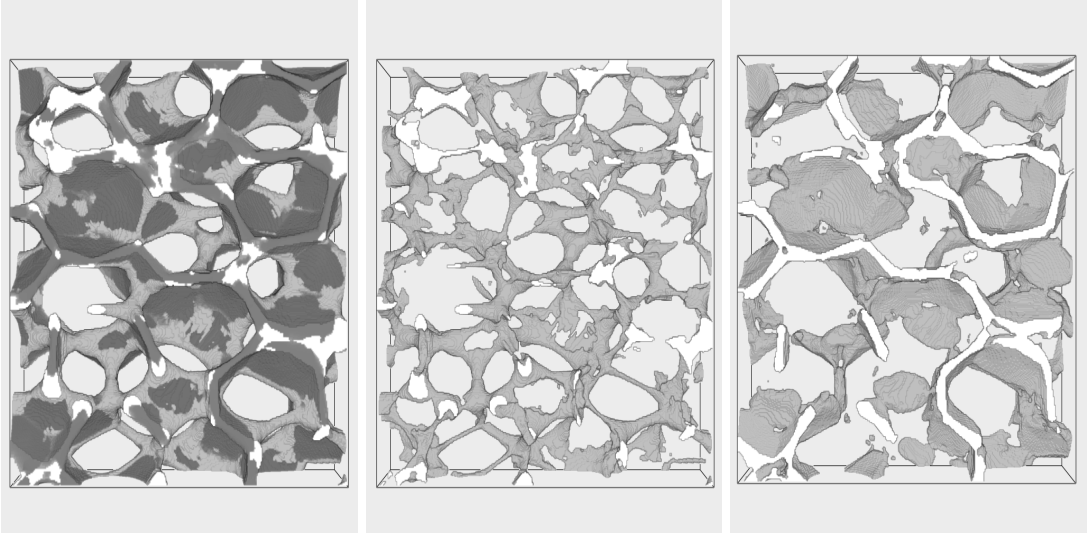


Figure 2.12: Use case 3, Example 1. Rendered results for the simulated foam: original image (white - struts, gray - facets), strut system, and facet system.

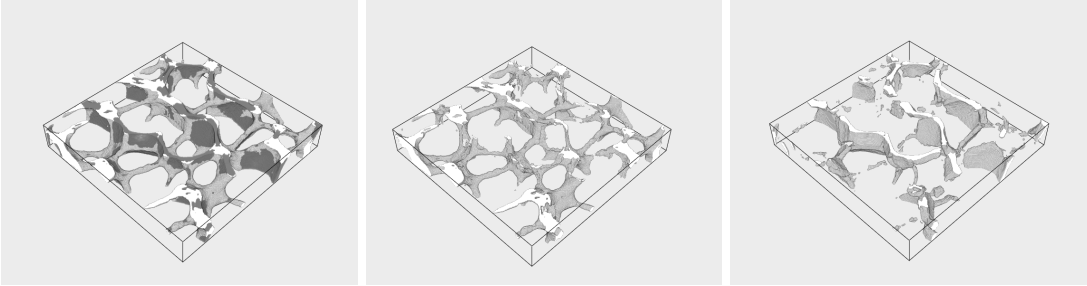


Figure 2.13: Use case 3, Example 2. Rendered results for the real ceramic foam: original image (white - struts, gray - facets), strut system, and facet system. The rendered sub-image consists of $400 \times 400 \times 50$ pixels with spacing $33.91\,\mu$m.

percentages and the visual evaluation prove that our framework gives reliable and satisfactory results.

Finally, our framework is applied to a CT image of a real silicon carbide foam. The sample was scanned at Fraunhofer ITWM with pixel edge length $33.91\,\mu$m (referred to as Example 2). Further details are described in [47]. On the sub-volume of $500 \times 500 \times 100$ pixels, our algorithm takes 9.5 minutes. Results are shown in Figure 2.13. In this case, no ground truth is available. Hence, the results can only be evaluated visually. The segmented wall system contains some strut pixels. The majority of the walls is however segmented correctly.

2.5 Discussion

We have presented a framework for adaptive directional filtering of 3D image data. The restriction of filter orientations to a cone centered in an initial input orientation avoids checking the full sampled orientation space. That results in significant run-time savings compared to previously

suggested methods.

Our algorithm has three main parameters: the half-length L of the SE, the opening angle of the cone δ_{max}, and the sphere discretization parameter n. The half-length parameter L depends strongly on the application. It has to approximate the size of the object of interest to be effective. The parameters δ_{max} and n balance run-time and accuracy. The SE size L limits the discretization density n since the number of discrete SEs of half-length L is restricted. For example, there are only 13 discretized lines with half-length 1 in the ℓ_∞ norm (base axis, plane diagonals, space diagonals).

Our method requires a map of input orientations at which the cone is centered. These have to be close to the correct orientation, but do not have to fit it very well. The adaptive filter on the search cone is in fact able to fix imprecision in input orientations or scale and finds the most appropriate orientation. This makes the method more robust and less dependent on the scale parameter σ than classical Hessian directional filtering [22].

Our framework is very flexible, can be adapted to specific tasks, and used as building block for various image processing pipelines. This is in particular demonstrated by the adaption of R_{aniso} to R^*_{aniso} for handling of crossing cracks.

We have validated our framework in use cases from materials science, it is however not restricted to that area. In biomedical applications, [13, 21, 22] have used approaches similar to ours but with line SE exclusively. Our algorithm can enhance and extract vessels straightforwardly, too. Therefore, our framework can be seen as a robust, flexible, and multi-functional method for filtering, enhancement, and separation of oriented structures in 3D. The two outputs – orientation information and filtered image – can be used to solve multiple tasks.

Future work will explore more complex SEs while preserving computational efficiency. Another topic of further research is a thresholding method for precise and reliable unsupervised segmentation of cracks based on $\Gamma^{SE}_{max}(I)$ from equation (2.4) and R^*_{aniso} from equation (2.7).

Chapter 3

3d adaptive line granulometry

In this chapter, we derive a method based on the concept of granulometry for analyzing both fiber length and orientation without single fiber separation. First, we explain the need for analyzing fiber length and orientation in fiber reinforced composites. Secondly, we show how to define granulometry on binary images from 3d adaptive line morphology from the previous chapter. We show that this extension indeed satisfies the properties of granulometric functions from mathematical morphology. Finally, we show an application example for quantifying length and orientation in two types of fiber reinforced polymers (FRP): injection moulded and 3d printed.

3.1 Analyzing the fiber length and orientation

The mechanical and physical properties of fiber reinforced materials are connected to the microstructural characteristics of the fiber system. For example, the tensile strength of a fiber reinforced material will be improved significantly in a direction orthogonal to the dominant orientation of the fiber system. The length of fibers and fiber volume fraction can have a similar role as well. Generally, fiber length is known in advance from the fiber material specifications given by the producer. However, during the production process, fibers can break and tear apart effectively changing their length. Furthermore, the orientation of the fiber system is highly dependent on the production process and its parameters. Hence, the characterization of fiber length and orientation from CT images is the most precise way to analyze the fiber length and orientation of the sample.

For the fiber orientation, a variety of well-approved and robust methods are available, see e.g. [6, 7]. Unbiased estimation of the fiber length distribution based on image data is however a much harder task. Various methods have been suggested, usually building on the separation of the fibers. They require therefore very high resolutions [49, 50, 51], nearly unidirectional fibers [49, 52, 53], or user interaction [53, 54]. For a more detailed discussion see [55]. Ambiguities due to the need to merge fiber segments [17] or to resolve fiber-fiber contacts [50] are unavoidable. Moreover, the notorious censoring problem of properly treating fibers not completely observed is taken into account only very rarely [49, 56, 57]. As a consequence, pyrolysis of the composite and subsequent analysis of the manually separated fibers is still advertised as the method of choice till today [58] although being highly error-prone as well.

3.2 Preliminaries

For our purpose, we define granulometry only for binary images. For $D \subset \mathbb{Z}^3$ a discrete grid, we define the space of binary images $\mathscr{L}_D^{\{0,1\}} = \{I | I : D \to \{0,1\}\}$. Binary images can be interpreted as a set $X \subset D$ or as a function $I(p) = \mathbb{1}_X(p)$. This means that $I(p) = 1$ if $p \in X$, and otherwise 0. We will use these two notations interchangeably in this chapter. When possible we will define morphological operators on the space of grayvalue images $\mathscr{L}_D$. However, all proofs in this chapter will be restricted to the space of binary images $\mathscr{L}_D^{\{0,1\}}$.

In this chapter, we allow for the slight abuse of notation by using d as a distance function on both spatial and orientation domains. When measuring the distance between voxels $d : \mathbb{R}^3 \times \mathbb{R}^3 \to \mathbb{R}_+$ denotes Chebysev (maximum) distance d_∞ from Section 2.3. Alternatively, for the orientations $d : S_+^2 \times S_+^2 \to [0, \pi]$ refers to angle distance metric from equation (2.2).

In the following sections we focus on the line as a structuring element. Similarly as in Section 2.3, let $\ell_L(u) \subset D$ be a discretized line of length L with orientation $u \in S_+^2$ centered at the origin. Then let $\ell_L(u)(p) = \{p^* | p^* - p \in l_L(u)\}$ be a translated version of $l_L(u)$ centered at voxel $p \in D$. Furthermore, let $L^* = d(p, p^*)$ be the distance between voxels p and p^*. Then $\ell_{L^*}^+(u)(p^*)$ is a half-line segment from $\ell_{L^*}(u)(p^*)$ between p and p^*.

The previous chapter was based on directional filtering with orientation map $v : D \cap X \to S_+^2$. In this chapter, we need the following **key assumption on the orientation map** v in order to construct operators from mathematical morphology in a mathematically sound way:

- For binary image $I = \mathbb{1}_X \in \mathscr{L}_D^{\{0,1\}}$, orientation is estimated in the beginning and the **orientation map** $v : D \cap X \to S_+^2$ **is kept fixed** and used to construct the oriented structuring element for all subsequent morphological operators.

The reason for this is that we want to achieve that our operators satisfy the key properties of classical morphological operators which usually use the same structuring element of fixed size in every voxel.

3.3 Extension from 3d adaptive line morphology

To characterize fiber length we will construct granulometry based on the line opening filter. Granulometry in mathematical morphology was originally designed to measure the size distribution of grains using balls of varying radius. The maximal fitting ball is taken as its size. Note that the result of this process is a voxel weighted size distribution. For our application example this size distribution is actually a length distribution of the fiber system. We will discuss later how to convert it to a more common type of size distribution, the number weighted one. First, we give an explanation of why we choose this setup.

Why do we characterize fibers with the line as a structuring element? Fibers are cylindrical elongated objects, i.e. we assume that the length of the fiber is at least two times larger than its diameter. Hence fibers can be deemed to be one dimensional objects with small thickness in 3d space. Since we are interested only in the characterization of the length, out of simple structuring elements (ball, plane, and line), the line fits best for this purpose.

Why do we use the opening as a filter? The assumption of this step is that the fiber system is segmented, binarized, or separated from the remaining material matrix. Then, line erosion preserves all the voxels in which a symmetric line of fixed length can fit. These correspond to the fiber centers. In the following step, line dilation reconstructs the remainder of the voxels

belonging to the same fiber from fiber centers, i.e. in these voxels a non-centered line of a given length can be fitted. All voxels belonging to the shorter fibers are hence removed.

Why do we introduce line granulometry? We are interested in characterizing the length distribution of the fiber system. Hence, it is intuitive to vary the length of the line and perform line opening with increasing line length until all voxels are removed. For every voxel the maximal length of the fitted non-centered oriented line is recorded as the fiber length at that voxel. This translates exactly into the concept of granulometry.

First, we give precise definitions of granulometry according to mathematical morphology in Section 3.3.1. Next, we aim at defining adaptive line opening as a morphological filter which will be used to construct 3d adaptive line granulometry. We do this gradually in three steps. As a first step, we define morphological operators on the constant orientation map for $u \in S^2_+$ (Section 3.3.2). These are needed to understand how to generalize these operators to voxelwise varying orientation map $v : D \cap X \to S^2_+$ (Section 3.3.3). As it turns out, naive generalizations of these operators can cause undesired effects when measuring fiber length. Finally, we construct an adaptive line opening and prove that it can be seen as a morphological opening (Section 3.3.4). This three-step process is depicted in Figure 3.1.
The final goal is to use these morphological line operators to construct 3d adaptive granulometry (Section 3.3.5). This is followed by the proof of granulometric axioms (Section 3.3.6) and implementation algorithm (Section 3.3.7). As an output of granulometry, we get the maximal length of the fitted line in every voxel. We show how to utilize this information to get the number weighted orientation and length distribution (without any correction for edge effects) in Section 3.3.8.

3.3.1 Definition of granulometry

The concept of granulometry is inspired by the sieving of granular material with sieves of increasing hole sizes. In mathematical morphology, sieves of different sizes can be seen as openings and closings with structuring elements whose size is defined through a parameter $l > 0$. Openings are used for bright objects on a dark background, while closings work with dark objects on a bright background. These morphological operators are suitable because they remove all structures in the image whose shape can not be matched with a structuring element of size l. In other words, their size in terms of structuring elements is smaller than l. This was first proposed by Matheron [59]. Hence, granulometry can be seen as a family of functions with the size parameter l that satisfy certain axioms motivated by the sieving of the grains, see e.g. [31].

Definition 1. *A family* $\{\Phi_l : \mathscr{L}_D^{\{0,1\}} \to \mathscr{L}_D^{\{0,1\}}\}_{l>0}$ *of functions is called* ***granulometry*** *if the following three axioms are fulfilled:*

1. ***anti-extensivity*** *implies that* Φ_l *shrinks* $X \subset D$*, i.e.*

$$\Phi_l(X) \subseteq X \text{ for all } l > 0. \tag{3.1}$$

2. ***increasingness*** *implies that for* $X, Y \subset D$ *for which* $X \subset Y$ *the following holds*

$$\Phi_l(X) \subseteq \Phi_l(Y) \text{ for all } l > 0. \tag{3.2}$$

3. ***absorption:*** *when applying two granulometric functions for* $l, m > 0$*, the following holds*

$$\Phi_l(\Phi_m(X)) = \Phi_m(\Phi_l(X)) = \Phi_{\max(m,l)}(X). \tag{3.3}$$

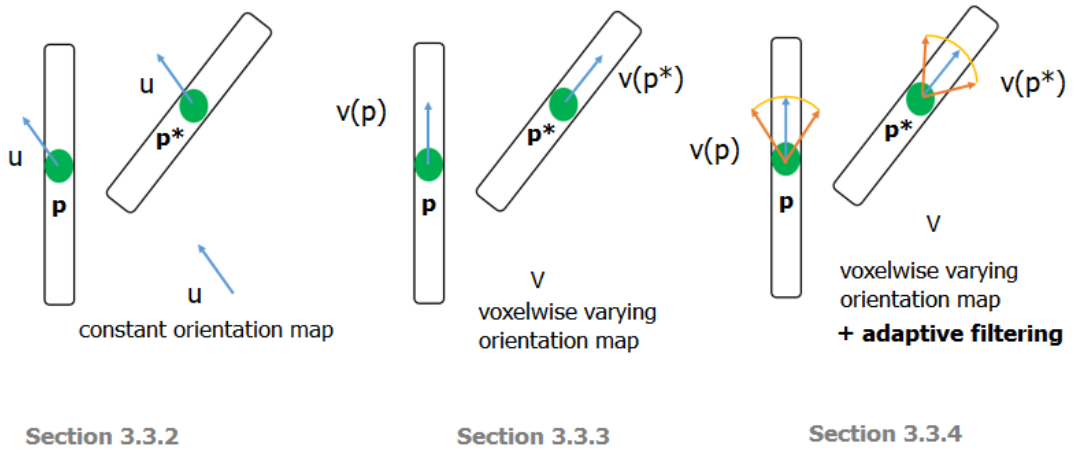


Figure 3.1: Comparison of three settings for defining morphological line operators (from left to right): constant orientation map, voxelwise varying orientation map, and voxelwise varying orientation map in the adaptive setting. Interesting voxels are marked with green circle, orientations in the voxels are marked with blue arrow. Adaptive filtering is marked with orange arrows and yellow curve.

To summarize, the goal of granulometry is to separate image structures based on their size. In mathematical morphology this is done by using a structuring element of fixed shape with varying size. In this chapter we aim to characterize fiber length in every voxel with line structuring element, i.e. we want to construct a line granulometry similar to [10, 35, 60, 61].

3.3.2 Morphological line operators on constant orientation map

First, we define the erosion and dilation on the line structuring element of length $L \in \mathbb{N}$ oriented at voxelwise constant orientation $u \in S^2_+$ (Figure 3.1, left). This means that we use the line of the fixed length L with the same fixed orientation u at every voxel.
Erosion $\mathcal{E}_{\ell_L(u)} : \mathscr{L}_D \to \mathscr{L}_D$ is defined via

$$\mathcal{E}_{\ell_L(u)}(I)(p) = \min_{p^* \in \ell_L(u)(p)} I(p^*).$$

Dilation $\delta_{\ell_L(u)} : \mathscr{L}_D \to \mathscr{L}_D$ can be defined in several ways. We give three definitions. The first two definitions are equivalent if the line of the same length L and orientation u is used as a structuring element. Under the same conditions, the third one is equivalent to the first two but it is defined only for binary images. The need for introducing these three definitions will be discussed in the following section for voxelwise varying orientation map. First, analogously as in erosion, in voxel p we can look for the maximum on the line passing through p:

$$\delta^{(1)}_{\ell_L(u)}(I)(p) = \max_{p^* \in \ell_L(u)(p)} I(p^*).$$

Secondly, we can look for the maximum of values of all pixels p^* whose line passes through voxel p

$$\delta^{(2)}_{\ell_L(u)}(I)(p) = \max_{p \in \ell_L(u)(p^*)} I(p^*).$$

Finally, we give the third equivalent[1] definition of line dilation on $\mathscr{L}_D^{\{0,1\}}$ that we refer to as **connectivity-preserving dilation**. It is defined on a mask $J = \mathbb{1}_Y$ for some Y so that $X \subset Y \subset D$:

$$\delta^{(3)}_{\ell_L(u)}(I, J)(p) = \max_{p \in \ell_L(u)(p^*)} \Big(I(p^*) \mathbb{1}_{\ell^+_{L^*}(u)(p^*) \subset Y} \Big), \tag{3.4}$$

where $L^* = d(p, p^*)$ and $\ell^+_{L^*}(u)(p^*)$ is a half-line segment from $\ell_{L^*}(u)(p^*)$ between p and p^* that must be contained in the mask J. From now onwards, we will use the connectivity-preserving dilation to construct an opening with the notation

$$\delta_{\ell_L(u)}(\cdot, J) := \delta^{(3)}_{\ell_L(u)}(\cdot, J)$$

for some $J \subset D$ as a prior. Since we aim to construct an opening, we will use the (binary) input image I as a prior to limit the reconstruction. An opening $\gamma^O_{\ell_L(u)} : \mathscr{L}_D^{\{0,1\}} \to \mathscr{L}_D^{\{0,1\}}$ is then the erosion followed by the dilation:

$$\gamma^O_{\ell_L(u)}(I)(p) = \delta_{\ell_L(u)}\big(\mathcal{E}_{\ell_L(u)}(I), I\big)(p) \tag{3.5}$$

The reason why we apply the mask is that with the dilation we want to reconstruct only the voxels belonging to the original image I. This will be very important in the context of our framework with voxelwise varying orientation. There is an additional characterization of our opening:

Lemma 1. *$p \in \gamma^O_{\ell_L(u)}(I)$ if and only if there exists a voxel $p^* \in X$ s.t. $\ell_L(u)(p^*) \subset X$ and $p \in \ell_L(u)(p^*)$.*

Proof. If $p \in \gamma^O_{\ell_L(u)}(I)$, then we have one of the two cases:

1. $p \in \mathcal{E}_{\ell_L(u)}(I)$.
   Then from the definition of erosion, it holds $\ell_L(u)(p) \subset X$. Hence, for $p^* = p$ the claim is satisfied.

2. $p \notin \mathcal{E}_{\ell_L(u)}(I)$.
   Then from the definition of connectivity preserving dilation, there exists $p_1 \in \mathcal{E}_{\ell_L(u)}(I)$ s.t. $p \in \ell^+_{L_1}(u)(p_1) \subset X$ for $L_1 = d(p, p_1) \leq L$. From the definition of erosion, we further have $\ell^+_{L_1}(u)(p_1) \subset \ell_L(u)(p_1) \subset X$. Hence, for $p^* = p_1$, the claim is satisfied.

For reverse, let p, p^* be voxels s.t. $\ell_L(u)(p^*) \subset X$ and $p \in \ell_L(u)(p^*)$.
We immediately have $p^* \in \mathcal{E}_{\ell_L(u)}(I)$ and $p^* \in \gamma^O_{\ell_L(u)}(I)$. Furthermore, we have $p \in \ell^+_{L^*}(u)(p^*) \subset \ell_L(u)(p^*) \subset X$ for $L^* = d(p, p^*)$. Hence, from the definition of connectivity preserving dilation it follows $p \in \delta_{\ell_L(u)}\big(\mathcal{E}_{\ell_L(u)}(I), I\big) = \gamma^O_{\ell_L(u)}(I)$.

□

This interpretation will be used in the proof that this operator is indeed an opening in the sense of mathematical morphology. There is an additional result as a consequence of the previous lemma:

Corollary 1. *If there exists a voxel $p^* \in X$ s.t. $\ell_L(u)(p^*) \subset X$, then $\ell_L(u)(p^*) \subset \gamma^O_{\ell_L(u)}(I)$.*

[1] Claim of the equivalence is obviously dependent on the selection of the mask J. For $J = \delta^{(2)}_{\ell_L(u)}(I)$, the equivalence holds true. Another type of equivalence is achieved on the opening (dilation-erosion) operator, i.e. $\delta^{(2)}_{\ell_L(u)}(\mathcal{E}_{\ell_L(u)}(I)) = \delta^{(3)}_{\ell_L(u)}(\mathcal{E}_{\ell_L(u)}(I), I)$.

Definition 2. *Any function* $\gamma : \mathscr{L}_D^{\{0,1\}} \to \mathscr{L}_D^{\{0,1\}}$ *is an* ***opening*** *as a morphological operator if it satisfies the following properties:*

- ***anti-extensivity:*** *for* $X \subset D$
$$\gamma(X) \subseteq X$$
.
- ***increasingness:*** *for* $X, Y \subset D$ *s.t.* $X \subset Y$ *it holds*
$$\gamma(X) \subseteq \gamma(Y).$$
- ***idempotence:***
$$\gamma(X) = \gamma(\gamma(X)).$$

Lemma 2. $\gamma^O_{\ell_L(u)}$ *is an opening on* $\mathscr{L}_D^{\{0,1\}}$ *according to Definition 2.*

Proof. Anti-extensivity for $\gamma^O_{\ell_L(u)}$ is ensured by the connectivity-preserving dilation operator. $\delta^{(3)}_{\ell_L(u)}$. Formally, let $p \notin X$. For every voxel $p^* \in X$ s.t. $p \in \ell_L(u)(p^*)$, it holds $\ell_L(u)(p^*) \not\subset X$ and hence $p \notin \gamma_{\ell_L(u)}(I)$.

Increasingness is satisfied in the context of the fixed orientation map u that is used for both images. This is shown in the following lines: assume $p \in \gamma^O_{\ell_L(u)}(X)$, then from Lemma 1 there exist a voxel $p^* \in X$ at which the line $\ell_L(u)(p^*) \subset X$ can be fitted and $p \in \ell_L(u)(p^*) \subset X$. From $X \subset Y$ implies that $p, p^* \in Y$ and $p \in \ell_L(u)(p^*) \subset Y$, and hence $p \in \gamma^O_{\ell_L(u)}(Y)$ using Lemma 1. Hence, the increasingness is satisfied.

To prove idempotence, by anti-extensivity we have $\gamma^O_{\ell_L(u)}(\gamma^O_{\ell_L(u)}(X)) \subseteq \gamma^O_{\ell_L(u)}(X)$.
Let $p \in \gamma^O_{\ell_L(u)}(X)$

- Then from Lemma 1 there exist $p^* \in X$ and line $\ell_L(u)(p^*) \subset X$ s.t. $p \in \ell_L(u)(p^*)$.
- However, from Corollary 1 it also holds $\ell_L(u)(p^*) \subset \gamma^O_{\ell_L(u)}(X)$ and $p^* \in \gamma^O_{\ell_L(u)}(X)$.

According to Lemma 1, this implies $p \in \gamma^O_{\ell_L(u)}(\gamma^O_{\ell_L(u)}(X))$. Hence, $\gamma^O_{\ell_L(u)}(X) \subseteq \gamma^O_{\ell_L(u)}(\gamma^O_{\ell_L(u)}(X))$. □

3.3.3 Morphological line operators on voxelwise varying orientation map

In this section we show generalizations of the line erosion and dilation from the previous section on the voxelwise varying orientation map $v : D \cap X \to S^2_+$, see Figure 3.1, center. Furthermore, we analyze the three definitions of line dilations in a non-constant voxelwise orientation setting and discuss which one is the most suitable for the application in mind: adaptive line granulometry based on adaptive line opening for measuring fiber length distribution.
First, erosion $\mathcal{E}_{\ell_L}(\cdot, v) : \mathscr{L}_D \to \mathscr{L}_D$ for the orientation map v is defined via

$$\mathcal{E}_{\ell_L}(I, v)(p) := \mathcal{E}_{\ell_L(v(p))}(I)(p) = \min_{p^* \in \ell_L(v(p))(p)} I(p^*).$$

The three dilation operators can be easily extended to the orientation map v. We refer to them as $\delta^{(1)}_{\ell_L}(\cdot, v), \delta^{(2)}_{\ell_L}(\cdot, v) : \mathscr{L}_D \to \mathscr{L}_D$, and $\delta^{(3)}_{\ell_L}(\cdot, J, v) : \mathscr{L}_D^{\{0,1\}} \to \mathscr{L}_D^{\{0,1\}}$, respectively

$$\delta^{(1)}_{\ell_L}(I, v)(p) := \delta^{(1)}_{\ell_L(v(p))}(I)(p) = \max_{p^* \in \ell_L(v(p))(p)} I(p^*).$$

$$\delta^{(2)}_{\ell_L}(I,v)(p) = \max_{p\in\ell_L(v(p^*))(p^*)} I(p^*).$$

$$\delta^{(3)}_{\ell_L}(I,J,v)(p) = \max_{p\in\ell_L(v(p^*))(p^*)} \Big(I(p^*)\mathbb{1}_{\ell_{L^*}(v(p^*))(p^*)\subset Y}\Big),$$

for $L^* = d(p,p^*)$.

Two interpretations of $\delta^{(2)}_{\ell_L}(\cdot,v)$ on binary images: Definition of $\delta^{(2)}_{\ell_L}(\cdot,v)$ can be seen literally: for every voxel p we search for all voxels p^* with lines $\ell_L(v(p^*))(p^*)$ and check if p belongs to that line. Alternatively, we can start at voxel p^* and update maximums for all voxels p on the line $\ell_L(v(p^*))(p^*)$. Furthermore, one can see that only active voxels affect the maximum. Hence, we can do this only for active (non-null) voxels p^*.
In the remaining sections we will use the second interpretation. Furthermore, this interpretation is also the one that will be used in practice due to efficiency (Section 3.3.7).

The difference between $\delta^{(1)}_{\ell_L}(\cdot,v)$ and $\delta^{(2)}_{\ell_L}(\cdot,v)$: We explain the definitions of these two dilations on a simple example. Imagine having an image of a fiber of length $2L$ oriented at u and applying a line erosion $\mathcal{E}_{\ell_L}(\cdot,v)$ on it. Here, only voxels in the fiber center are still active. Dilation $\delta^{(1)}_{\ell_L}(\cdot,v)$ would check every voxel and its corresponding orientation in an attempt to reconstruct the whole fiber. On the other hand, $\delta^{(2)}_{\ell_L}(\cdot,v)$ would only use the orientation in the center of the fiber, i.e. in the active voxels to reconstruct the whole fiber. In this case inactive voxels do not depend on their own voxel orientation but rather on the orientation of the active voxels in their vicinity. Hence, the principles behind these two dilations are very different when using non-constant orientation in every voxel. Figure 3.2 shows an example of when these two dilations can result in very different results for a voxelwise varying orientation map. In this example, a completely disconnected fiber is reconstructed using $\delta^{(1)}_{\ell_L}(\cdot,v)$. In granulometry this is an undesirable property which is avoided by $\delta^{(2)}_{\ell_L}(\cdot,v)$. It can also occur that fibers are touching, $\delta^{(2)}_{\ell_L}(\cdot,v)$ handles similar situations better than $\delta^{(1)}_{\ell_L}(\cdot,v)$, and hence $\delta^{(2)}_{\ell_L}(\cdot,v)$ is preferred.

The difference between $\delta^{(2)}_{\ell_L}(\cdot,v)$ and $\delta^{(3)}_{\ell_L}(\cdot,v)$: These two definitions of line dilations are similar. The only difference is the connectivity-preserving property. However, it is not obvious which one would be preferred in the context of line opening on the voxelwise varying orientation map.

3.3.4 Adaptive morphological line operators

Selection of the line dilation operator

First, we give a short and intuitive comment on **the difference between $\delta^{(2)}_{\ell_L}(\cdot,v)$ and $\delta^{(3)}_{\ell_L}(\cdot,v)$ in the adaptive setting.** We do this without giving explicit formal definitions for the sake of brevity.
Adaptive sampling around the input orientation is shown to contribute to the robustness of filtering in the previous chapter. However, Figure 3.3 shows an example where an adaptive version of $\delta^{(2)}_{\ell_L}(\cdot,v)$ can reconstruct a close-by nearly parallel fiber. Similar effects happen for the orientation maps with estimation errors. This serves as motivation to use an alternative connectivity-preserving dilation $\delta^{(3)}_{\ell_L}(\cdot,v)$. This operator does not guarantee to correctly handle the case when nearly parallel fibers are touching. However, $\delta^{(3)}_{\ell_L}(\cdot,v)$ seems to be the most suitable

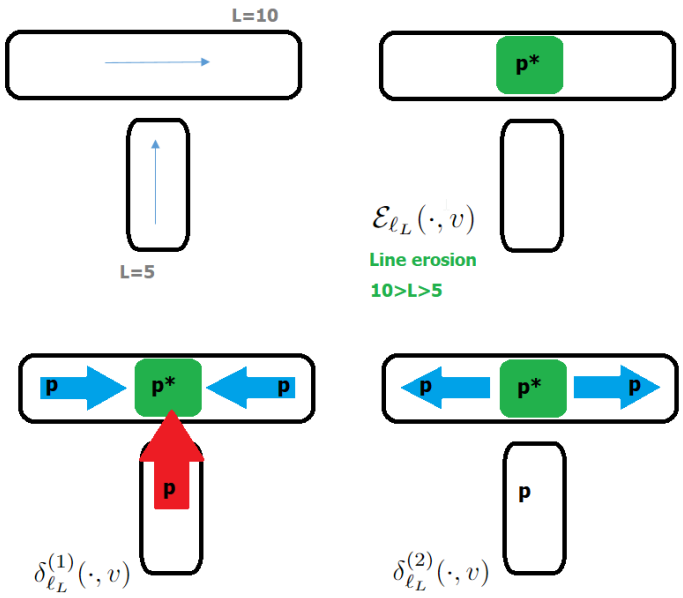


Figure 3.2: Illustration of the principles behind the morphological line operators on the voxelwise varying orientation map. On input image with two orthogonal fibers of lengths 5 and 10 voxels (top left) is applied an erosion operator $\mathcal{E}_{\ell_L}(\cdot, v)$ for $L \in \langle 5, 10\rangle \cap \mathbb{N}$ as illustrated in the green circle (top right). Afterwards, the two dilation operators $\delta^{(1)}_{\ell_L}(\cdot, v)$ (bottom left) and $\delta^{(2)}_{\ell_L}(\cdot, v)$ (bottom right) are applied to the eroded image. For $\delta^{(1)}_{\ell_L}(\cdot, v)$ we start at every voxel (e.g. see voxels marked with $\mathbf{p}$) and check if the oriented line (blue or red arrows) in that voxel passes through the green square (e.g. voxel $\mathbf{p}^*$). For $\delta^{(2)}_{\ell_L}(\cdot, v)$ we start only at the green square (e.g. voxel $\mathbf{p}^*$) and check which voxels can be reached from there (blue arrows). Red arrow indicates the difference between $\delta^{(1)}_{\ell_L}(\cdot, v)$ and $\delta^{(2)}_{\ell_L}(\cdot, v)$.

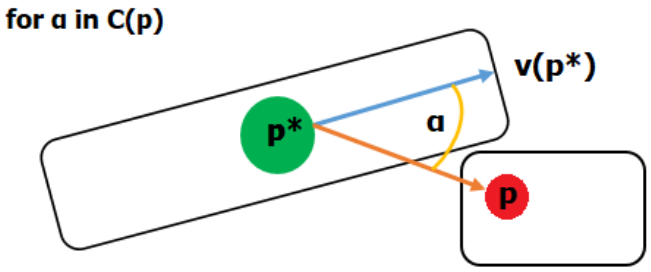


Figure 3.3: Example for the adaptive line opening with the dilation operator $\delta^{(2)}_{\ell_{L(u)}}$ on the eroded fiber image (green). Voxels (e.g. voxel p in red circle) not belonging to the original fiber can be reconstructed by the dilation in the case of nearly parallel nearby fibers. Voxel p^* is marked in the green circle, while input orientation $v(p^*)$ is marked with blue arrow. Orange arrow is a selected orientation from the search cone $C(p)$ which reconstructs the nearby fiber. This serves as a motivation for the definition of $\delta^{(3)}_{\ell_{L(u)}}$.

dilation out of all three proposed dilation operators to be used on a voxelwise varying orientation map in the adaptive setting.

Definitions of adaptive morphological line operators

After selecting the appropriate line dilation definition for the voxelwise varying orientation map $v : D \cap X \to S^2_+$ in the previous section, we can extend line dilation and erosion to the adaptive setting and define adaptive morphological line opening (Figure 3.1, right).

First, we define adaptive line erosion and dilation according to the definitions from the previous section. Adaptive line erosion $\mathcal{E}_L(\cdot, v) : \mathscr{L}_D^{\{0,1\}} \to \mathscr{L}_D^{\{0,1\}}$ is trivially defined by plugging in the erosion operator $\mathcal{E}_{\ell_L(u)}$ in equation (3.4).

$$\mathcal{E}_L(I, v)(p) = \max_{u \in C(p)} \Big(\mathcal{E}_{\ell_L(u)}(I)(p) \Big).$$

The maximum operator across all orientations in the search cone $C(p)$ is used because we want to have at least one oriented centered line to be fitted in voxel p in order to keep it active. Note that the search cone $C(p)$ depends on the orientation map v by $C(p) = \{u \in S : d(u, v(p)) \leq \delta_{max}\}$. For the adaptive dilation $\mathcal{D}_L(\cdot, v, J) : \mathscr{L}_D^{\{0,1\}} \to \mathscr{L}_D^{\{0,1\}}$ more care is required due to the voxelwise varying orientation map v. Here, the notation is slightly different because the adaptivity is related to the voxel p^* in the vicinity of the targeted voxel p. Hence, it can not be written elegantly in terms of $\delta^{(3)}_{\ell_L(u)}$, but the similarity is still visible

$$\mathcal{D}_L(I, v, J) = \max_{p \in \ell_L(u)(p^*), u \in C(p^*)} \Big(I(p^*) \mathbb{1}_{\ell^+_{L^*}(u)(p^*) \subset J} \Big),$$

for $L^* = d(p^*, p)$. Finally, our adaptive opening $\Gamma^{\ell_L}_{max}(\cdot, v) : \mathscr{L}_D^{\{0,1\}} \to \mathscr{L}_D^{\{0,1\}}$ is defined as

$$\Gamma^{\ell_L}_{max}(I, v)(p) = \mathcal{D}_L(\mathcal{E}_L(I, v), v, I). \tag{3.6}$$

There is an additional characterization of our adaptive opening:

Lemma 3. *$p \in \Gamma^{\ell_L}_{max}(I, v)$ if and only if there exist a voxel $p^* \in X$ s.t. $\ell_L(u)(p^*) \subset X$ and $p \in \ell^+_{d(p,p^*)}(u^*)(p^*) \subset X$ for $u, u^* \in C(p^*)$ and $d(p, p^*) \leq L$.*

The proof goes similar as in Lemma 1 but slightly more technical. Here, voxel p is reconstructed in the dilation step from voxel p^*. This implies that the erosion preserves voxel p^*. Hence, condition $\ell_L(u)(p^*) \subset X$ comes from the adaptive erosion $\mathcal{E}_L$. The condition $p \in \ell^+_{d(p,p^*)}(u^*)(p^*) \subset X$ comes from the adaptive dilation $\mathcal{D}_L$. This interpretation will be used in the proof that this operator is indeed an opening in the sense of mathematical morphology. There is an additional result as a consequence of the previous lemma:

Corollary 2. *If there exists a voxel $p^* \in X$ s.t. $\ell_L(u)(p^*) \subset X$ and $\ell^+_{L^*}(u^*)(p^*) \subset X$ for some $u, u^* \in C(p^*)$ and $L^* \leq L$, then $\ell^+_{L^*}(u^*)(p^*) \subset \Gamma^{\ell_L}_{max}(I, v)$.*

Lemma 4. *$\Gamma^{\ell_L}_{max}(I, v)$ is indeed an opening for the fixed orientation map v on $\mathscr{L}_D^{\{0,1\}}$ according to Definition 2.*

Proof. Anti-extensivity is a consequence of connectivity-preserving dilation $\mathcal{D}_L$. Let $p \notin X$. Then for every $p^* \in X$ and for all $u \in C(p^*)$ s.t. $p \in \ell_L(p^*)(u)$, it holds $\ell_L(p^*)(u) \not\subset X$. Hence, $p \notin \Gamma^{\ell_L}_{max}(I, v)$.

The increasingness is satisfied in the context of a fixed orientation map $v : D \to S^2_+$ that is used for both images. This is shown in the following lines: assume $p \in \Gamma^{\ell_L}_{max}(X, v)$, then there exist $p^* \in X$ and $u, u^* \in C(p^*)$ at which line $\ell_L(u)(p^*) \subset X$ can be fitted and $p \in \ell^+_{d(p,p^*)}(u^*)(p^*) \subset X$. From $X \subset Y$ implies that $p, p^* \in Y$ and $\ell_L(u)(p^*), \ell^+_{d(p,p^*)}(u^*)(p^*) \subset Y$. Hence, from Lemma 3 $p \in \Gamma^{\ell_L}_{max}(Y, v)$. Hence, the increasingness is satisfied. However, here we stress the importance of the fixed orientation map v. For the different orientation maps $v_1, v_2 : D \to S^2_+$ for $X \subset Y$, respectively, the increasingness may not be satisfied.

The proof of idempotence is also similar. First, $\Gamma^{\ell_L}_{max}(\Gamma^{\ell_L}_{max}(I, v), v) \subseteq \Gamma^{\ell_L}_{max}(I, v)$ holds from anti-extensitivity.
Second, let $p \in \Gamma^{\ell_L}_{max}(I, v)$.

- Then from Lemma 3 there exist $p^* \in X$ and $u, u^* \in C(p^*)$ so that $\ell_L(u)(p^*) \subset X$ and $p \in \ell^+_{d(p,p^+)}(u^*)(p^*) \subset X$.
- From Corrolary 2 it also holds $p^* \in \ell_L(u)(p^*) \subset \Gamma^{\ell_L}_{max}(I, v)$ and $\ell^+_{d(p,p^+)}(u^*)(p^*) \subset \Gamma^{\ell_L}_{max}(I, v)$.

Hence, from Lemma 3 $p \in \Gamma^{\ell_L}_{max}(\Gamma^{\ell_L}_{max}(I, v), v)$. □

3.3.5 Adaptive line granulometry

For the fixed orientation map v, the family of functions $\{\Gamma^{\ell_L}_{max}(\cdot, v), L \in \mathbb{N}\}$ is a granulometry generating family, i.e. it satisfies the axioms from Definition 1. This will be shown in the following section. In practice, L is limited by some $L_{max} \in \mathbb{N}$ and the length of the longest fitting line in the voxel p is then found to be a function $L_{granulometry}(\cdot, v) : D \to [0, L_{max}] \cap \mathbb{N}$

$$L_{granulometry}(I, v)(p) = \underset{L \in [0, L_{max}] \cap \mathbb{N}}{\arg\max} \Big(\Gamma^{\ell_L}_{max}(I, v)(p) - \Gamma^{\ell_{L+1}}_{max}(I, v)(p) \Big). \tag{3.7}$$

For binary images, line opening at the step $L \in [1, L_{max}]$ of granulometry removes all the fibers shorter than L. The difference of line openings $\Gamma^{\ell_L}_{max}(I, v)(p) - \Gamma^{\ell_{L+1}}_{max}(I, v)(p) = 1$ only in the case when our adaptive line opening at the step L preserves voxel p and our adaptive line opening at the step $L + 1$ removes it. Otherwise, this difference always equals 0. Hence, $L_{granulometry}$ aims at recording the argument of the spike in the difference of line openings $\Gamma^{\ell_L}_{max}(I, v)(p) - \Gamma^{\ell_{L+1}}_{max}(I, v)(p)$. Keeping track of L through granulometry at which opening step each fiber voxel disappears, yields what can be interpreted as a local fiber length distribution.

This new voxel based method is similar to the path openings [62] in avoiding explicit fiber separation but differs in not adapting to curves. This reduces the effort but clearly induces an underestimation of the length of curved fiber fractions. The line segment granulometry suffers from the well-known problems in discretizing the 3d orientation space sufficiently finely if implemented naively [7]. Our adaptive morphology framework yields however a way out by restricting the searched orientations to a conic subset of the full orientation space centered about a roughly estimated preferred orientation. Within this cone, sampling the orientations very finely is affordable. Here, this framework is applied for the first time for fiber length estimation.

To summarize, our method needs the following inputs: The orientation map v and the width of the orientation cone δ_{max} about it, the maximal half-length L_{max} of the line segments used as structuring elements, and a parameter controlling fineness of the discretization n.

In the applied part of the chapter, the orientation map is obtained from eigenvalue analysis of the Hessian matrix of second order gray value derivatives as described in [7], with $\sigma = 4$. Since we aim at measuring the local fiber length, the half-length parameter $L \in [1, 350]$ for the structuring element is large, in particular increased compared to [1]. The parameters controlling

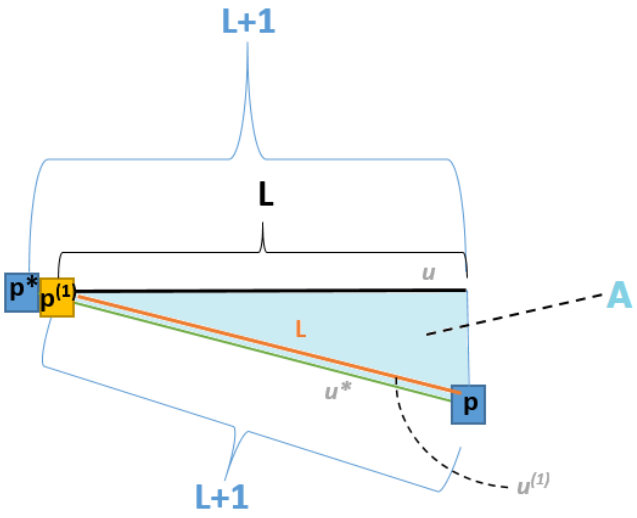


Figure 3.4: Lemma 5: graphical proof on the existence of $p^{(1)}$ and $\ell_L^+(u^{(1)}(p^{(1)}) \subset X$ (orange line) from $\ell_L(u)(p^*) \subset X$ (black line) and $\ell_L^+(u^*)(p^*) \subset X$ (green line). A denotes the area contained in X due to the convexity of the connected components of X.

the fineness of discretization $n = 1\,400$ and the width of the orientation cone $\delta_{max} = 0.2$ are kept. The latter two result in 368 201 discrete orientations for the whole orientation space restricted to 7 340 falling into the cone.

3.3.6 Proof of granulometry axioms

Adaptive line granulometry is specific in the sense that every voxel uses the initial orientation map v as an input which varies across the image domain. This implies that the granulometry axioms from Section 3.3.1 are not trivially satisfied. The key condition is that the orientation map v is estimated at the beginning and the same orientation map v is used through the family of granulometric functions.
However, we need additional assumptions on the image X and orientation map v. Motivated by the observation that fibers are often modelled as cylinders [7], we assume that every connected component of X is convex. Additionally, we expect that the orientation map v allows only for gradual changes across the spatial domain. In the context of measuring fiber length, one would expect that the orientation map v would be similar in the voxels which belong to the same fiber. These two assumptions are used in the following lemma.

Lemma 5. *Let $X \in \mathscr{L}_D^{\{0,1\}}$ be a binary image s.t. every connected component is convex and v be its orientation map which is sufficiently smooth, i.e. for $p_1, p_2 \in X$ s.t. $d(p_1, p_2) = 1$ it holds $d(v(p_1), v(p_2)) < C << \delta_{max}$ for some $C > 0$. Then, it holds*

$$\Gamma_{max}^{\ell_{L+1}}(X, v) \subseteq \Gamma_{max}^{\ell_L}(X, v).$$

Proof. Let $p \in \Gamma_{max}^{\ell_{L+1}}(X, v)$, then from Lemma 3 there exist $p^* \in X$ and $u, u^* \in C(p)$ s.t. $\ell_{L+1}(u)(p^*) \subset X$ and $\ell^+_{d(p,p^*)}(u^*)(p^*) \subset X$ for $d(p, p^*) \leq L + 1$.

- If $d(p,p^*) \leq L$, then by using $\ell_L(u)(p^*) \subset \ell_{L+1}(u)(p^*)$ in Lemma 3, it implies $p \in \Gamma^{\ell_L}_{max}(X,v)$.
- On the other hand, let $d(p,p^*) = L+1$.
  - Let $\overset{\frown}{uu^*} = \{w \in S^2_+ \mid d(w,u), d(w,u^*) \leq d(u,u^*)\} \subset S^2_+$ be a circle arc[2] between u and u^*. Then $\overset{\frown}{uu^*} \subset C(p^*)$.
  - Since $\ell^+_{L+1}(u)(p^*), \ell^+_{L+1}(u^*)(p^*) \subset X$ and connected components of X are convex, it holds $A = \{q \in \ell^+_{L+1}(w)(p^*) \mid w \in \overset{\frown}{uu^*}\} \subset X$ (Figure 3.4).
  - Then there exists $p^{(1)} \in \ell_{L+1}(u)(p^*)$ s.t. $d(p^{(1)},p^*) = 1$ and $d(p^{(1)},p) = L$ (Figure 3.4). Because of the smoothness of the orientation map, it holds $u \in C(p^{(1)})$, $\ell_L(u)(p^{(1)}) \subset X$, and there exists $u^{(1)} \in C(p^{(1)})$ s.t. $p \in \ell^+_L(u^{(1)})(p^{(1)})$. From the convexity assumption it follows that $\ell^+_L(u^{(1)})(p^{(1)}) \subset A \subset X$.
  - Now, we apply Lemma 3 for $p^{(1)}, u$, and $u^{(1)}$ to show $p \in \Gamma^{\ell_L}_{max}(X,v)$.

□

Lemma 6. *Under the assumptions from Lemma 5,* $\left(\Gamma^{\ell_L}_{max}(\cdot,v)\right)_{L>0}$ *is a granulometry for fixed orientation map* v *on* $\mathscr{L}_D^{\{0,1\}}$ *according to Definition 1.*

Proof. 1. **anti-extensivity** from equation (3.1) is an immediate consequence of the anti-extensivity of the adaptive line opening $\Gamma^{\ell_L}_{max}(\cdot,v)$.

2. **increasingness** from equation (3.2) is an immediate consequence of the increasingness of the adaptive line opening $\Gamma^{\ell_L}_{max}(\cdot,v)$.

3. **absorption** is also satisfied. If we assume $M < L$, then $\ell_M(u) \subset \ell_L(u)$ for all $u \in S^2_+$.
   - Lemma 5 gives the following result for $L > M$:
   $$\Gamma^{\ell_L}_{max}(X,v) \subseteq \Gamma^{\ell_M}_{max}(X,v). \tag{3.8}$$
   - Also, applying $\Gamma^{\ell_M}_{max}(\cdot,v)$ on $\Gamma^{\ell_L}_{max}(X,v)$ does not affect the image. First, from anti-extensivity of the opening we have $\Gamma^{\ell_M}_{max}(\Gamma^{\ell_L}_{max}(X,v),v) \subseteq \Gamma^{\ell_L}_{max}(X,v)$.
   Second, from idempotence of the opening and equation (3.8), it follows $\Gamma^{\ell_L}_{max}(X,v) = \Gamma^{\ell_L}_{max}(\Gamma^{\ell_L}_{max}(X,v),v) \subseteq \Gamma^{\ell_M}_{max}(\Gamma^{\ell_L}_{max}(X,v),v)$. Hence, it holds
   $$\Gamma^{\ell_M}_{max}(\Gamma^{\ell_L}_{max}(X,v),v) = \Gamma^{\ell_L}_{max}(X,v).$$
   - On the other hand, we have $\Gamma^{\ell_L}_{max}(\Gamma^{\ell_L}_{max}(X,v),v) \subseteq \Gamma^{\ell_L}_{max}(\Gamma^{\ell_M}_{max}(X,v),v)$ from increasingness of the opening and equation (3.8). Now, from idempotence of the opening $\Gamma^{\ell_L}_{max}(X,v) = \Gamma^{\ell_L}_{max}(\Gamma^{\ell_L}_{max}(X,v),v)$, it follows
   $$\Gamma^{\ell_L}_{max}(X,v) \subseteq \Gamma^{\ell_L}_{max}(\Gamma^{\ell_M}_{max}(X,v),v).$$
   - Reverse also holds from $\Gamma^{\ell_M}_{max}(X,v) \subseteq X$. Hence, due to the increasingness of the opening
   $$\Gamma^{\ell_L}_{max}(\Gamma^{\ell_M}_{max}(X,v),v) \subseteq \Gamma^{\ell_L}_{max}(X,v).$$

   This proves equation (3.3).

□

[2]Here, $d : S^2_+ \times S^2_+ \to [0,\pi]$ refers to angle distance metric from equation (2.2). When measuring the distance between voxels we slightly abuse notation by again using d as the distance function. However, in this case $d : \mathbb{R}^3 \times \mathbb{R}^3 \to \mathbb{R}_+$ denotes Chebysev (maximum) distance d_∞ from Section 2.3.

3.3.7 Implementation details

Here, we give an efficient algorithm for our adaptive line granulometry, see Algorithm 1. Our algorithm is split in two steps: erosion and dilation step.
In the erosion step, for voxel p we walk on the every line in its search cone starting from p until we reach the inactive (null, non-binary) voxel or until L_{max}-th voxel. We record the number of voxels on the line between p and this voxel, i.e. the distance on the line. We do the same for the reflected line as well and record the minimum of these two distances of voxels. Finally, we take the maximum of these distances over all the lines in the search cone. This represents the maximal length of the symmetric line that can be fitted in voxel p. We refer to this length map as an erosion length map or $E : D \to [0, L_{max}] \cap \mathbb{N}$ in Algorithm 1.
In the dilation step, we use the erosion length map E as an input. The erosion length map tells us the length of the line that should be used in the dilation step for every voxel. For every voxel p, we record the maximum of $E(p^*)$ for those pixels p^* for which there exists a line in their search cone so that the line passes through p. In practice, we do this by initializing the granulometry length map $L_{granulometry}$ with E (denoted with G in Algorithm 1). Then starting at every voxel p^* and walking over voxels p in every line of length $E(p^*)$ in the search cone, we compare $E(p^*)$ with the granulometry length $G(p)$ and update it with $E(p^*)$ if it is larger than the current granulometry length in p. Walking on a line is stopped if we reach a null pixel and moved on to the next line in the search cone.

3.3.8 Length weighted length and orientation distributions

The number weighted fiber length distribution refers to the distribution for which only one length is recorded per fiber. This is intuitive since we see a fiber as one unit with one length. The number weighted fiber length distribution motivates single fiber segmentation as a method of choice in many applications. However, in this section we explain how to alternatively estimate the number weighted length distribution from the adaptive line granulometry.

Any probability distribution of a random variable (e.g. length or orientation) can be estimated empirically from the observations as a set of pairs $H = \{(val_i, w_i); i \in \{1, \cdots, N\}\}$, where val_i denotes the value of the measured characteristic, w_i its weight in the distribution, i is an index, and $N \in \mathbb{N}$ is the number of pairs. Based on this, one can define a normalized frequency of the interval $\langle x_1, x_2]$ as $F_H(x_1, x_2) = \frac{\sum_{i, x_1 < val_i \leq x_2} w_i}{\sum_i w_i}$. Usually, when sampling voxelwise length or orientation in the image, equal weights are given to every voxel, i.e. $w_i = 1$.

Length weighted length distribution

The output of granulometry $L_{granulometry}$ is a map which estimates the length for every voxel in the image. The resulting fiber length histogram is not the empirical number weighted fiber length distribution. On the one hand, the histogram is length weighted as longer fibers cover more voxels. On the other hand, sample cutting results in cut fibers, too, and longer fibers are more likely to be affected. The length weighting is removed by weighing each histogram bin with the respective fiber length under the assumption of the constant diameter of fibers in the sample. Formally, the voxel weighted length distribution is estimated from a set

$$H^{len}_{voxel} = \{(L_g(p), 1) \mid p \in D_0\},$$

Algorithm 1 Adaptive line granulometry

procedure ADAPTIVE-LINE-GRANULOMETRY($X, n, \delta_{max}, L_{max}, v$)
Input binary image X,
sphere discretization n,
cone width δ_{max},
maximal length L_{max},
orientation map v.
Output image G, # to be initialized later
Temporary image E, set to 0 for all voxels
# Erosion step
for voxel $p \in X$ **do**
for orientation $u \in C(p)$ **do**
for $i = 1 : L_{max}$ **do**
if $\ell_{L_{max}}(u)(p)[i] \notin X$ or $-\ell_{L_{max}}(u)(p)[i] \notin X$ **then**
if $i > E(p)$ **then**
E(p) = i-1;
Break;
end if
end if
# Case we reached L_{max}
if $i = L_{max}$ **then**
$E(p) = L_{max}$;
end if
end for
end for
end for

#Dilation step
G= E.copy();
for voxel $p^* \in X$ **do**
for orientation $u \in C(p^*)$ **do**
for $i = 1 : E(p^*)$ **do**
$p = \ell_{E(p^*)}(u)(p^*)[i]$
if $p \notin X$ **then**
Break;
end if
if $G(p) < E(p^*)$ **then**
$G(p) = E(p^*)$;
end if
end for
# Reflection of line around p
for $i = 1 : E(p^*)$ **do**
$p = -\ell_{E(p^*)}(u)(p^*)[i]$
if $p \notin X$ **then**
Break;
end if
if $G(p) < E(p^*)$ **then**
$G(p) = E(p^*)$;
end if
end for
end for
end for
Return G
end procedure

where $L_g := L_{granulometry}(I, v)$ is a granulometry from the previous sections[3] and $D_0 = \{p \in D \mid L_g(p) > 0\} \subseteq D$. Now, the number weighted length distribution can be estimated as

$$H^{len}_{number} = \{(L_g(p), \frac{1}{L_g(p)}) \mid p \in D_0\}.$$

The bias due to the incomplete observation could be reduced by so-called minus-sampling or weighing the lengths with the reciprocal of the probability of observing fibers of this length at all (Miles-Lantouejoul method [63]). However, the focus here lies on the differences of the fiber components in the two materials under consideration in the next section. This means that we implicitly assume that the edge effects for both types of materials are either similar or negligible and do not disturb the comparability.

Length weighted orientation distribution

Similarly as for granulometry, we get voxelwise information on orientation. Note that this is again the length weighted distribution because it is defined per voxel and the orientation distribution is shifted towards the orientation of the long fiber since they consist of more voxels than short ones. For conversion to the number weighted orientation distribution, adaptive line granulometry is needed: orientation in every voxel is given a weight inversely proportional to the value of the adaptive line granulometry in that voxel. Given the weights, the number weighted orientation distribution is estimated.
Formally, the voxel weighted orientation distribution is estimated as a set

$$H^{o}_{voxel} = \{(v(p), 1) \mid p \in D_0\}.$$

Now, the number weighted orientation distribution can be estimated as

$$H^{o}_{number} = \{(v(p), \frac{1}{L_g(p)}) \mid p \in D_0\}.$$

3.4 Materials and imaging: injection moulded vs 3d printed

3.4.1 Fiber material and specimen production

Materials under investigation are fiber reinforced polymers (FRP) which are made using two different production techniques: injection moulding and 3d printing[4]. Both types of material under investigation have 30 % nominal fiber mass content.

Injection moulded FRP The moulded material is a Sabic STAMAX 30YK270E[5]. Blocks of 26 mm × 7 mm × 2 mm were cut out via micro water jet cutting from 280 mm × 80 mm × 2 mm moulded plates 97 mm distal from film gate in the center (see Figure 3.5, left). The blocks were divided in two stripes, which were polished afterwards. Five micro specimens in 0° and 30° directions to the moulding direction were cut out per stripe. The upper left corner of each specimen was trimmed by a 45° angle, to reproduce the extraction directions and position within the plate.

[3] This is introduced for more compact notation.
[4] Here, a Fused Deposition Modeling (FDM) 3d printer is used which works by depositing melted filament material made of thermoplastic polymers layer by layer until the full object is formed.
[5] See www.sabic.com for further information on material and manufacturing process.

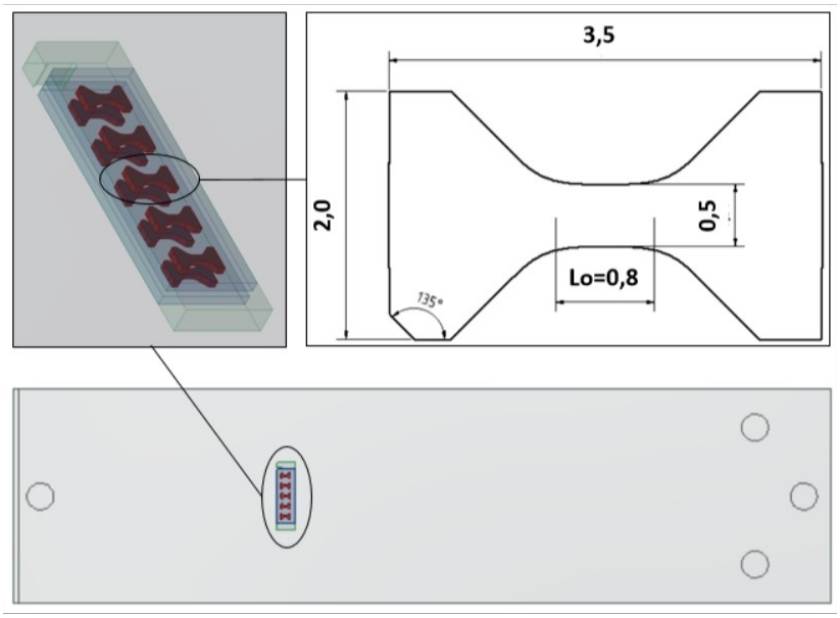


Figure 3.5: Micro sample extraction from injection molded (IM) plates (bottom), slicing extracted blocks in two stripes (upper left) and sample geometry in mm (upper right).

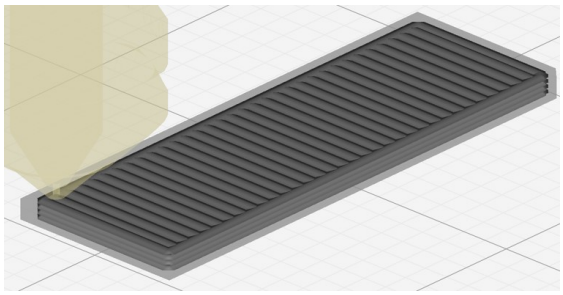


Figure 3.6: Producing stripes for sample extraction via 3d printing.

3d printed FRP The material for 3d printing is BASF filament Ultrafuse PP GF3[6]. Stripes were laid down in four layers, each 0.2 mm high, using an Ultimaker S5. The printing trajectory was chosen such that the fiber orientation in all layers is similar to 0° in injection moulding. The printing direction, except the wall layer, was unidirectional in all layers (see Figure 3.6). Apart from these lay-up adjustments, the filament manufacturer's printing parameters for the Ultimaker S5 were applied. After printing and polishing in exactly the same way as the moulded samples, micro specimens were cut as described in Section 3.4.2.

3.4.2 Micro specimens

As shown in Figure 3.5 (right), two stripes were cut out from the block (see Figure 3.5, green transparent) with a water jet cutting machine Daetwyler WJ F4B2, followed by polishing symmetrically from both sides with diamond paste (3 μm) to $\approx$ 500 μm thin stripes. Figure 3.7 shows

[6]See www.forward-am.com for further information on material and manufacturing process.

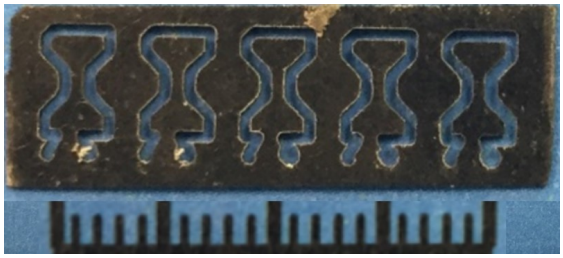


Figure 3.7: Samples extracted from polished stripes via micro water-jet cutting

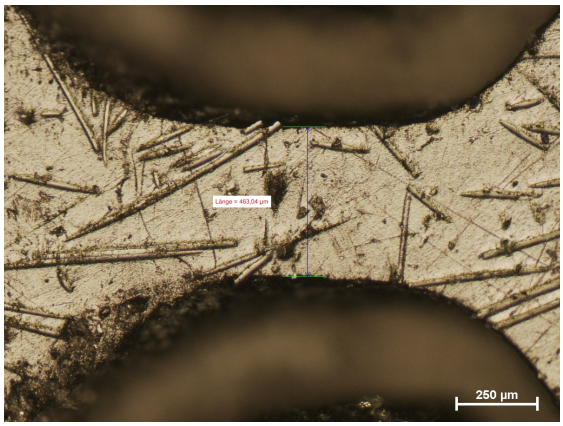


Figure 3.8: Microscopic view on the gauge area of a micro sample used in this work

the water jet cut micro specimens before the final manual extraction from the polished stripe. The width of the specimens in the center is measured by light microscopy (see Figure 3.8). The thickness of the stripes is determined before cutting using a micrometer gauge.

3.4.3 3d imaging by computed tomography

Fraunhofer ITWM's CT device is used for 3d imaging of the samples. The CT device is equipped with a Feinfocus FXE 225.51 X-ray tube with maximum acceleration voltage 225 kV, maximum power 20 W, and a Perkin Elmer flat bed detector XRD 1621 with 2 048 × 2 048 pixels. The tube voltage of 110 kV is rather low and the integration time of 1 s high, in particular to enable separation of air (pores and background) and matrix material by gray value. Tomographic reconstructions are obtained from 1 200 projections, each averaged over five, resulting in an overall integration time of 5 s. The voxel edge length of 2 μm ensures proper resolution of the fiber diameter. A representative slice as well as a volume rendering are shown in Figure 3.9.

Additional to the samples described above, a sample of the original, unprocessed Ultrafuse PP GF30 filament was imaged at voxel edge length 1.8 μm (Figure 3.11).

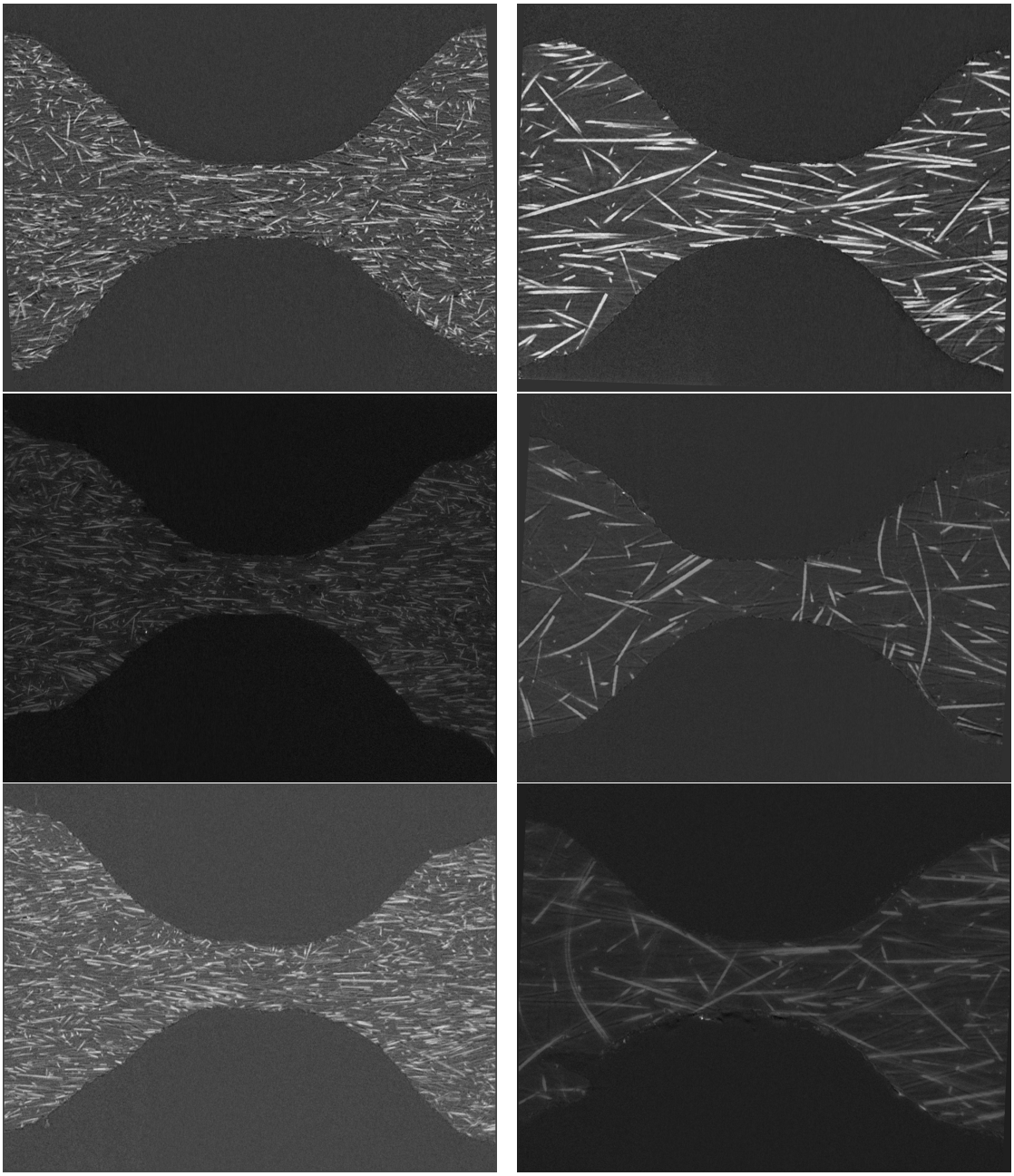


Figure 3.9: Representative reconstructed CT images: slice views. Left: 3d printed sample. Right: Injection molded sample. Visualized are approximately $1\,200 \times 1\,500 \times 100$ voxels corresponding to $2.4\,\text{mm} \times 3\,\text{mm} \times 0.2\,\text{mm}$. Note that the fiber component of the printed material (left) appears denser in the volume rendering in spite of identical fiber volume fractions. This is a purely visual effect due to the thinner and therefore more fibers.

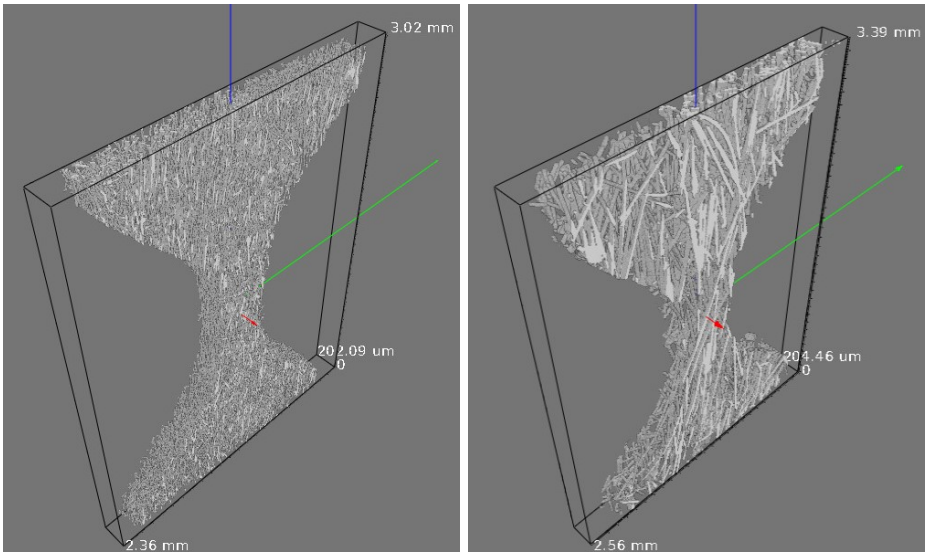


Figure 3.10: Representative reconstructed CT images. Renderings of the fiber component. Left: 3d printed sample. Right: Injection molded sample. Visualized are approximately $1\,200 \times 1\,500 \times 100$ voxels corresponding to $2.4\,\text{mm} \times 3\,\text{mm} \times 0.2\,\text{mm}$.

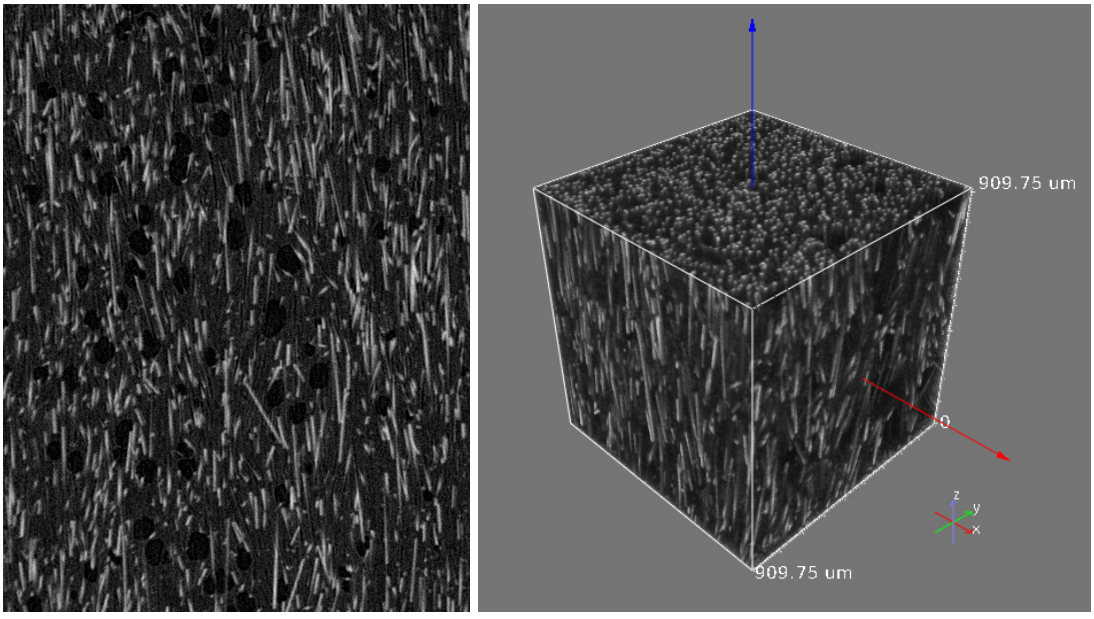


Figure 3.11: CT image of filament used for 3d printed samples: xy slice ($1\,000 \times 1\,300$) and 3d rendering ($500 \times 500 \times 500$ voxels) corresponding to representative subvolume of size $0.9\,\text{mm} \times 0.9\,\text{mm} \times 0.9\,\text{mm}$. Voxel edge length is 1.8 μm. During 3d printing the filament is melted and passed through the heated extrusion nozzle to create a layer of 3d printed material.

3.5 Application: comparing fiber length and orientation

Here, we report the results of our analysis on three samples for each of these two types of FRP. In the following figures and tables, we will use 3DP-1, 3DP-2, and 3DP-3 to mark 3d printed samples. Three injection moulded samples will be referred to as IM-1, IM-2, and IM-3. For every sample $L_{granulometry}$ is calculated for the parameter configuration from Section 3.3.4. The length weighted length and orientation distributions are corrected to number weighted counterparts according to Section 3.3.8.

3.5.1 Fiber Lengths

	25% quantile (µm)	Median (µm)	75% quantile (µm)
filament	52	103	182
3DP-1	54	94	166
3DP-2	50	94	166
3DP-3	50	90	158
IM-1	178	374	590
IM-2	182	378	594
IM-3	174	374	590
mean(IM)/mean(3DP)	3.47	4.05	3.62

Table 3.1: Estimates for fiber length quantiles per sample.

The analysis confirms the visual impression: fibers in the 3d printed samples are in general shorter than in the injection molded ones. However, 3d adaptive granulometry enables more in-depth analysis. Length histograms are shown in Figure 3.12, 2d slices with the color mapped lengths in Figure 3.13. Quantiles of the empirical length weighted distributions are given in Table 3.1. The last row of Table 3.1 shows the ratio of mean quantile length of IM samples to mean quantile length of 3DP samples. This gives a quantification that in general fibers in IM samples are between 3.5 and 4 times longer than the fibers in 3DP samples. Length histograms from Figure 3.12 shows the different shapes of distributions of 3DP and IM samples. Length histograms of 3DP samples are highly concentrated up to 100 μm with fast decay towards higher lengths, e.g. very low number of fibers exists for length around 300 μm. On the other hand, length histograms of IM samples have fatter tails at both ends of the length histograms, i.e. in IM samples we observe both very long (above 600 μm) and very short (below 100 μm) fibers.

Table 3.1 contains the quantiles for the filament sample as well (Figure 3.11). The filament image was cropped to $1\,000 \times 1\,300 \times 1\,000$ voxels. Contrary to the printed samples, the filament is quite porous. As a consequence, the fiber volume fraction in the filament sample is in the range $\langle 8.26\%, 12.08\% \rangle$ depending on the gray value threshold chosen for binarization, which is lower compared to the average volume fraction of the 3DP printed sample $\langle 11.67\%, 18.13\% \rangle$. In spite of these slight ambiguities the fiber length analysis proves what had to be expected: There are more longer fibers in the filament. Note that the 75 % quantile might be even higher in reality, but the longer the fibers are, the more likely they are cut off by the sample or image edges.

3.5.2 Fiber Orientation

In order to be consistent with the fiber length analysis, the line granulometry map was used for the orientation estimation, too. That means, the local fiber orientation in a voxel is the one of

sample	mean μ	concentration κ	shape index	strength index
3DP-1	(-0.03, 0.99, 0.004)	5.31	1.71	4.87
3DP-2	(-0.12, 0.99, 0.006)	5.46	1.71	4.86
3DP-3	(-0.10, 0.99, 0.009)	4.96	1.71	4.86
IM-1	(-0.07, 0.99, -0.010)	5.97	3.61	5.16
IM-2	(-0.11, 0.99, 0.015)	5.47	3.61	5.15
IM-3	(-0.08, 0.99, -0.004)	5.87	3.61	5.15

Table 3.2: Comparison of orientation distributions: estimation of the parameters.

the longest line segment passing through this voxel and fitting into the fiber system.

The samples do not differ strongly with respect to fiber orientation. All feature a strong prevalence of the y-direction (the horizontal one in the slices visualized in Figures 3.9 and 3.13), see also the mean fiber orientations in the 2nd column of Table 3.2. Fitted von Mises-Fisher distributions[7] differ only slightly in the concentration parameter as reported in the 3rd column of Table 3.2, with the injection moulded samples featuring higher concentration. The spherical density plots in Figure 3.14 reveal a tendency of the fiber orientations in the 3d printed samples to deviate out-of-plane. This becomes manifest in the red to yellow blobs indicating higher concentrations being more roundish, while in the injection moulded samples orientations rather vary in-plane and thus the high concentration areas are rather elliptical. It would be interesting to further link the orientation distribution to the length distribution and differences in the production process (injection moulding vs 3d printing). However, this is beyond the scope of the thesis.

These observations are backed by the shape and strength indices of the orientation distributions according to [64] given in Table 3.2, too. These distribution free summary statistics for spherical data capture the shape varying from girdle (0) to clustered (∞) and how strong this shape actually is, i.e. the greater the strength statistics, the better it is described by the shape index. The fiber orientations in the IM samples are more clustered and this shape is slightly clearer.

[7] This was done using Maximum Likelihood Estimate (MLE) using R-package function *vmf.mle* from library *CRAN - Package Directional*, see https://cran.r-project.org/web/packages/Directional/index.html.

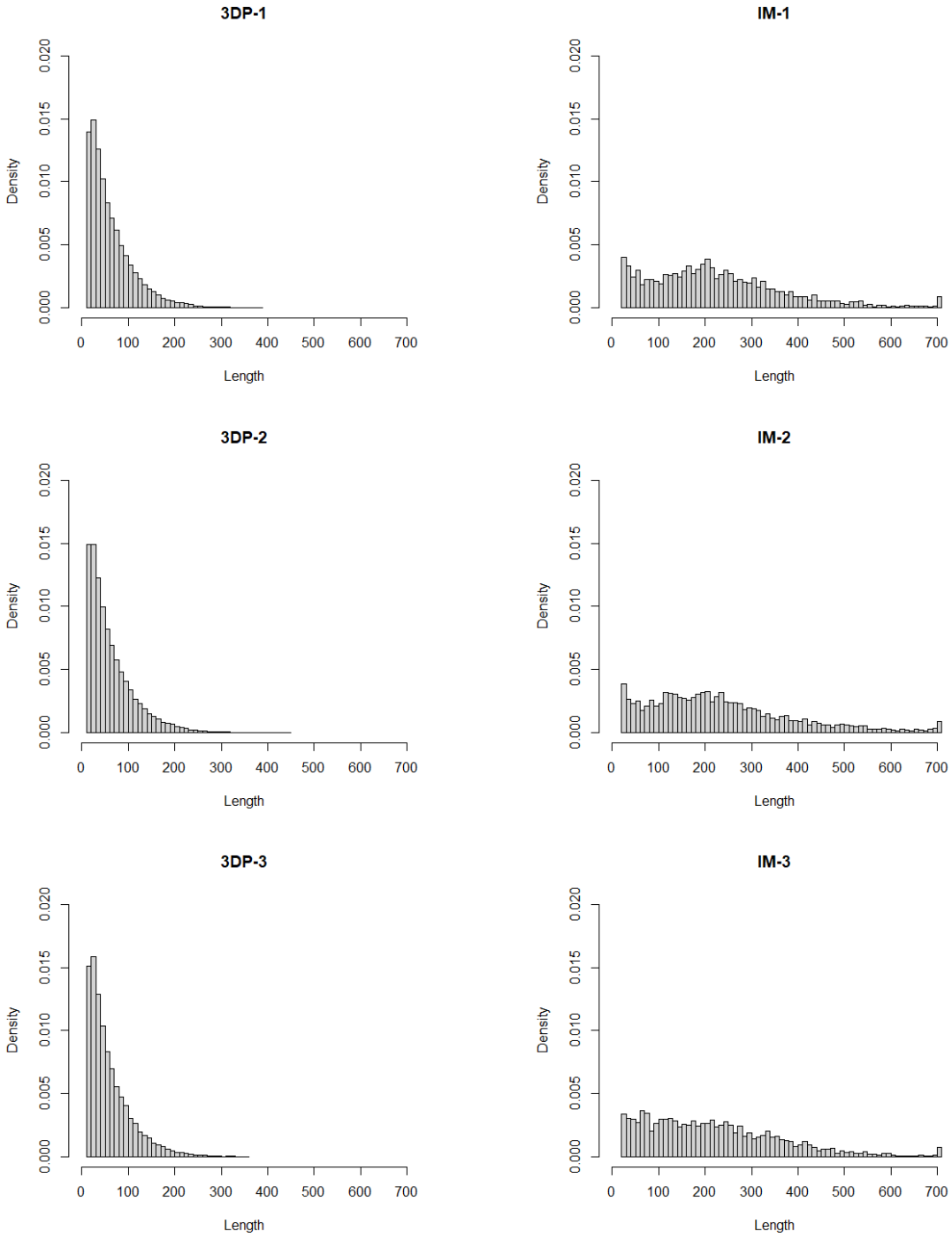


Figure 3.12: Length weighted, uncorrected empirical length distributions for the two samples shown in Figure 3.9. Left: 3d printed sample. Right: injection molded sample.

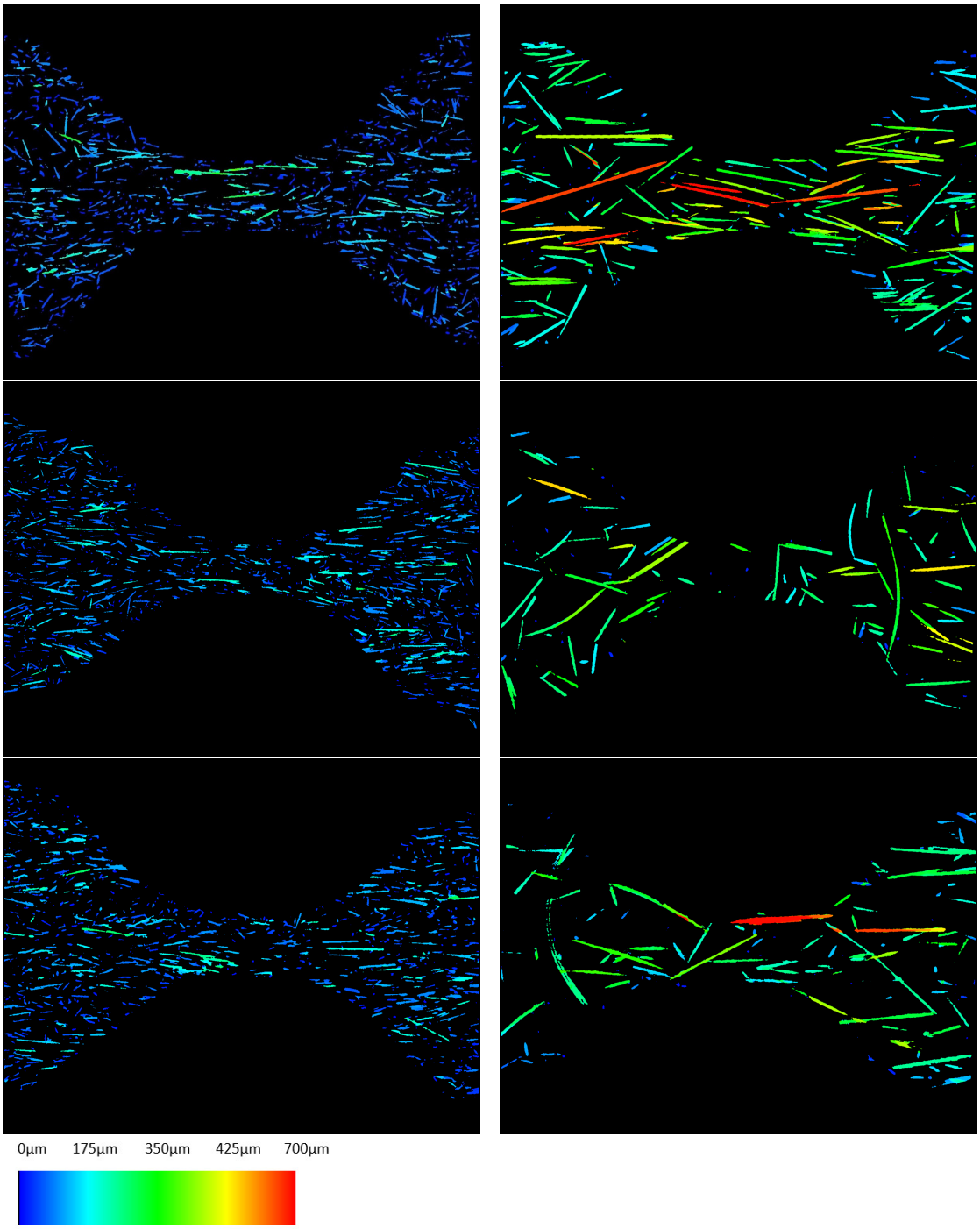


Figure 3.13: Color coded length map for the two samples shown in Figure 3.9. Left: 3d printed samples (from top to bottom: 3DP-1, 3DP-2, and 3DP-3). Right: injection molded samples (from top to bottom IM-1, IM-2, and IM-3).

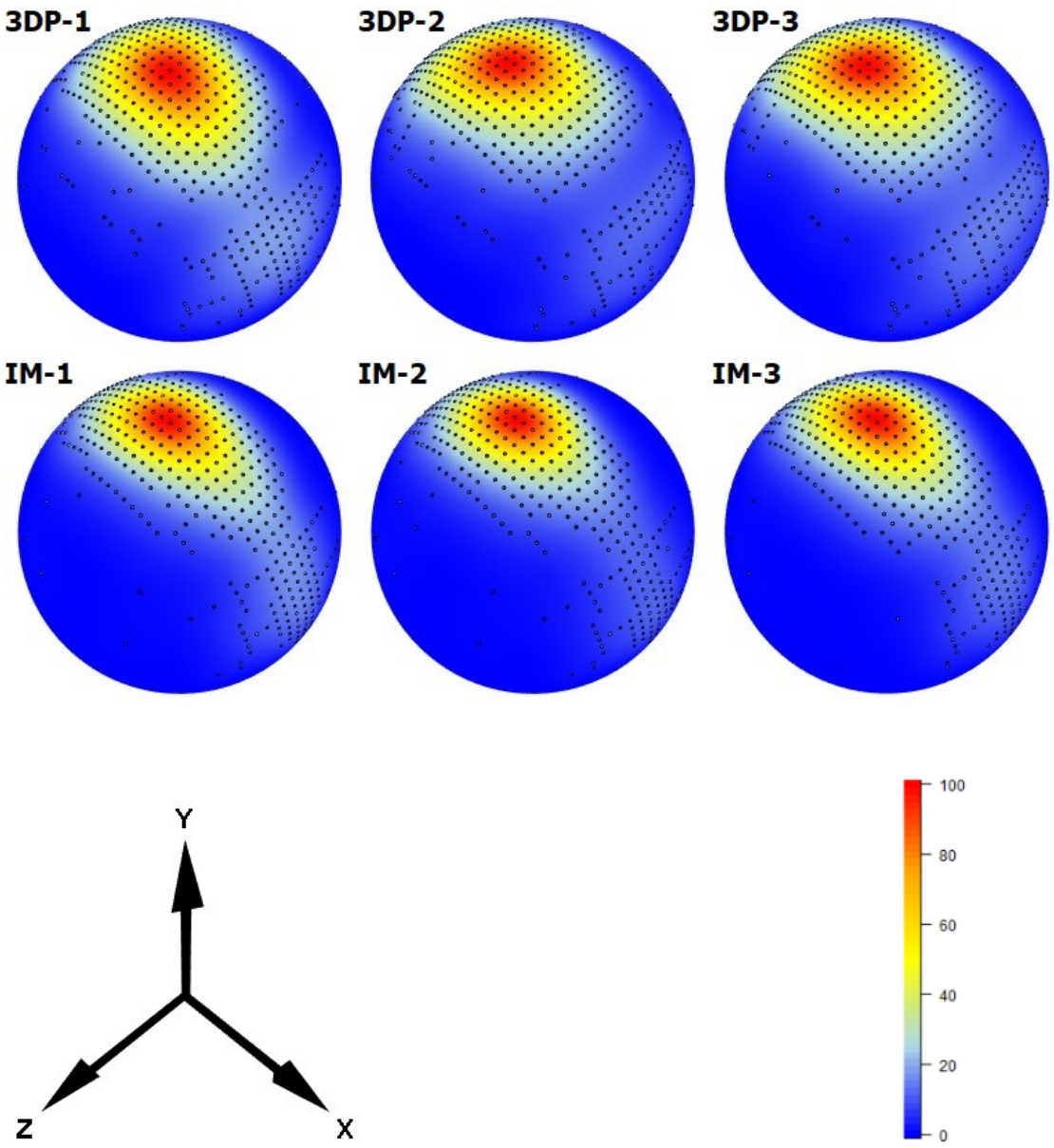


Figure 3.14: Color coded empirical orientation distributions. Top: 3d printed samples (3DP-1, 3DP-2, and 3DP-3). Bottom: injection molded samples (IM-1, IM-2, and IM-3).

3.6 Discussion

This chapter constructs 3d adaptive line granulometry based on the idea of 3d adaptive morphology from the previous chapter. This method relies on voxelwise information and avoids single fiber segmentation. Fiber length distribution is measured using line structuring element. Mathematical formulation of the method and its relationship with the concepts from mathematical morphology are thoroughly presented and discussed. 3d adaptive line granulometry extension enables the estimation of the number weighted fiber length and orientation distribution, as shown in the comparison study between injection moulded and 3d printed fiber reinforced polymers.

The major drawback of the introduced method is the inability to measure curved fibers due to the use of a straight (line) structuring element. Literature suggests that path openings should be suitable for this task [62]. The possible direction for future research could be the investigation if the path opening algorithm can be combined with the idea of orientation estimation from the local image information to reduce the number of paths that need to be covered. This is particularly relevant for 3d applications due to the large number of possible paths.

Another missing ingredient of this method is the adjustment for the edge effect due to the cutting of the fiber during the specimen preparation. To gain the correct and unbiased fiber length and orientation distribution these effects need to be accounted for. This could be done by reweighting the voxel weighted length distribution based on their distance to the boundary, i.e. as fiber is closer to the boundary, the more likely it will be cut during the sample preparation and hence altering the original length distribution.

Part II

Construction of scale equivariant deep and scattering networks based on the Riesz transform

Chapter 4

Introduction

The Riesz transform is a singular integral operator[1] which enables the decomposition of the high dimensional signal into instantaneous phase, frequency, and amplitude. Furthermore, the Riesz transform has two properties which make it a good candidate for feature extraction: a differential interpretation and scale equivariance. Generally, there are only few studies in the image processing and computer vision community regarding the Riesz transform. So far, major research efforts related to the Riesz transform were mainly concentrated in the signal processing community. In the following chapters we aim at bridging the gap in understanding the Riesz transform and its properties and relating them to possible applications. Furthermore, we show how to embed the Riesz transform in a popular deep learning framework to achieve scale invariance (Chapter 6) and how to design a Riesz scattering representation (Chapter 8), a simple mathematical model of neural networks which requires a small number of training examples. Chapter 6 is based on [3], while Chapter 8 is contained in [4]. These methods based on the Riesz transform are able to achieve generalization to scales that are previously unseen during training. This is useful for many applications where objects occur at different scales. Hence, only a few scales have to be present in the training set for Riesz-based methods to achieve a good performance on a wider range of scales. This is in contrast with standard methods in deep learning.

Our main motivation for using the Riesz transform is based on the observation that cracks in CT images occur in a large range of scales. Hence, scale invariance is identified as one of the key properties of crack segmentation methods. The main goal of Part II is to design methods that can segment cracks of arbitrary width. Although crack segmentation of CT images is a 3d problem, we start by developing methods based on the Riesz transform for 2d. There are several reasons for this. Firstly, scale equivariance in 2d images is an active field of research where a unique solution currently does not exist. We believe the Riesz transform is a viable solution to this problem. Secondly, development of novel methods is easier for 2d, since 2d images are more tractable, smaller (less memory hungry), and easier to analyze than the 3d counterpart. Afterwards, we switch to 3d, as a main goal of our research. From theoretical perspective this switch to the 3d is trivial. However, from a practical point of view memory requirements for 3d case are significantly higher. Hence, we introduce adjustments to deal with this issue.

In this chapter we give details on potential theory related to the invention of the Riesz transform, introduce basic notions of equivariance and invariance that will be extensively used in Part II, and give details on the Hilbert transform and the analytic signal as 1d counterpart of the Riesz transform and the monogenic signal. In this chapter we do not give a formal (explicit) definition of the Riesz transform, but rather an implicit one through the historical context of

[1] Singular integral operator is a convolution operator with a function that has a singularity at the origin.

Section 4.1. However, although the explicit definition is not a hard requirement to understand the content of this chapter, one is encouraged to skip forward to Section 5.2 of the following chapter for the formal definition.

4.1 Origin of the Riesz transform

The Riesz transform is called after Marcel Riesz, a Hungarian mathematician who made several contributions to harmonic analysis and potential theory in the 20th century. Marcel Riesz [65] proved that the Riesz potential is an inverse for the power of the Laplace operator on Euclidean space. The significance of this is that Laplace operators occur in many differential equations describing physical phenomena. For example, the diffusion equation describes heat and fluid flow, while the wave equation models wave propagation.

Formally, let $f : \mathbb{R}^d \to \mathbb{R}$ be a locally integrable function. Then for $0 < \alpha < d$, the **Riesz potential** $I_\alpha(f) : \mathbb{R}^d \to \mathbb{R}$ is defined as

$$I_\alpha(f)(x) = \frac{1}{c_\alpha} \int_{\mathbb{R}^d} \frac{f(y)}{|x-y|^{d-\alpha}} dy, \tag{4.1}$$

where $c_\alpha = \pi^{n/2} \frac{\Gamma(\alpha/2)}{\Gamma((d-\alpha)/2)}$ is a constant with Γ denoting the gamma function. This is also written as

$$I_\alpha(f) = (-\triangle)^{-\alpha/2} f \tag{4.2}$$

in a sense of an inverse of the Laplace operator $\triangle$. The connection between Riesz potential and Riesz transform is the following: the Riesz transform $\mathcal{R}$ is the differential of the Riesz potential for $\alpha = 1$:

$$\mathcal{R} = \frac{\partial}{\partial_x} I_1, \tag{4.3}$$

where $\frac{\partial}{\partial_x}$ is the differentiation operator. The Riesz transform is the only scale (dilation) equivariant operator among all differential operators of the Riesz potential DI_α for $0 < \alpha < d$. Connections between function f, its Riesz potentials and Riesz transform are given by the Sobolev inequalities. The first result is also known as Hardy–Littlewood–Sobolev fractional integration theorem.

Theorem 1. *Let $f \in L_p(\mathbb{R}^d)$, $0 < \alpha < d$, and $1 < p < d/\alpha$. Then for $q > 0$ defined via*

$$\frac{1}{q} = \frac{1}{p} - \frac{\alpha}{d},$$

there exists a constant C depending only on p such that

$$||I_\alpha(f)||_q \leq C||f||_p. \tag{4.4}$$

The proof of the theorem can be found in [66](Chapter 5, §1.3, Theorem 1). Furthermore, authors in [67] establish L_1-type estimates for I_α, known as L_1-Sobolev inequalities. That is, for $p = 1$ and $\frac{1}{q} = 1 - \frac{\alpha}{d}$, there exist a constant C' s.t.

$$||I_\alpha(f)||_q \leq C'||\mathcal{R}(f)||_1. \tag{4.5}$$

The significance of L_1-Sobolev inequalities is in the fact that boundedness of the Riesz transform implies boundedness of the Riesz potential, i.e. of the whole family of functions for $0 < \alpha < d$.
The scale equivariance of the Riesz transform, the fractional Laplacian interpretation of the Riesz potential from equation (4.2) and L_1-Sobolev inequalities (4.5) motivate the need to explore the Riesz transform as a tool for feature extraction and image analysis.

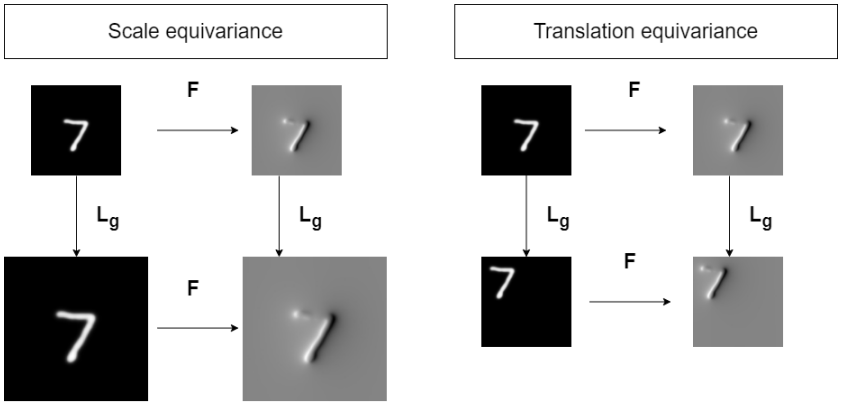


Figure 4.1: Illustration of equivariances on the MNIST dataset [68]. F represents the first order Riesz transform $\mathcal{R}_1$ from equation (5.1), while L_g is a (semi-)group operation. We show the examples for rescaling and translation.

4.2 Equivariance and invariance

Here, we give a formal definition and an intuitive interpretation of equivariance and invariance. Let $\mathscr{L} = \{f : \mathbb{R}^d \to \mathbb{R}\}$ be a set of functions that we refer to as images. Let $(G, .)$ be a group or semigroup. Let $L_g : \mathscr{L} \to \mathscr{L}$ be a family of operators that transform an image based on group element $g \in G$ defined as $L_g(f) = f(x.g^{-1})$ for $f \in \mathscr{L}$. The operator $F : \mathscr{L} \to \mathscr{L}$ is said to be equivariant to L_g if

$$F(L_g(f)) = L_g(F(f)), \quad \forall f \in \mathscr{L},$$

or invariant to L_g if

$$F(L_g(f)) = F(f), \quad \forall f \in \mathscr{L}.$$

Visualizations of these properties are given in Figure 4.1 and Figure 4.2. An additional example on equivariance is given in Figure 5.2. Invariance is a stronger property than equivariance. It implies that applying operator L_g does not have any effect on the result of operator F. In the context of images, equivariance implies that the order in which we apply the operators F and L_g on the input image does not matter. Invariance implies that operator F removes the effect of operator L_g. In applications, it is useful to have equivariant feature extractors, which serve as a input to a classifier which should be invariant to certain transformations, i.e. classification should not be sensitive to applying these transformations. For example, usually we want to achieve invariance to translation, rotation, or rescaling (Figure 4.2).

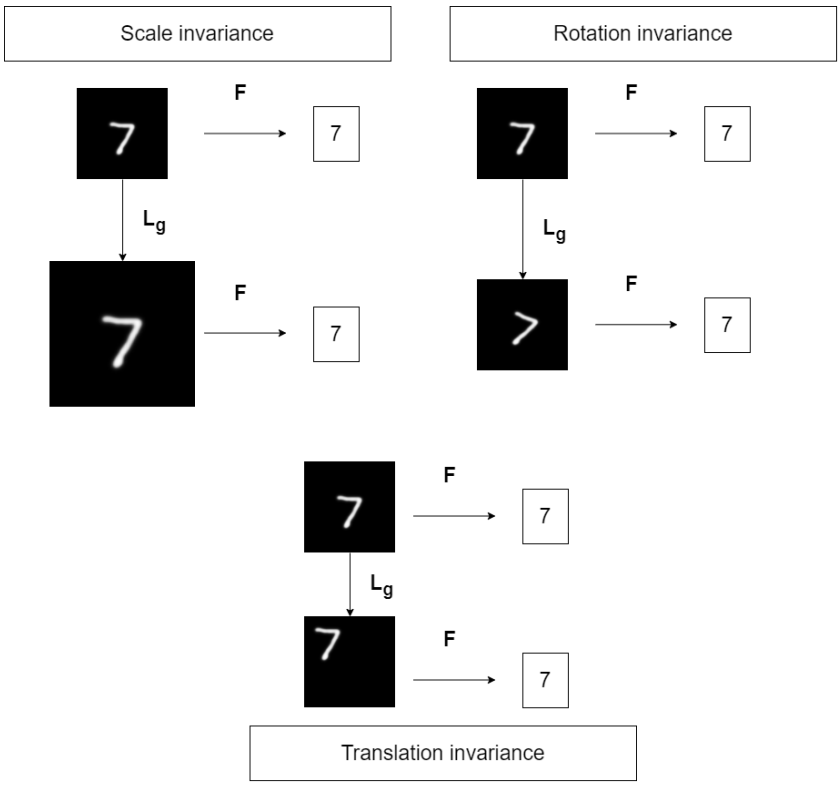


Figure 4.2: Illustration of invariances on the MNIST dataset [68]. F represents a classifier, while L_g is a (semi-)group operation. We show the examples rescaling, rotation, and translation.

4.3 Hilbert transform and analytic signal

Since the Riesz transform is a high dimensional generalization of the Hilbert transform, we give a short introduction to the Hilbert transform and the analytic signal. The goal is to understand the motivation behind the Riesz transform. We based this section on an overview on this topic from [69].

4.3.1 Hilbert transform and analytic signal

The analytic signal was originally introduced by Gabor [70] in the 1940s. It is related to the processing of 1d signals such as speech or music, where compression of signals based on frequency bands was shown to preserve most of the information contained in the signal. Since the analytic signal is constructed from the Hilbert transform, we start by definition of the Hilbert transform.

Hilbert transform: For $f \in L_2(\mathbb{R}) = \{g : \mathbb{R} \to \mathbb{R} \mid \int_{\mathbb{R}} |g(x)|^2 dx < \infty\}$, the Hilbert transform $\mathcal{H} : L_2(\mathbb{R}) \to L_2(\mathbb{R})$ is defined in the spatial domain as a convolution with the Hilbert kernel $h(t) = \frac{1}{\pi t}$:

$$\mathcal{H}(f) = f * h.$$

In the Fourier domain it is obtained by multiplication with $\mathcal{F}(h)(u) = -i \operatorname{sign}(u)$:

$$\mathcal{F}(\mathcal{H}(f))(u) = -i \operatorname{sign}(u)\mathcal{F}(f)(u),$$

where $\operatorname{sign} : \mathbb{R} \to \mathbb{R}$ is an operator which is equal to 1 for $u > 0$, 0 if $u = 0$, and otherwise -1. The Hilbert transform shifts the phase of the signal by $\frac{\pi}{2}$. We illustrate this in the following example.

Example Hilbert transform of the cosine function: Let $f : \mathbb{R} \to \mathbb{R}$ be

$$f(t) = \cos(wt)$$

for $w > 0$, then

$$\mathcal{F}(\mathcal{H}(f))(u) = -i \operatorname{sign}(u)\ \pi[\delta(u - w) + \delta(u + w)],$$

where $\delta(t)$ denotes Dirac delta function, i.e. $\delta(t) = \lim_{b \to 0} \frac{e^{-(x/b)^2}}{|b|\sqrt{\pi}}$. Now, for $u = w$ we have

$$\mathcal{F}(\mathcal{H}(f))(w) = -i\pi,$$

while for $u = -w$ we have

$$\mathcal{F}(\mathcal{H}(f))(-w) = i\pi.$$

For $u \notin \{-w, w\}$, it holds

$$\mathcal{F}(\mathcal{H}(f))(u) = 0.$$

This can be written more compactly as

$$\begin{aligned}\mathcal{F}(\mathcal{H}(f))(u) &= i\pi(-\delta(u - w) + \delta(u + w)) = \frac{\pi}{i}(-1)(-\delta(u - w) + \delta(u + w)) = \\ &= \frac{\pi}{i}(\delta(u - w) - \delta(u + w)) = \mathcal{F}(\sin(w\cdot))(u).\end{aligned}$$

Hence,

$$\mathcal{H}(\cos(w\cdot)) = \sin(w\cdot) = \cos(w \cdot + \frac{\pi}{2}).$$

Analytic signal: The analytic signal of a real-valued signal is a complex signal constructed by adding the Hilbert transform as an imaginary part of the input signal:

$$f_A = f + i\mathcal{H}(f).$$

In the Fourier domain this turns out to be

$$\mathcal{F}(f_A)(\omega) = \mathcal{F}(f)(\omega)(1 + \text{sign}(\omega)).$$

Hence, the analytic signal removes all negative frequencies of the real-valued functions by extending it to the complex domain. The original signal can be of course reconstructed by taking the real part of the analytic signal. Additional properties include: **orthogonality** $\langle f, \mathcal{H}f \rangle = 0$ and **energy preservation** $||f||_2 = ||\mathcal{H}(f)||_2$. Since f_A is a complex signal, it allows for representation in polar coordinates:

$$f(t) = A(t)e^{i\phi(t)}.$$

Here, $A(t) = \sqrt{f(t)^2 + \mathcal{H}(f)(t)^2}$ is known as local amplitude, while $\phi(t) = \arctan(\frac{\mathcal{H}(f)(t)}{f(t)})$ is called local phase. According to [71], amplitude and phase enable orthogonal decomposition into structural (phase) and energetic (amplitude) information. This means that if one changes one of these characteristics, this will not affect the other one.

4.3.2 Analytic function

Next, we look at the characterization of the analytic signal through analytic functions. This can be best understood through the Riemann-Hilbert problem:

$$\frac{\partial F}{\partial z}(z) = 0, \quad \text{for } z = x + iy \in \mathbb{C}, y > 0 \tag{4.6}$$

with boundary condition

$$\text{Re}(F(x)) = f(x), \quad x \in \mathbb{R} \tag{4.7}$$

The solution is given by the Cauchy integral up to a constant $C \in \mathbb{C}$

$$F(z) = \frac{1}{2\pi i} \int_{\mathbb{R}} \frac{1}{\tau - z} f(\tau) d\tau + C. \tag{4.8}$$

To show connections between the Cauchy integral and the analytic signal we need the **Sokhotski-Plemelj formula**. It is defined for a complex-valued function g which is defined and continuous on the real line, and states that for $a < 0 < b$ it holds:

$$\lim_{\epsilon \to 0^+} \int_a^b \frac{g(\tau)}{(\tau - i\epsilon)} d\tau = i\pi g(0) + \text{p.v.} \int_a^b \frac{g(\tau)}{\tau} d\tau, \tag{4.9}$$

where p.v. denotes the Cauchy principal value of the integral because the Hilbert kernel is not defined at 0. Then, we look at the Cauchy integral (for $C = 0$) at the boundary value of the real (upper) half-plane in the complex space, i.e. as $y \to 0$

$$\begin{aligned}
\lim_{y \to 0} F(x + iy) &= \frac{1}{2\pi i} \lim_{y \to 0} \int_{\mathbb{R}} \frac{1}{(\tau - x) - iy} f(\tau) d\tau \overset{(4.9)}{=} \\
&= \frac{1}{2\pi i} \Big(i\pi f(x) + p.v. \int_{\mathbb{R}} \frac{f(\tau)}{\tau - x} d\tau \Big) = \frac{1}{2} \Big(f(x) + p.v. \int_{\mathbb{R}} \frac{f(\tau)}{\pi(x - \tau)} d\tau \Big) = \\
&= \frac{1}{2}(f(x) + i\mathcal{H}(f)(x)) = \frac{1}{2} f_A(x).
\end{aligned}$$

In our proof we apply the Sokhotski-Plemelj formula to the function $g(\tau) = f(\tau + x)$ for $x \in \mathbb{R}$. Hence, we have established a connection between the analytic signal as a boundary value of the analytic function from the Riemann-Hilbert problem in the upper half-plane. The Riemann-Hilbert problem represents a main guideline on how to generalize the Hilbert transform to higher dimensions.

4.3.3 Derivation of Riesz transform from Riemann-Hilbert problem

This derivation is based on Clifford algebra. Since this is outside the scope of the thesis, we just state the main result for $d = 2$. More details on this can be found in [69]. The Dirac operator on $\mathbb{R}^d$ is a first order differential operator needed to generalize its counterpart in 1d

$$D = \sum_{k=1}^{d} \frac{\partial}{\partial x_k}$$

The Riemann-Hilbert problem based on the Dirac operator can be defined in 2d as

$$DF(x) = 0, \quad x = (x_1, x_2, x_3) \in \mathbb{R}^3, x_3 > 0 \tag{4.10}$$

with boundary condition

$$\mathrm{Re}(F(x_1, x_2)) = f(x_1, x_2), \quad (x_1, x_2) \in \mathbb{R}^2. \tag{4.11}$$

Similarly, as in 1d it can be shown that the boundary value of the solution ends up being the monogenic signal $f_M = f + i\mathcal{R}_1(f) + j\mathcal{R}_2(f)$ from [71] up to factor $\frac{1}{2}$, where $\mathcal{R} = (\mathcal{R}_1, \mathcal{R}_2)$ is the Riesz transform. More details on the monogenic signal are given in Section 5.3.1. Hence, the Riesz transform serves as an analogue to the Hilbert transform when generalizing the Rieman-Hilbert problem to higher dimensions.

Chapter 5

The Riesz transform

5.1 Overview of the Riesz transform in image and signal processing

The Riesz transform is a generalization of the Hilbert transform to higher dimensional spaces, as shown in Chapter 4. First practical applications of the Riesz transform arise in signal processing through the definition of the monogenic signal [71] which enables a decomposition of higher dimensional signals into local phase and local amplitude. First, a bandpass filter is applied to the signal to separate the band of frequencies. Using the Riesz transform, the local phase and amplitude can be calculated for a selected range of frequencies. For more details we refer to [71, 72].

As images are 2d or 3d signals, applications of the Riesz transform naturally extend to the fields of image processing and computer vision through the Poisson scale space [73, 74] which is an alternative to the well-known Gaussian scale space. Köthe [75] compared the Riesz transform with the structure tensor from a signal processing perspective. Unser [76] related higher order Riesz transforms and derivatives. Furthermore, they give a reason for preferring the Riesz transform over the standard derivative operator: The Riesz transform does not amplify high frequencies.

Higher order Riesz transforms were also used for analysis of local image structures using ideas from differential geometry [77, 78]. Benefits of using the first and second order Riesz transforms as low level features have also been shown in measuring similarity [79], analyzing and classification of textures [69, 80], and orientation estimation [81, 82].

Since wavelets have been proven to be very useful for multiresolution signal analysis, it was natural to extend the concept of monogenic signal to wavelets. This has been done through generalization of the analytic wavelet transform to higher dimensions [83] by preserving locality of the wavelet and introducing monogenicity through the Riesz transform. Furthermore, in [84, 85] the monogenic signal is applied to an isotropic wavelet that generates a wavelet frame. This results in a new mother wavelet that is directed, steerable, and also generates a wavelet frame. All these results enabled a multiresolution monogenic signal analysis. In [76] Unser derives an Nth order extension of [84] using the high order Riesz transforms. An example of this construction is illustrated on so called Riesz-Laplace wavelets. Unser and colleagues [76, 85] are the first ones utilizing the scale equivariance property of the Riesz transform, and have inspired the design of *quasi monogenic shearlets* [86].

Interestingly, in early works on the Riesz transform in signal processing or image processing

[71, 72, 74], scale equivariance has not been noticed as a feature of the Riesz transform and hence remained sidelined. Usefulness of the scale equivariance has been shown later in [76, 78].

Recently, the Riesz transform found its way into the field of deep learning: Riesz transform features are used as supplementary features in classical CNNs to improve robustness [87]. In our work, we will use the Riesz transforms for extracting low-level features from images and use them as basis functions which replace trainable convolutional filters in CNNs or Gaussian derivatives.

5.2 Definition of the Riesz transform

Let $L_2(\mathbb{R}^d) = \{f : \mathbb{R}^d \to \mathbb{R} \mid \int_{\mathbb{R}^d} |f(x)|^2 dx < \infty\}$ be the subset of functions from $\mathscr{L}$ with finite L_2-norm. Formally, for a d-dimensional signal $f \in L_2(\mathbb{R}^d)$ (i.e. an image or a feature map), the Riesz transform of first order $\mathcal{R} = (\mathcal{R}_1, \cdots, \mathcal{R}_d)$ is defined in the spatial domain as $\mathcal{R}_j : L_2(\mathbb{R}^d) \to L_2(\mathbb{R}^d)$

$$\mathcal{R}_j(f)(x) = C_d \lim_{\epsilon \to 0} \int_{\mathbb{R}^d \setminus B_\epsilon} \frac{y_j f(x-y)}{|y|^{d+1}} dy, \tag{5.1}$$

where $C_d = \Gamma((d+1)/2)/\pi^{(d+1)/2}$ is a normalizing constant and $B_\epsilon \subset \mathbb{R}^d$ is ball of radius ϵ centered at the origin. Alternatively, the Riesz transform can be defined in the frequency domain via the Fourier transform $\mathcal{F}$

$$\mathcal{F}(\mathcal{R}_j(f))(u) = -i\frac{u_j}{|u|}\mathcal{F}(f)(u) = \frac{1}{|u|}\mathcal{F}(\partial_j f)(u), \tag{5.2}$$

for $j \in \{1, \cdots, d\}$. High order Riesz transforms are defined by applying a sequence of first order Riesz transforms. That is, for $n_1, n_2, ..., n_d \in \mathbb{N} \cup \{0\}$ we set

$$\mathcal{R}^{(n_1,n_2,\ldots,n_d)}(f)(x) := \mathcal{R}_1^{n_1}(\mathcal{R}_2^{n_2}(\cdots(\mathcal{R}_d^{n_d}(f)))(x), \tag{5.3}$$

where $\mathcal{R}_j^{n_j}$ refers to applying the Riesz transform $\mathcal{R}_j$ n_j times in a sequence. Let $N = \sum_{i=1}^d n_i$ for the rest of Part II. Then, we refer to (5.3) as an N-th order Riesz transform.

The Riesz transform kernels of first and second order resemble those of the corresponding Gaussian derivatives (Figure 5.1). This can be explained by the following relations

$$\mathcal{R}(f) = (-1)(-\triangle)^{-1/2}\nabla f \tag{5.4}$$

$$\mathcal{R}^{(n_1,n_2,\ldots,n_d)}(f)(x) = (-1)^N(-\triangle)^{-N/2}\frac{\partial^N f(x)}{\partial^{n_1}x_1 \cdots \partial^{n_d}x_d}. \tag{5.5}$$

The fractional Laplace operator $\triangle^{N/2}$ acts as an isotropic low-pass filter. The main properties of the Riesz transform can be summarized in the following way [76]:

- **translation equivariance:**

  $$\mathcal{R}_j(\mathcal{T}_{x_0}(f))(x) = \mathcal{T}_{x_0}(\mathcal{R}_j(f))(x),$$

  where $x_0 \in \mathbb{R}^d$ and $j \in \{1, \cdots, d\}$. $\mathcal{T}_{x_0}(f)(x) : L_2(\mathbb{R}^d) \to L_2(\mathbb{R}^d)$ is a translation operator defined as $\mathcal{T}_{x_0}(f)(x) = f(x - x_0)$. This property reflects the fact that the Riesz transform commutes with the translation operator.

- **steerability:**

  Here, steering refers to the process of calculating an arbitrarily oriented filter as a linear combination of basis filters [9]. The Riesz transform is defined only for the (orthogonal) base

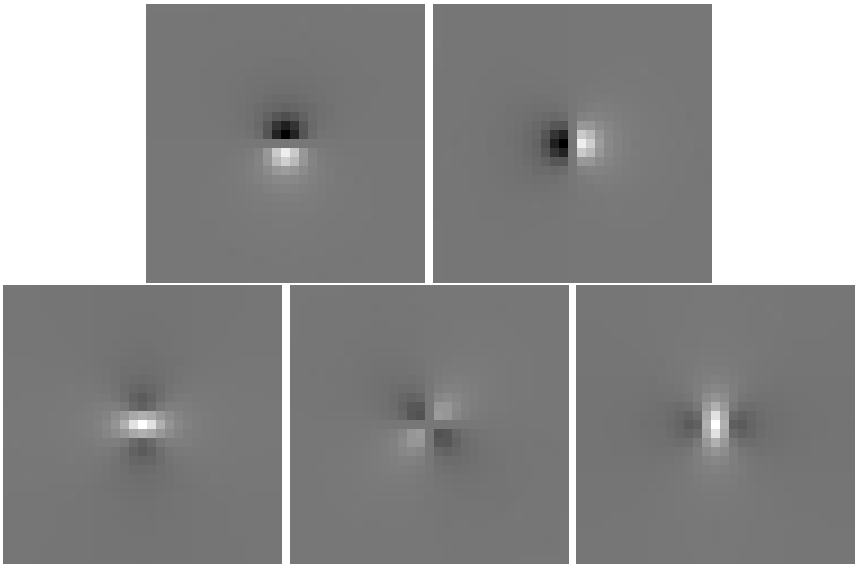


Figure 5.1: Visualizations of Riesz transform kernels of first and second order. First row (from left to right): $\mathcal{R}_1$ and $\mathcal{R}_2$. Second row (from left to right): $\mathcal{R}^{(1,1)}$, $\mathcal{R}^{(1,2)}$, and $\mathcal{R}^{(2,2)}$.

axes of $\mathbb{R}^n$. However, in practice the relevant features can be oriented arbitrarily. Hence, it is beneficial to orient the Riesz transform. This is achieved through the directional Hilbert transform $\mathcal{H}_v : L_2(\mathbb{R}^d) \to L_2(\mathbb{R}^d)$ in direction $v \in \mathbb{R}^d$, $||v|| = 1$ which is defined as $\mathcal{F}(\mathcal{H}_v(f))(u) = i\text{sign}(\langle u, v\rangle)$. $\mathcal{H}_v$ is steerable in terms of the Riesz transform, that is

$$\mathcal{H}_v(f)(x) = \sum_{j=1}^{d} v_j \mathcal{R}_j(f)(x) = \langle \mathcal{R}(f)(x), v\rangle.$$

In 2d, the directional Hilbert transform for a unit vector $v = (\cos\phi, \sin\phi)$, $\phi \in [0, 2\pi]$ becomes $\mathcal{H}_v(f)(x) = \cos\phi\mathcal{R}_1(f)(x) + \sin\phi\mathcal{R}_2(f)(x)$. This is equivalent to the link between gradient and directional derivatives [76] and a very useful property for learning oriented features.

High order steerability: similarly, higher order directional Hilbert transforms $\mathcal{H}_v^{(N)} : L_2(\mathbb{R}^d) \to L_2(\mathbb{R}^d)$ can be steered by higher order Riesz transforms:

$$\mathcal{H}_v^{(N)}(f)(x) = \sum_{|n|=N} \frac{N!}{n!} v^n \mathcal{R}^n(f)(x).$$

where $v = (v_1, \cdots, v_d) \in \mathbb{R}^d$ is a unit vector and $v^n := \prod_{i=1}^{N}(v_i)^{n_i}$. For example, for $d = N = 2$ and $v = (\cos\phi, \sin\phi)$ we have:

$$\mathcal{H}_v^{(2)}(f)(x) = \cos^2(\phi)\mathcal{R}^{(2,0)}(f)(x) + \sin^2(\phi)\mathcal{R}^{(0,2)}(f)(x) + 2\cos(\phi)\sin(\phi)\mathcal{R}^{(1,1)}(f)(x).$$

- **all-pass filter [71]:** Let $H = (H_1, \cdots, H_d)$ be the Fourier transform of the Riesz kernel, i.e. $\mathcal{F}(\mathcal{R}_j(f))(u) = i\frac{u_j}{|u|}\mathcal{F}(f)(u) = H_j(u)\mathcal{F}(f)(u)$. The energy of the Riesz transform for frequency $u \in \mathbb{R}^d$ is defined as the norm of the d-dimensional vector $H(u)$ and has value 1 for all non-zero frequencies $u \neq 0$, i.e.

$$|H(u)| = 1, \quad u \neq 0.$$

The all-pass filter property reflects the fact that the Riesz transform is a non-local operator and that every frequency is treated fairly and equally. Combined with scale equivariance, this eliminates the need for multiscale analysis or multiscale feature extraction.

- **scale (dilation) equivariance:**

$$\mathcal{R}_j(L_a(f))(x) = L_a(\mathcal{R}_j(f))(x),$$

for $a > 0$ and $j \in \{1, \cdots, d\}$. $L_a : L_2(\mathbb{R}^d) \to L_2(\mathbb{R}^d)$ is a dilation or rescaling operator defined as $L_a(f)(x) = f(\frac{x}{a})$. That is, the Riesz transform does not only commute with translations but also with scaling.

Scale equivariance enables an equal treatment of the same objects at different scales. As this is the key property of the Riesz transform for our application, we will briefly present a proof. We restrict to the first order in the Fourier domain. The proof for higher orders follows directly from the one for the first order. We have to show that

$$[\mathcal{R}_i f\Big(\frac{\cdot}{a}\Big)](x) = [\mathcal{R}_i f]\Big(\frac{x}{a}\Big). \tag{5.6}$$

Proof. Remember that the Fourier transform of the dilated function is given by $\mathcal{F}(f(\alpha\cdot))(u) = \frac{1}{\alpha^d}\mathcal{F}(f)(\frac{u}{\alpha})$. Setting $g(x) = f(\frac{x}{a})$, we have $\mathcal{F}(g)(u) = a^d\mathcal{F}(f)(au)$. This yields

$$\mathcal{F}\left(\mathcal{R}_j(f\Big(\frac{\cdot}{a}\Big))\right)(u) = \mathcal{F}\Big(\mathcal{R}_j(g)\Big)(u) = i\frac{u_j}{|u|}\mathcal{F}(g)(u) = i\frac{u_j}{|u|}a^d\mathcal{F}(f)(au) =$$

$$= a^d\Big(i\frac{au_j}{a|u|}\Big)\mathcal{F}(f)(au) = a^d\mathcal{F}\Big(\mathcal{R}_j(f)\Big)(au) = \mathcal{F}\left(\mathcal{R}_j(f)\Big(\frac{\cdot}{a}\Big)\right)(u).$$

□

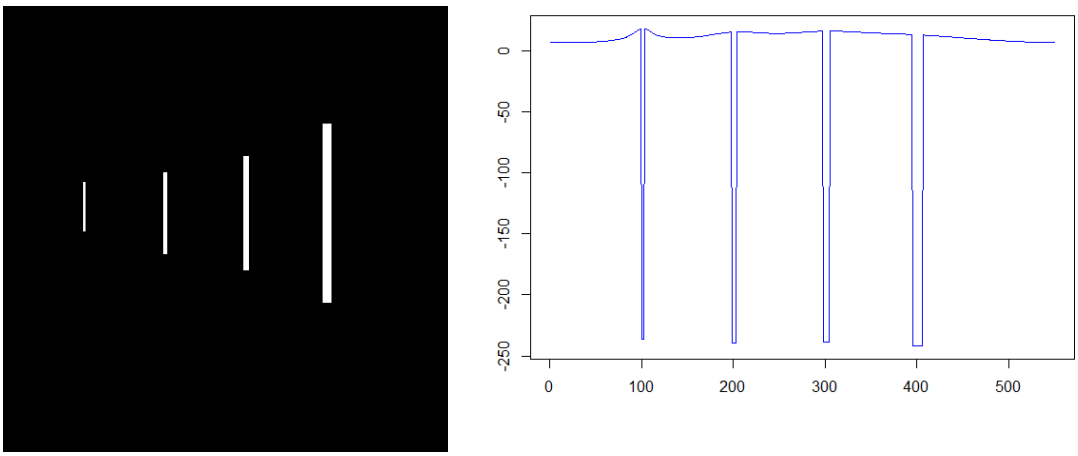


Figure 5.2: Interpretation of the Riesz transform on a mock example of 550 × 550 pixels: aligned rectangles with equal aspect ratio and constant gray value 255 (left) and response of the second order Riesz transform $\mathcal{R}^{(2,0)}$ of the left image sampled horizontally through the centers of the rectangles (right).

Figure 5.2 provides an illustration of the scale equivariance. It shows four rectangles with length-to-width ratio 20 and varying width (3, 5, 7, and 11 pixels) together with the gray value profile of the second order Riesz transform $R^{(2,0)}$ along a linear section through the centers of the rectangles. In spite of the different widths, the Riesz transform yields equal filter responses for each rectangle (up to rescaling). In contrast, to achieve the same behaviour in Gaussian scale space, the scale space has to be sampled (i.e. a subset of scales has to be selected), the γ-normalized derivative [88] has to be calculated for every scale, and finally the scale yielding the maximal absolute value has to be selected. In comparison, the simplicity of the Riesz transform achieving the same in just one transform without sampling scale space and without need for a scale parameter is striking.

5.3 Riesz transform and signal decomposition

5.3.1 Decomposition using the monogenic signal

Generally, the goal of the analytic or the monogenic signal is to extract useful information from a signal by means of decomposition. For example, a 2d signal f can be decomposed into local amplitude, orientation, and phase. Here, we summarize results for the 2d case from [71]. First, the monogenic representation $f_M(x)$ of the signal $f \in L_2(\mathbb{R}^2)$ at point $x \in \mathbb{R}^2$ is calculated:

$$f_M(x) = f(x) + i\mathcal{R}_1 f(x) + j\mathcal{R}_2 f(x) = f(x) + if_1(x) + jf_2(x),$$

where i and j can be seen as imaginary units and are related to the notation and calculus from Clifford analysis [71]. Here, we use the notation: $f_1(x) := \mathcal{R}_1 f(x)$ and $f_2(x) := \mathcal{R}_2 f(x)$. Then, local amplitude $|f_M(x)|$ is given by:

$$|f_M(x)| = \sqrt{f(x)^2 + f_1(x)^2 + f_2(x)^2}.$$

Local orientation and local phase refer to structural information at the point x. Local orientation is determined from $(f_1(x), f_2(x))$ following the differential interpretation of the Riesz transform:

$$v(x) = \tan^{-1}\left(\frac{f_2(x)}{f_1(x)}\right).$$

Local phase is defined in the direction orthogonal to the local orientation:

$$\phi(x) = \frac{f_D(x)}{|f_D(x)|}\tan^{-1}\left(\frac{\sqrt{f_1(x)^2 + f_2(x)^2}}{f(x)}\right),$$

where $f_D(x) = (-f_2(x), f_1(x), 0)$. Then, given local phase and local amplitude of the monogenic signal, the full signal can be reconstructed:

$$f_M(x) = |f_M(x)|e^{(-j,i,0)\cdot\phi(x)}.$$

5.3.2 Poisson scale space

The most well-known linear scale space is Gaussian scale space [89, 90, 91, 92]. It was thought to be the only scale space until works from Felsberg and Sommer [73, 74] where Poisson scale space was defined from the Laplace equation. They have shown that Poisson scale space satisfies most of the axioms of Gaussian scale space. Later it was shown in [93, 94] that the functions

$u \in L_2(\mathbb{R}^{d+1})$ that satisfy reasonable axioms based on these two scale spaces are the solutions of the family of pseudo differential equations based on parameter $\alpha \in \langle 0, 1]$:

$$\frac{\partial}{\partial s} u = -(-\triangle)^{\alpha} u$$
$$\lim_{s \to 0} u(x, s) = f(x),$$

for image $f \in L_2(\mathbb{R}^d)$ and $(-\triangle)^{\alpha}$ is a fractional Laplacian operator[1]. This class of infinitely many new scale spaces based on parameter $\alpha \in \langle 0, 1]$ is referred to as α-scale spaces [93, 94]. Interestingly, α-scale spaces connect Poisson and Gaussian scale space. For $\alpha = 1$, we get the diffusion equation and Gaussian scale space, while for $\alpha = \frac{1}{2}$ we get Poisson scale space. According to [74] Poisson scale space, also known as monogenic scale space, unites two fundamental concepts: scale space and phase-based image processing. Since phase-based image processing is related to the Riesz transform and monogenic signal, we give a short overview of Poisson scale space.

Felsberg and Sommer [73] looked at the solution of the Laplace equation on $\mathbb{R}^2$ with an additional scale dimension:

$$\triangle u(x, y, s) = \frac{\partial}{\partial_{xx}} u(x, y, s) + \frac{\partial}{\partial_{yy}} u(x, y, s) + \frac{\partial}{\partial_{ss}} u(x, y, s) = 0. \tag{5.7}$$

The fundamental solution of the Laplace equation is known as a Newton potential

$$u(x, y, s) = \frac{c}{\sqrt{x^2 + y^2 + s^2}},$$

where c is an arbitrary constant. Since differential operators commute, derivatives of u also satisfy the Laplace equation

$$g(x, y, s) = \frac{\partial}{\partial_s} u(x, y, s) = \frac{s}{(x^2 + y^2 + s^2)^{\frac{3}{2}}},$$
$$h_x(x, y, s) = \frac{\partial}{\partial_x} u(x, y, s) = \frac{x}{(x^2 + y^2 + s^2)^{\frac{3}{2}}},$$
$$h_y(x, y, s) = \frac{\partial}{\partial_y} u(x, y, s) = \frac{y}{(x^2 + y^2 + s^2)^{\frac{3}{2}}}.$$

In the previous step we set the constant c to -1. Here, g acts as an isotropic lowpass filter and it is known as a Poisson kernel. The Poisson kernel satisfies the five axioms of Iijima [89, 95] which define linear scale space. Furthermore, it holds

$$\lim_{s \to 0} h_x(x, y, s) = \frac{x}{(x^2 + y^2)^{3/2}},$$
$$\lim_{s \to 0} h_y(x, y, s) = \frac{y}{(x^2 + y^2)^{3/2}},$$

which are exactly the kernels of the Riesz transform for $d = 2$, i.e. for image f it holds $\lim_{s \to 0}(f * h_x(\cdot, \cdot, s)) = \mathcal{R}_1(f)$ and $\lim_{s \to 0}(f * h_y(\cdot, \cdot, s)) = \mathcal{R}_2(f)$. This means that at scale 0 $(f, f * h_x, f * h_y)$ is a monogenic signal. For scale $s > 0$, $(f * g, f * h_x, f * h_y)$ can be seen as a monogenic signal of the lowpass filtered signal $f * g$. This allows for a monogenic decomposition of the scale decomposed signal.

[1] Here, fractional spatial Laplacian is defined only in terms of variables from the spatial domain $\mathbb{R}^d$ rather than on the joint spatial and scale domain $\mathbb{R}^{d+1}$.

5.3.3 Decomposition using high order Riesz transform

The high order Riesz transform also enables useful signal decomposition. Here, we follow the results given in [85]. The first theorem is a result of the invertibility of the high order Riesz transform. Throughout this section, let $n = (n_1, \cdots, n_d)$ be a multivector s.t. $|n| = \sum_{i=1}^{d} n_i = N$ and $N \in \mathbb{N}$.

Theorem 2. *The N-th order Riesz transform allows for the following decomposition:*

$$\sum_{|n|=N} \frac{N!}{n!} (\mathcal{R}_1^{n_1} \cdots \mathcal{R}_d^{n_d})^* (\mathcal{R}_1^{n_1} \cdots \mathcal{R}_d^{n_d}) = Id. \tag{5.8}$$

Proof. The proof is done based on [76] in the Fourier domain.
For input signal f we have

$$\begin{aligned}\mathcal{F}((\mathcal{R}_1^{n_1} \cdots \mathcal{R}_d^{n_d})^* (\mathcal{R}_1^{n_1} \cdots \mathcal{R}_d^{n_d})(f))(w) &= \Big(\frac{-iw}{||w||}\Big)^n \Big(\frac{iw}{||w||}\Big)^n \mathcal{F}(f)(w) \\ &= \Big(\frac{w_1^2}{||w||^2}\Big)^{n_1} \cdots \Big(\frac{w_d^2}{||w||^2}\Big)^{n_d} \mathcal{F}(f)(w).\end{aligned}$$

Using this equation on the left side of equation (5.8) in the Fourier domain

$$\mathcal{F}\Big(\sum_{|n|=N} \frac{N!}{n!} (\mathcal{R}_1^{n_1} \cdots \mathcal{R}_d^{n_d})^* (\mathcal{R}_1^{n_1} \cdots \mathcal{R}_d^{n_d}) f\Big)(w) = \sum_{|n|=N} \frac{N!}{n!} \mathcal{F}\Big((\mathcal{R}_1^{n_1} \cdots \mathcal{R}_d^{n_d})^* (\mathcal{R}_1^{n_1} \cdots \mathcal{R}_d^{n_d}) f\Big)(w)$$

$$= \sum_{|n|=N} \frac{N!}{n!} \Big(\frac{w_1^2}{||w||^2}\Big)^{n_1} \cdots \Big(\frac{w_d^2}{||w||^2}\Big)^{n_d} \mathcal{F}(f)(w) = \sum_{|n|=N} \frac{N!}{n!} \Big(\frac{(w_1^2)^{n_1} \cdots (w_d^2)^{n_d}}{||w||^{2N}}\Big) \mathcal{F}(f)(w)$$

$$= [\text{from multinomial theorem}] \; = \frac{1}{||w||^{2N}} (w_1^2 + \cdots w_d^2)^N \mathcal{F}(f)(w)` = \mathcal{F}(f)(w).$$

□

Hence, the Nth order Riesz transform decomposes a signal into $p(N, d) = \binom{N+d-1}{d-1}$ components from which the original signal can be reconstructed. The next theorem states that this decomposition preserves the L_2-norm of the signal.

Theorem 3. *The N-th order Riesz transform satisfies the following Parseval-like identity:*

$$\sum_{|n|=N} \frac{N!}{n!} \langle \mathcal{R}^n f, \mathcal{R}^n g \rangle_{L_2} = \langle f, g \rangle_{L_2}, \tag{5.9}$$

for every $f, g \in L_2(\mathbb{R}^d)$.

This implies the conservation of signal energy $||f||$

$$\sum_{|n|=N} \frac{N!}{n!} ||\mathcal{R}^n f||^2 = ||f||^2. \tag{5.10}$$

and the contraction property

$$||\mathcal{R}^n f||^2 \leq ||f||^2, \tag{5.11}$$

for any multivector n. For $N = 1$, this turns out to be $||f||^2 = ||\mathcal{R}_1(f)||^2 + ||\mathcal{R}_2(f)||^2$ or in terms of the previous section $|f_M(x)|^2 = 2|f(x)|^2$.

Nonexpansiveness of the Riesz transform: For multivector n it holds:

$$||\mathcal{R}^n(f) - \mathcal{R}^n(g)||^2 \leq ||f - g||^2, \tag{5.12}$$

Proof. Since $\mathcal{R}^n$ is a linear operator, it holds $\mathcal{R}^n(f-g) = \mathcal{R}^n(f) - \mathcal{R}^n(g)$. Hence, the contraction property implies nonexpansiveness:

$$||\mathcal{R}^n(f) - \mathcal{R}^n(g)||^2 = ||\mathcal{R}^n(f - g)||^2 \leq ||f - g||^2.$$

□

These properties will be useful for understanding the decay of energy, i.e. amplitude with the increasing depth of the Riesz scattering representation in Chapter 8.

Chapter 6

The Riesz network

6.1 Introduction

In image data, the same objects may occur at highly varying scales. Examples are cars or pedestrians at different distances from the camera, cracks in concrete of varying thickness or imaged at different resolution, or blood vessels in biomedical applications (see Figure 6.1). It is natural to assume that the same object or structure at different scales should be treated equally, i.e. should have equal or at least similar features. This property is called scale or dilation invariance and has been very well investigated in classical image processing [88, 96, 97].

Neural networks have proven to segment and classify robustly and well in many computer vision tasks. Nowadays, the most popular and successful neural networks are Convolutional Neural Networks (CNNs). It would be desirable that neural networks share typical properties of human vision such as translation, rotation, or scale invariance. While this is true for translation invariance, CNNs are not scale or rotation invariant by default. This is due to the excessive use of convolutions which are local operators. Moreover, training sets often contain a very limited number of scales. To overcome this problem, CNNs are often trained with rescaled images through data augmentation. However, when a CNN is given input whose scale is outside the range covered by the training set, it will not be able to generalize [98, 99]. To overcome this problem, a CNN trained at a fixed scale can be applied to several rescaled versions of the input image and the results can be combined. This, however, requires multiple runs of the network.

One application example, where the just described challenges naturally occur is the task of segmenting cracks in 2d or 3d gray scale images of concrete. Crack segmentation in 2d has been a vividly researched topic in civil engineering, see [42] for an overview. Cracks are naturally multiscale structures (Fig. 6.1, top) and hence require multiscale treatment. Nevertheless, adaption to scale (crack thickness[1]) has not been treated explicitly so far.

Recently, crack segmentation in 3d images obtained by computed tomography (CT) has become a subject of interest [44, 42]. Here, the effect of varying scales is even more pronounced [100]: crack thicknesses can vary from a single pixel to more than 100 pixels. Hence, the aim is to design and evaluate crack segmentation methods that work equally well on all possible crack widths without complicated adjustment by the user.

In this chapter, we focus on 2d multiscale crack segmentation in images of concrete samples. We design the Riesz network which replaces the popular 2d convolutions by first and second order Riesz transforms to allow for a scale invariant spatial operation. The resulting neural network

[1] Crack scale, thickness, and width refer to the same characteristic and will be interchangeably used throughout the paper.

Figure 6.1: Examples of similar objects appearing on different scales: section of a CT image of concrete showing a crack of locally varying thickness (top) and pedestrians at different distances from the camera (bottom, taken from [101]).

is provably scale invariant in only one forward pass. It is sufficient to train the Riesz network on one scale or crack thickness, only. The network then generalizes automatically without any adjustments or rescaling to completely unseen scales. We validate the network performance using images with simulated cracks of constant and varying widths generated as described in [42, 102]. Our network is compared with competing methods for multiscale segmentation and finally applied to real multiscale cracks observed in 2d slices of tomographic images.

There is just one publicly available dataset which allows for testing scale equivariance – MNIST Large Scale [99]. We analyze the performance of the Riesz network on this dataset.

Next, we give an overview of deep learning methods from the literature that are robust to variations in scale. Deep learning methods which have mechanisms to handle variations in scale effectively can be split in two groups based on their scale generalization ability.

6.1.1 Scale invariant deep learning methods for a limited range of scales

The first group can handle multiscale data but is limited to the scales represented either by the training set or by the neural network architecture. The simplest approach to learn multiscale features is to apply the convolutions to several rescaled versions of the images or feature maps in every layer and to combine the results by maximum pooling [98] or keeping the scale with maximal activation [103] before passing it to the next layer. In [104, 105], several approaches based on downscaling images or feature maps with the goal of designing robust multiscale object detectors are summarized. However, scaling is included in the network architecture such that scales have to be selected a priori. Therefore, this approach only yields local scale invariance, i.e. an adaption to the scale observed in a given input image is not possible after training.

Another intuitive approach is to rescale trainable filters, i.e. convolutions, by interpola-

tion [106]. Another intuitive approach is to rescale trainable filters, i.e. convolutions, by interpolation [106]. In [104], a new multiscale strategy was derived which uses convolution blocks of varying sizes sequenced in several downscaling layers creating a pyramidal shape. The pyramidal structure is utilized for learning scale dependent features and making predictions in every downsampling layer. Layers can be trained according to object size. That is, only the part of the network relevant for the object size is optimized. This guarantees robustness to a large range of object scales. Similarly, in [105], a network consisting of a downsampling pyramid followed by an upsampling pyramid is proposed. Here, connections between pyramid levels are devised for combining low and high resolution features and predictions are also made independently on every pyramid level. However, in both cases, scale generalization properties of the networks are restricted by their architecture, i.e. by the depth of the network (number of levels in the image pyramid), the size of convolutions as spatial operators as well as the size of the input image.

Spatial transformer networks [107] focus on invariance to affine transformations including scale. This is achieved by using a so-called *localisation network* which learns transformation parameters. Finally, using these transformation parameters, a new sampling grid can be created and feature maps are resampled to it. These parts form a trainable module which is able to handle and correct the effect of the affine transformations. However, spatial transformer networks do not necessarily achieve invariant recognition [108]. Also, it is not clear how this type of network would generalize to scales not represented in the training set.

In [109], so-called *structured receptive fields* are introduced. Linear combinations (1×1 convolutions) of basis functions (in this case Gaussian derivatives up to 4th order) are used to learn complex features and to replace convolutions (e.g. of size 3×3 or 5×5). As a consequence, the number of parameters is reduced, while the expressiveness of the neural network is preserved. This type of network works better than classical CNNs in the case where less training data is available. However, the standard deviation parameters of the Gaussian kernels are manually selected and kept fixed. Hence, the scale generalization ability remains limited.

Making use of the semi-group property of scale spaces, *scale equivariant neural networks* motivate the use of *dilated convolutions* [110] to define scale equivariant convolutions on the Gaussian scale space [111] or morphological scale spaces [112]. Unfortunately, these neural networks are unable to generalize to scales outside those determined by their architecture and are only suitable for downscale factors which are powers of 2, i.e. $\{2, 4, 8, 16, \cdots\}$. Furthermore, scale equivariant steerable networks [113] show how to design scale invariant networks on the scale-translation group without using standard or dilated convolutions. Following an idea from [109], convolutions are replaced by linear combinations of basis filters (Hermite polynomials with Gaussian envelope). While this allows for non-integer scales, scales are still limited to powers of a positive scaling factor a. Scale space is again discretized and sampled. Hence, a generalization to arbitrary scales is not guaranteed.

6.1.2 Scale invariant deep learning methods for arbitrary scales

The second group of methods can generalize to arbitrary scales, i.e. any scales that are in a range bounded from below by image resolution and from above by image size, but not necessarily contained in the training set. Our Riesz network also belongs to this second group of methods.

An intuitive approach is to train standard CNNs on a fixed range of scales and enhance their scale generalization ability by the following three step procedure based on image pyramids: downsample by a factor $a > 1$, forward pass of the CNN, upsample the result by $\frac{1}{a}$ to the original image size [99, 100]. Finally, forward passes of the CNN from several downsampling factors $\{a_1, \cdots, a_n > 0 \mid n \in \mathbb{N}\}$ are aggregated by using the maximum or average operator across the scale dimension. This approach indeed guarantees generalization to unseen scales as scales

can be adapted to the input image and share the weights of the network [99]. However, it requires multiple forward passes of the CNN and the downsampling factors have to be selected by the user.

Inspired by Scattering Networks [114, 115], normalized differential operators based on first and second order Gaussian derivatives stacked in layers or a cascade of a network can be used to extract more complex features [116]. Subsequently, these features serve as an input for a classifier such as a support vector machine. Varying the standard deviation parameter σ of the Gaussian kernel, generalization to new scales can be achieved. However, this type of network is useful for creating *handcrafted* complex scale invariant features, only, and hence is not trainable. A more comprehensive overview on this subfield is given in Chapter 8.

Its expansion to trainable networks by creating so-called Gaussian derivative networks [117] is one of the main inspirations for our work. For combining spatial information, γ-normalized Gaussian derivatives are used as scale equivariant operators ($\gamma = 1$). Similarly as in [109], linear combinations of normalized derivatives are used to learn more complex features in the spirit of deep learning. During the training step, prior knowledge of the scale for every instance in the training set is required, i.e. the standard deviation parameter σ of the Gaussian kernel is set to reflect the scale of every instance, while the trainable weights are shared. In the inference step, the scale dimension needs to be discretized, sampled, and for each scale σ, the forward pass of the network has to be executed.

6.2 Riesz network

In the spirit of structured receptive fields [109] and Gaussian derivative networks [117], we use the Riesz transforms of first and second order instead of standard convolutions to define Riesz layers.

As a result, Riesz layers are scale equivariant in a single forward pass. Replacing standard derivatives with the Riesz transform has been motivated by [75], while using a linear combination of Riesz transforms of several order follows [80].

6.2.1 Riesz layers

The base layer of the Riesz networks is defined as a linear combination of the Riesz transforms of several orders implemented as 1d convolution across feature channels (Figure 6.2). Here, we limit ourselves to first and second order Riesz transforms. Thus, the linear combination reads as

$$J_{\mathcal{R}}(f) = C_0 + \sum_{k=1}^{d} C_k \cdot \mathcal{R}_k(f) + \sum_{k,l \in \mathbb{N}_0, k+l=2} C_{k,l} \cdot \mathcal{R}^{(k,l)}(f), \tag{6.1}$$

where $\{C_0, C_k | k \in \{1, \cdots, d\}\} \cup \{C_{k,l} | l, k \in \mathbb{N}_0, l+k=2\}$ are parameters that are learned during training.

Now we can define the general layer of the network (Figure 6.2). Let us assume that the Kth network layer takes input $F^{(K)} = (F_1^{(K)}, \cdots, F_{c^{(K)}}^{(K)}) \in \left(L_2(\mathbb{R}^d)\right)^{c^{(K)}}$ with $c^{(K)}$ feature channels and has output $F^{(K+1)} = (F_1^{(K+1)}, \cdots, F_{c^{(K+1)}}^{(K+1)}) \in \left(L_2(\mathbb{R}^d)\right)^{c^{(K+1)}}$ with $c^{(K+1)}$ channels. Then the output in channel $j \in \{1, \cdots, c^{(K+1)}\}$ is given by

$$F_j^{(K+1)} = \sum_{i=1}^{c^{(K)}} J_K^{(j,i)}(F_i^{(K)}). \tag{6.2}$$

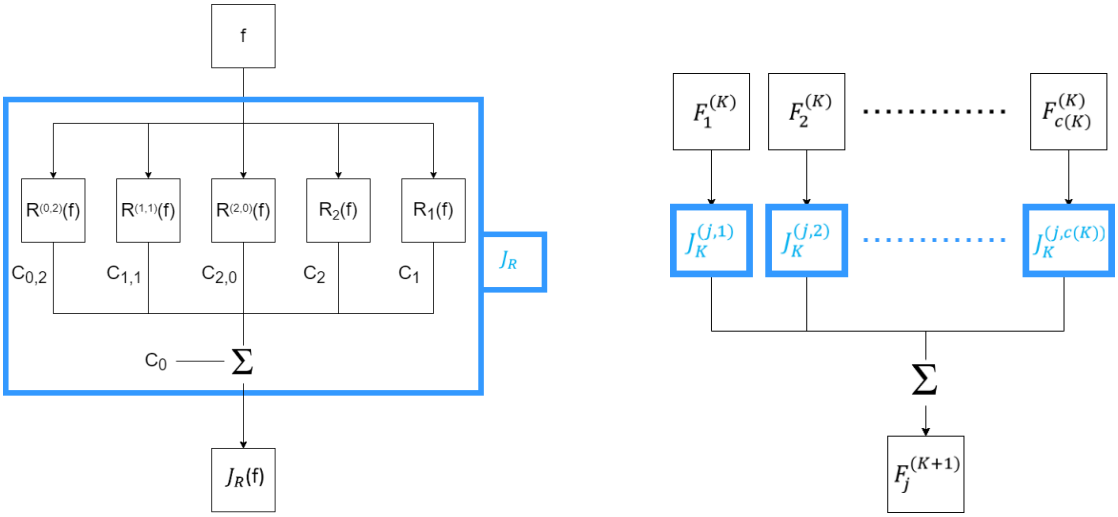


Figure 6.2: Building blocks of Riesz networks: the base Riesz layer from equation (6.1) (left) and the full Riesz layer from equation (6.2) (right).

Here, $J_K^{(j,i)}$ is defined in the same way as $J_{\mathcal{R}}(f)$ from equation (6.1), but trainable parameters may vary for different input channels i and output channels j, i.e.

$$J_K^{(j,i)}(f) = C_0^{(j,i,K)} + \sum_{k=1}^{d} C_k^{(j,i,K)} \cdot \mathcal{R}_k(f) + \sum_{k,l \in \mathbb{N}_0, k+l=2} C_{k,l}^{(j,i,K)} \cdot \mathcal{R}^{(k,l)}(f). \tag{6.3}$$

In practice, the offset parameters $C_0^{(j,i,K)}$, $i = 1, \cdots, c^{(K)}$ are replaced with a single parameter defined as $C_0^{(j,K)} := \sum_{i=1}^{c^{(K)}} C_0^{(j,i,K)}$.

6.2.2 Proof of scale equivariance

We prove the scale equivariance for $J_{\mathcal{R}}(f)$. That implies scale equivariance for $J_K^{(j,i)}(f)$ and consequently for $F_j^{(K+1)}$ for arbitrary layers of the network. By construction (see Section 6.2.3), this will result in provable scale equivariance for the whole network. For any scaling parameter $a > 0$ and $x \in \mathbb{R}^d$, we have

$$\begin{aligned} J_{\mathcal{R}}\left(f(\tfrac{\cdot}{a})\right)(x) &= C_0 + \sum_{k=1}^{d} C_k \cdot \mathcal{R}_k\left(f(\tfrac{\cdot}{a})\right)(x) + \sum_{k=1}^{d}\sum_{l=k}^{d} C_{k,l} \cdot \mathcal{R}^{(k,l)}\left(f(\tfrac{\cdot}{a})\right)(x) \\ &\overset{(5.6)}{=} C_0 + \sum_{k=1}^{d} C_k \cdot \mathcal{R}_k(f)(\tfrac{x}{a}) + \sum_{k=1}^{d}\sum_{l=k}^{d} C_{k,l} \cdot \mathcal{R}^{(k,l)}(f)(\tfrac{x}{a}) \\ &= J_{\mathcal{R}}(f)(\tfrac{x}{a}). \qquad \blacksquare \end{aligned}$$

6.2.3 Network design

The basic building block of modern CNNs is a sequence of the following operations: batch normalization, spatial convolution (e.g. 3×3 or 5×5), the Rectified Linear Unit (ReLU) activation

function, and Max Pooling. For more details on these operations, the reader is referred to Appendix A. Spatial convolutions have by default a limited size of the receptive field and Max Pooling is a downsampling operation performed on a window of fixed size. For this reason, these two operations are not scale equivariant and consequently CNNs are sensitive to variations in the scale. Hence, among the classical operations, only batch normalization [118] and ReLU activation preserve scale equivariance. To build our neural network from scale equivariant transformations, only, we restrict to using batch normalization, ReLUs, and Riesz layers, which serve as a replacement for spatial convolutions. In our setting, Max Pooling can completely be avoided since its main purpose is to combine it with spatial convolutions in cascades to increase the size of the receptive field while reducing the number of parameters.

Generally, a layer consists of the following sequence of transformations: batch normalization, Riesz layer, and ReLU. Batch normalization improves the training capabilites and avoids overfitting, ReLUs introduce non-linearity, and the Riesz layers extract scale equivariant spatial features. For every layer, the number of feature channels has to be selected. Hence, our network with $K \in \mathbb{N}$ layers can be simply defined by a $(K+2)$-tuple specifying the channel sizes[2] e.g. $(c^{(0)}, c^{(1)}, \cdots c^{(K)}, c^{(K+1)})$. The final layer is defined as a linear combination of the features from the previous layer followed by a sigmoid function yielding the desired probability map as output.

The four layer Riesz network we apply here can be schematically written as $1 \to 16 \to 32 \to 40 \to 48 \to 1$ and has $(1{\cdot}5{\cdot}16{+}16){+}(16{\cdot}5{\cdot}32{+}32){+}(32{\cdot}5{\cdot}40{+}40){+}(40{\cdot}5{\cdot}48{+}48){+}(48{\cdot}1{+}1) = 18\,825$ trainable parameters.

6.3 Applications in 2d crack segmentation

In this section we evaluate the four layer Riesz network defined above on the task of segmenting cracks in 2d slices from CT images of concrete. Particular emphasis is put on the network's ability to segment multiscale cracks and to generalize to crack scales unseen during training. To quantify these properties, we use images with simulated cracks. Being accompanied by an unambiguous ground truth, they allow for an objective evaluation of the segmentation results.

Data generation: Cracks are generated by the fractional Brownian motion (Experiment 1) or minimal surfaces induced by the facet system of a Voronoi tessellation (Experiment 2). Dilated cracks are then integrated into CT images of concrete without cracks. As pores and cracks are both air-filled, their gray value distribution should be similar. Hence, the gray value distribution of crack pixels is estimated from the gray value distribution observed in air pores. The crack thickness is kept fixed (Experiment 1) or varies (Experiment 2) depending on the objective of the experiment. As a result, realistic semi-synthetic images can be generated (see Figure 6.3). For more details on the simulation procedure, we refer to [42, 102]. Details on number and size of the images can be found below. Finally, we show applicability of the Riesz network for real data containing cracks generated by tensile and pull-out tests.

Quality metrics: As metrics for evaluation of the segmentation results we use precision (P), recall (R), F1-score (or dice coefficient, Dice), and Intersection over Union (IoU). We refer to Appendix D for more details.

[2] Channel dimension $c(0)$ denotes the dimension of input image, e.g. for grayvalue images $c(0) = 1$. Channel dimension $c(K+1)$ denotes the dimension of the final output of the network. For the crack segmentation the output map is the binary image, e.g. $c(K+1) = 1$.

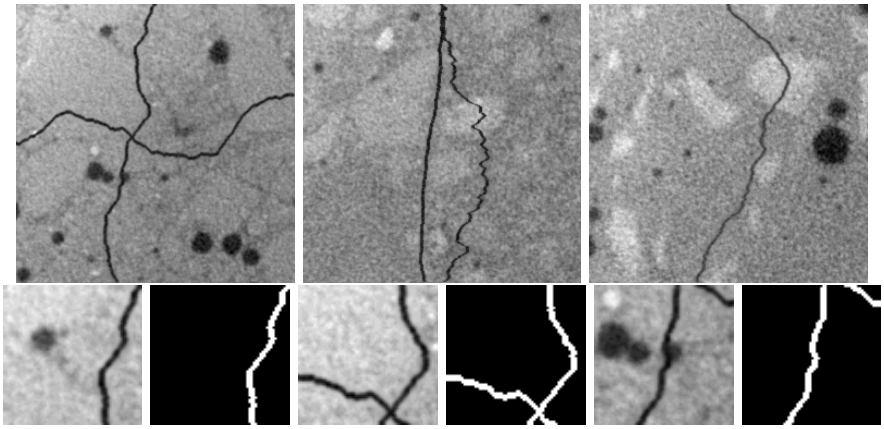


Figure 6.3: Cracks of width 3 used for training: before (first row) and after cropping (gray value images in the second row). Binary images in the second row represent ground truth images of the gray value images in the same row. Image sizes are 256×256 (first row) and 64×64 (second row).

Training parameters: If not specified otherwise, all models are trained on cracks of fixed width of 3 pixels. Cracks for the training are generated in the same way as for Experiment 1 on 256×256 sections of CT images of concrete. Then, 16 images of size 64×64 are cropped without overlap from each of the generated images. In this way, images without cracks are present in the training set. After data augmentation by flipping and rotation, the training set consists of 1 947 images of cracks. Some examples are shown in Figure 6.3. For validation, another set of images with cracks of width 3 is used. The validation data set's size is a third of the size of the training set.

All models are trained for 50 epochs with initial learning rate 0.001 which is halved every 20 epochs. ADAM optimization [119] is used, while the cost function is set to binary cross entropy loss.

Crack pixels are labelled with 1, while the background is labelled with 0. As there are far more background than crack pixels, we deal with a highly imbalanced data set. Therefore, crack and pore pixels are given a weight of 40 to compensate for class imbalance and to help distinguishing between these two types of structures which do not differ in their gray values.

6.3.1 Measuring scale equivariance

Measures for assessing scale equivariance have been introduced in [111, 113]. For an image or feature map f, a mapping function Φ (e.g. a neural network or a subpart of a network), and a scaling function L_a we define

$$\Delta_a(\Phi) := \frac{||L_a(\Phi(f)) - \Phi(L_a(f))||_2}{||L_a(\Phi(f))||_2}.$$

Ideally, this measure should be 0 for perfect scale equivariance. In practice, due to scaling and discretization errors we expect it to be positive yet very small.

To measure scale equvariance of the full Riesz network with randomly initialized weights, we use a data set consisting of 85 images of size 512×512 pixels with crack width 11 and use downscaling factors $a \in \{2, 4, 8, 16, 32, 64\}$. The evaluation was repeated for 20 randomly initialized Riesz networks. The resulting values of Δ_a are given in Figure 6.4.

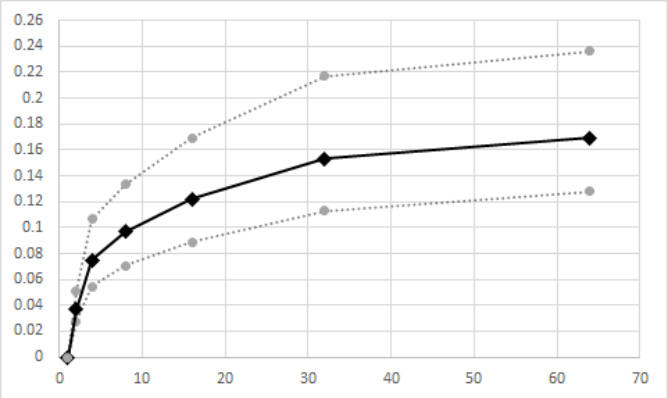


Figure 6.4: Measure of scale equivariance Δ_a for the four layer Riesz network with randomly initialized parameters w.r.t. the downsacling factor a. Mean (black), minimum, and maximum (gray) of 20 repetitions. Points on the line correspond to $a \in \{1, 2, 4, 8, 16, 32, 64\}$.

The measure Δ_a was used to validate the scale equivariance of Deep Scale-spaces (DSS) in [111] and scale steerable equivariant networks in [113]. In both works, a steep increase in Δ_a is observed for downscaling factors larger than 16, while for very small downscaling factors, Δ_a is reported to be below 0.01. In [113], Δ_a reaches 1 for downscaling factor 45. The application scenario studied here differs from those of [111, 113]. Results are thus not directly comparable but can be considered only as an approximate baseline. For small downscaling factors, we find Δ_a to be higher than in [113] (up to 0.075). However, for larger downscaling factors ($a > 32$), Δ_a increases more slowly e.g. $\Delta_{64} = 0.169$. This proves the resilience of Riesz networks to very high downscaling factors, i.e. large changes in scale.

6.3.2 Experiment 1: Generalization to unseen scales

Our models are trained on images of fixed crack width 3. To investigate their behaviour on crack widths outside of the training set, we generate cracks of widths $\{1, 3, 5, 7, 9, 11\}$ pixels in images of size 512×512, see Figure 6.9. Each class contains 85 images. Besides scale generalization properties of the Riesz network, we check how well it generalizes to random variations in crack topology or shapes, too.

Ablation study on the Riesz network

We investigate how network parameters and composition of the training set affect the quality of the results, in order to learn how to design this type of neural networks efficiently.

Size of training set: First, we investigate robustness of the Riesz network to the size of the training set. Literature [109] suggests that neural networks based on *structure receptive fields* are less data hungry, i.e. their performance with respect to the size of the training set is more stable than that of conventional CNNs. Since the Riesz network uses the Riesz transform instead of a Gaussian derivative as in [109], it is expected that the same would hold here, too.

The use of smaller training sets has two main benefits. First, obviously, smaller data sets reduce the effort for data collection, i.e. annotation or simulation. Second, smaller data sets reduce the training time for the network if we do not increase the number of epochs during training.

Method	w1	w3	w5	w7	w9	w11
	Dice	Dice	Dice	Dice	Dice	Dice
baseline	0.352	0.895	0.941	0.954	0.962	0.964
width 1	0.535	0.761	0.738	0.678	0.634	0.631
width 5	0.317	0.891	0.935	0.951	0.959	0.957
mixed width	0.297	0.865	0.905	0.935	0.954	0.962
trainset 489	0.356	0.877	0.929	0.945	0.954	0.958
trainset 975	0.365	0.919	0.942	0.954	0.964	0.966
layer 2	0.297	0.865	0.905	0.935	0.954	0.962
layer 3	0.366	0.915	0.940	0.954	0.966	0.971
layer 5	0.390	0.914	0.950	0.960	0.969	0.972

Table 6.1: Experiment 1. Ablation study: scale generalization ability of Riesz networks. Baseline is trained on 1 947 images with cracks of width 3 and has 4 layers. Cells are colored in lightgray if the metric is better than for the baseline, but not by more than 0.02. Dark gray color is used for metrics being more than 0.02 better compared to the baseline.

We constrain ourselves to three sizes of training sets: 1 947, 975, and 489. These numbers refer to the sets after data augmentation by flipping and rotation. Hence, the number of original images is three times smaller. In all three cases we train the Riesz network for 50 epochs and with similar batch sizes (11, 13, and 11, respectively). Results on unseen scales with respect to data set size are shown in Table 6.1 and Figure 6.5. We observe that the Riesz network trained on the smallest data set is competitive with counterparts trained on larger data sets albeit featuring generally $1-2\%$ lower Dice and IoU.

Choice of crack width for training: There are two interesting questions with respect to crack width. Which crack width is suitable for training of the Riesz network? Do varying crack thicknesses in the training set improve performance significantly?

To investigate these questions, we choose three training data sets with cracks of fixed widths 1, 3, or 5. A fourth data set combines crack widths 1, 3, and 5. We train the Riesz network with these sets and evaluate its generalization performance across scales. Results are summarized in Figure 6.6 and Table 6.1. Crack widths 3 and 5 yield similar results, while crack width 1 seems not to be suitable, except when trying to segment cracks of width 1. Cracks of width 1 are very thin, subtle, and in some cases even disconnected. Hence, they differ significantly from thicker cracks which are 8-connected and have a better contrast to the concrete background. This indicates that very thin cracks should be considered a special case which requires somewhat different treatment. Rather surprisingly, using the mixed training data set does not improve the metrics. Diversity with respect to scale in the training set seems not to be a decisive factor when designing Riesz networks.

Number of layers: Finally, we investigate the explanatory power of the Riesz network depending on network depth and the number of parameters. We train four networks with $2-5$ layers and 2 721, 9 169, 18 825, and 34 265 parameters, respectively, on the same data set for 50 epochs. The network with 5 layers has structure $16 \to 32 \to 40 \to 48 \to 64$ and every other network is constructed from this one by removing the required number of layers at the end. Results are shown in Table 6.1 and in Figure 6.7. The differences between the networks with 3, 4, and 5 layers are rather subtle. For the Riesz network with only 2 layers, performance deteriorates considerably ($3-5\%$ in Dice and IoU).

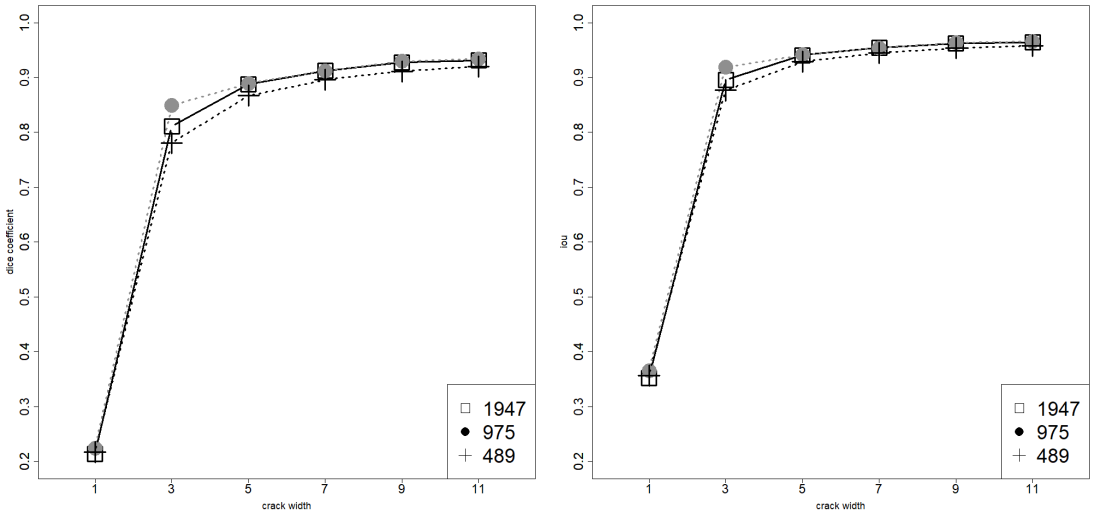


Figure 6.5: Experiment 1. Effect of the training set size on generalization to unseen scales. The baseline Riesz network is marked with 1 947, w3, and layer 4 and with square symbol □. Left: Dice, right: IoU.

In general, Riesz networks appear to be robust with respect to training set size, depth of network, and number of parameters. Hence, it is not necessary to tune many parameters or to collect thousands of images to achieve good performance, in particular for generalization to unseen scales. For the choice of crack width, 3 and 5 seem appropriate while crack width 1 should be avoided.

Comparison with competing methods

Competing methods: The four layer Riesz network is compared to two other methods – Gaussian derivative networks [117] and U-net [120] on either rescaled images [99] or an image pyramid [100]. The Gaussian derivative network uses scale space theory based on the Gaussian kernel and the diffusion equation. Using the γ-normalized Gaussian derivatives from [88], layers of first and second order Gaussian derivatives are constructed [117]. U-net has around 2.7 million parameters, while the Gaussian derivative network has the same architecture as the Riesz network and hence the same number of parameters (18k).

We design an experiment for detailed analysis and comparison of the ability of the methods to generalize to scales unseen during training. In typical applications, the thickness of the cracks would not be known. Here, crack thickness is kept fixed such that the correct scale of cracks is a priori known. This allows for a selection of an optimal scale (or range of scales) such that we have a best case comparison. For the Gaussian derivative network, scale is controlled by the standard deviation parameter σ which is set to the half width of the crack. For the U-net, scale is adjusted by downscaling the image to match the crack width used in the training data. Here, we restrict the downscaling to discrete factors in the set $\{2, 4, 8, ...\}$. For widths 1 and 3, no downscaling is needed. For width 5, the images are downscaled by 2, for width 7 by 4, and for widths 9 and 11 by 8. For completeness, we include results for the U-net without downscaling denoted by "U-net plain". Table 6.2 yields the prediction quality measured by the Dice coefficient, while the other quality measures are shown in Figure 6.8. Exemplary segmentation results are shown

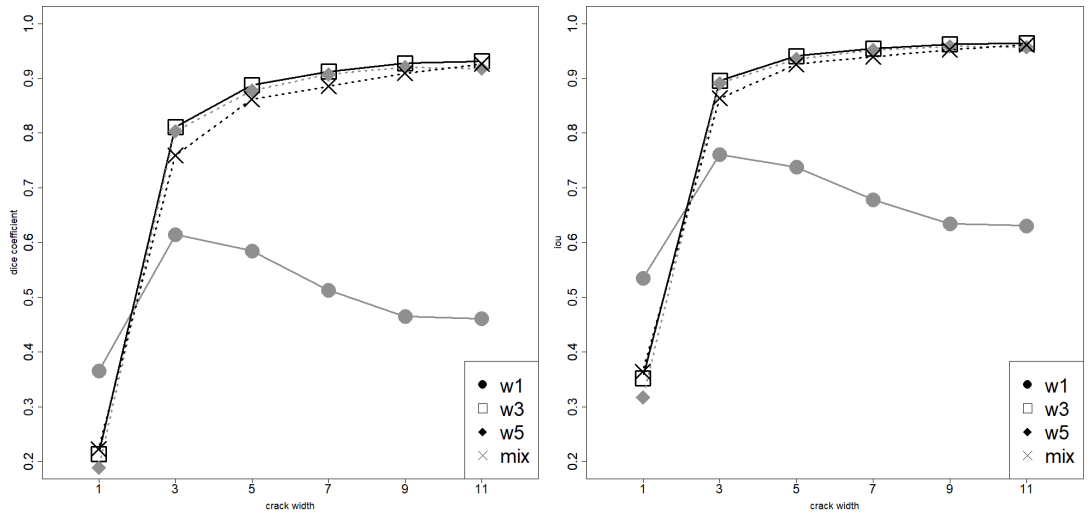


Figure 6.6: Experiment 1. Effect of the crack width in the training set on generalization to unseen scales. The baseline Riesz network is marked with 1 947, w3, and layer 4 and with square symbol □. Left: Dice, right: IoU.

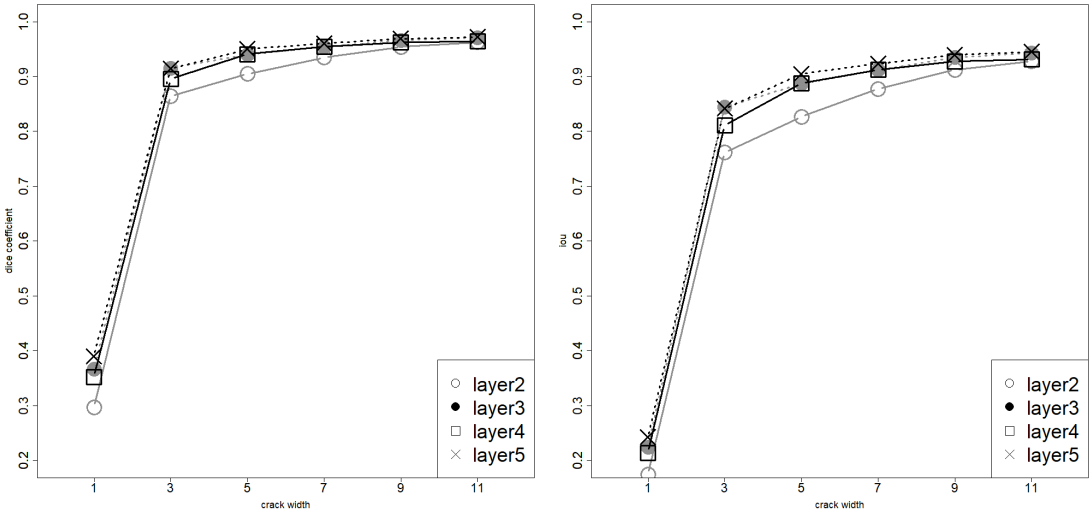


Figure 6.7: Experiment 1. Effect of the network depth on generalization to unseen scales. The baseline Riesz network is marked with 1 947, w3, and layer 4 and with square symbol □. Left: Dice, right: IoU.

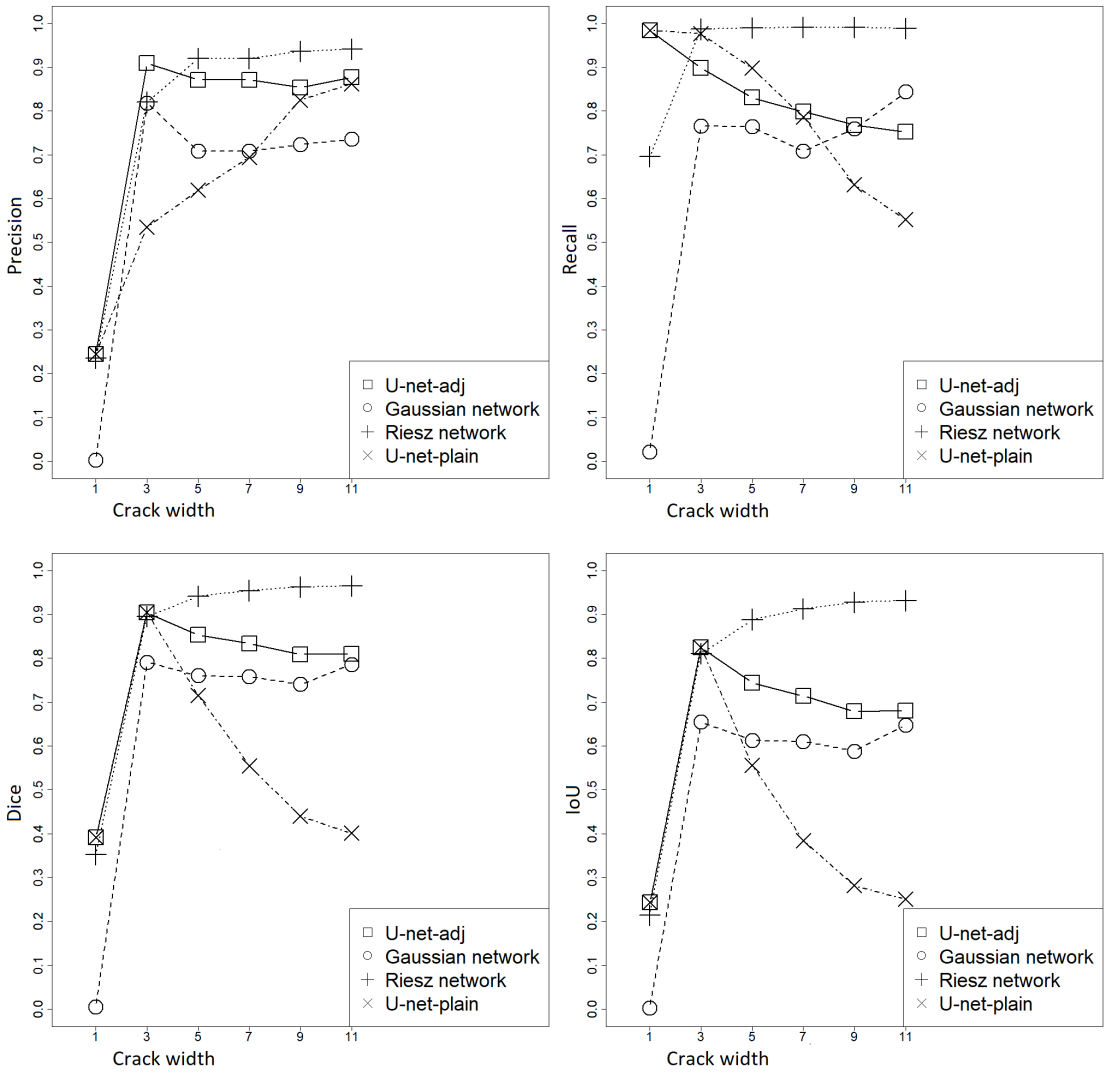


Figure 6.8: Experiment 1. Comparison of the competing methods. Results of the simulation study with respect to crack width. Training on crack width 3.

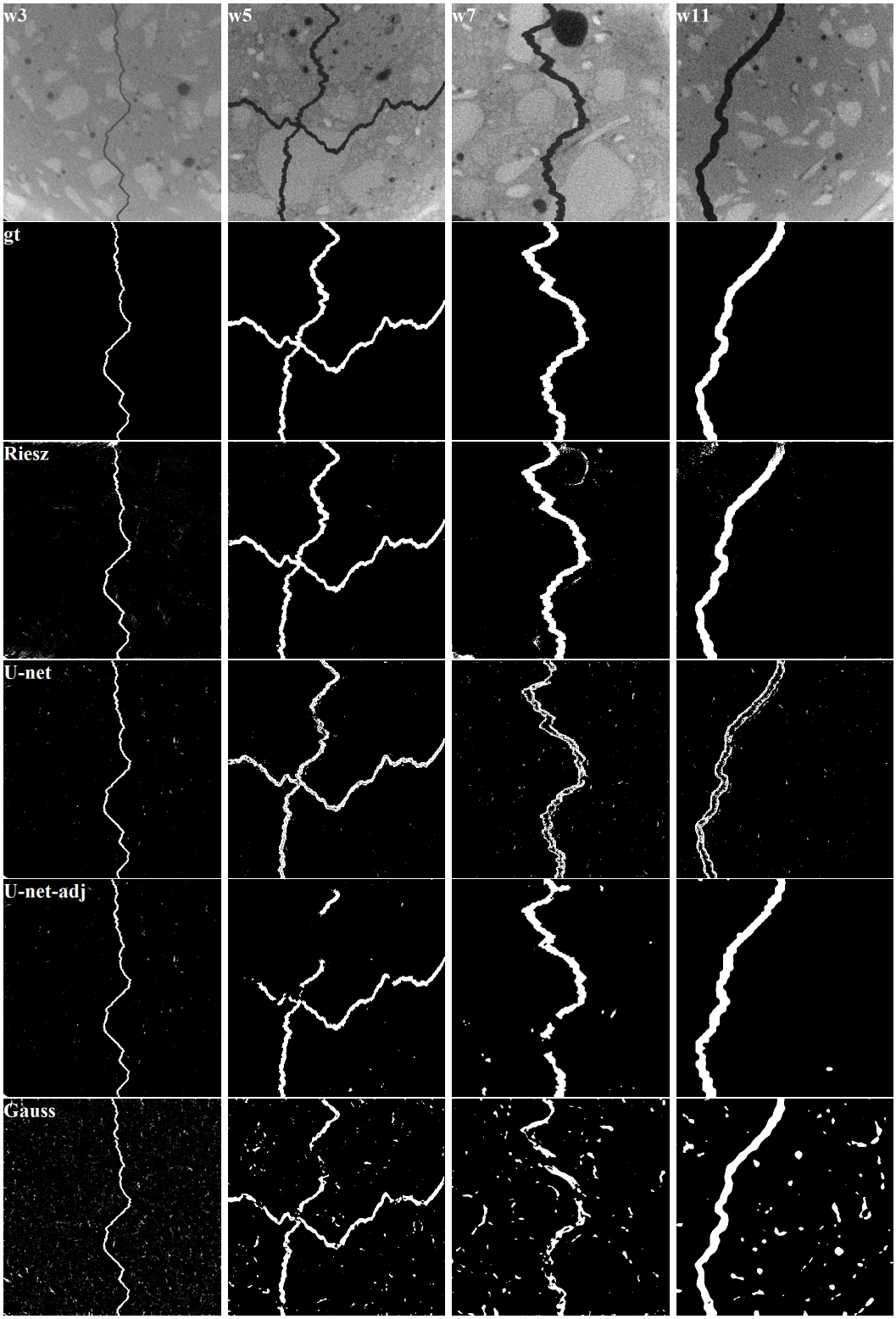


Figure 6.9: Experiment 1. Columns (from left to right): crack of widths 3, 5, 7, and 11. Rows (from top to bottom): input image, ground truth, Riesz network, plain U-net, U-net with scale adjustment, and Gaussian derivative network. All images have size 512×512 pixels.

Method	w1	w3	w5	w7	w9	w11
	Dice	Dice	Dice	Dice	Dice	Dice
U-net plain	0.391	0.904	0.715	0.555	0.440	0.401
U-net scale adj.	0.391	0.904	0.853	0.833	0.809	0.810
U-net-mix scale adj.	**0.420**	**0.917**	0.929	0.916	0.921	0.921
Gaussian network	0.004	0.765	0.764	0.709	0.759	0.843
Riesz network	0.352	0.895	**0.941**	**0.954**	**0.962**	**0.964**

Table 6.2: Experiment 1. Comparison with competing methods: Dice coefficients for segmentation of cracks of differing width. Training was performed on crack width 3. Best performing method bold.

in Figure 6.9.

As expected, the performance of the plain U-net decreases with increasing scale. Scale adjustment stabilizes U-net's performance but requires manual selection of scales. Moreover, the interpolation in upsampling and downsampling might induce additional errors. The decrease in performance with growing scale is still apparent ($10-15\%$) but significantly reduced compared to the plain U-net (55%). To get more insight into performance and characteristics of the U-net, we add an experiment similar to the one from [99]: We train the U-net on crack widths 1, 3, and 5 on the same number of images as for one single crack width. This case is referred to "U-net-mix scale adj." in Table 6.2. Scales are adjusted similarly: w5 and w7 are downscaled by factor 2, w9 and w11 are downscaled by factor 4. The results are significantly better than those obtained by the U-net trained on the single width ($10-15\%$ in Dice and IoU on unseen scales), but still remain worse than the Riesz network trained on a single scale (around 7% in Dice and IoU on unseen scales).

The Gaussian derivatives network is able to generalize steadily across the scales (Dice and IoU 74%) but nevertheless performs worse than the scale adjusted U-net (around 10% in IoU). Moreover, it is very sensitive to noise and typical CT imaging artifacts (Figure 6.9).

On the other hand, the Riesz network's performance is very steady with the growing scale. We even observe improving performance in IoU and Dice with the increase in crack thickness. This is due to pixels at the edge of the crack influencing the performance metrics less and less the thicker the crack gets. The Riesz network is unable to precisely localize cracks of width 1 as, due to the partial volume effect, such thin cracks appear to be discontinuous. With the exception of the thinnest crack, the Riesz network has Dice coefficients above 94% and IoU over 88% for completely unseen scales. This even holds for cases when the crack is more than 3 times thicker than the one used for training.

6.3.3 Experiment 2: Performance on multiscale data

Since cracks are naturally multiscale structures, i.e. crack thickness varies as the crack propagates, the performance of the considered methods on multiscale data is analyzed as well. On the one hand, we want to test on data with an underlying ground truth without relying on manual annotation prone to errors and subjectivity. On the other hand, the experiment should be conducted in a more realistic and less controlled setting than the previous one, with cracks as similar as possible to real ones.

We therefore use again simulated cracks, this time however with varying width [102]. The thickness is modeled by an adaptive dilation. See Figure 6.10 for realization examples. The change rendering our experiment more realistic than the first one is to exploit no prior information about the scale. The Riesz network does not have to be adjusted for this experiment while the competing methods require scale selection as described in Section 6.3.2. Without knowing the

scale, testing several configurations is the only option. Note that in this experiment we used a different crack simulation technique [102] than in Experiment 1. In principle, we cannot claim that either of the two techniques generates more realistic cracks. However, this change serves as an additional goodness check for the methods since these simulation techniques can be seen as independent.

We adjust the U-net as follows: We downscale the image by several factors from $\{2, 4, 8, 16...\}$. The forward pass of the U-net is applied to the original and every downscaled image. Subsequently, the downscaled images are upscaled back to the original size. All predictions are joined by the maximum operator. We report results for several downscaling factor combinations specified by a number N, which is the number of consecutive downscaling factors used, starting at the smallest factor 2. Similarly as in Experiment 1, we report results of two U-net models: the first model is trained on cracks of width 3 as the other models in the comparison. The second model is trained on cracks with mixed widths. Including more crack widths in the training set has proven to improve the scale generalization ability in Experiment 1. Hence, the second model represents a more realistic setting that would be used in practice where the crack width is typically unknown. We denote the respective networks as "U-net pyramid" N and "U-net-mix pyramid" N.

For the Gaussian network, we vary the standard deviation parameter σ in the set $\{1.5, 3, 6, 12\}$. This selection of scales is motivated by the network having been trained on crack width 3 with $\sigma = 1.5$. We start with the original σ and double it in each step. As for the U-net, we test several configurations, now specified by the number N of consecutive σ values used, starting at the smallest (1.5). We denote the respective network "Gaussian network" N.

Results are reported in Table 6.3 and Figure 6.10. We observe a clear weakness of the Riesz network in segmenting thin cracks (Figure 6.10, first and last row). Despite of this, the recall is still quite high (90%). However, this could be due to thicker cracks - which are handled very well - contributing stronger to these statistics as they occupy more pixels. Nevertheless, the Riesz network deals with the problem of the wide range scales in an elegant way, just with a single forward pass of the network.

The performance of the U-net improves with including more levels in the pyramid, too. However, this applies only up to a certain number of levels after which the additional gain becomes minimal. Moreover, applying the U-net on downscaled images seems to induce oversegmentation of the cracks (Figure 6.10, second and third row). Including a variety of crack widths in the training set improves the overall performance of U-net in all metrics. This confirms the hypothesis that U-net significantly benefits from variations in the training set. However, this model of U-net is still outperformed by the Riesz network trained on a single crack width. The Gaussian network behaves similarly as the U-net, with slightly worse performance (according to Dice or IoU) but better crack coverage (Recall). As the number of σ values grows, the recall increases but at the same time artifacts accumulate across scales reducing precision. The best balance on this data set is found to be three scales.

6.3.4 Experiment 3: Application to cracks in CT images of concrete

Finally, we check the methods' performance on real data: cracks in concrete samples generated by tensile and pull-out tests. In these examples, the crack thickness varies from 1 or 2 pixels to more than 20 pixels (Figure 6.11). This motivates the search for methods that automatically generalize to completely unseen scales. Here, we can assess the segmentation results qualitatively, only, as no ground truth is available. Manual segmentation of cracks in high resolution images is time consuming and prone to individual biases.

The first sample (Figure 6.11, first row) is a concrete cylinder with a glass fiber reinforced

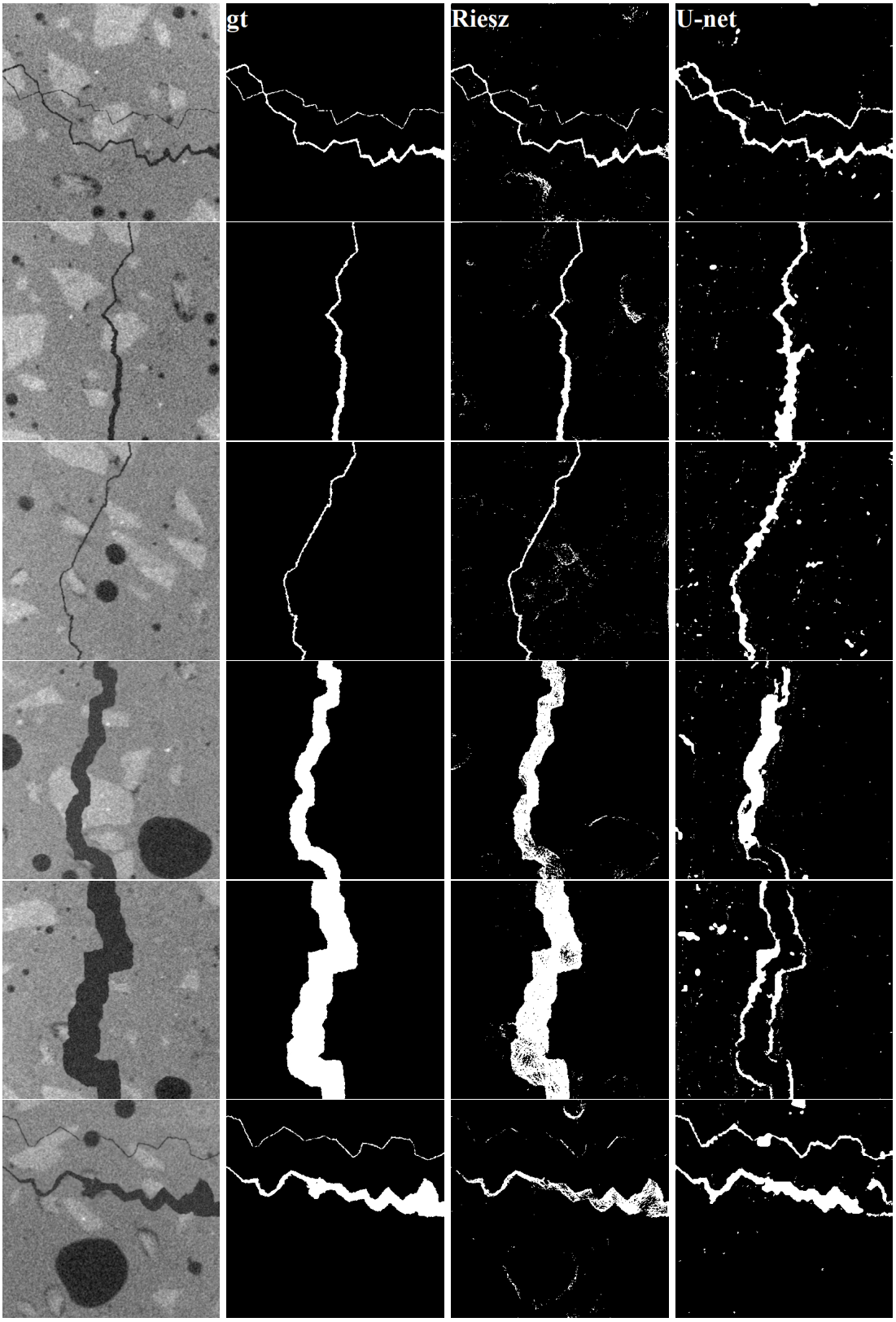


Figure 6.10: Experiment 2. Cracks with varying width. From left to right: input image, ground truth, results of the Riesz network, and the U-net with 4 pyramid levels. Image size 400×400 pixels.

Method	Multiscale cracks			
	Precision	Recall	Dice	IoU
U-net, plain	0.655	0.322	0.432	0.275
U-net pyramid 2	0.598	0.518	0.555	0.384
U-net pyramid 3	0.553	0.623	0.586	0.414
U-net pyramid 4	0.496	0.705	0.582	0.411
U-net-mix, plain	0.471	0.288	0.358	0.218
U-net-mix pyramid 2	0.626	0.646	0.635	0.466
U-net-mix pyramid 3	0.624	0.804	0.703	0.542
U-net-mix pyramid 4	0.583	0.899	0.707	0.547
Gaussian network 2	0.553	0.503	0.527	0.358
Gaussian network 3	0.418	0.735	0.533	0.364
Gaussian network 4	0.306	0.857	0.451	0.291
Riesz network	**0.901**	**0.902**	**0.902**	**0.821**

Table 6.3: Experiment 2. Performance on simulated multiscale cracks.

composite bar embedded along the center line. A force is applied to this bar to pull it out of the sample and thus initiate cracking. Cracking starts around the bar and branches in three directions: left, right diagonal, and down (very subtle, thin crack). Crack thicknesses and thus scales vary depending on the crack branch. As before, our Riesz network is able to handle all but the finest crack thicknesses efficiently in a single forward pass without specifying the scale range. The U-net on the image pyramid requires a selection of downsampling steps, accumulates artifacts from all levels of the pyramid, and slightly oversegments thin cracks (left branch).

The second sample (Figure 6.11, second row) features a horizontal crack induced by a tensile test. Here we observe permanently changing scales, similar to our simulated multiscale data. The crack thickness varies from a few to more than 20 pixels. Once more, the Riesz network handles the scale variation well and segments almost all cracks with minimal artifacts. In this example, U-net covers the cracks well, too, even the very subtle ones. However, it accumulates more false positives in the areas of concrete without any cracks than the Riesz network.

6.3.5 Robustness to additive Gaussian noise

In this section, we investigate how performance of methods changes with respect to the random Gaussian process with increasing standard deviation. The goal of this experiment is to analyze performance of the Riesz network to "poor" imaging conditions when trained on the original data.

The dataset with fixed crack width 3 from above was used. First, images are normalized with min-max normalization to be in the range $[0, 1]$, then random Gaussian noise is added, and again min-max normalized. We test it for six different Gaussian random processes with fixed seed 0, i.e. for six standard deviations $\sigma \in \{0, 0.01, 0.05, 0.1, 0.2, 0.5\}$ (Figure 6.12 and Figure 6.13). Here, standard deviation 0 means that no noise has been added to the image. The Riesz network is compared with U-net-mix, a version of the U-net from before that was trained on all three types of fixed crack widths. This model of the U-net is preferred rather than the one trained on a single crack width because the goal here is to test the models that would be used in practice. However, we use fixed crack width to remove the effect of the scale selection for the U-net and to measure only the effect of the noise.
Results are shown in Table 6.4. Segmentation results are shown in Figure 6.12 and Figure 6.13. For a very small amount of noise ($\sigma = 0.01$) both methods are stable. However, as more noise

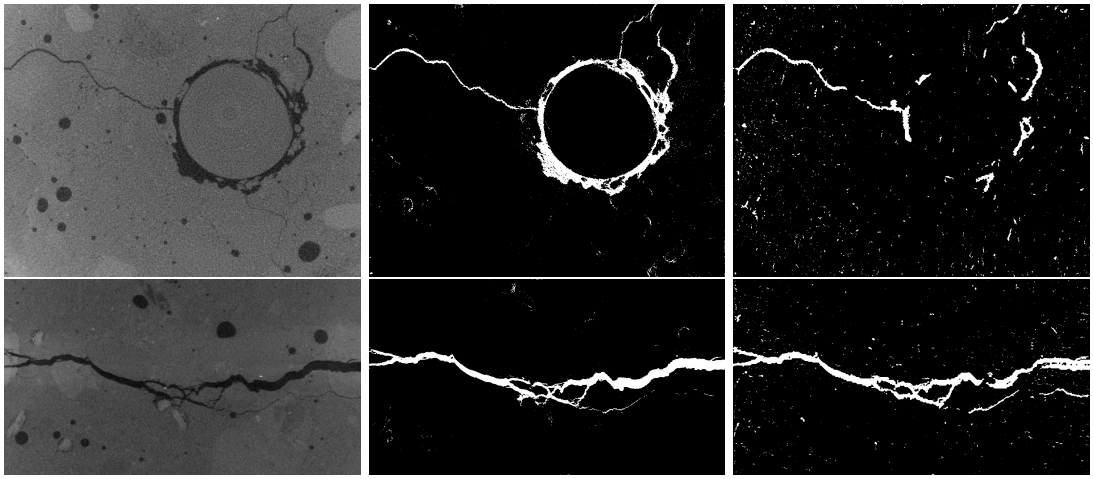


Figure 6.11: Experiment 3. Real cracks in concrete from pull-out test (first row) and tensile test (second row): slice from input CT image, results of the Riesz network and of U-net with 2 pyramid levels. Image sizes are $832 \times 1\,088$ (1st row) and 544×992 (2nd row).

seed 0	0	0.01	0.05	0.1	0.2	0.5
Riesz-precision	0.8198	0.8177	**0.7310**	**0.5175**	**0.2399**	**0.0799**
U-net-mix-precision	**0.9062**	**0.8883**	0.4415	0.1344	0.0368	0.0228
Riesz-recall	**0.9866**	**0.9857**	**0.9490**	**0.7745**	**0.3545**	**0.1145**
U-net-mix-recall	0.9273	0.9064	0.4584	0.1410	0.0406	0.02424
Riesz-Dice	0.8955	0.8939	**0.8259**	**0.6205**	**0.2861**	**0.0941**
U-net-mix-Dice	**0.9167**	**0.8973**	0.4498	0.1376	0.0386	0.0242

Table 6.4: Robustness to additive Gaussian noise on a dataset with fixed crack width 3: comparison of the Riesz network with U-net-mix with growing standard deviation of σ of Gaussian random noise.

is added, the performance of the U-net-mix steeply decays. Even for $\sigma = 0.05$, the performance of the U-net-mix degrades with Dice close to 0.45. For the same σ, the Riesz network is much better (Figure 6.12): Dice is almost 0.83, while recall is still high, i.e. close to 0.95. With further increase in σ, the performance of the Riesz network also decreases, however the decay is more gradual than for U-net-mix. Already at $\sigma = 0.2$, both methods are not able to produce meaningful results anymore (Figure 6.13).
This experiment implies that the Riesz network is more robust to additive Gaussian noise and in general indicates that the Riesz network could perform better with respect to changes in imaging conditions, e.g. contrast, noise, and illumination.

6.3.6 Application to fiber reinforced concrete

It is a well-known weakness of concrete that it has low tensile strength, i.e. under high tensile force it fails abruptly and explosively. For that reason, reinforcement material is mixed with the cement paste creating a composite material. Most common reinforcements are steel rebars. Nowadays, fibers have become widely used as reinforcement in concrete creating a new class of reinforced concrete materials, e.g. ultra high performance fiber-reinforced concrete [121, 122, 123]. A variety of materials can be used for fiber material, including glass, carbon, and basalt. Since

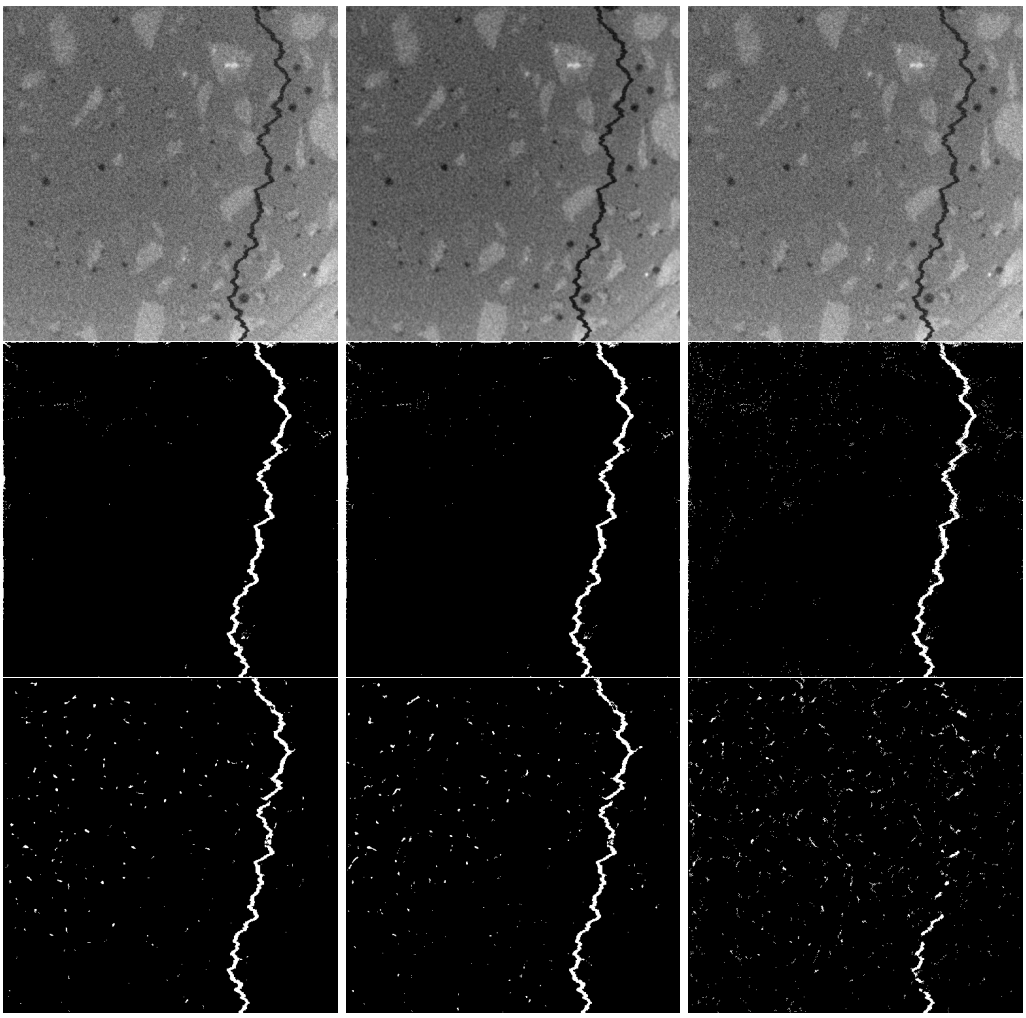


Figure 6.12: Robustness to additive Gaussian noise. First row: input image, second row: crack segmentation from the Riesz network, third row: crack segmentation from U-net-mix. From left to right: $\sigma = 0, 0.01, 0.05$.

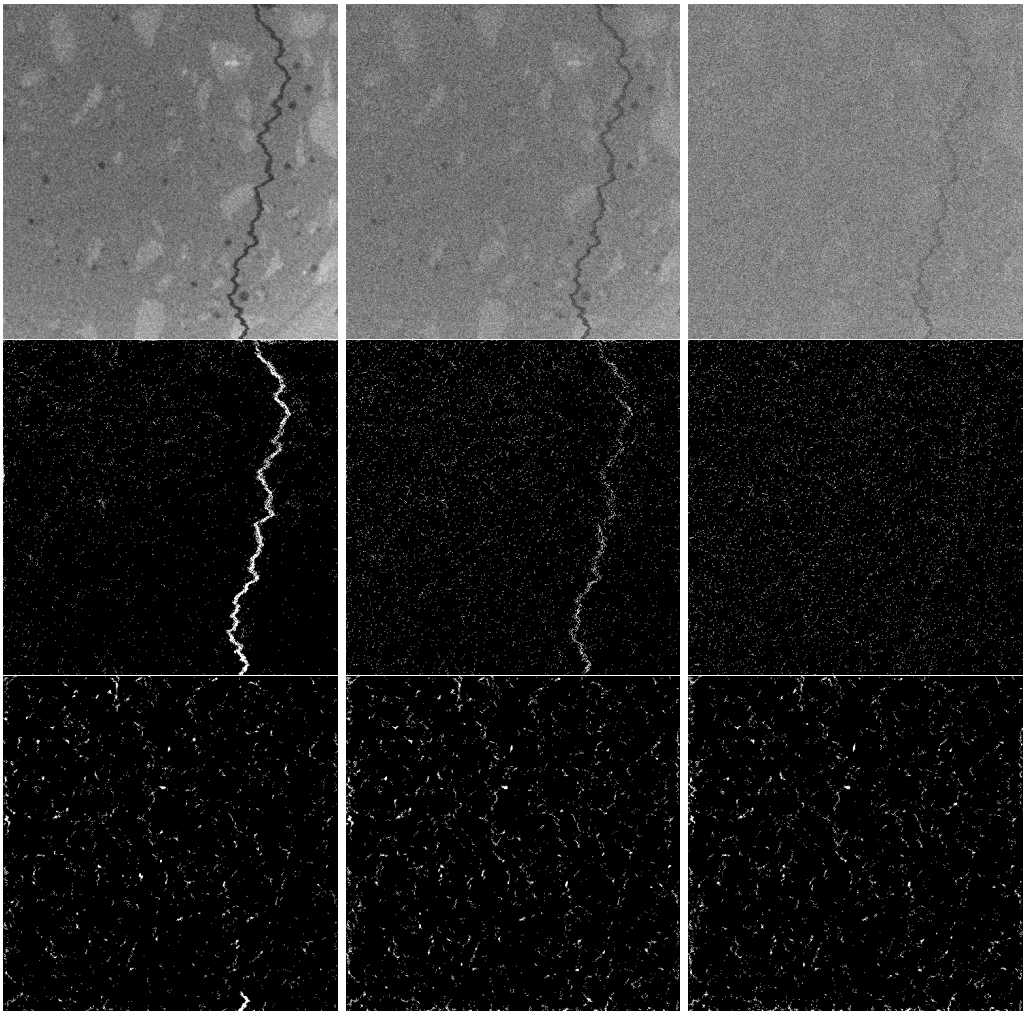


Figure 6.13: Robustness to additive Gaussian noise. First row: input image, second row: crack segmentation from the Riesz network, third row: crack segmentation from U-net-mix. From left to right: $\sigma = 0.1, 0.2, 0.5$.

all of these materials have different mechanical properties, the properties of fiber reinforced concrete are connected to the properties of the concrete mixture, including the fiber material. Hence, a lot of effort has recently been invested in the investigation of fiber reinforced concrete samples with various material configurations. In the context of CT imaging, different materials mean different energy absorption properties, i.e. fibers can appear both brighter or darker than concrete, which can result in very different images. In the context of crack segmentation, this means that our methods should be able to efficiently handle these variations. This section compares the performance of the Riesz network, U-net, and U-net-mix from the previous sections on three different fiber reinforced concrete images: Figure 6.14, Figure 6.15, and Figure 6.16. We comment on possible post-processing steps to improve results and discuss the robustness of the methods in the context of fiber reinforced concrete.

In Figure 6.14 high performance concrete (HPC) with polypropylene fibers as reinforcement is shown. See [124] for more details on sample and crack initiation. In this image, fibers are long and appear dark and hence interfere with the crack in the center. All three methods are able to extract the central and dominant crack in the middle. The Riesz network is not able to segment the thin crack on the left from the main crack, contrary to both U-nets. However, both U-nets accumulate a much larger amount of misclassified noise compared to Riesz network.

In Figure 6.15 sample reinforced with steel fibers is shown. For more details see [55, 125]. In this image, fibers appear bright and create uneven illumination effects. We use a simple pre-processing step to understand if we can reduce this effect and improve the performance of the methods. Simple morphological openings with square structuring elements of half-sizes 2 and 5 are used for that purpose. As the size of the structuring element increases, segmentation results improve for all three methods. While the Riesz network struggles with low contrast cracks on the right, both types of U-net segment falsely many non-crack voxels.

The CT image from Figure 6.16 originates from ultra high performance fiber reinforced concrete (UHPFRC) with steel fibers [122]. Here, fibers turn out to be bright structures in the images. These extremely highly X-ray absorbing fibers affect the gray value dynamics of the CT images. Morphological openings with square structuring elements of half-sizes 2 and 5 are applied to reduce this effect. As we increase the size, crack segmentation improves for the Riesz networks. Both types of U-net segment large amount of noise, even with opening as a pre-processing step, rendering them as ineffective for this sample.

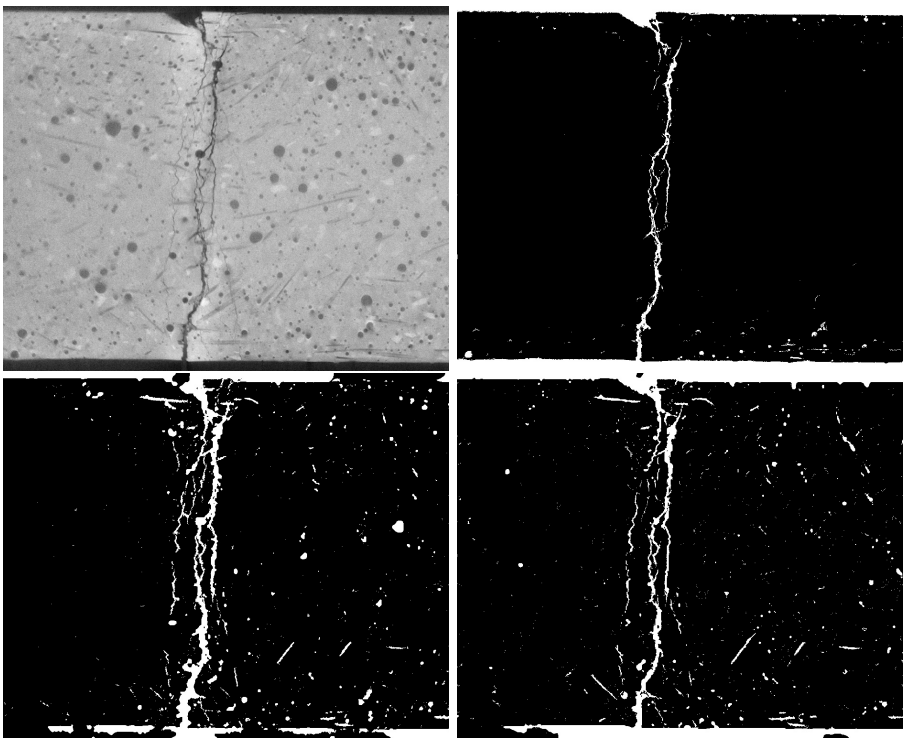


Figure 6.14: Cracks in high-performance concrete with polypropylene fibers. First row: input image (left), segmentation results from the Riesz network (right). Second row: U-net (left) and U-net mix (right). Image size is 933×764.

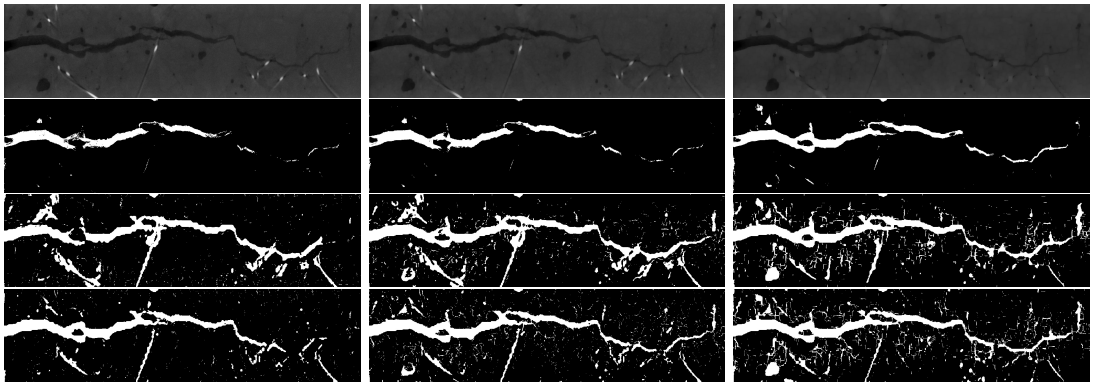


Figure 6.15: Cracks in concrete with steel fibers. Rows: input image, segmentation results from the Riesz network, U-net and U-net mix, respectively. Columns: original images, images after applying square closing of half-size 2, and images after applying square opening of half-size 5. Image size is $1\,295 \times 336$.

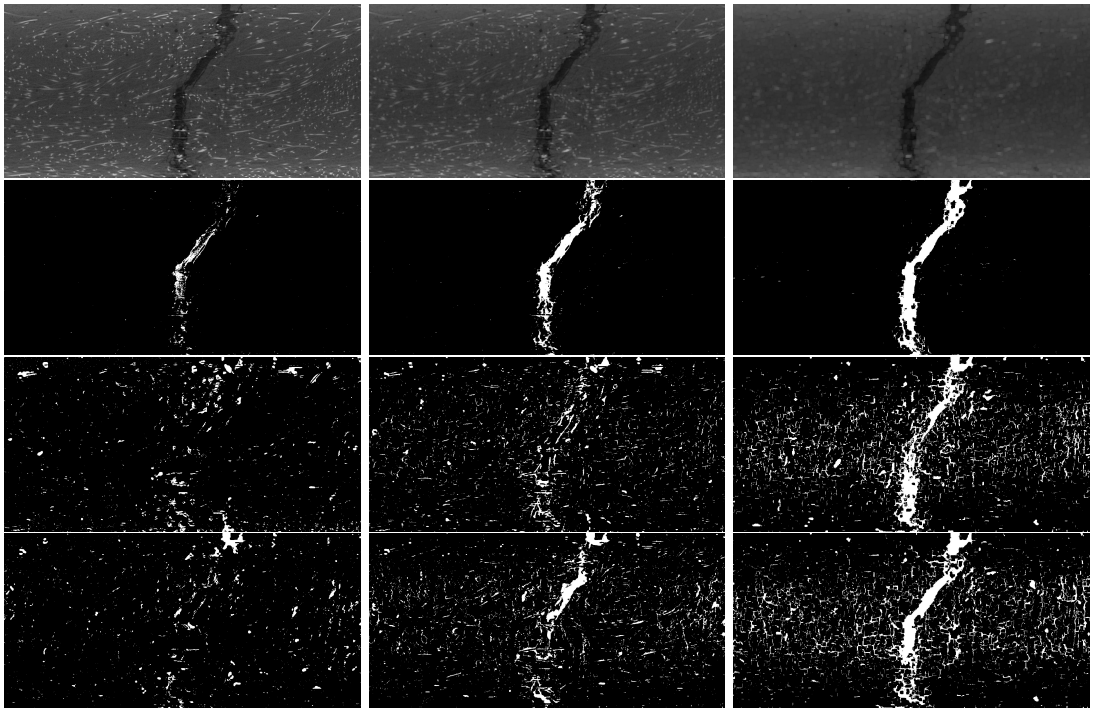


Figure 6.16: Cracks in concrete with ultra high performance fiber-reinforced concrete (UHPFRC) with steel fibers. Rows (from left to right): input image, segmentation results from the Riesz network, U-net and U-net mix, respectively. Column: original images, images after applying square opening of half-size 2, and images after applying square closing of half-size 5. Image size is $1\,579 \times 772$.

6.4 Results on MNIST Large Scale dataset

We test the Riesz networks on a classification task on the MNIST Large Scale[3] [99] to test wider applicability of Riesz networks outside of crack segmentation task. This data set was derived from the MNIST data set [68] and it consists of images of digits between 0 and 9 belonging to one of ten classes (Figure 6.17) which are rescaled to a wide range of scales to test scale generalization abilities of neural networks (Figure 6.18).

Figure 6.17: 10 classes in MNIST Large Scale data set. All images have size 112×112.

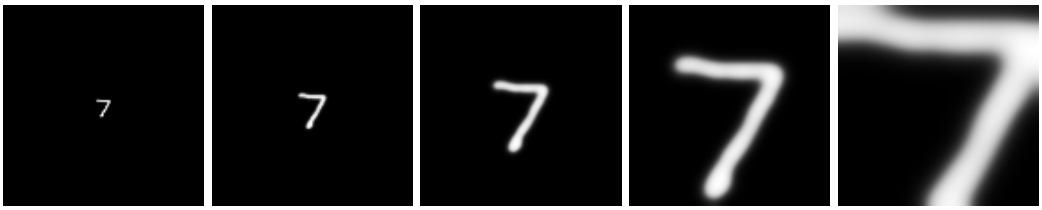


Figure 6.18: Variation of scales in MNIST Large Scale data set (from left to right): scales 0.5, 1, 2, 4 and 8. All images have size 112×112.

Our Riesz network has the channel structure 12-16-24-32-80-10 with the softmax function at the end. In total, it has 20 882 parameters. Following [117], only the central pixel in the image is used for classification. We use the standard CNN described in [99] but without any scale adjustments as a baseline to illustrate the limited scale generalization property. This CNN has the channel structure 16-16-32-32-100-10 with the softmax function at the end and in total 574 278 parameters. The training set has 50 000 images of the single scale 1. We used a validation set of 1 000 images. The test set consists of scales ranging in $[0.5, 8]$ with 10 000 images per scale. All images have size 112×112. Models are trained using the ADAM optimizer [119] with default parameters for 20 epochs with a learning rate 0.001 which is halved every 3 epochs. Cross-entropy is used as loss function.

Figure 6.19 shows validation and training loss during 20 epochs. Interestingly, the Riesz network converges faster and even its validation loss remains lower than the training loss of CNN. Accuracies for the different scales are shown in Table 6.5.

[3] https://www.zenodo.org/record/3820247

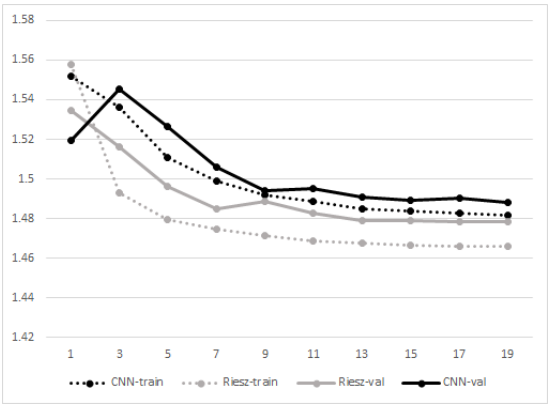


Figure 6.19: Train and validation loss per epoch for the Riesz network and CNN (as a baseline).

scale	0.5	0.595	0.707	0.841	1	1.189	1.414	1.682
CNN	40.74	64.49	88.35	96.87	97.77	96.08	80.06	38.68
Riesz	96.34	97.59	98.06	98.54	98.58	98.50	98.45	98.40
Riesz-pad20	96.33	97.57	98.07	98.48	98.63	98.54	98.49	98.46
Riesz-pad40	96.34	97.55	98.07	98.47	98.63	98.58	98.53	98.44
FovAvg 17ch tr1 [99]	98.58	99.05	**99.33**	**99.39**	**99.40**	**99.39**	**99.38**	**99.36**
FovMax 17ch tr1 [99]	**98.71**	**99.07**	99.27	99.34	99.37	99.35	99.36	99.34

scale	2	2.378	2.828	3.364	4	4.757	5.657	6.727	8
CNN	25.90	24.91	23.64	21.34	19.91	18.87	18.04	15.64	11.79
Riesz	98.39	98.24	98.01	97.51	96.42	93.5	81.58	67.66	51.82
Riesz-pad20	98.39	98.35	98.33	98.16	97.78	97.08	95.48	91.10	79.78
Riesz-pad40	98.46	98.39	98.34	98.29	98.16	97.80	96.82	**93.75**	**83.6**
FovAvg 17ch tr1 [99]	**99.35**	99.31	99.22	99.12	98.94	98.47	96.20	89.17	71.31
FovMax 17ch tr1 [99]	99.33	**99.35**	**99.34**	**99.35**	**99.34**	**99.27**	**97.88**	92.76	79.23

Table 6.5: Classification accuracy (in %) of MNIST Large Scale data set. Best performing method bold.

The Riesz network shows stable accuracy for scales in the range $[0.5, 4]$. The CNN, which has way more degrees of freedom, is only competitive for scales close to the training scale. Results for two scale adjusted versions of the CNN as reported in [99] are also given in Table 6.5. Their performance is slightly superior to the Riesz network (around $1-2\%$). However, it is important to note that this approach uses (max or average) pooling over 17 scales.

Further works considering the MNIST Large Scale data set are [112, 117]. Unfortunately, no numeric values of the accuracies are provided, so we can compare the results only qualitatively. The Riesz network's accuracy varies less on a larger range of scales than those of the scale-equivariant networks on Gaussian or morphological scale spaces from [112] that were trained on scale 2. The Gaussian derivative network [117] trained on scale 1 yields results in a range between 98% and 99% for medium scales $[0.7, 4.7]$ using pooling over 8 scales. The Riesz network yields similar values but without the need for scale selection.

On the smallest scale of 0.5, the Riesz network seems to give a better result than [117], while it is outperformed on the largest scales. The reason for the latter is that digits start to reach the

boundary of the image. To reduce that effect, we pad the images by 20 and 40 pixels with the minimal gray value. Indeed, this improves the accuracy significantly for larger scales (Table 6.5), while it remains equal for the rest of the scales. For example, for scale 8, accuracy increases from 51.8% to 79.8% (padding 20) and 83.6% (padding 40). This is a better accuracy than that reported in [117] and [99] for models trained on scale 1.

6.5 Discussion

In this chapter we introduced a new type of scale invariant neural network based on the Riesz transform as filter basis instead of standard convolutions. Our Riesz neural network is scale invariant in one forward pass without specifying scales or discretizing and sampling the scale dimension. Its ability to generalize to scales differing from those trained on is tested and validated in segmenting cracks in 2d slices from CT images of concrete. Usefulness and elegance of the method become manifest in the fact that only one fixed scale is needed for training, while preserving generalization to completely unseen scales. This reduces the effort for data collection, generation or simulation. Furthermore, our network has relatively few parameters (around 18k) which reduces the danger of overfitting.

Experiments on simulated yet realistic multiscale cracks as well as on real cracks corroborate the Riesz network's potential. Compared to other deep learning methods that can generalize to unseen scales, the Riesz network yields improved, more robust, and more stable results.

A detailed ablation study on the network parameters reveals several interesting features: This type of networks requires relatively few data to generalize well. The Riesz network proves to perform well on a data set of approximately 200 images before augmentation. This is particularly useful for deep learning tasks where data acquisition is exceptionally complex or expensive. The performance based on the depth of the network and the number of parameters has been analyzed. Only three layers of the network suffice to achieve good performance on cracks in 2d slices of CT images. Furthermore, the choice of crack thickness in the training set is found to be not decisive for the performance. Training sets with crack widths 3 and 5 yield very similar results.

The two main weaknesses of our approach in the crack segmentation task are undersegmentation of thin cracks and edge effects around pores. In CT images, thin cracks appear darker than thicker cracks due to the partial volume effect. For the same reason thin cracks look disconnected. Thin cracks might therefore require special treatment. In some situations, pore edge regions get erroneously segmented as crack. These can however be removed by a post-processing step and are no serious problem.

To unlock the full potential of the Riesz transform, validation on other types of problems is needed. In the following chapter, the method is adjusted for 3d since CT data is originally 3d.

An interesting topic for further research is to join translation and scale invariance with rotation invariance to design a new generation of neural networks with encoded basic computer vision properties [115]. This type of neural network could be very efficient because it would have even less parameters and hence would require less training data, too.

Chapter 7

The Riesz network in 3d: crack segmentation in CT images of concrete

Concrete is the base material in civil engineering, e. g. in buildings and bridges. It is a combination of aggregates, cement, and water. Its quality, durability, and mechanical stability have to be ensured. This is connected to its material composition, e.g. water-to-cement ratio, type of cement paste used, type of aggregates used, drying conditions, the addition of reinforcements, etc. For investigating concrete behaviour under load, various stress tests can be applied to concrete specimens. Concrete is known to be a brittle material with high compressive strength but low tensile strength. Hence, concrete tends to develop cracks under load. For example, during tensile testing, cracks occur when the applied force exceeds the tensile strength of the concrete. With the increase of force, the specimen may eventually fail. To increase tensile strength, reinforcement is added during the mixing phase. This improves tensile strength, but still mechanical properties of new materials have to be thoroughly studied before they can be used in critical applications such as building or bridge construction. Shape, size, and location of the cracks, as well as the sample's post-cracking mechanical behaviour provide valuable information on the specific properties of the studied concrete. Characterization of cracks can help to understand and analyze the effect of adding reinforcements in the concrete. With the naked eye, the cracks are only visible on the sample surface and neither their full 3d structure nor the correlation of the crack with local structural features can be observed.

Micro-computed tomography (µCT) can provide the missing information. This imaging method enables unique, non-destructive insight into 3d microstructures. The most important components of a typical laboratory CT device are the X-ray emitting tube, the turntable holding the sample, and the flat bed detector (Figure 7.1). During the scan, the sample is rotated by a predefined angular increment. In each position, a 2d X-ray projection image is taken. Finally, the 3d image of the sample is reconstructed computationally from the projections. In laboratory CT devices, the X-rays usually form a cone. The position of the sample between the X-ray tube and the detector therefore controls the magnification and together with the number and size of the detector pixels the voxel size. The voxel gray values represent essentially local X-ray attenuation, which in turn depends on the atomic numbers of the elements passed by the beam. Concrete matrix, aggregates, and air filled pores absorb X-rays differently. Thus, the corresponding voxels should feature different gray values, too. The heterogeneous composition and varying density of

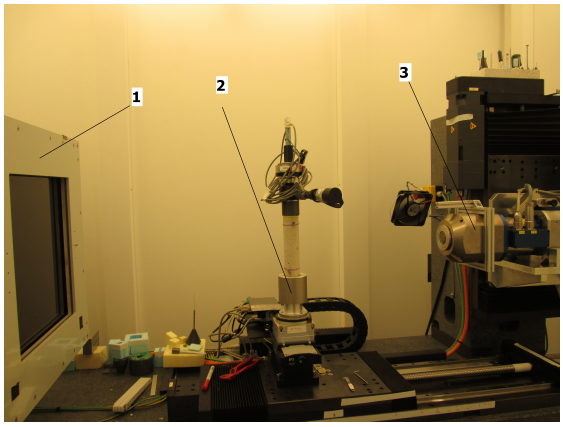


Figure 7.1: CT setup at Fraunhofer ITWM, Kaiserslautern, Germany. 1 - flat bed detector, 2 - turntable holding the sample, 3 - X-ray emitting tube.

the concrete however result in varying voxel gray values within the concrete matrix.

A major benefit of CT being a non-destructive method is the following: the same concrete specimen can be scanned several times during the mechanical test and even in-situ. However, to gain the desired structural information, the reconstructed µCT images need to be processed and analyzed. These images can be very large (up to $2\,000^2 \times 10\,000$ voxels). Hence, manual crack segmentation is not feasible and automatic image processing is needed.

This chapter deals with automatic crack segmentation in 3d CT images of concrete using Riesz networks. Here, we consider a crack to be a two dimensional air filled structure or surface with varying width, shape, topology, and orientation. Since cracks are essentially empty spaces, they appear dark in CT images. We designed computational studies based on semi-synthetic data with simulated cracks that enable objective comparison of the Riesz network to the established methods. This is followed by the application to real crack data on various types of concrete samples and various types of cracks.

7.1 Riesz network: adjustments for 3d

Extension of the Riesz network from 2d to 3d is straightforward: only the number of Riesz transforms increases. In 2d there are five Riesz transforms of first and second order

$$\{\mathcal{R}_1, \mathcal{R}_2, \mathcal{R}^{(2,0)}, \mathcal{R}^{(1,1)}, \mathcal{R}^{(0,2)}\},$$

while in 3d there are nine Riesz transform of first and second order

$$\{\mathcal{R}_1, \mathcal{R}_2, \mathcal{R}_3, \mathcal{R}^{(2,0,0)}, \mathcal{R}^{(0,2,0)}, \mathcal{R}^{(0,0,2)}, \mathcal{R}^{(1,1,0)}, \mathcal{R}^{(1,0,1)}, \mathcal{R}^{(0,1,1)}\}.$$

The three layer Riesz network we apply here can be written as $(1, 16, 16, 32, 1)$. It has $(1 \cdot 9 \cdot 16 + 16) + (16 \cdot 9 \cdot 16 + 16) + (16 \cdot 9 \cdot 32 + 32) + (32 \cdot 1 + 1) = 7\,153$ trainable parameters. This is most remarkable when comparing to the 3d U-net from [126] with more than 2 million parameters.

Details on training For the training set we use semi-synthetic realistic crack images with simulated cracks [42] and hence we have unambiguous ground truth available. The training set

consists of three crack images of width 3 and three crack images of width 5 of size 256^3 cropped into non-overlapping 64^3 images. This training set corresponds to the same crack images as in [100, 126], but without the crack images of width 1, see also Section 7.3.1 for more details. In total, the training set has 1302 images of size 64^3. As loss function binary cross entropy is used and optimized with ADAM [119] with batch size 2 for 20 epochs with an initial learning rate of 0.005 which halves every 5 epochs.

7.1.1 Need for window adjustment in 3d

As a reminder our Riesz network in 2d from the previous chapter has 18 825 trainable parameters and can be written as $(1, 16, 32, 40, 48, 1)$. In 3d it has 7 153 trainable parameters and can be written as $(1, 16, 16, 32, 1)$. It is logical to assume that when moving from 2d to 3d, more parameters would be needed rather than less. The reason for this is that the cracks in 3d can have larger variations in crack topology, shape, orientation, etc. than the cracks in 2d due to an extra degree of freedom which 3d has in comparison to 2d. However, for the Riesz networks we are limited with memory during computations in 3d. In practice, 3d images often have a larger number of voxels than 2d and hence it is easier to run out of memory. For example, for a 400×400 image, in 2d the Riesz network required memory is bounded by the number of feature maps in the fifth step of the networks, i.e. $48 \times 400 \times 400 = 768 \times 10^4 = 7.68$ million float variables are needed. On the other hand, for a $400 \times 400 \times 400$ image in 3d the Riesz network required memory is bounded by $32 \times 400 \times 400 \times 400 = 1\,024 \times 10^6 = 2\,048$ million float variables. Hence, memory requirements increase fast from 2d to 3d. These memory limitations make the Riesz network unable to work on large 3d CT images, e.g. images of size $1\,000^3$. Hence, we make adjustments by cropping the input image into nonoverlapping 400^3 cubes and segmenting cracks on these subwindows. Finally, the original image is assembled from subwindows. This adjustment by limiting the window size disables the Riesz network to be fully scale invariant with respect to scales larger than a cropped image window. However, scale invariance to all the scales smaller than the cropped window size is preserved. The experimental part of this chapter shows that in most cases taking the window size of 400^3 is enough to capture all the interesting cracks.

7.2 Crack segmentation on CT images: previosuly known methods

First studies [44, 127, 128] on crack segmentation methods for 3d CT images date back to 2011 and focus on classical methods from image processing, e.g. sheet filter, template matching, etc. Recently, methods from machine and deep learning such as 3d U-net and random forest have successfully been used for crack segmentation tasks [42], too. Another important contribution of [42] is the ability to simulate realistic crack structures with fractorial Brownian motion. Furthermore, this study analyzed and compared several methods from classical image processing and machine learning on semi-synthetic image data with the corresponding ground truths. The best-performing methods were Hessian-based percolation [44] and the 3d U-net [129]. Subsequently, both approaches were successfully adapted to real CT data of concrete [100, 124, 126].

We take these two methods used in several studies [100, 124, 126], namely Hessian-based percolation [44] and 3d U-net [42]. Since these are well-tested methods, they serve as a baseline to which the Riesz network can be compared to. We refer to Appendix C for details on the methods. For simulation studies we compare the Riesz network only to the 3d U-net because they belong to the same class of methods, i.e. deep learning. For real data the Riesz network

is compared to both methods. Here, the goal is to analyze the usefulness of the Riesz network among all methods that could potentially be used for automatic crack segmentation in 3d CT images of concrete.

7.3 Experiments on simulated data

Quality metrics for experiments on simulated data are described in Appendix D.

7.3.1 Comparison with neural networks on a simulated dataset with fixed width cracks

In this part, we test our methods on a dataset from [42]. For the ground truth image of the crack, a fractional Brownian surface is used. We create a dataset of 60 semi-synthetic crack images of size 256^3 voxels (Figure 7.3). It consists of 20 images per crack width 1, 3, and 5. Each of the three groups consists of eight images with one crack, six images with two cracks in parallel planes, and six images with two cracks in orthogonal planes (Figure 7.2). The background patches are extracted from four 3d CT images of concrete. The concrete samples represent the same concrete type consisting of aggregates, cement matrix, and air pores. The size of air pores and aggregates varies in each concrete specimen. Since the 3d CT images are sufficiently large, we can use different background patches for each image.
Out of the 60 images, nine (three per group) are reserved for training the learning methods. One image of each group is used for validation. The remaining images are used for the evaluation of the methods. This dataset was designed to evaluate the performance of the methods with respect to the varying crack shape, topology, orientations, and simple branching. By keeping the crack width fixed, this dataset removes the effect of the scale on the performance. Nevertheless, this dataset serves as a valuable benchmark to understand how Riesz network performs compared to 3d U-net, a well-tested and well-investigated method.

Details on the models The Riesz network is compared to the two models of 3d U-net: U-net single scale and U-net multiple scale. Both of these models have the same architecture and number of parameters but differ on the training data, i.e. on the subset of the set that was reserved for training.

U-net single scale is trained only on a single scale depending on the crack width in the test set. Hence, it represents three models trained on crack widths 1, 3, and 5, each trained on 768 images. U-net multiple scale is one model which is trained on all three scales simultaneously, i.e. in total on 2304 images. This model is optimized for several crack widths and hence is a more realistic type of model which could be used in practice. Both of these models have around 2 million trainable parameters and details on training can be found in [42]. Our Riesz network is trained on crack widths 3 and 5 simultaneously, see Section 7.1 for details. Crack width 1 seems to be different from the remaining two due to the fact that strict connectivity may not be preserved (Figure 7.4) and hence end up being harder to segment.

Results Table 7.1 and Figure 7.5 show the results of the Riesz network in comparison with two 3d U-net models trained on different data [42, 100]. The Riesz network does not perform well on crack width 1, while on the other two crack widths performance is significantly better. For width 3 and 5 recall from the Riesz network is very competitive with both U-nets, outperforming them in recall for crack width 3. This indicates that the Riesz network is indeed able to recognize crack structure in the image. However, the main problem is low precision, i.e. high sensitivity to noise.

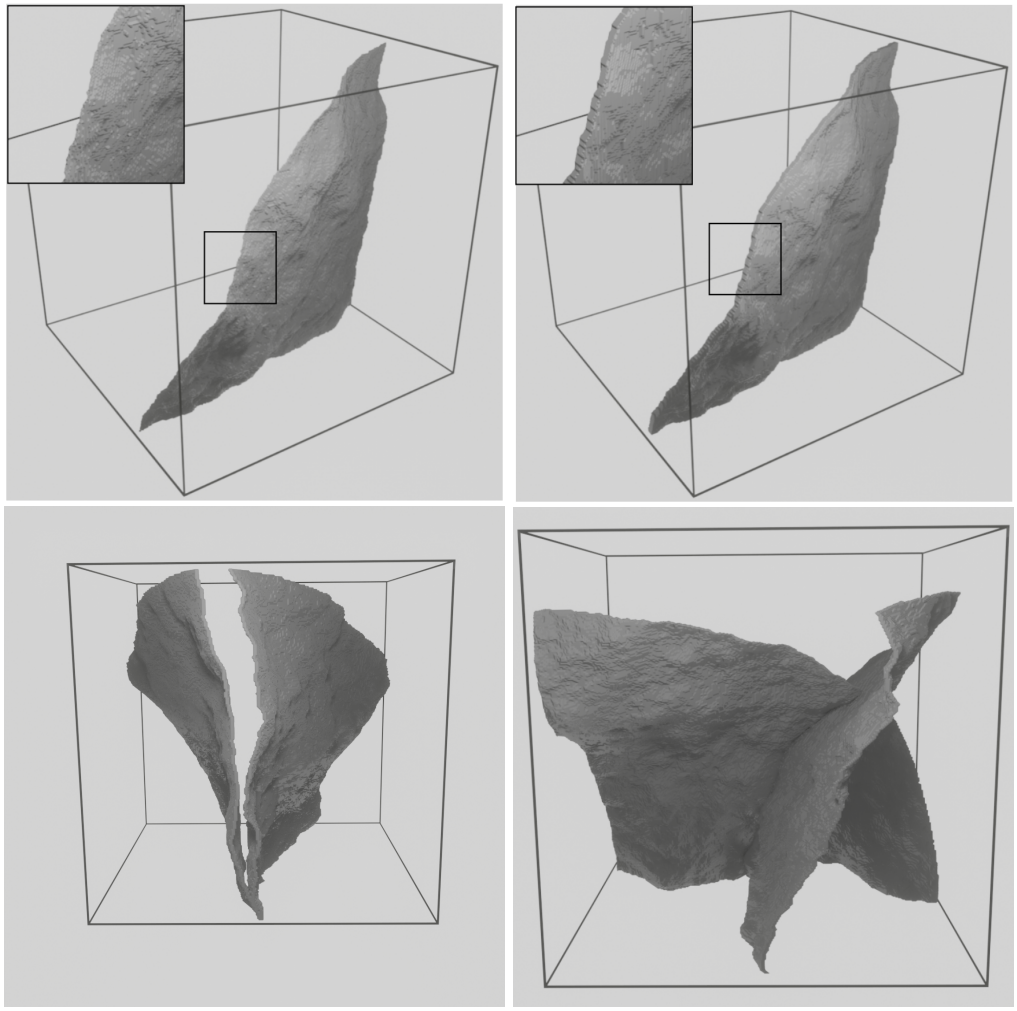


Figure 7.2: 3d renderings of simulated crack surfaces in an image of size 256^3. First row: crack width 1 and crack width 5. Second row: two cracks of width three in parallel planes and two cracks of width three in orthogonal planes.

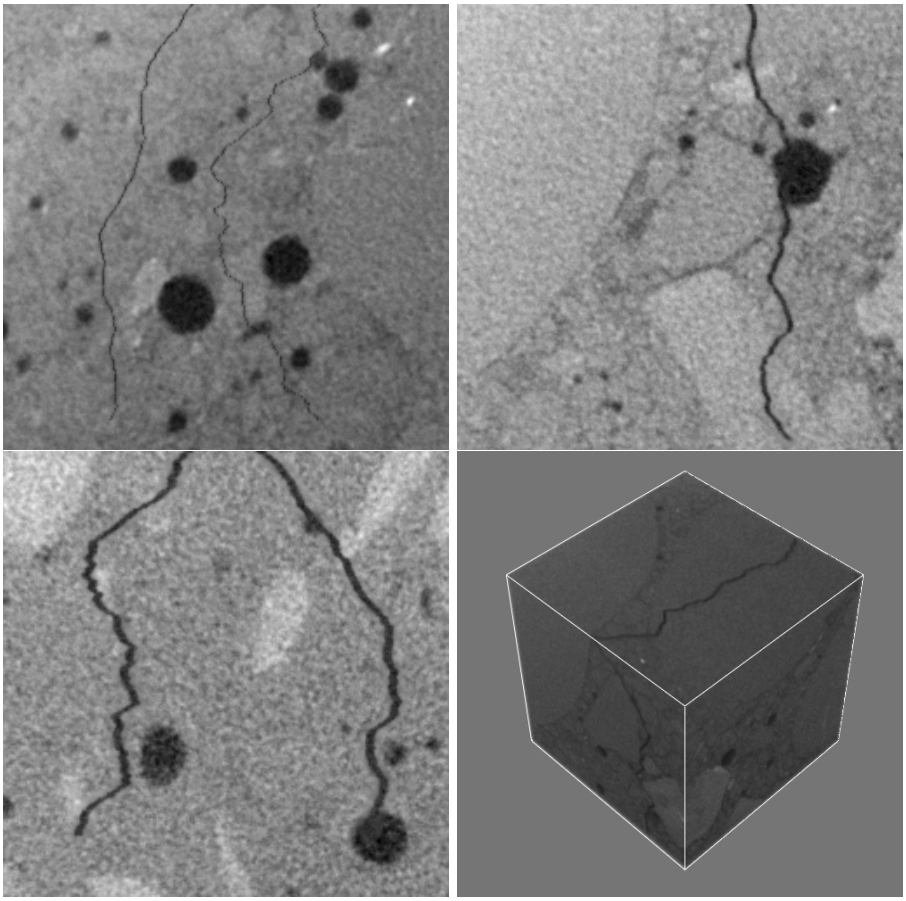


Figure 7.3: From left to right: 2d slices of the images featuring simulated cracks (crack width 1, 3, and 5 voxels, image size 256^2 pixels), 3d rendering of a simulated crack (image size 256^3 voxels).

This could be due to a low number of parameters which makes precise crack localization and distinction from noise hard. When using tolerance[1] 1, the median F1 score of the Riesz network reaches 0.83 for crack width 3 and 0.9089 for crack width 3, while the median recall of 0.99 for both crack widths implies that the whole crack surfaces have been detected. With tolerance 1, the median F1 score of the Riesz network is still lower than those of both U-nets in the median F1-score (Figure 7.5).

Adding more parameters to the Riesz network A previous study on the Riesz networks in 2d indicates that $7k$ parameters may not be enough to achieve satisfying results. Here, we experiment with the 3d Riesz network with more parameters. We double the number of channels in each layer resulting in a network architecture which can be written as $(1, 32, 32, 64, 1)$ with in total 28 129 parameters. This model was trained analogously as the smaller Riesz network, i.e. with the same training set, number of epochs, batch size, learning rate, and its decay (Section 7.1.1). However, in *pytorch* this network is not able to run on full 256^3 voxel images without crashing. Hence, we run it on non-overlapping 128^3 windows and check if the increased number

[1] See Appendix D for an explanation of tolerance.

Crack width 1									
	3d U-Net single scales trained [42]			3d U-Net, multiple scale trained [100]			Riesz network *(crack width previously unseen)*		
	min	median	mean	min	median	mean	min	median	mean
precision	**0.9842**	**0.9956**	**0.9949**	0.5983	0.7768	0.7883	0.0480	0.1966	0.1907
(tol1)	(0.995)	(0.9996)	(0.9994)	(0.9159)	(0.9946)	(0.9891)	(0.0838)	(0.3250)	(0.3187)
recall	**0.6879**	**0.9539**	**0.9152**	0.3570	0.7872	0.7429	0.3277	0.8510	0.79108
(tol1)	(0.8847)	(0.9847)	(0.9710)	(0.7063)	(0.9725)	(0.9332)	(0.6677)	(0.966)	(0.9388)
F1 score	**0.8129**	**0.9739**	**0.9509**	0.4955	0.7854	0.7468	0.0838	0.3233	0.3397
(tol1)	(0.9387)	(0.9921)	(0.9846)	(0.8271)	(0.9827)	(0.9581)	(0.1820)	(0.4799)	(0.4691)

Crack width 3									
	3d U-Net single scale trained [42]			3d U-Net, multiple scales trained [100]			Riesz network		
	min	median	mean	min	median	mean	min	median	mean
precision	0.6494	**0.9574**	0.9383	**0.9173**	0.9504	**0.9544**	0.2474	0.5334	0.5101
(tol1)	(0.7091)	(0.9962)	(0.9728)	(0.9592)	(0.9984)	(0.9952)	(0.3624)	(0.7135)	(0.6655)
recall	0.9028	0.9578	0.9565	**0.9044**	0.9580	0.9553	**0.9044**	**0.9757**	**0.9694**
(tol1)	(0.9409)	(0.991)	(0.9838)	(0.9561)	(0.9917)	(0.9879)	(0.9519)	(0.9959)	(0.9879)
F1 score	0.7824	**0.9516**	**0.9451**	**0.92533**	0.9483	0.9546	0.3958	0.7063	0.6673
(tol1)	(0.8294)	(0.9900)	(0.9767)	(0.9654)	(0.9956)	(0.9925)	(0.5320)	(0.8328)	(0.7947)

Crack width 5									
	3d U-Net single scale trained [42]			3d U-Net, multiple scales trained [100]			Riesz network		
	min	median	mean	min	median	mean	min	median	mean
precision	0.895	0.9743	0.9692	**0.9503**	**0.9806**	**0.9803**	0.4452	0.7077	0.6882
(tol1)	(0.9828)	(0.9988)	(0.9972)	(0.9940)	(0.9996)	(0.9991)	(0.6023)	(0.8198)	(0.8012)
recall	0.616	0.9354	0.9036	**0.7675**	0.948	**0.936**	0.4369	**0.9735**	0.9307
(tol1)	(0.7877)	(0.9927)	(0.9720)	(0.9560)	(0.9969)	(0.9917)	(0.7880)	(0.9950)	(0.9809)
F1 score	0.7517	0.9537	0.9334	**0.8605**	**0.968**	**0.957**	0.4825	0.8281	0.7897
(tol1)	(0.8745)	(0.9954)	(0.9822)	(0.9774)	(0.9979)	(0.9954)	(0.7422)	(0.9089)	(0.8854)

Table 7.1: Results of the deep learning methods on the simulated datasets with fixed width cracks. The highest value is given in bold. Metrics with tolerance 1 are given in brackets. Visualizations on F1 score and recall are shown in Figure 7.5.

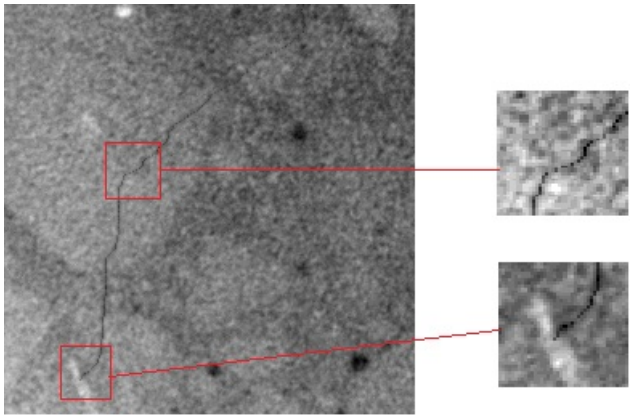


Figure 7.4: CT image with a simulated crack of width 1. Red boxes show disconnectivity of the thin crack structure which makes the correct segmentation harder.

of parameters improves precision on cracks on this dataset.

Table 7.2 and Figure 7.5 compare the two types of Riesz networks on the same dataset with three crack widths. We notice that recall remains roughly the same as in the smaller Riesz network, while precision significantly improves $(10 - 20\%)$ and consequently the F1 score. However, these results still lag the precision and F1 metrics of the U-nets in Table 7.1. A possible explanation is that memory restrictions for 3d images disable the Riesz network from having enough degrees of freedom (i.e. parameters) to improve localization of the segmented cracks in 3d settings.

Adding post processing: keep 5 largest connected components Extracting the largest connected components can reduce the effect of noise in the segmentation results and hence improve precision. Here, we keep the 5 largest connected components as a simple post-processing step. Results are also shown in Table 7.2. With this simple post-processing step, precision rises around $8 - 9\%$ without tolerance for all three crack widths. This further improves the F1 score. Most importantly, recall remains roughly the same in all three cases. This experiment shows that the Riesz network could benefit even more with more sophisticated post-processing and close the gap between two types of U-net on these datasets with fixed crack widths. When using tolerance 1, for both post-processing strategies F1-score comes close to 0.89 for crack width 1 and 0.94 indicating overall good performance in this task.

7.3.2 Comparison with neural networks on a simulated dataset with multiscale cracks

As alternative to the fractional Brownian surface, crack structures can be modelled as a minimal surfaces from a realization of a random spatial Voronoi tessellation [102]. The locally varying thickness featured by real cracks can be captured by adaptive dilation of the crack. That is, a morphological dilation with the size of the structuring element varying according to a size map is applied. Here, the size map is kept constant in the yz plane while changing in x direction. This means that in each 2d x slice of the 3d image of the crack, the foreground is dilated N_i times by a 2×2 pixel square structuring element, where i is the slice index. The $N_i,\ i = 0, \ldots, n$ are incremented following a Bernoulli distribution with parameter $0 < p < 1$. This crack is integrated into the real concrete background in the same way as for cracks from fractional Brownian surface.

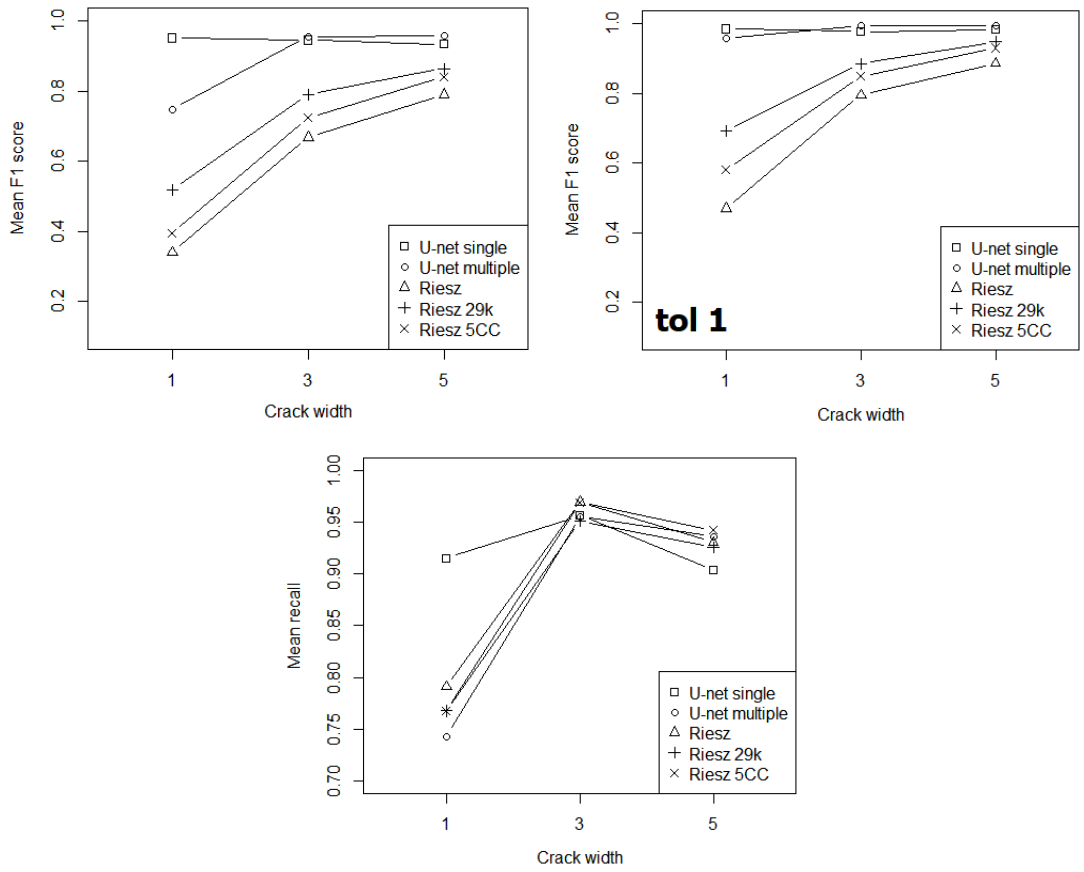


Figure 7.5: Visualization of the mean F1 score (first row) and recall (second row) for the models from Table 7.1 and Table 7.2. The Riesz network lags the two U-net models in the F1 score, while giving similar recalls (except for width 1). Improvements in the F1 score can be achieved by using models from Table 7.2 (first row left) with a small decrease in recall (second row). When using tolerance 1 (first row right) this gap in the F1 score is even smaller.

Crack width 1									
	Riesz network $29k$ parameters			Riesz network $7k$ parameters			Riesz network 5 largest connected components		
	min	median	mean	min	median	mean	min	median	mean
precision	0.1524	0.3897	0.3973	0.0480	0.1966	0.1907	0.0529	0.2752	0.2722
(tol1)	(0.2826)	(0.5656)	(0.5485)	(0.0838)	(0.3250)	(0.3187)	(0.0991)	(0.4132)	(0.4454)
recall	0.2922	0.8635	0.7675	0.3277	0.8510	0.79108	0.2211	0.84151	0.7677
(tol1)	(0.67)	(0.9564)	(0.9206)	(0.6677)	(0.966)	(0.9388)	(0.3862)	(0.9526)	(0.8865)
F1 score	0.2134	0.5401	0.51834	0.0838	0.3233	0.3397	0.0854	0.4172	0.39420
(tol1)	(0.3982)	(0.7032)	(0.6906)	(0.1820)	(0.4799)	(0.4691)	(0.1578)	(0.5813)	(0.5793)

Crack width 3									
	Riesz network $29k$ parameters			Riesz network $7k$ parameters			Riesz network 5 largest connected components		
	min	median	mean	min	median	mean	min	median	mean
precision	0.3358	0.7223	0.6903	0.2474	0.5334	0.5101	0.2760	0.628	0.5905
(tol1)	(0.4748)	(0.8566)	(0.8144)	(0.3624)	(0.7135)	(0.6655)	(0.4042)	(0.8072)	(0.7587)
recall	0.75302	0.9692	0.9515	0.9044	0.9757	0.9694	0.9043	0.9756	0.9693
(tol1)	(0.8627)	(0.9918)	(0.9785)	(0.9519)	(0.9959)	(0.9879)	(0.9511)	(0.9959)	(0.9875)
F1 score	0.5017	0.8182	0.78955	0.3958	0.7063	0.6673	0.4315	0.7509	0.7233
(tol1)	(0.6437)	(0.9185)	(0.8847)	(0.5320)	(0.8328)	(0.7947)	(0.5756)	(0.8889)	(0.8478)

Crack width 5									
	Riesz network $29k$ parameters			Riesz network $7k$ parameters			Riesz network 5 largest connected components		
	min	median	mean	min	median	mean	min	median	mean
precision	0.6367	0.8557	0.8181	0.4452	0.7077	0.6882	0.5716	0.7976	0.7709
(tol1)	(0.7479)	(0.8942)	(0.8885)	(0.6023)	(0.8198)	(0.8012)	(0.7698)	(0.8995)	(0.8916)
recall	0.3481	0.9674	0.9257	0.4369	0.9735	0.9307	0.42481	0.9778	0.9422
(tol1)	(0.6049)	(0.9902)	(0.9614)	(0.7880)	(0.9950)	(0.9809)	(0.7168)	(0.9945)	(0.9764)
F1 score	0.4787	0.9017	0.8638	0.4825	0.8281	0.7897	0.5474	0.8803	0.8407
(tol1)	(0.7114)	(0.9476)	(0.9340)	(0.7422)	(0.9089)	(0.8854)	(0.7903)	(0.9443)	(0.9301)

Table 7.2: Effect of the increasing the number of parameters in the Riesz network on the simulated datasets with fixed width cracks. Metrics with tolerance 1 are given in brackets. Visualizations on F1 score and recall are shown in Figure 7.5.

Tolerance 0									
	3d U-Net, multiscale [100]			3d U-Net, fine-tuned			Riesz network		
	min	median	mean	min	median	mean	min	median	mean
precision	0.822 (10)	0.920	0.931	**0.880** (10)	**0.976**	**0.984**	0.671 (10)	0.848	0.854
recall	0.182 (7)	0.859	0.929	0.046 (7)	0.878	**0.963**	**0.805** (7)	**0.918**	0.920
F1 score	0.307 (7)	0.869	0.918	0.088 (7)	**0.897**	**0.970**	**0.773** (10)	0.872	0.898

Tolerance 1									
	3d U-Net, multiscale [100]			3d U-Net, fine-tuned			Riesz network		
	min	median	mean	min	median	mean	min	median	mean
precision	0.962 (12)	0.989	0.992	**0.991** (4)	**0.997**	**0.998**	0.846 (10)	0.923	0.917
recall	0.374 (7)	0.923	0.984	0.096 (7)	0.913	**0.989**	**0.942** (7)	**0.983**	0.985
F1 score	0.544 (7)	0.946	0.986	0.175 (7)	0.930	**0.994**	**0.912** (10)	**0.957**	0.961

Table 7.3: Results of the multiscale and the fine-tuned 3d U-Net and the Riesz network on 15 images with synthetic multiscale cracks for tolerances 0 and 1. The images yielding the minimal values are named in parentheses. Results of this table are summarized in Figure 7.6.

We compare the Riesz network with two types of 3d U-net models. The first 3d U-net model is first pretrained on cracks generated by fractional Brownian surfaces and later trained further on additional crack images generated by Voronoi tessellations that are not in the test set. This second step of training is often called fine-tuning or calibration. Hence, we refer to this model as *3d U-net fine-tuned*. The second model is a standard 3d U-net from [100] trained on cracks generated by the fractional Brownian surface [42]. The difference between this 3d U-net model and the one from the previous section is in the inference step: during crack segmentation it is applied to several images downscaled by factors $\{1, 0.5, 0.25\}$, which are afterwards upscaled to the original image size. The final segmentation result is obtained by taking a voxel-wise maximum over all images. We refer to this as *3d U-net multiscale*. We compare the Riesz network with these two U-net models on 15 images (numbered 1 to 15) featuring Voronoi tessellation based multiscale cracks.

Results Results of this experiment are shown in Table 7.3, Figure 7.6, and Figure 7.7. 3d U-net fine-tuned achieves the highest values in precision and F1 score for this dataset. The Riesz network lags 3d U-net fine-tuned in median F1-score by only 0.02. This is especially encouraging since the Riesz network was not tuned on this type of data, contrary to this 3d U-net model. However, when given a crack image that deviates a lot in contrast or noise level from the training set (Figure 7.7, right), the performance of 3d U-net fine-tuned drops by a large factor as can be seen in the minimal values for these metrics in Table 7.3 and Figure 7.6. The same holds true for 3d U-net multiscale. On the contrary, the Riesz network seems to be more stable to variations in image quality, imaging conditions, etc. (Figure 7.7, right). When it comes to recall, the Riesz network dominates the two U-net models. When we use a tolerance of 1 voxel in the metrics, the Riesz network seems to become even more competitive in F1 score with 3d U-net fine-tuned.

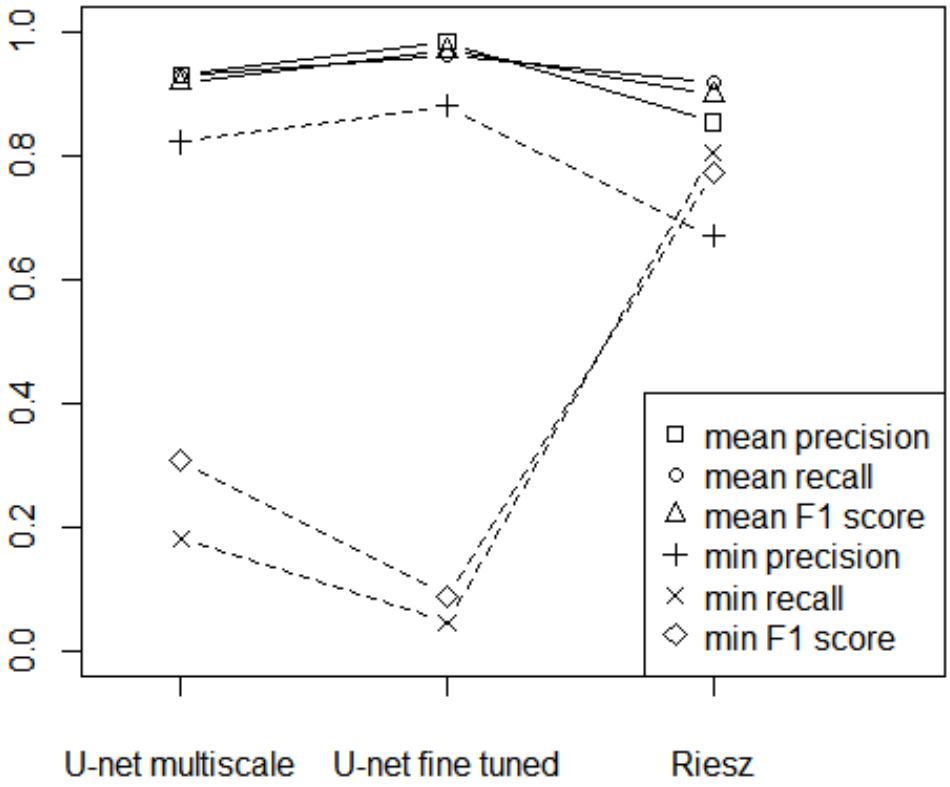
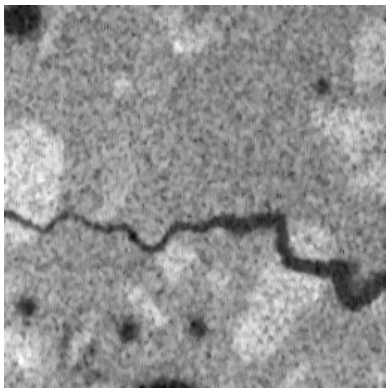
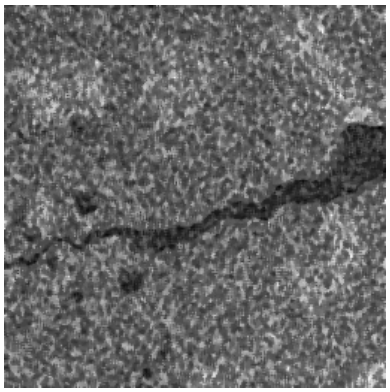


Figure 7.6: Visualization of the minimum (dashed line) and mean (solid line) values of the performance metrics for the three models from Table 7.3. The Riesz network has slightly worse mean values and significantly better minimum values of the performance metrics.

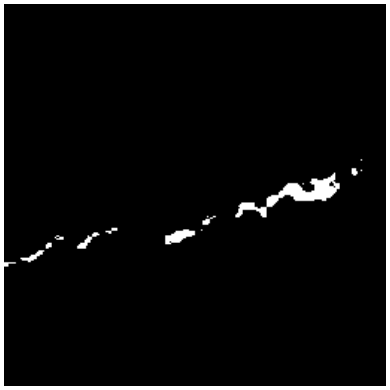


(a) Image 10 from the test dataset.



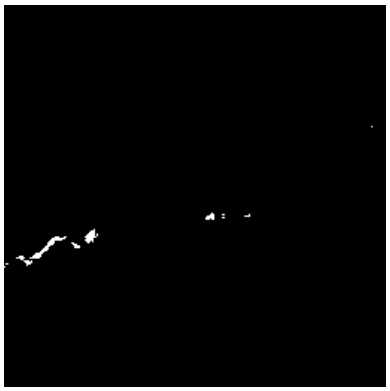
(b) Image 7 from the test dataset.

(c) Segmentation result from 3d U-net multiscale.

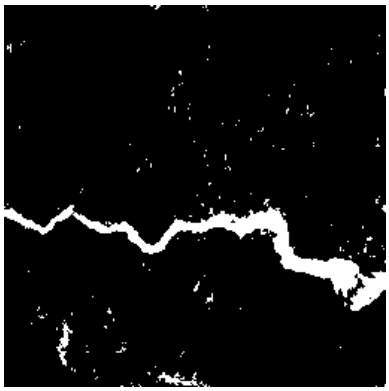


(d) Segmentation result from 3d U-net multiscale.

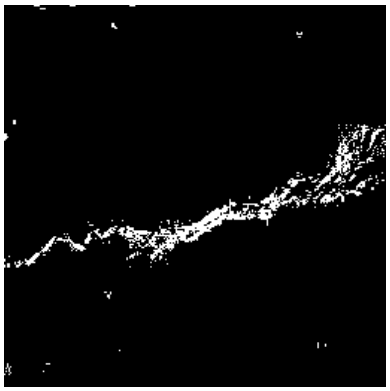
(e) Segmentation result from 3d U-net fine-tuned.



(f) Segmentation result from 3d U-net fine-tuned.



(g) Segmentation result from the Riesz network.



(h) Segmentation result from the Riesz network.

Figure 7.7: Segmentation results on the multiscale cracks dataset.

7.4 Application to real cracks in 3d CT images

Conclusion from simulation studies

The simulation studies show that the Riesz network represents a viable alternative to 3d U-net models. Riesz network achieves high recall on both simulated datasets with the exception of crack width 1. Precision is unfortunately lagging, probably due to a lower number of parameters than in the 2d case from Chapter 6. Precision can be improved by post-processing techniques such as extraction of the largest connected components.

An interesting observation about the Riesz network was made in the second simulation study on multiscale cracks: the Riesz network gave a more stable performance on images whose appearance deviates a lot from most of the images in the training set. This suggests that the Riesz network could be potentially better on varying types of concrete compositions, imaging conditions, etc. We keep this hypothesis in mind for the following sections of the chapter.

From simulated to real cracks

In this section we move from semi-synthetic images with available ground truth to real crack images with no ground truth. For real datasets it is hard to acquire reliable ground truth. Here, the only possibility remains manual or semi-automatic annotation. However, these images potentially consist of hundreds of slices, which are time consuming and exhausting to label slicewise. Tools for semi-automatic annotation for 3d data usually rely on annotating 2d slices, ignoring the depth, i.e. the third dimension. The reliability or quality of semi-automatic or manual annotation is also subject to doubt due to human factor[2]. For this reason, we evaluate results in this chapter only qualitatively by analyzing 2d slices and 3d renderings. 2d sliceview enables us to see the finer structure of segmented cracks, e.g. if very thin cracks are not segmented or if the crack is oversegmented. 3d renderings tell us about the global shape and structure of cracks and help us see if the segmented crack is salient or if the major part of the crack is not segmented. This chapter is based on eight CT images with real cracks which were originally segmented with 3d U-net and Hessian-based segmentation in the following studies [100, 124, 126]. Hence, these methods serve as a baseline to which Riesz network can be compared to. Parameter configurations for these two methods can be found in original publications (Appendix C)

7.4.1 Crack segmentation in normal and high performance concrete from tensile tests

This dataset was first used in [100]. Cracks in this dataset have large variations in crack width or scale requiring similar adaptation to multi-scale cracks as for the simulated multiscale cracks. However, here crack branching is completely random, i.e. less controlled than in simulated cracks, e.g. compare Figure 7.3 with Figure 7.8. Furthermore, transitions in crack scale are smoother and allow for changes in crack growth phases (thinning vs thickening) within the same crack, e.g. compare Figure 7.7 with Figure 7.8.

Concrete samples

The cylindrical test specimens with a diameter of 48.0 mm consist of a normal-strength (normal concrete - NC) and a high-strength (high performance concrete - HPC) fiber reinforced concrete.

[2]For example, there are differences in perceptions between annotators which can affect the consistency of the annotations. There are also possible variations in the consistency for the same annotator, e.g. at the beginning and at the end of the working day.

An amount of 2.0% by volume of fibers made from glass fiber reinforced polymer (GFRP) are added to each concrete mix. The test specimens are clamped in a tensile testing machine with a maximum tensile load of 10 kN. The objectives of the test series are to open the initial crack and to achieve a defined crack width between 0.1 and 0.4 mm.

CT imaging

Each sample is scanned by the laboratory μCT device at Fraunhofer ITWM, Kaiserslautern, Germany. A Feinfocus FXE 225.51 X-ray tube with a maximum acceleration voltage 180.3 kV and a maximum power of 12.7 W, and a Perkin Elmer flat-bed detector XRD 1621 with $2\,048 \times 2\,048$ pixels were used. The tube voltage is set to 180 kV and the integration time is 1 second. Tomographic reconstructions are obtained from 1 200 projections. Details on the resulting images are given in Table 7.4. To evaluate our methods, cropped versions of the images that contain most of the crack structure are considered.

	Size [voxels]	Cropped size [voxels]	Voxel edge length
NC12	$1\,912 \times 1\,538 \times 1\,913$	$1\,000 \times 550 \times 880$	22.7 μm
HPC11	$1\,905 \times 1\,593 \times 1\,905$	$1\,000 \times 500 \times 880$	22.7 μm
HPC1	$1\,981 \times 1\,981 \times 1\,677$	$1\,050 \times 1\,050 \times 600$	20.4 μm
NC2	$2\,009 \times 2\,009 \times 1\,665$	$1\,100 \times 1\,000 \times 520$	20.4 μm

Table 7.4: Information on the μCT image data.

Results

We compare the Riesz network from the simulation study to the 3d U-net and Hessian-based percolation, see Appendix C for details. Figure 7.8 gives a comparison of sliceview shown in *Figure 7* of [102], while Figure 7.9 and Figure 7.10 compare renderings from the three methods after post-processing. This includes the extraction of the largest connected components followed by the median filter on $3 \times 3 \times 3$. Additional sliceviews on the Riesz network's results before and after applying the median filter as a simple post-processing step are given in Figures 7.11, 7.12, 7.13, and 7.14.

From the renderings in Figure 7.9 we observe a similar appearance of cracks for all three methods. For sample NC12 the Riesz network seems to give the surface with the least holes, i.e. the most connected. This does not hold in general for HPC11. Here, the Riesz network's crack has holes at similar positions as the one from the 3d U-net and they are even bigger but it seems to perform better than both methods at the front left corner where all three methods struggle to segment a connected surface. In the renderings in Figure 7.10 the Riesz network's crack seems to have more holes in the crack surface than the cracks from the other two methods, but again segments more connected crack surface at the front left corner in the sample NC2. Results for HPC1 are almost identical for all three methods.

From the sliceview in Figure 7.8 segmentation results seem similar on the high level, but the Riesz network seems to give small improvements in connectivity, recall, and localization of cracks in the areas where crack orientation changes as marked by red boxes. This comes with a small expense of additional noise getting segmented. Figure 7.11 reflects the ability to handle large range variations in scale without any assumption on crack width except for the window adjustment due to computational reasons. The remaining two methods require the selection of the scale range and discretization of scales either through downsampling (3d U-net) or through the sampling of Gaussian scale space (Hessian-based percolation). Furthermore, we can notice

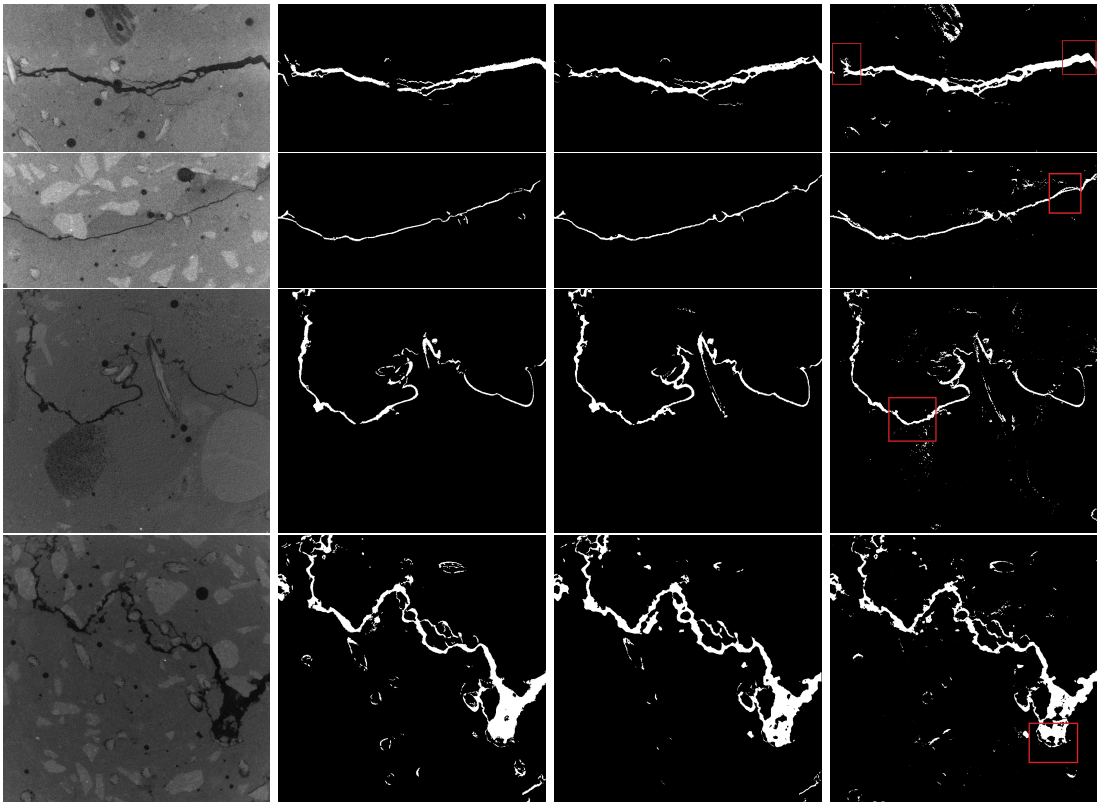


Figure 7.8: Sliceview comparison from *Figure 7* of [100]. From left to right: input image, segmentation results with Hessian-based percolation, 3d U-net, and the Riesz network+median filter. Red boxes mark improvements in connectivity, recall, and localization of cracks in the areas where crack orientation changes with respect to the baselines.

that the Riesz network is partially able to handle thin cracks until they turn to subpixel or very low contrast pattern (Figure 7.11 and Figure 7.14). Figure 7.12 reflects the lower precision of the Riesz network as noticed in computational studies, i.e. a lot of noise near the crack is misclassified as crack. This could be partially resolved by simple post-processing with a median filter. Figure 7.13 shows the ability to segment cracks with many orientation changes that look very different than the training set (Figure 7.3). Here, together with Figure 7.14, we can observe weakness of the Riesz network to missclassifying the borders of fibers as cracks.

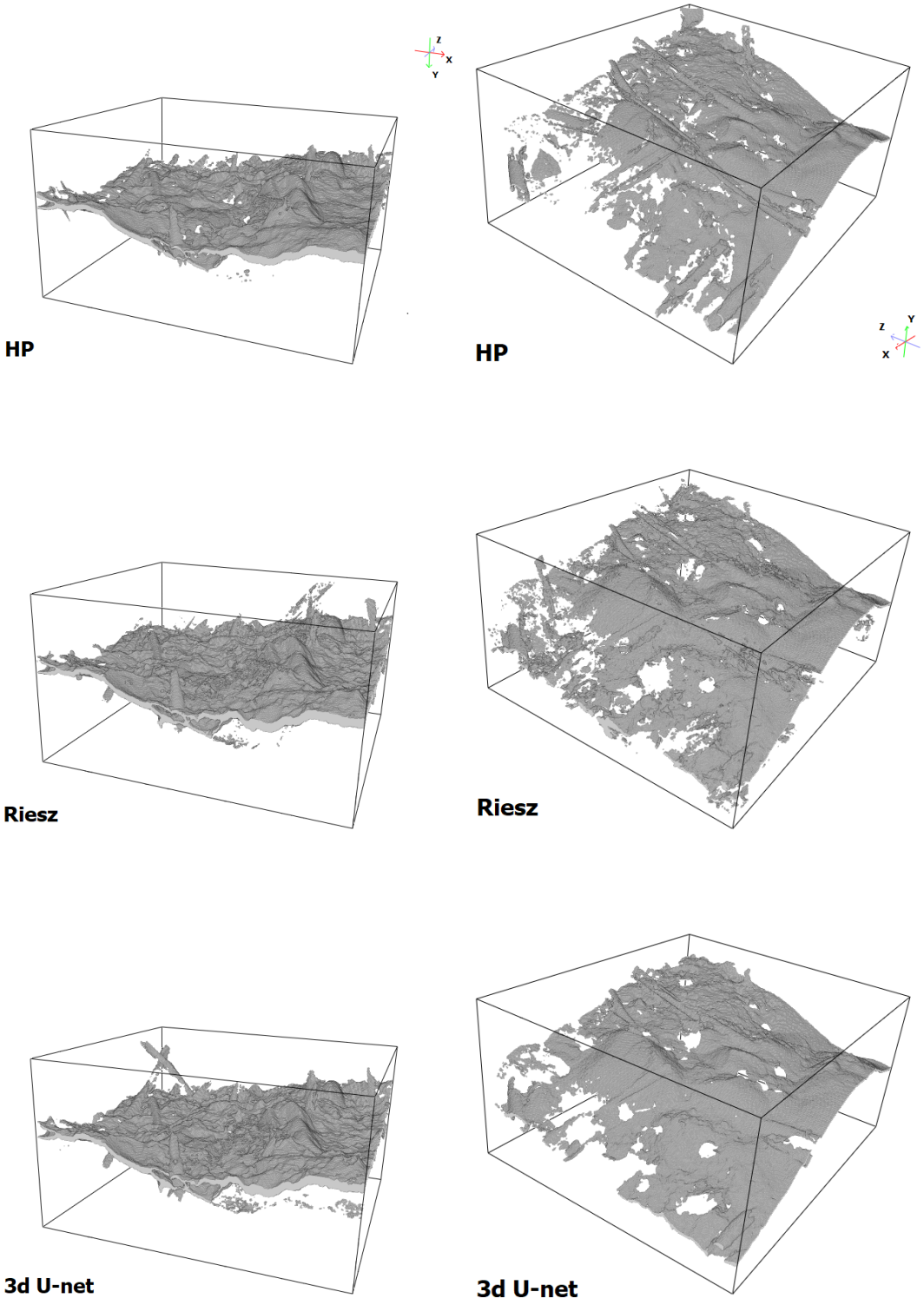


Figure 7.9: Renderings of the cracks segmented by (from top to bottom): Hessian-based percolation, 3d U-net, and the Riesz network on NC12 (left) and HPC11 (right) samples (Table 7.4).

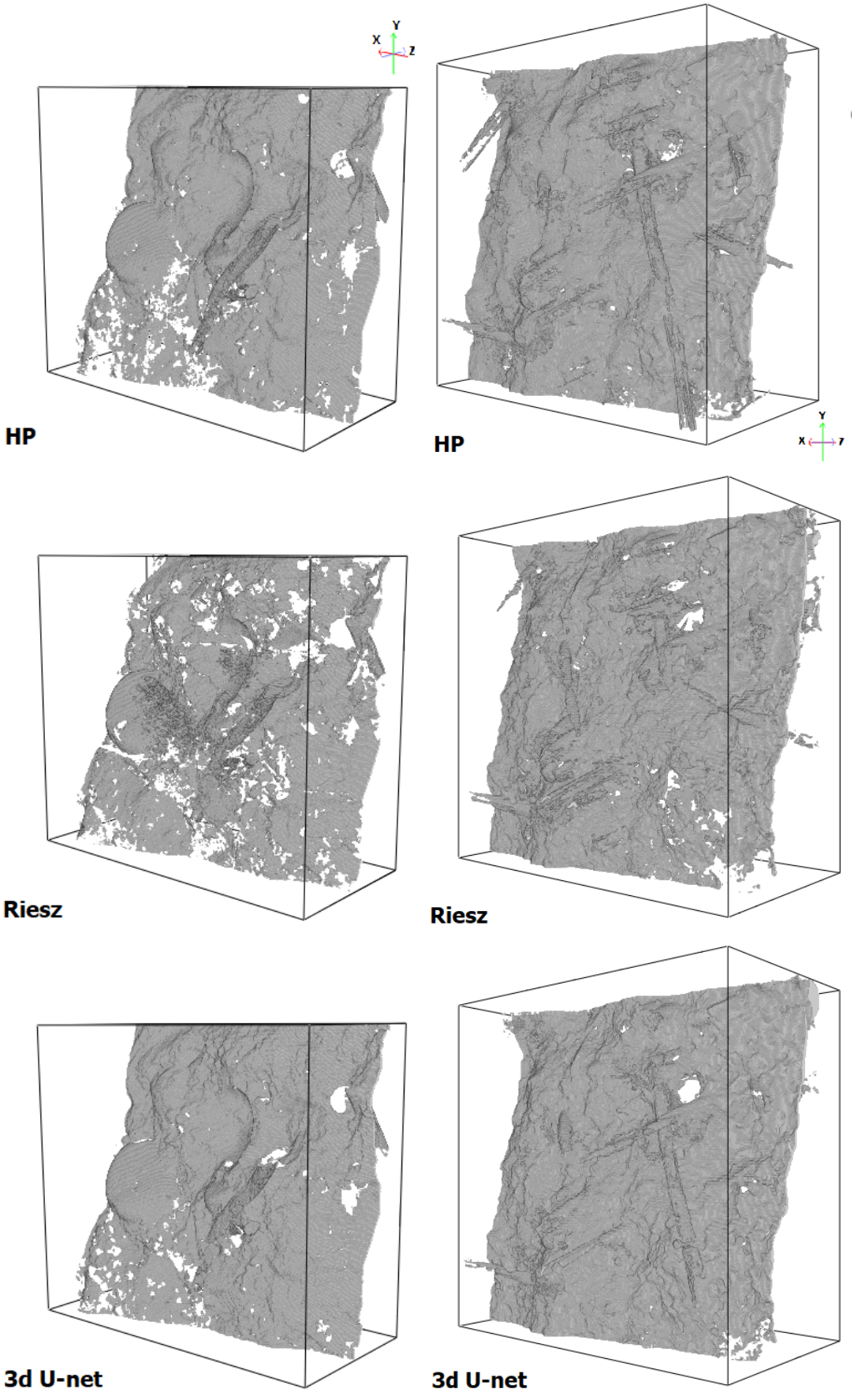


Figure 7.10: Renderings of the cracks segmented by (from top to bottom): Hessian-based percolation, 3d U-net, and the Riesz network on NC2 (left) and HPC1 (right) samples (Table 7.4).

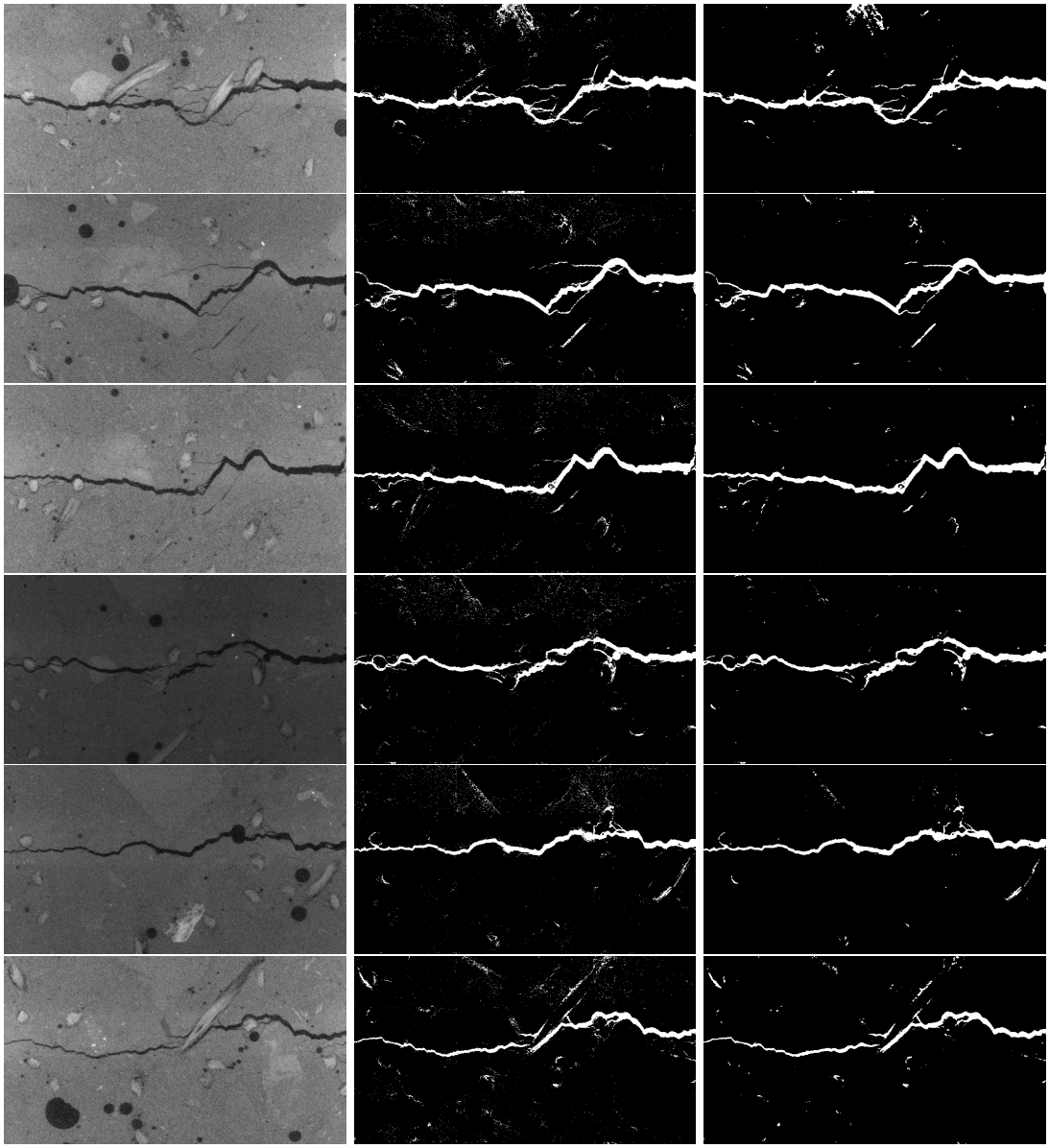


Figure 7.11: Sample NC12: input image (left), segmentation results from the Riesz network before (center) and after (right) post-processing with the median filter with mask $3 \times 3 \times 3$.

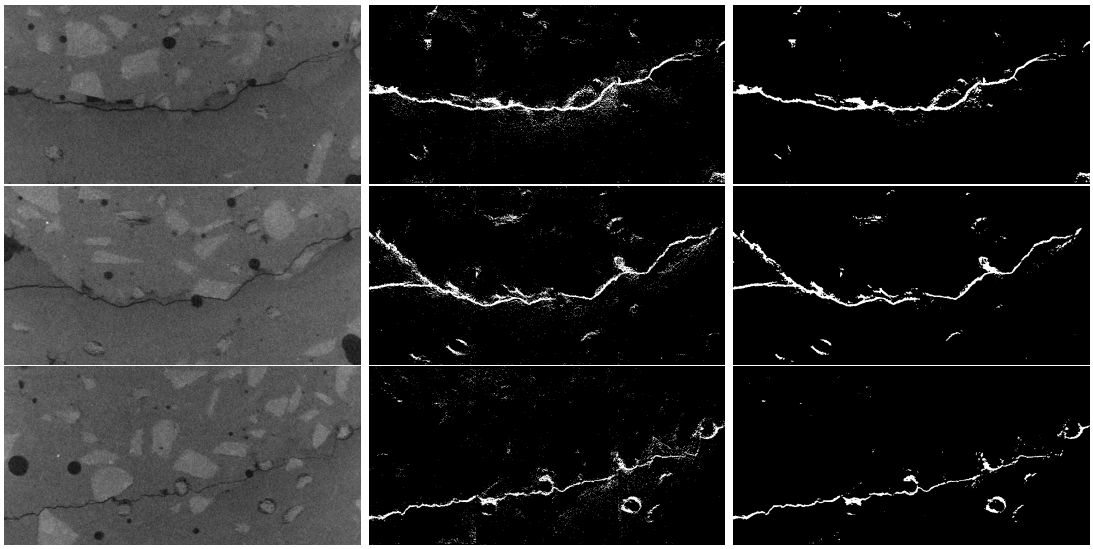


Figure 7.12: Sample HPC11: input image (left), segmentation results from the Riesz network before (center) and after (right) post-processing with the median filter with mask $3 \times 3 \times 3$.

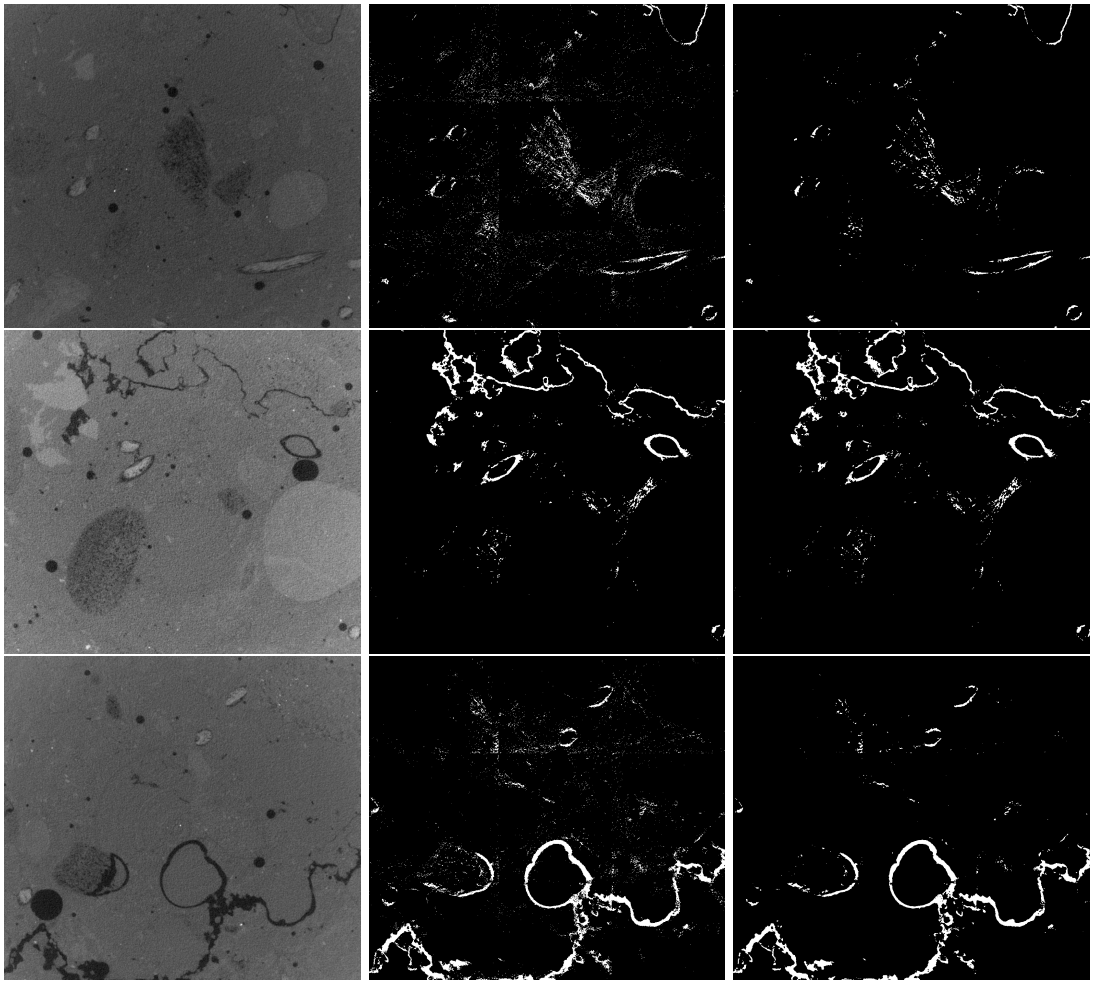


Figure 7.13: Sample NC2:input image (left), segmentation results from the Riesz network before (center) and after (right) post-processing with the median filter with mask $3 \times 3 \times 3$.

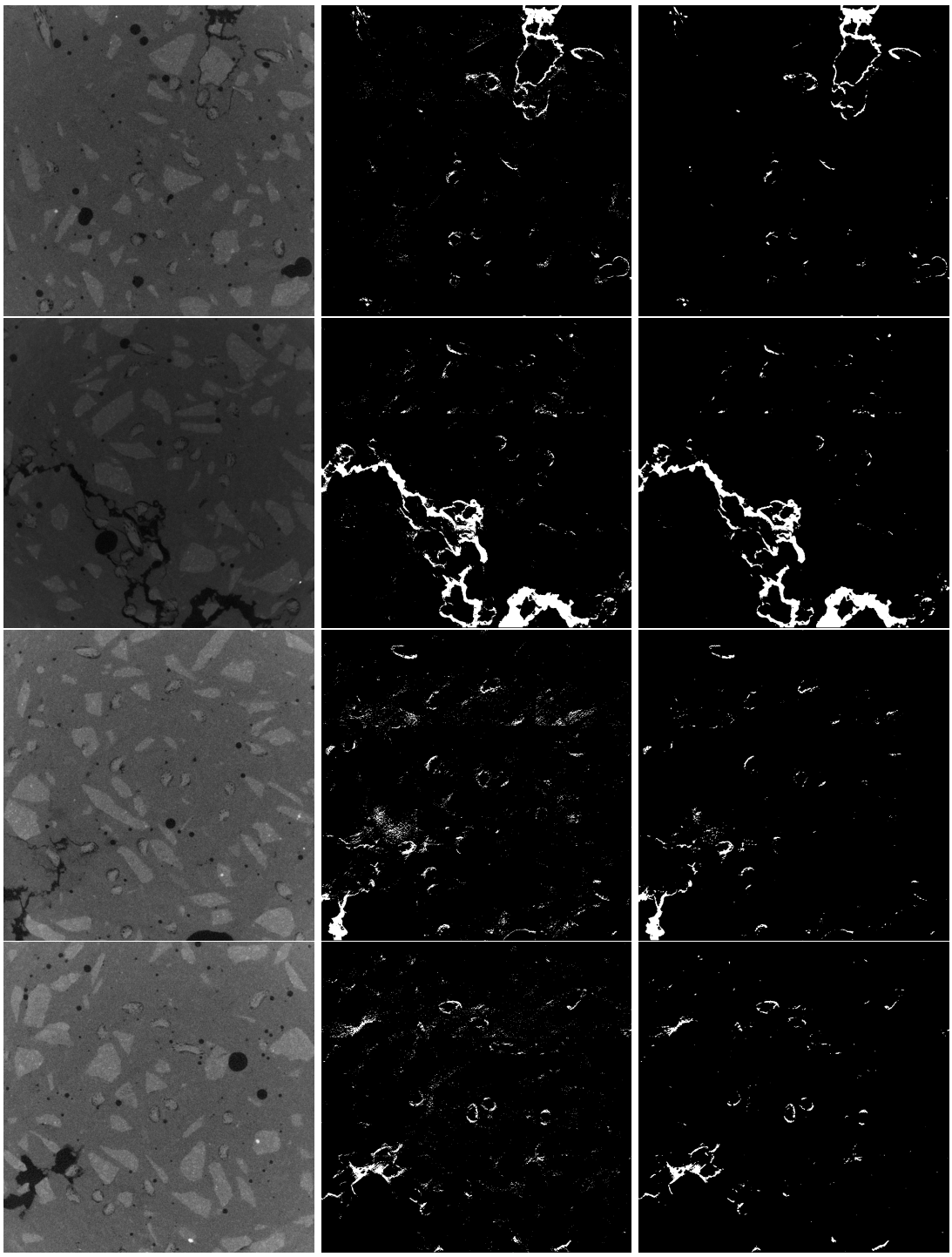


Figure 7.14: Sample HPC1: input image (left), segmentation results from the Riesz network before (center) and after (right) post-processing with the median filter with mask $3 \times 3 \times 3$.

7.4.2 Crack segmentation in concrete from pull-out tests

This dataset was originally used in [126], as a follow-up study from [100]. Cracks were created in a concrete sample with embedded rebar by pulling out the rebar. Hence, these cracks differ from the ones from [100]. These two works together give an idea of how reliable crack segmentation methods are in segmenting various types of cracks.

Concrete samples

Three concrete samples with embedded rebar were prepared. On two of these specimens, pull-out tests were applied. The first specimen (Image 1) was loaded until failure, i.e. until the bar was completely pulled out of the concrete. The second specimen shows cracking but the bar stays in place. For this specimen, two scans were taken. Image 2 refers to the central part of the specimen, while Image 3 is a scan of the bottom part. Interestingly, the cracks in the two scans exhibit very different characteristics regarding shape and width. Finally, the third specimen (Image 4) has not been subjected to a pull-out test at all but contains a crack which occurred during the concreting process.

	Size [voxels]	Cropped size [voxels]	Voxel edge length
Image 1 (I1)	$1\,965 \times 1\,965 \times 1\,716$	$1\,000 \times 900 \times 500$	26.8µm
Image 2 (I2)	$1\,979 \times 1\,979 \times 1\,547$	$1\,000 \times 1000 \times 700$	25.8µm
Image 3 (I3)	$1\,987 \times 1\,987 \times 1\,577$	$1\,000 \times 1\,000 \times 800$	25.8µm
Image 4 (I4)	$959 \times 959 \times 1\,849$	$368 \times 268 \times 1\,500$	80.0µm

Table 7.5: The four reconstructed µCT images used in the following.

CT imaging

The concrete specimens were scanned at the Fraunhofer ITWM, Kaiserslautern, Germany. The µCT device consists of a Feinfocus FXE 225.51 X-ray tube with maximum acceleration voltage 180.3 kV and maximum power of 12.7 W. The detector is a Perkin Elmer flat-bed detector XRD 1621 with $2\,048 \times 2\,048$ pixels. For Image 1 and Image 4, a tube voltage of 180 kV was used. The integration time was set to 1 second. 1 200 projections were used for acquiring tomographic reconstructions. For Image 2 and Image 3, a tube voltage of 160 kV, an integration time of 0.5 seconds and 400 projections were used. The CT setup is shown in Figure 7.15. Details on the reconstructed images are given in Table 7.5. The segmentation methods are applied to cropped images that contain most of the crack structure.

Results

Figure 7.16 gives a comparison of sliceview shown in *Figure 4* of [126], while Figure 7.17 and Figure 7.18 compare renderings from the three methods after post-processing. This includes the extraction of the largest connected components followed by the median filter on $3 \times 3 \times 3$. Additional sliceviews of the Riesz network are given in Figures 7.19, 7.20, 7.21, and 7.22.

Based on renderings in Figure 7.17 and Figure 7.18, and sliceview comparison Figure 7.16, we can observe that the Riesz network works significantly better than the baselines on one out of four samples while having high recall on all four samples. We see that the Riesz network segments more crack surfaces for sample I2 and more connected crack surface for sample I4. For sample I1 and I3 the Riesz network segments what seems to be too many noise voxels misclassified as a crack.

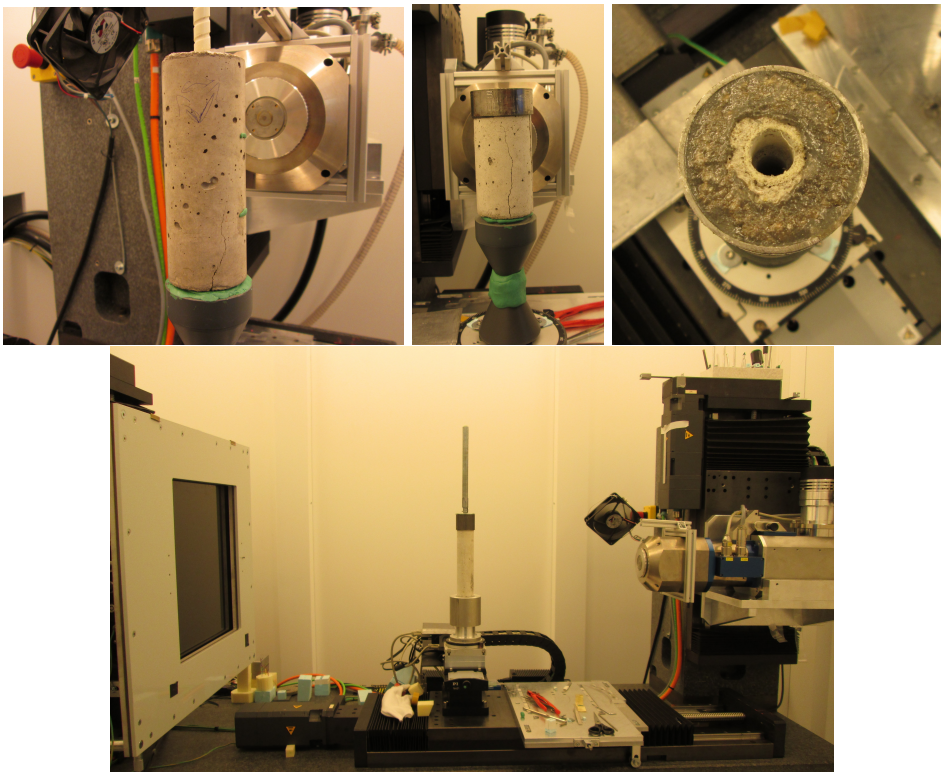


Figure 7.15: From left to right: side view of the specimen for Image 2 and 3, side and top view of Image 1, and CT imaging setup when scanning Image 4.

For sample I1 the Riesz network segments part of the rebar as a crack due to the cropping of the input image into non-overlapping subwindows and assembling them back after the inference (Section 7.1.1). Due to subwindow size of 400^3 rebar is not evenly cut, i.e. smaller part of the rebar is cut as the elongated structure at the edge which reminds of a crack in shape and hence gets misclassified. This can be seen in Figure 7.19 where the left boundary of segmented rebar appears straight up to the edge effects of the method as marked with red boxes. Figure 7.19 and Figure 7.21 also show that sometimes grain edges in samples I1 and I3 get classified as cracks because they seem darker than the surrounding. Figure 7.22 shows that in the case of sample I4 oversegmentation and poor localization can happen to the Riesz network. In contrast, Figure 7.21 and sample I3 give examples of good localization of crack structures (red boxes) and the ability to detect very thin cracks albeit disconnectedly (blue box). Figure 7.20 (red boxes) of sample I2 shows the impressive performance of the Riesz network in segmenting very thin cracks. From Figure 7.16 (red boxes) it appears that the Riesz network gives better segmentation results than the competing methods on sample I2: cracks appear more connected and compact, even the air ring between rebar and concrete is fully classified as a crack since it is a 2d surface with thickness in 3d space.

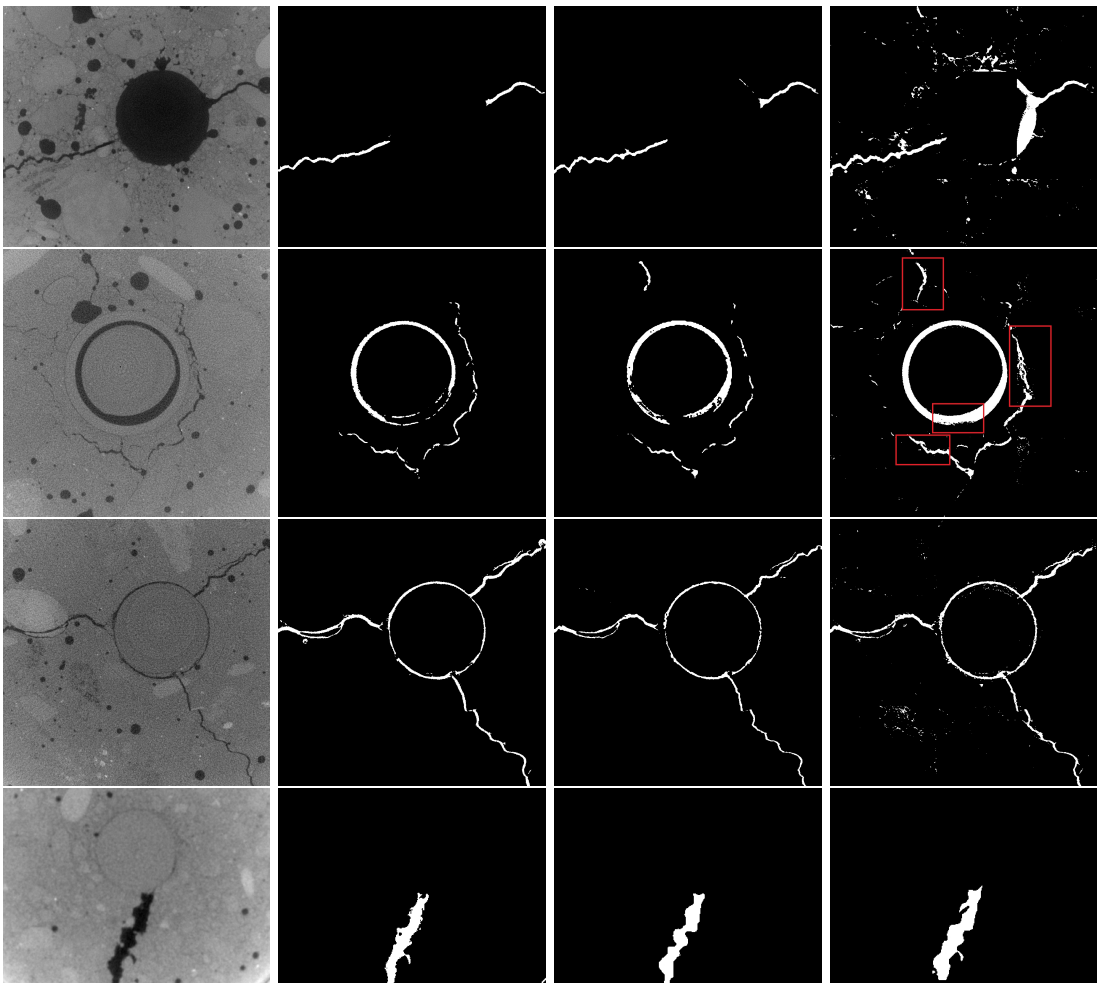


Figure 7.16: Sliceview comparison from *Figure 4* of [126]. From left to right: input image, segmentation results with Hessian-based percolation, 3d U-net, and Riesz network+median filter. Red boxes mark the improvements in the segmentation results of the Riesz networks compared to the remaining two methods.

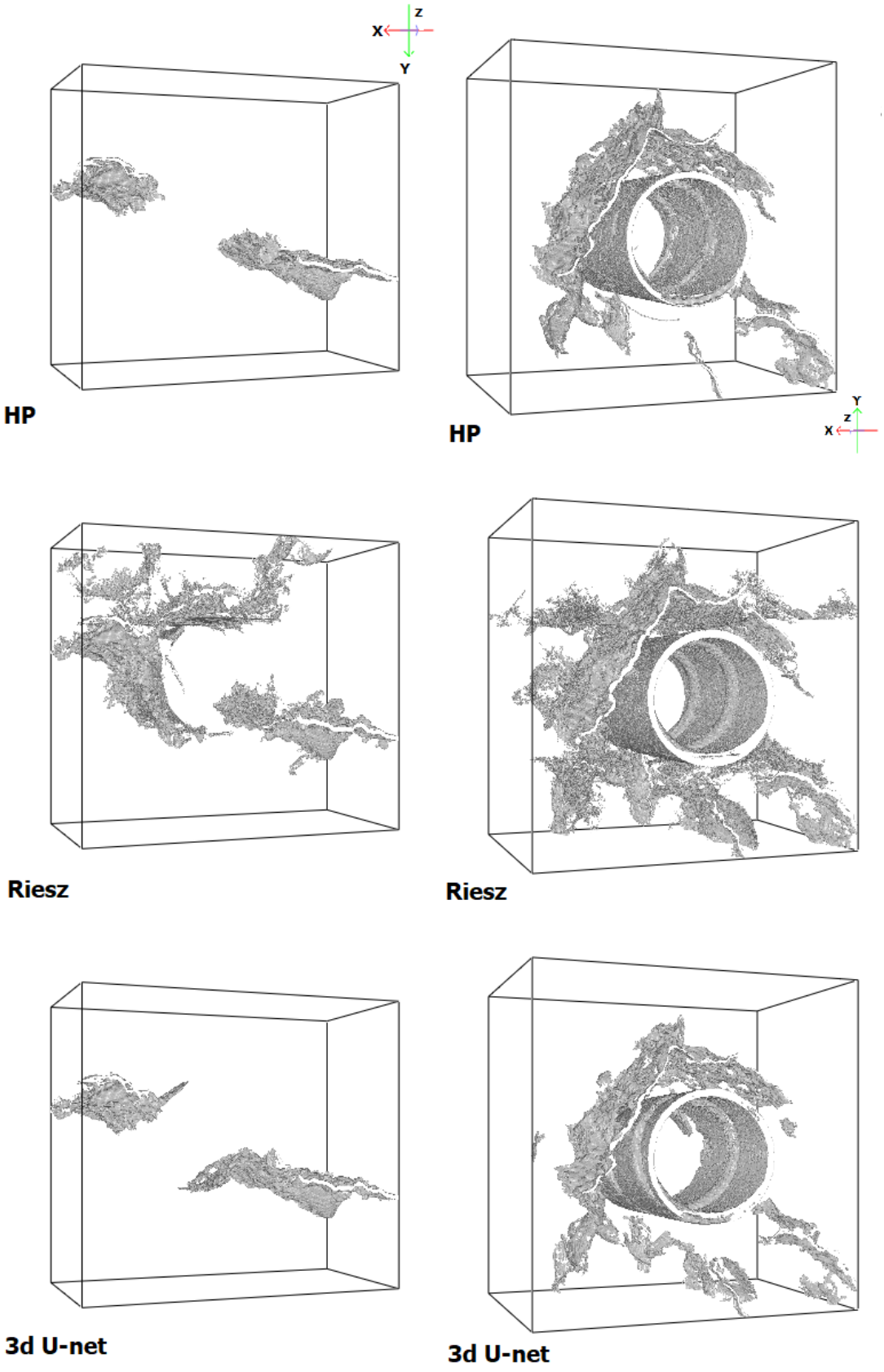


Figure 7.17: Renderings of the cracks segmented by (from top to bottom): Hessian-based percolation, 3d U-net, and the Riesz network on I1 (left) and I2 (right) samples (Table 7.5).

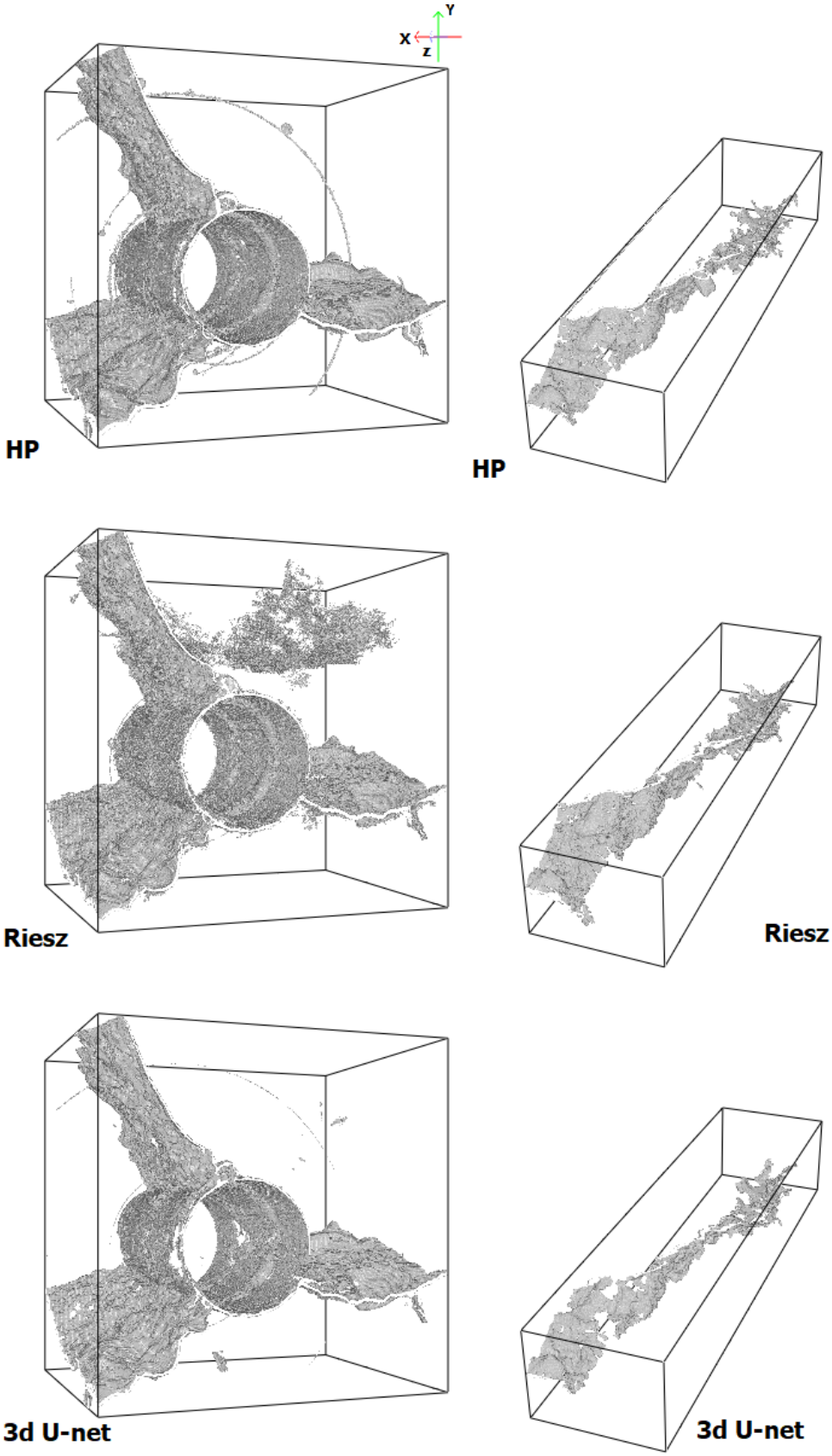


Figure 7.18: Renderings of the cracks segmented by (from top to bottom): Hessian-based percolation, 3d U-net, and the Riesz network on I3 (left) and I4 (right) samples (Table 7.5).

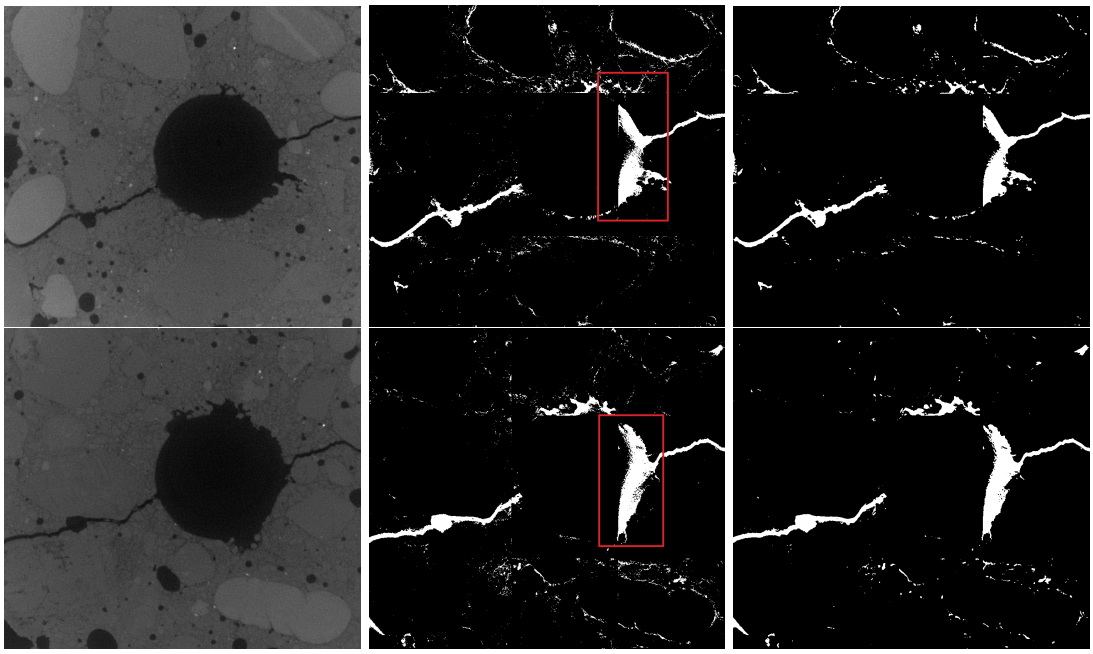


Figure 7.19: Sample I1: input image (left), segmentation results from the Riesz network before (center) and after (right) post-processing with the median filter with mask $3 \times 3 \times 3$. Red boxes show the edge effects which occur due to the cropping of the input image into non-overlapping 400^3 subwindows.

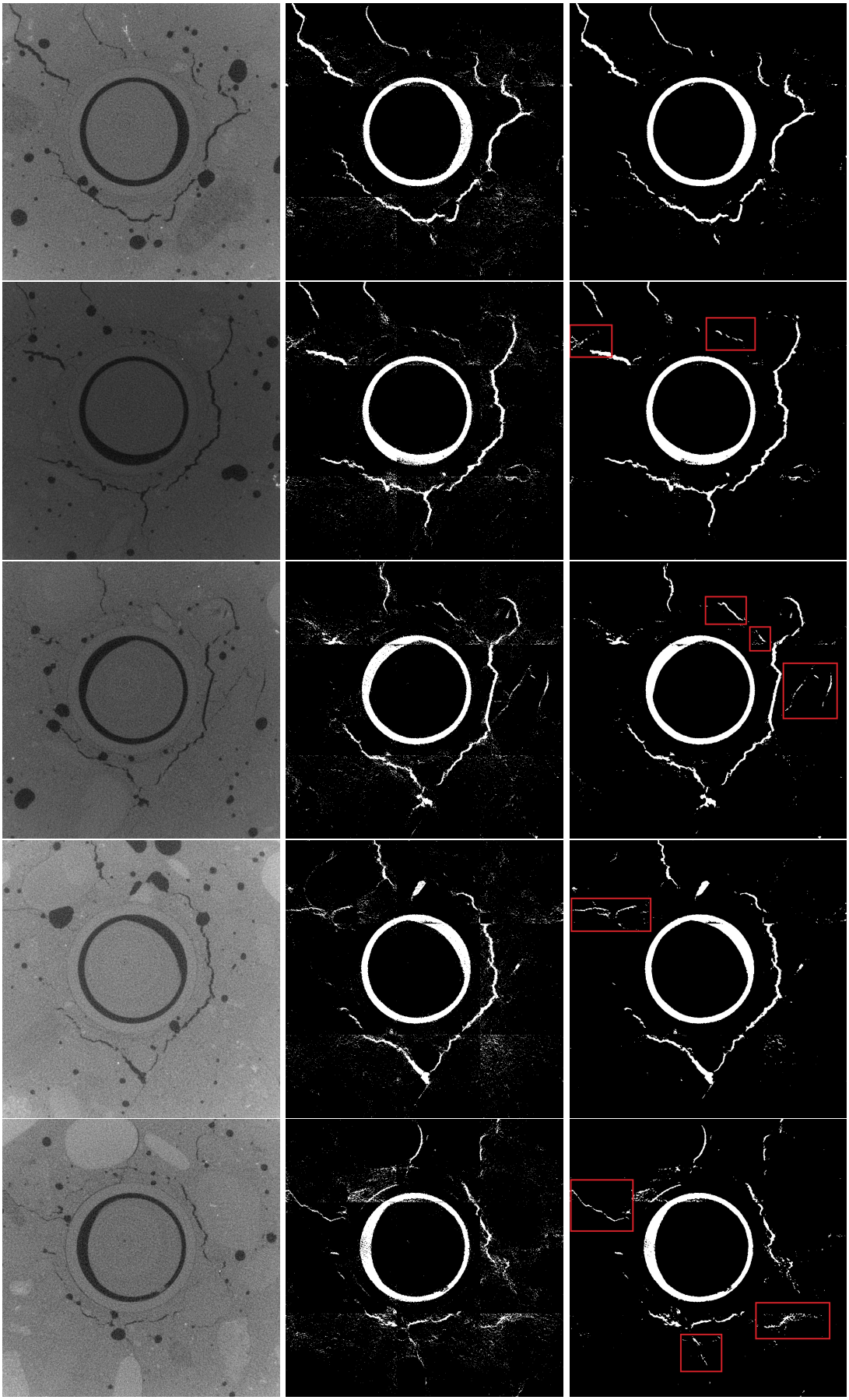


Figure 7.20: Sample I2: input image (left), segmentation results from the Riesz network before (center) and after (right) post-processing with the median filter with mask $3 \times 3 \times 3$. Red boxes mark thin cracks that were correctly segmented by the Riesz network.

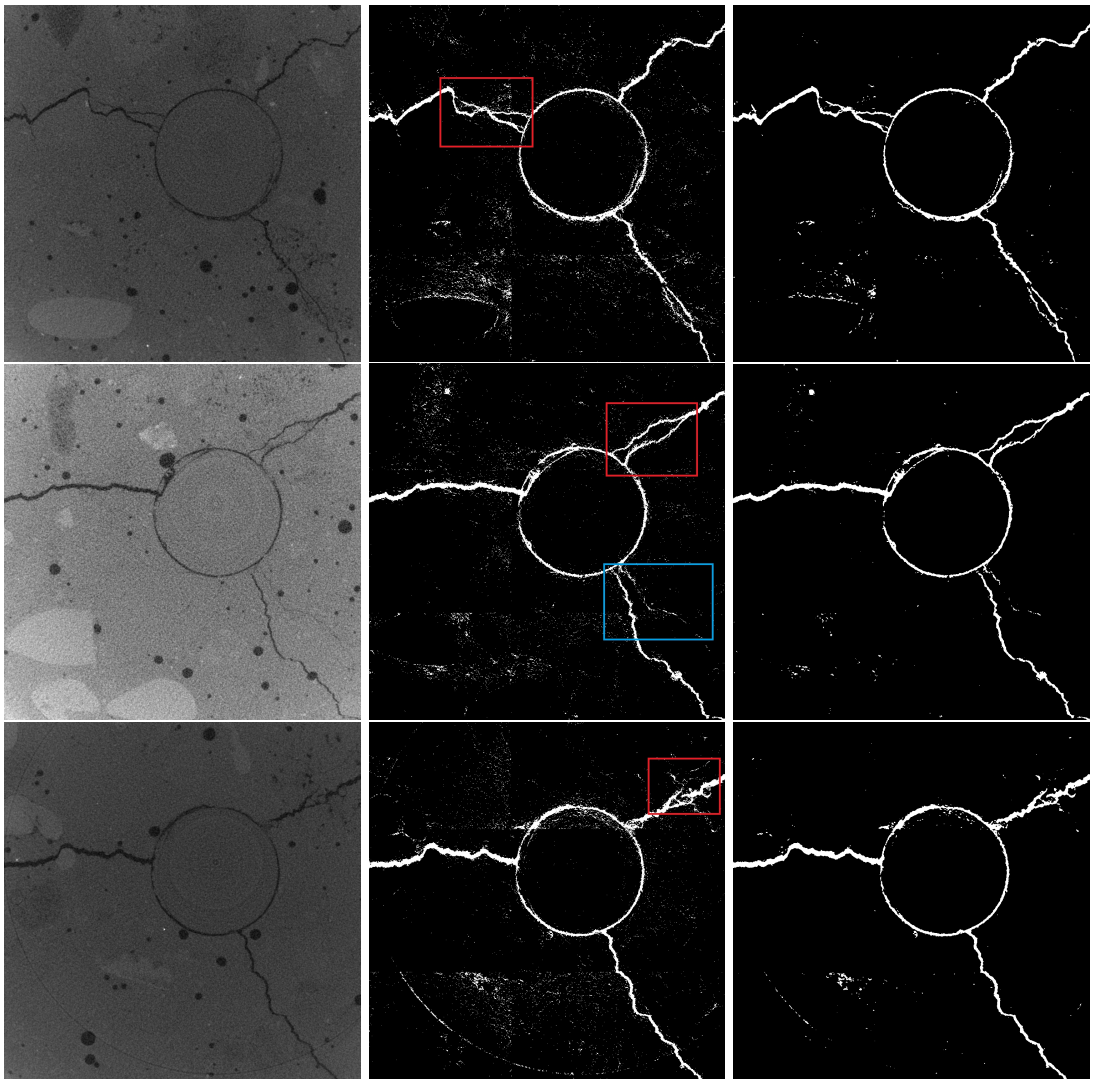


Figure 7.21: Sample I3: input image (left), segmentation results from the Riesz network before (center) and after (right) post-processing with the median filter with mask $3 \times 3 \times 3$. Red boxes shows examples of good localization of crack structures. Blue box shows an example where the Riesz network was able to detect very thin cracks albeit disconnectedly.

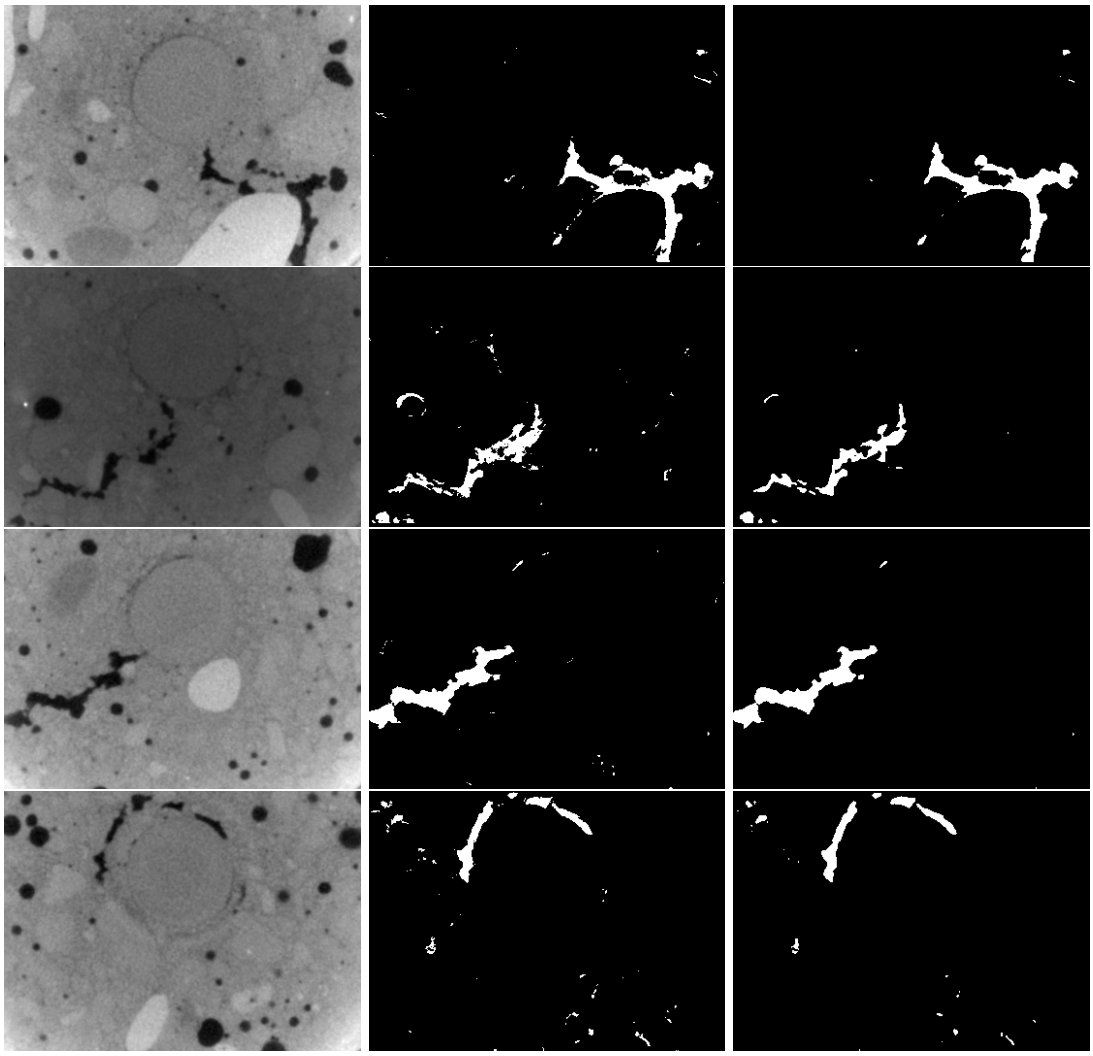


Figure 7.22: Sample I4: input image (left), segmentation results from the Riesz network before (center) and after (right) post-processing with the median filter with mask $3 \times 3 \times 3$.

7.4.3 Fiber reinforced concrete sample

This experiment on CT images of real cracks involves one sample with a very different material composition and a different method of crack initiation, see [124] for details. Another important aspect is that a different CT device (other than the one from Fraunhofer ITWM) was used for imaging. This experiment is the final experiment on 3d CT images to assess the quality of the Riesz network.

Concrete samples

Fiber reinforced concrete sample Bending tensile strength tests were carried out on hardened high performance concrete with polypropylene fibers (PP) MasterFiber 235 SPA [130]. The tests were conducted on beam specimens of dimensions $100 \times 100 \times 400$ mm^3 in accordance with EN 14651 [131]. A sample before and after testing is shown in Figure 7.23.

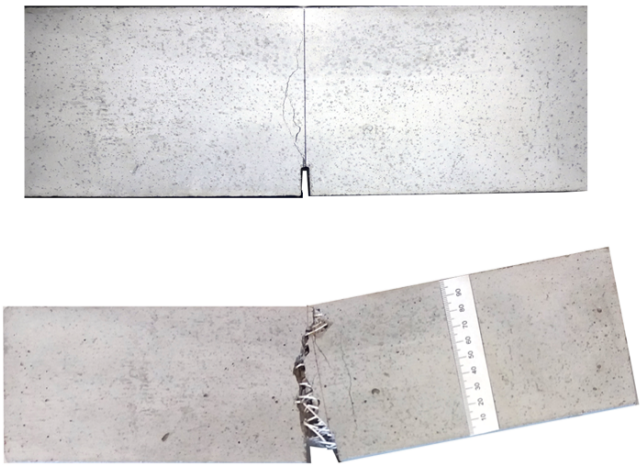


Figure 7.23: Fiber reinforced concrete before and after tensile test.

CT imaging

The fiber reinforced concrete sample was scanned at the Fraunhofer EZRT, Fürth, Germany. The dimension of the sample allowed for the application of a 300 kV microfocus X-ray tube from Hamamatsu Photonics. The sample was scanned at 290 kV. The detector has a sensitive area of approx. 42 x 42 cm^2 and a pixel size of 139 µm. The sample was positioned on a rotary axis for measurement and imaged onto the detector with a magnification factor of 2.3. This results in an effective voxel size of 60.4 µm in the object. For the 3d reconstruction, 3 600 angles were acquired along a complete rotation of the sample with an exposure time of 0.8 s each. The total measurement time was 48 min. Details on the sample are given in Table 7.6.

Size [voxels]	Cropped size [voxels]	Voxel edge length
$2\,145 \times 2\,001 \times 2\,427$	$1\,600 \times 1\,600 \times 1600$	60.4 μm

Table 7.6: Information on the µCT image data for fiber reinforced concrete sample (RPTU-EZRT).

Results

In this sample crack thickness varies from a few voxels to a couple of hundreds of voxels. Hence, Hessian-based percolation and 3d U-net require additional adjustments [124]. For example, the Frangi filter uses 38σ's (or 38 scales) in the range $[0, 75]$ in the selection of candidate voxels[3]. 3d U-net model uses downscaling factors $\{0.0625, 0.09375, 0.125\}$ for big cracks and $\{0.125, 0.25, 0.375, 0.5\}$ for smaller cracks. Note that these smaller downscaling factors are necessary because large ones tend to remove very thin cracks. These adjustments have to be done manually with prior analysis of crack properties, i.e. mostly crack width. For the Riesz network we test two variants. The first variant downscales the original image with only a single downscaling factor 0.5 and uses the same Riesz network with two different window sizes: 100^3 and 400^3. The smaller window size is used to segment thinner cracks which are harder to segment on larger windows because of thicker cracks and the partial volume effect. The second variant uses the Riesz network with downscaling factor 0.25 only on window size 400^3.

Figure 7.24 gives a comparison of sliceview shown in *Figure 5* of [124], while Figure 7.25 compares renderings from the three methods after post-processing. This includes the extraction of the largest connected components followed by the median filter on $3 \times 3 \times 3$. Additional sliceview on the Riesz network are given in Figures 7.26, 7.27, and 7.28.

Renderings in Figure 7.25 reveal similar crack structures for all three methods. Hessian-based percolation seems to be the most sensitive to misclassifying the dark fibers as cracks, while the Riesz network and 3d U-net seem to be equally resistant to this effect.

Figure 7.24 and Figure 7.27 show the problem with the Riesz network in the middle of the cracks due to the edge effect from non-overlapping window cropping i.e. crack ends up being on the border of cropping and hence this effect occurs. Figure 7.28 shows results from the second variant of the Riesz network and proves that the Riesz network is able to segment the largest crack opening without edge effects when the crack is fitting to the window size. This further stresses the drawback of the need for window adjustment from Section 7.1.1. Furthermore, the Riesz network seems to recognize part of the grains in the crack opening and not classify them as crack voxels in Figure 7.24. Other methods tend to remove or oversmooth them. This is further stressed in Figure 7.26. Segmentation of very thin cracks which are parallel to the main crack opening is only partial and disconnected as marked by red boxes in Figure 7.27. Hessian-based percolation seems to perform better on thin cracks (Figure 7.24). The reason for this can be two-fold. Hessian-based percolation is the only method that uses full-sized CT image of this sample where no downscaling interpolation error exists. Secondly, Hessian-based percolation is the only method that has implicitly included a post-processing step as a part of the method. Here, the region growing algorithm could potentially reconstruct a crack even if it was partially detected such as in Figure 7.27 (red boxes) for the Riesz network. However, the cost of the region growing algorithm is that parts of the dark fiber get misclasified as cracks.

[3]For the comparison HP used 14σ's for range $[0.5, 7.5]$ in Section 7.4.1 and 28σ's for range $[0.5, 15]$ in Section 7.4.2. For more details on parameters see Appendix C.

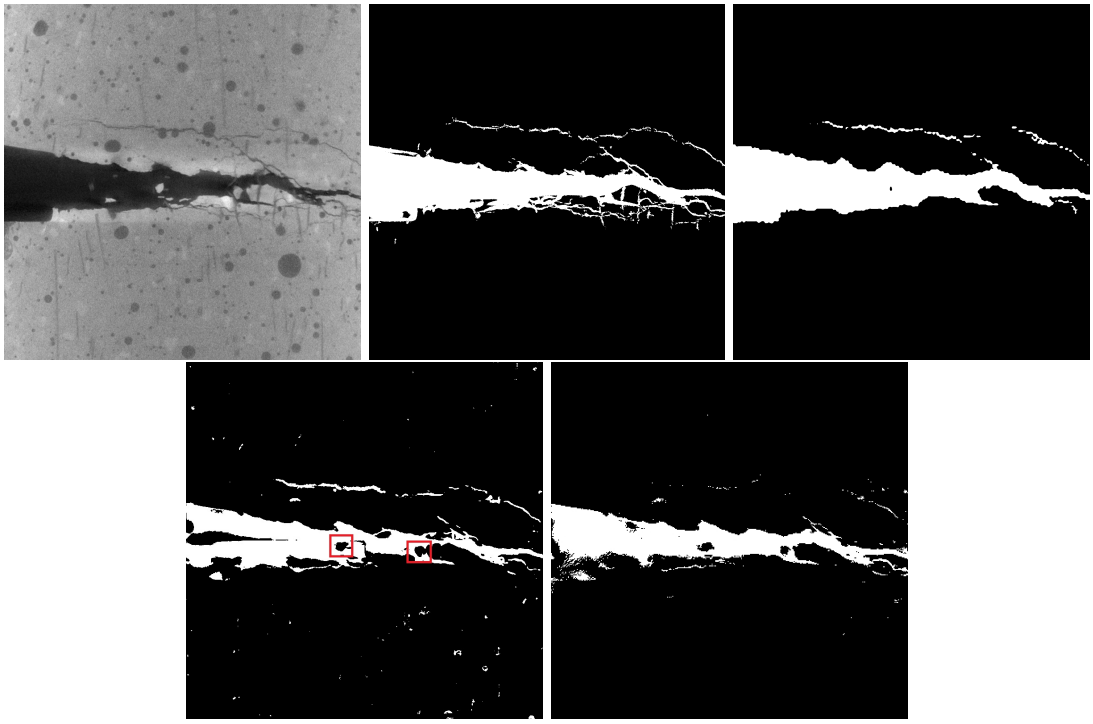


Figure 7.24: Sliceview comparison from *Figure 5* of [124]. From left to right: input image, segmentation results with Hessian-based percolation, 3d U-net (all in the first row), the Riesz network+median filter on input image downscaled with factor 0.5, and the Riesz network on input image downscaled with factor 0.25 (all in second row). The Riesz network recognizes part of the grains in the crack opening and does not classify them as crack voxels (red boxes).

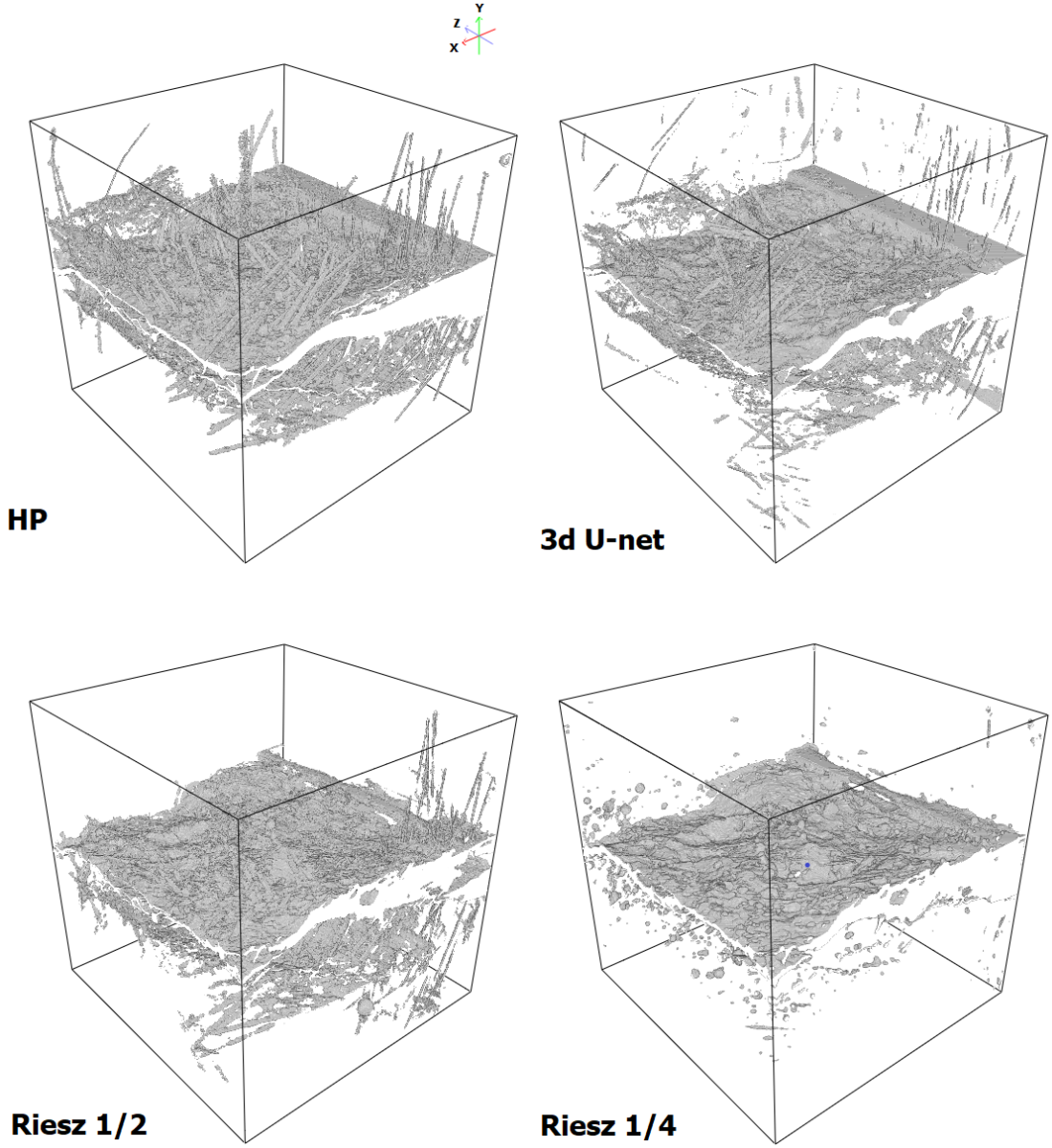


Figure 7.25: Renderings of the cracks segmented by (from top to bottom): Hessian-based percolation, 3d U-net, and the two versions of the Riesz network on EZRT-RPU sample (Table 7.6).

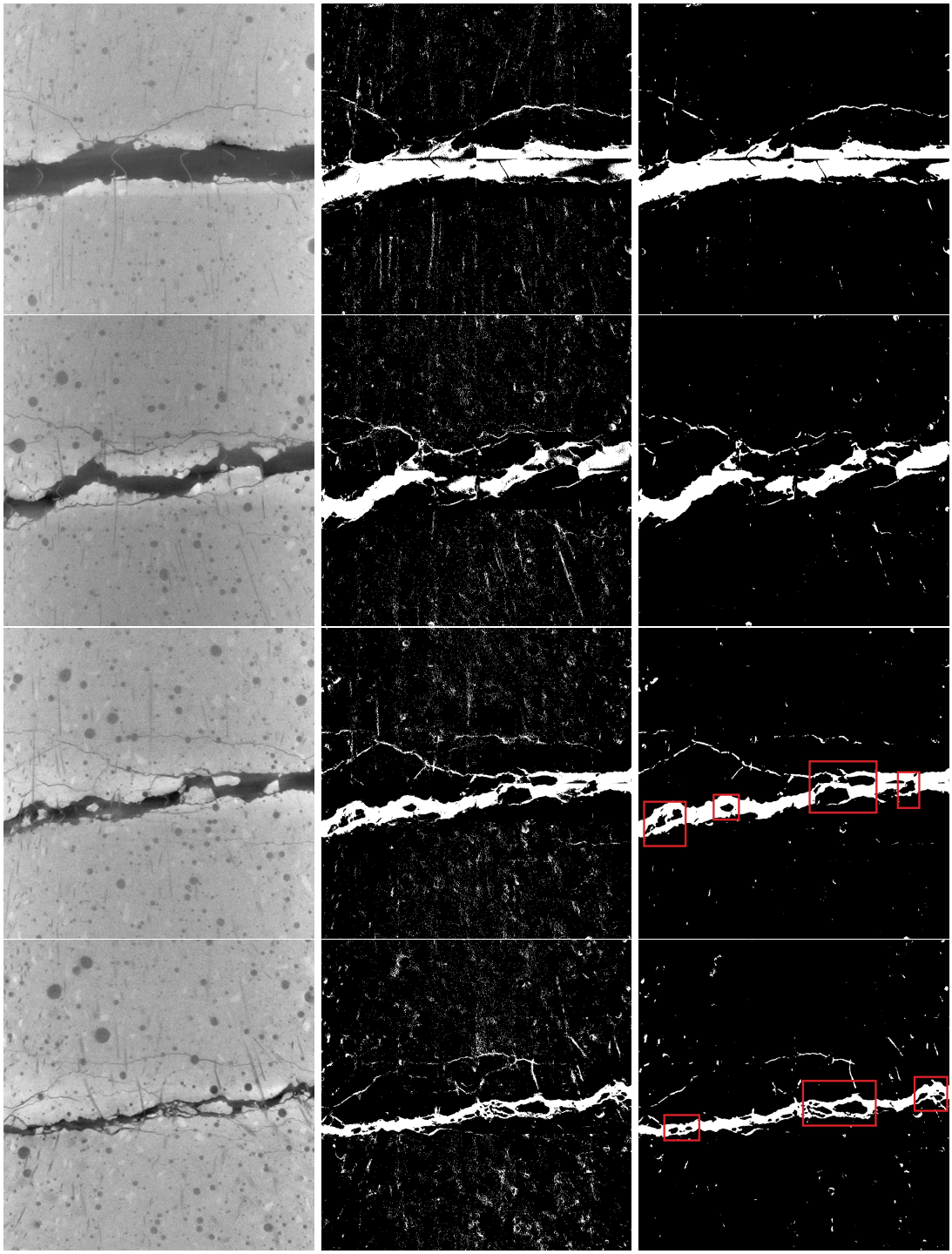


Figure 7.26: Sample EZRT-RPTU: input image (left), segmentation results from the Riesz network before (center) and after (right) post-processing with the median filter with mask $3 \times 3 \times 3$ applied to input image downscaled with factor 0.5. Slices in x-direction. Red boxes mark examples of good localization of crack structures in the segmentation results of the Riesz network.

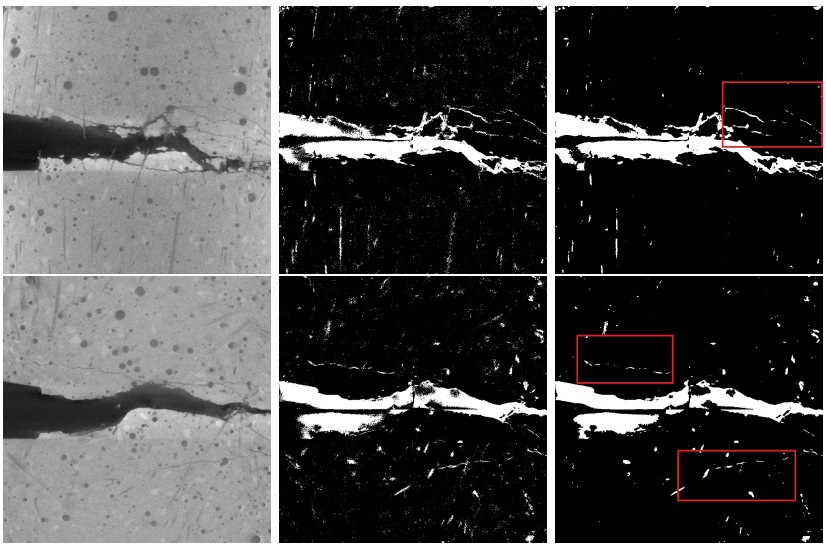


Figure 7.27: Sample EZRT-RPTU: input image (left), segmentation results from the Riesz network before (center) and after (right) post-processing with the median filter with mask $3 \times 3 \times 3$ applied to input image downscaled with factor 0.5. Slices in y-direction. Red boxes mark thin cracks that were partially segmented by the Riesz network.

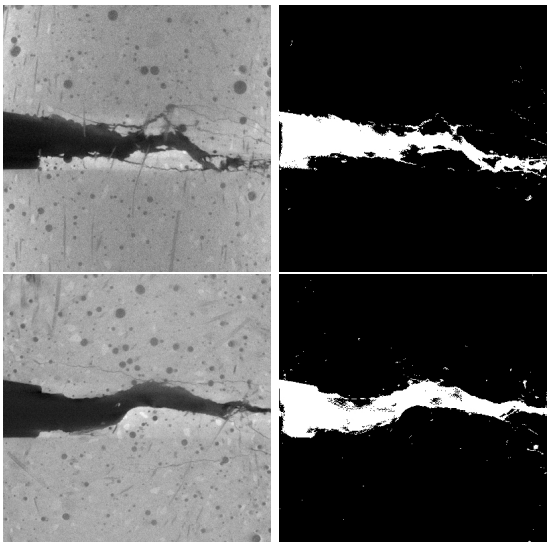


Figure 7.28: Sample EZRT-RPTU: input image (left), segmentation results from the Riesz network before (center) and after (right) post-processing with the median filter with mask $3 \times 3 \times 3$ applied to input image downscaled with factor 0.25. Slices in y-direction from Figure 7.27.

7.5 Conclusion

In this chapter, the Riesz network has been generalized to 3d and applied to crack segmentation tasks in 3d CT images. When moving from 2d to 3d, the Riesz network is adjusted to work only on 400^3 subwindows because of the memory issues. Afterwards, the 3d Riesz network has been extensively tested and compared to the relevant methods on two simulated datasets with available ground truth and on eight 3d CT images with real cracks. Conceptually, the Riesz network positions itself between classical methods (e.g. Hessian-based percolation) and deep learning methods (e.g. 3d U-net) in the following sense: it uses a determinstic (non-trainable, parameter-free) feature extractor which is combined with 1d convolutions and non-linearities creating a deep layered structure. As a consequence, the number of parameters is significantly reduced from millions (e.g. 3d U-net) to thousands. The most interesting property that the Riesz network possesses theoretically and others do not is scale invariance. In practice, the Riesz network requires less user interference compared to the compared methods, see Section 7.4.3.

A simulation study shows that the Riesz network has a high recall, i.e. good coverage of the crack structure. However, precision is significantly lower compared to 3d U-net. Precision is further improved with a simple post-processing technique: extraction of the largest connected components. This implies that the Riesz network would benefit the most from more sophisticated post-processing methods which remove noise. A simulation study shows the weakness of the Riesz network to segment thin cracks correctly. This has been a known weakness of the competing methods as well.

Application to real data confirms the hypothesis on high recall with lower precision. Here, better localization than the segmentation results from the competing methods can be observed in a few samples. The high recall is proven to be particularly useful on sample I2 from Section 7.4.2, where way more crack structures than for competing methods have been segmented. On the remaining datasets, the verdict is similar as in the simulation study: most of the crack structures are recognized but with the cost of more noisy voxels being misclassified than for the baselines. Hence, the same conclusion on the post-processing strategy seems to be valid here, too.

From a theoretical perspective, the Riesz network is both scale and translation invariant. While simulation studies on the Riesz network systematically cover the problem of scale invariance, the problem of rotation invariance has not been explicitly studied. Rotation invariance needs to be ensured as well for the crack segmentation methods. The questions for future research remain: should the Riesz network learn rotation invariance through rotation diversity of the training set? Should the Riesz network be modified to be theoretically both scale and rotation equivariant? See e.g. [132, 133] as an example of the rotation equivariant architectures.

As a final conclusion of the chapter, one can claim that the Riesz network is useful for crack segmentation task in 3d CT images. An important feature is that scale invariance can simply be achieved up to the window size. Only crack widths of 3 and 5 are used for training, while inference can be achieved for arbitrary crack widths up to the window size. This reduces the effort for data collection and very few scales need to be present in the training set for the method to achieve scale generalization. However, due to the low number of parameters in 3d, post-processing steps are necessary to get noise-free results.

Chapter 8

The Riesz scattering representation

8.1 Introduction

Deep learning offers very powerful tools for various tasks in computer vision and image processing, e.g. segmentation or classification. However, one of its main drawbacks is that it is data-hungry, i.e. it requires a large amount of annotated data with enough inter-class variability for the training part to be successful. For that purpose when very few data is available, manual feature extraction combined with a classifier (e.g. PCA or SVM) becomes more practical and useful.

Hierarchical feature representations are an efficient way to extract universal features that can be used for a wide range of tasks. The idea originates from so-called scattering networks [134], where a complex wavelet is used to extract the local amplitude. Rotated and rescaled versions of this wavelet capture diverse features and serve as one layer of the representation. The hierarchical representation then arises from applying the same set of wavelets on the amplitudes from the previous layer. In this way, higher order nonlinear features can be extracted and utilized.

Generally, it is desirable that any feature representation satisfies basic properties of human vision and is able to handle acquisition variability of 2d digital cameras (Section 4.2). However, achieving that is not trivial at all. Here, we impose translation invariance as a basic condition that the representation should satisfy and investigate how to achieve robustness to scale changes or even scale invariance.

Mathematically, scale space is a continuous semi-group, so scales are unlimited. However, it is important to note that images are discretized and bounded representations of the real world and for that reason scale in the images is bounded from two sides [116]. From below it is bounded by image resolution or pixel size (inner scale). Details smaller than the pixel size cannot be recorded. From above, image scale is bounded by the image size (outer scale). What happens outside of the image cannot be observed. A typical approach for obtaining scale invariance is to sample the scale dimension and to extract features at each sampled scale, e.g. in the SIFT descriptor [96]. However, different scales in the image are not independent and many of them are not relevant for the given task such that a uniform sampling of the scale dimension results in highly redundant representations. Another problem arises from scale sampling when the window size is changed. In that case, we might need more or less scales for the image, depending on whether the window size is decreased or increased. Furthermore, when the pixel size changes, scale sampling has to be adjusted to derive a comparable representation. For these reasons, scale sampling is inflexible

and bears potential risks and instabilities during automatic feature extraction.

In this chapter, we design a hierarchical feature representation based on the Riesz transform as an alternative to scattering networks. The Riesz transform can be used to construct a quadrature filter for signal decomposition into amplitude and phase. Computing the amplitude acts as a nonlinear transformation and is used to build layers in depth. Our representation inherits the scale equivariance property of the Riesz transform. Global average pooling at the end results in a scale and translation invariant hierarchical representation. Theoretical implications of this representation are discussed and its application for texture and digit classification is presented. In these applications, robustness to scale variations is observed. Furthermore, our representation generalizes to completely unseen scales that are not covered by the training set.

8.1.1 Related work on scattering networks

Scattering networks and related work:
The idea behind scattering networks is to design hierarchical image descriptors similar to CNNs which are robust or (locally) invariant to transformations from a fixed compact Lie group [134]. Details on scattering networks can be found in Appendix G.

The design is based on constructing a nearly quadrature filter[1] from Morlet or other types of wavelets, i.e., a filter that decomposes the signal into amplitude and phase. The filtering is repeated for several rescaled and rotated versions of the Morlet wavelet. Only amplitude information is kept, and the same set of wavelets is applied to the amplitudes sequentially to derive higher order features.

An important result of Mallat [134] is that the scattering network is Lipschitz continuous with respect to the actions from the diffeomorphism group. This means, the scattering network yields very similar outputs for the original image and "slightly locally" deformed versions of it. As a first application example, a scattering network on the translation group was used to construct a (locally) translation invariant feature representation which was applied to texture and digit classification problems [136, 114]. This representation was further extended to local invariance for translations, rotations, and scalings [115], which proves to be useful for unknown viewing conditions in texture classification. Furthermore, a scattering network that is stable to rotations and deformations along the rotation axes was applied to solve object classification tasks in [137].

While scattering networks have shown to perform well on datasets with little training data (e.g. with 46 images per class on the CUReT dataset in [114]), this class of methods is still outperformed by CNNs on large datasets such as Caltech or CIFAR [137]. A possible explanation for this has been suggested in [138]: scattering networks are not able to represent complex, edge-like patterns, e.g. checker-boards patterns. Later, it was shown in [139, 140] that hybrids of scattering networks and deep CNNs can compete with end-to-end trained fully convolutional deep CNNs and outperform them in the small data regime. The authors replace early trainable convolutional layers with pre-defined scattering layers. Furthermore, Cotter and Kinsbury [141] design another hybrid version by using a scattering transform followed by a trainable 1d convolution as a building block of their scattering representation. The latest work on scattering networks enables parametrization of the Morlet wavelets [142], i.e. parameters are trainable and hence can be optimized for a suitable task. This results in a more problem adapted wavelet filter bank which is not necessarily a wavelet frame but is experimentally proven to be as stable to deformations (shear, scaling, and rotations) as classic scattering networks.

[1] Since the estimation of these characteristics is intrinsically noisy, various quadrature filters have been devised, see [135] for an overview and Appendix F for basics on quadrature filters. Appendix G introduces the Morlet wavelet. The reason for being *nearly* quadrature filter is due to the parameter β on the real part of the filter, i.e. it is not reflexive in the wavelet orientation.

The importance of scattering networks lies in the fact that they represent the simplest non-trainable and nonlinear feature extractor which arranges convolutional filters and non-linearities in cascades. This constitutes the closest well-studied and mathematically sound proxy model [143, 144] for trainable neural networks (e.g. CNNs) which are nowadays commonly used in a wide range of applications.

The so-called monogenic wavelet scattering network combines a scattering network with monogenic wavelets [145]. However, scale equivariance is not utilized and translation equivariance is not assumed. Hence, it is a highly redundant and scale-dependent representation which extracts $90,000$ features for a 200×200 image. The latest work [146] uses higher order Riesz transforms to classify rocks based on textures. However, here non-linearity is not applied and no feature hierarchy has been constructed. Hence, the model resembles classical manual feature extractors. Nevertheless, this work serves as a proof of concept that with a relatively low number of Riesz-based features (between 100 and 400) good performance can still be achieved.

Lindeberg's paper [116] is the only work in literature that considers scale equivariance (or covariance) in the same context as we do. An oriented quasi quadrature measure is defined from first and second order Gaussian derivatives and used for subsequent classification of textures. This representation depends on the sampling of the scale dimension which results in feature vectors of dimension 4,000. This feature dimension seems to be too redundant compared to the standard scattering network [114].

8.2 Riesz hierarchical representation

The construction of the Riesz hierarchical network is similar to that of scattering networks [134] (Appendix G). This is can be summarized as a three step process: apply complex filter bank to image, apply modulus operator as non-linearity, and repeat previous two steps to construct hierarchies. The main novelty is in fact that we use a smaller and less redundant filter bank compared to [114] based on the Riesz transform which ensures stability to variations in scales. The properties of the Riesz transform as stated above ensure that the most important properties of scattering networks such as nonexpansiveness and translation equivariance are preserved.

8.2.1 Base function and its properties

Instead of the Morlet wavelet, we use a scale-free Riesz transform kernel to construct a complex base function and eliminate the need for scale selection. For $x = (x_1, x_2) \in \mathbb{R}^2$ the complex base function $\psi : \mathbb{R}^2 \to \mathbb{C}$ is defined as

$$\psi(x) = ir_1(x) + r^{(2,0)}(x), \tag{8.1}$$

where r_1 and $r^{(2,0)}$ are kernels of the Riesz transform of first order $\mathcal{R}_1$ and second order $\mathcal{R}^{(2,0)}$, respectively. In other words, for $f \in L_2(\mathbb{R}^2)$ it holds:

$$(f * \psi)(x) = i\mathcal{R}_1(f)(x) + \mathcal{R}^{(2,0)}(f)(x),$$

Note that convolution with ψ is **scale equivariant** and a **quadrature filter**[2]. The latter is easily deduced from its **differential interpretation** since the first (second) order Riesz transform resembles the first (second) order derivative operator (see Equations (5.4) and (5.5)) and can be seen as an edge (line) detector.

[2]See Appendix F for details.

Steerability: An image can be thought of as a composition of differently oriented components, signals, or patterns. Note that the base function ψ is not an isotropic function as it is oriented vertically. Hence, the convolution with the base function preserves (amplifies) all patterns rotated in (or close to) the vertical axis, while it diminishes signals oriented horizontally. For this reason, it is useful to define a rotated version of the base function ψ. This is particularly important for images that are not uni-directional, i.e. that have multiple relevant orientations. Our base function and the rotated filters derived from it are steerable, i.e. there exist basis filters from which any rotated filter can be steered (Section 5.2, Figure 8.1, top).

Formally, let G be a finite rotation group. For group element $r = (\cos\phi, \sin\phi) \in G$, the base function ψ can be rotated using the steerability of the Riesz transform (Section 5.2) via

$$\psi[r](x) := \psi(rx) = ih_r(x) + h_r^{(2)}(x)$$

where h_r and $h_r^{(2)}$ are kernels of directional Hilbert transform or steered Riesz transform of first order $\mathcal{H}_v$ and second order $\mathcal{H}_v^{(2)}$ from Section 5.2. Then, the final filter bank can be schematically visualized (Figure 8.1, top). This yields

$$(f * \psi[r])(x) = i\mathcal{H}_r(f)(x) + \mathcal{H}_r^{(2)}(f)(x),$$

where

$$\mathcal{H}_r(f)(x) = \cos\phi\mathcal{R}_1(f)(x) + \sin\phi\mathcal{R}_2(f)(x),$$
$$\mathcal{H}_r^{(2)}(f)(x) = \cos^2(\phi)\mathcal{R}^{(2,0)}(f)(x) + \sin^2(\phi)\mathcal{R}^{(0,2)}(f)(x) + 2\cos(\phi)\sin(\phi)\mathcal{R}^{(1,1)}(f)(x).$$

requires computation of five Riesz transforms: two of the first order $(\mathcal{R}_1, \mathcal{R}_2)$ and three of the second order $(\mathcal{R}^{(2,0)}, \mathcal{R}^{(0,2)}, \mathcal{R}^{(1,1)})$. A visualization of the extracted features from the steered base functions based on the Riesz transform can be found in Figure 16 of Appendix H.

Nonexpansiveness of the base function ψ: Nonexpansiveness is preserved if the complex base function is multiplied by the real scalar $C \leq \frac{1}{2}$. Generally, it holds

$$||C\cdot(f*\psi) - C\cdot(g*\psi)||^2 \leq C||(f-g)*\psi||^2 = C||\mathcal{R}_1(f-g)||^2 + C||\mathcal{R}^{(2,0)}(f-g)||^2 \leq 2C||f-g||^2.$$

Similarly, this holds for the rotated version of ψ. This follows from the following lemma:

Lemma 7. *The following two inequalities hold for $f \in L_2(\mathbb{R}^d)$:*

- $||\mathcal{H}_r(f)||^2 \leq ||f||^2$,
- $||\mathcal{H}_r^{(2)}(f)||^2 \leq ||f||^2$.

Proof. A short and more concise version of the proof can be found in [4]. Here, we prove the stronger results for the first and second order directional Hilbert transform:

$$||\mathcal{H}_r(f)||^2 + ||\mathcal{H}_{r+\pi/2}(f)||^2 = ||f||^2$$

and

$$||\mathcal{H}_r^{(2)}(f)||^2 + ||\mathcal{H}_{r+\pi/2}^{(2)}(f)||^2 + 2||\mathcal{H}_r\mathcal{H}_{r+\pi/2}(f)||^2 = ||f||^2.$$

These results imply that the orthogonal directional Hilbert transforms preserve the total energy $||f||$ of the signal f.
Here, we will use the following notation: let $r = (\cos\phi, \sin\phi)$, then $r+\pi/2 := (\cos(\phi+\frac{\pi}{2}), \sin(\phi+\frac{\pi}{2}))$.

- Proof of the inequality $||\mathcal{H}_r(f)||^2 \leq ||f||^2$.:

$$
\begin{aligned}
&||\mathcal{H}_r(f)||^2 + ||\mathcal{H}_{r+\pi/2}(f)||^2 = \\
&= ||\cos\phi\mathcal{R}_1(f) + \sin\phi\mathcal{R}_2(f)||^2 + ||\cos(\phi+\pi/2)\mathcal{R}_1(f) + \sin(\phi+\pi/2)\mathcal{R}_2(f)||^2 = \\
&= \cos^2\phi||\mathcal{R}_1(f)||^2 + \sin^2\phi||\mathcal{R}_2(f)||^2 + 2\sin\phi\cos\phi\langle\mathcal{R}_1(f),\mathcal{R}_2(f)\rangle + \\
&+ \sin^2\phi||\mathcal{R}_1(f)||^2 + \cos^2\phi||\mathcal{R}_2(f)||^2 + 2\sin(\phi+\pi/2)\cos(\phi+\pi/2)\langle\mathcal{R}_1(f),\mathcal{R}_2(f)\rangle = \\
&= ||\mathcal{R}_1(f)||^2 + ||\mathcal{R}_2(f)||^2 + \Big(\sin(2\phi) + \sin(2\phi+\pi)\Big)\langle\mathcal{R}_1(f),\mathcal{R}_2(f)\rangle \\
&= ||\mathcal{R}_1(f)||^2 + ||\mathcal{R}_2(f)||^2 \overset{(5.9)}{=} ||f||^2.
\end{aligned}
$$

- Proof of the inequality $||\mathcal{H}_r^{(2)}(f)||^2 \leq ||f||^2$.:

$$
\begin{aligned}
&||\mathcal{H}_r^{(2)}(f)||^2 + ||\mathcal{H}_{r+\pi/2}^{(2)}(f)||^2 + 2||\mathcal{H}_r\mathcal{H}_{r+\pi/2}(f)||^2 = \\
&\overset{(\bigcirc)}{=} ||\cos^2(\phi)\mathcal{R}^{(2,0)}(f) + \sin^2(\phi)\mathcal{R}^{(0,2)}(f) + 2\cos(\phi)\sin(\phi)\mathcal{R}^{(1,1)}(f)||^2 + \\
&+ ||\cos^2(\phi+\pi/2)\mathcal{R}^{(2,0)}(f) + \sin^2(\phi+\pi/2)\mathcal{R}^{(0,2)}(f) + 2\cos(\phi+\pi/2)\sin(\phi+\pi/2)\mathcal{R}^{(1,1)}(f)||^2 + \\
&+ 2||\cos(\phi)\cos(\phi+\pi/2)\mathcal{R}^{(2,0)}(f) + \Big(\cos(\phi)\sin(\phi+\pi/2) + \sin(\phi)\cos(\phi+\pi/2)\Big)\mathcal{R}^{(1,1)}(f) + \\
&+ \sin(\phi)\sin(\phi+\pi/2)\mathcal{R}^{(0,2)}(f)||^2 = \\
&= \Big(\cos^2(\phi) + \sin^2(\phi)\Big)^2||\mathcal{R}^{(2,0)}(f)||^2 + \Big(\cos^2(\phi) + \sin^2(\phi)\Big)^2||\mathcal{R}^{(0,2)}(f)||^2 \\
&= +2\Big(\sin^2(2\phi) + \cos^2(2\phi)\Big)^2||\mathcal{R}^{(1,1)}(f)||^2 + \\
&+ 2\Big(\cos^2(\phi) + \cos^2(\phi+\pi/2)\Big)\Big(\sin(2\phi) + \sin(2\phi+\pi)\Big)\langle\mathcal{R}^{(1,1)}(f),\mathcal{R}^{(2,0)}(f)\rangle + \\
&+ 2\Big(\sin^2(\phi) + \sin^2(\phi+\pi/2)\Big)\Big(\sin(2\phi) + \sin(2\phi+\pi)\Big)\langle\mathcal{R}^{(1,1)}(f),\mathcal{R}^{(0,2)}(f)\rangle \\
&+ 2\Big(\cos(\phi)\sin(\phi) + \cos(\phi+\pi/2)\sin(\phi+\pi/2)\Big)^2\langle\mathcal{R}^{(0,2)}(f),\mathcal{R}^{(2,0)}(f)\rangle = \\
&\overset{(\triangle)}{=} ||\mathcal{R}^{(2,0)}(f)||^2 + 2||\mathcal{R}^{(1,1)}(f)||^2 + ||\mathcal{R}^{(0,2)}(f)||^2 \overset{(5.9)}{=} ||f||^2.
\end{aligned}
$$

($\bigcirc$) Here, we give details on how term $\mathcal{H}_r\mathcal{H}_{r+\pi/2}(f)$ is calculated.

$$
\begin{aligned}
&\mathcal{H}_r\mathcal{H}_{r+\pi/2}(f) = \mathcal{H}_r\Big(\mathcal{H}_{r+\pi/2}(f)\Big) = \mathcal{H}_r\Big(\cos(\phi+\pi/2)\mathcal{R}_1(f) + \sin(\phi+\pi/2)\mathcal{R}_2(f)\Big) = \\
&= \cos(\phi+\pi/2)\mathcal{H}_r\Big(\mathcal{R}_1(f)\Big) + \sin(\phi+\pi/2)\mathcal{H}_r\Big(\mathcal{R}_2(f)\Big) = \\
&= \cos(\phi+\pi/2)\Big(\cos(\phi)\mathcal{R}_1\mathcal{R}_1(f) + \sin(\phi)\mathcal{R}_2\mathcal{R}_1(f)\Big) + \\
&+ \sin(\phi+\pi/2)\Big(\cos(\phi)\mathcal{R}_1\mathcal{R}_2(f) + \sin(\phi)\mathcal{R}_2\mathcal{R}_2(f)\Big) = \\
&= \cos(\phi+\pi/2)\cos(\phi)\mathcal{R}^{(2,0)}(f) + \cos(\phi+\pi/2)\sin(\phi)\mathcal{R}^{(1,1)}(f) + \\
&+ \sin(\phi+\pi/2)\cos(\phi)\mathcal{R}^{(1,1)}(f) + \sin(\phi+\pi/2)\sin(\phi)\mathcal{R}^{(0,2)}(f) = \\
&= \cos(\phi+\pi/2)\cos(\phi)\mathcal{R}^{(2,0)}(f) + \Big(\cos(\phi+\pi/2)\sin(\phi) + \sin(\phi+\pi/2)\cos(\phi)\Big)\mathcal{R}^{(1,1)} \\
&+ \sin(\phi+\pi/2)\sin(\phi)\mathcal{R}^{(0,2)}(f).
\end{aligned}
$$

($\triangle$) Terms with $\langle \mathcal{R}^{(1,1)}(f), \mathcal{R}^{(2,0)}(f) \rangle$ and $\langle \mathcal{R}^{(1,1)}(f), \mathcal{R}^{(0,2)}(f) \rangle$ disappear due to $\sin(2\phi) + \sin(2\phi + \pi) = \sin(2\phi) - \sin(2\phi) = 0$.
Term with $\langle \mathcal{R}^{(0,2)}(f), \mathcal{R}^{(2,0)}(f) \rangle$ also disappear because $\cos(\phi)\sin(\phi) + \cos(\phi + \pi/2)\sin(\phi + \pi/2) = 0$. This is consequence of $\cos(\phi + \pi/2)\sin(\phi + \pi/2) = -\cos(\phi)\sin(\phi)$ for every $\phi \in [0, 2\pi]$. For $\phi \in [0, \pi/2] \cup [\pi, 3\pi/2]$ it holds $\sin(\phi + \pi/2) = -\cos(\phi)$ and $\cos(\phi + \pi/2) = \sin(\phi)$. For $\phi \in [\pi/2, \pi] \cup [3\pi/2, 2\pi]$ it holds $\sin(\phi + \pi/2) = \cos(\phi)$ and $\cos(\phi + \pi/2) = -\sin(\phi)$.

□

Now, a similar proof shows nonexpansiveness for steered base function $\psi[r]$.
The reason for requiring nonexpansiveness in [134] is that the representation should be stable (Lipschitz continuous) to small deformations, i.e. it should not amplify their effect [143]. Note that the selection of the coefficient C determines the decay of energy as we increase the depth of the Riesz representation. Its role in the Riesz representation will be analyzed later.

Zero integral of base function ψ: This result serves as a motivation to introduce non-linearity in the scenario where we assume (global) translation invariance. To achieve translation invariance, pooling operators must compute pooling statistics on the features map across the whole image domain[3]. If one applies global average pooling without non-linearity, the result would inevitably be 0 for every feature removing all the discriminative characteristics of the transformed signal. Zero integral of base function ψ also guarantees that representation is invariant to constant shifts in grayvalues. This result is given in the following lemma:

Lemma 8. *Integral of base function ψ equals to 0, i.e.*

$$p.v. \int_{\mathbb{R}^2} \psi(x)dx = 0$$

.

Proof. Since from equation (8.1) we have $\int_{\mathbb{R}^2} \psi(x)dx = i \cdot \int_{\mathbb{R}^2} r_1(x)dx + \int_{\mathbb{R}^2} r^{(2,0)}(x)dx$, this can be done separately for real and imaginary part. The kernel r_1 in the real part is anti-symmetric, i.e. $r_1(x) = -r_1(-x)$ and hence p.v. $\int_{\mathbb{R}} r_1(x)dx = 0$. In the imaginary part, kernel $r^{(2,0)} = r_1 * r_1$ and hence

$$\text{p.v.} \int_{\mathbb{R}^2} r^{(2,0)}(x)dx = \text{p.v.} \int_{\mathbb{R}^2} \text{p.v.} \int_{\mathbb{R}^2} r_1(t)r_1(x-t)dtdx = \text{p.v.} \int_{\mathbb{R}^2} r_1(t)\Big(\text{p.v.} \int_{\mathbb{R}^2} r_1(x-t)dx\Big)dt =$$
$$= [y = x - t \text{ change of variables}] = \text{p.v.} \int_{\mathbb{R}^2} r_1(t)\Big(\int_{\mathbb{R}^2} r_1(y)dy\Big)dt = [r_1 \text{ is anti-symmetric}] = 0.$$

Above p.v. refers to the Cauchy principal integral since the convolution with the Riesz kernel is not defined at 0. □

8.2.2 Path ordered scattering

The goal of this section is to design a universal hierarchical representation (Figure 8.1, top) of the image structure which would be useful in the sense that a simple classifier (e.g. PCA or SVM) can be constructed on top of the representation without the need to train a deep representation or adjust parameters.

[3]Section 8.2.3 is focused on pooling operators for global translation and scale invariance.

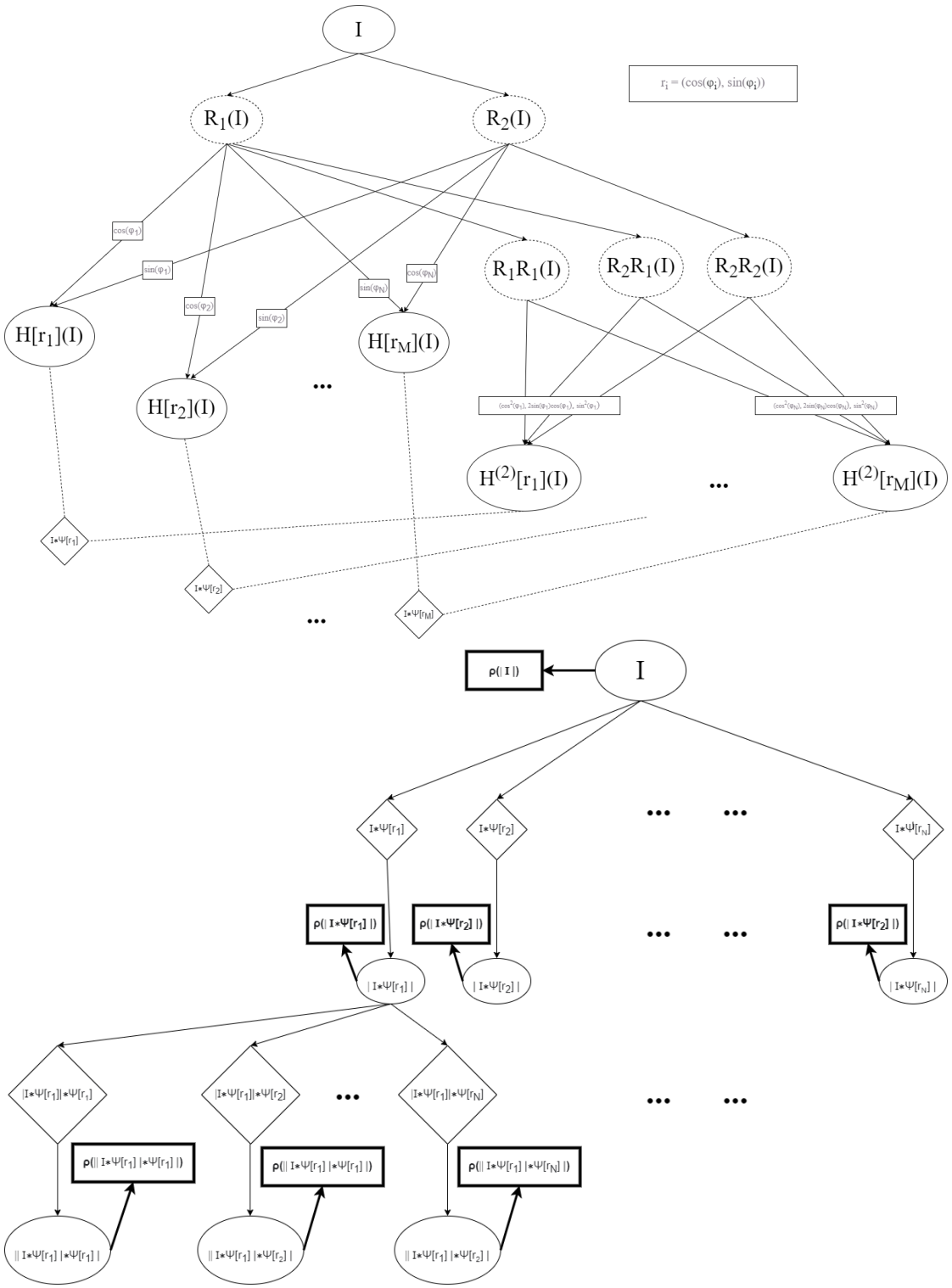


Figure 8.1: Top: extraction of features from quadrature filter based on the Riesz transform. Bottom: construction of Riesz representation.

Alternative to Scattering networks

Let $G_M = \{k\frac{\pi}{M}, k \in \{0, 1, \cdots, M-1\}\}$ be a finite rotation group where $M \in \mathbb{N}$ controls the discretization of the group. Then we define the base transformation layer:

$$W(f) := \{C(f * \psi[r]), r \in G_M\}$$

where $C \leq \frac{1}{2}$ is a scaling constant which is needed to preserve nonexpansiveness or control energy decay. To ensure that both the coordinate axes and the diagonal directions are included in G_M, we choose $M = 4m$ for some $m \in \mathbb{N}$.

Similar to scattering networks, we then apply a non-linearity operator. Here, we apply the amplitude operator to every feature map from $W(f)$ and discard the phase. This yields

$$S(f) = A(W(f)) := \{A(g), g \in W(f)\},$$

where $A : L^2(\mathbb{R}^d, \mathbb{C}) \to L^2(\mathbb{R}^d)$ is the pointwise amplitude operator of the complex functions i.e. $A(g_1(x) + ig_2(x)) = \sqrt{g_1(x)^2 + g_2(x)^2}$. To create a multilayer deep representation, we apply the operator S sequentially. That is, for the k-th layer:

$$S_k(f) := S^k(f) = S(\cdots S(f)), \tag{8.2}$$

and let $S_0 := f$. Selected feature maps up to depth 3 from the Riesz feature representation prior to pooling are visualized on two examples in Figure 17 and Figure 18 of Appendix H. The final representation Φ with $K \in \mathbb{N}$ layers is the ordered list of feature maps from all depths

$$\Phi(f) = \Big(S_k(f) \quad | \quad k = 0, \cdots, K\Big), \tag{8.3}$$

This representation yields in total $\sum_{k=0}^{K} M^k$ features per pixel.

Nonexpansiveness of Riesz representation $\Phi(f)$: A requirement from [114, 134] is that the total energy of the layer equals the energy of the input f. Although this is needed to achieve Lipschitz continuity to diffeomorphism actions, this results in fast energy decay and reduces the importance of features in deeper layers. For this reason the depth of scattering networks is usually limited to 2 [134, 114]. However, this property is only relevant when using classifiers such as PCA that do not use any feature normalization or preprocessing. On the other hand, prior to training a SVM on the features from the Riesz representation, the feature vectors are normalized coordinatewise with maximal absolute value. This normalization removes the effect of energy decay.

Strictly theoretically, to achieve nonexpansiveness of $\frac{1}{K+1}\Phi$ for $M = 4m$ for $m \in \mathbb{N}$, one can choose $C = \frac{1}{M}$ by the following arguments.

First, we analyze nonexpansiveness of the operator $S(f) = A(W(f))$. Here, we have

$$\begin{aligned}
||S(f) - S(g)||^2 = ||A(W(f)) - A(W(g))||^2 \leq ||A\big(W(f) - W(g)\big)||^2 = \\
= ||W(f-g)||^2 = C\Big(\sum_{r \in G_M} ||i \cdot \mathcal{H}_r(f-g) + \mathcal{H}_r^{(2)}(f-g)||^2\Big) = \\
= C\Big(\sum_{r \in G_M} ||\mathcal{H}_r(f-g)||^2 + \sum_{r \in G_M} ||\mathcal{H}_r^{(2)}(f-g)||^2\Big).
\end{aligned}$$

From Lemma 7 we have $\sum_{r \in G_M} ||\mathcal{H}_r(f-g)||^2 \leq \frac{M}{2}||f-g||$. A similar statement can be shown for the second summand.

For $C = \frac{1}{M}$, this yields

$$||S(f) - S(g)||^2 \leq C \left(\frac{M}{2}||f-g||^2 + \frac{M}{2}||f-g||^2 \right) = ||f-g||^2.$$

Since Φ defined for some $K \in \mathbb{N}$ has depth K, it needs to be scaled by $\frac{1}{K+1}$ to achieve nonexpansiveness. We experimentally investigate the effect of the constant C on the overall representation performance in Section 8.3.2.

Equivariance properties

As stated in Section 5.2 above, the Riesz transform is equivariant to two types of transformations: translation and scaling. In this section, we show that the Riesz representation inherits these equivariances. To prove this, it is enough to prove that the operator S is equivariant to these transformations since the Riesz representation consists of applying this operator in sequence. The operator S consists of two types of transformations: a convolution with complex base functions ψ and applying the pointwise amplitude operator A. By construction the base functions preserve both types of equivariances. The amplitude operator A is a pointwise function and hence preserves equivariance to scaling and translation. Next, we sketch the proof of the last claim.

Theorem 4. *Let $A : L_2(\mathbb{R}^d, \mathbb{C}) \to L_2(\mathbb{R}^d)$ be a (continuous) pointwise operator that only depends on the pixel values. Then A is equivariant to translation and scaling.*

Proof. Being a pointwise operator formally means there exists a function $F : \mathbb{C} \to \mathbb{R}$ such that for every $x \in \mathbb{R}^d$ it holds that $A(f)(x) = F(f(x))$. For a translation operator $T_{x_0} : L_2(\mathbb{R}^d, \mathbb{C}) \to L_2(\mathbb{R}^d, \mathbb{C})$ for $x_0 \in \mathbb{R}^d$ defined as $T_{x_0}(f)(x) = f(x - x_0)$, it follows

$$T_{x_0}(A(f))(x) = A(f)(x - x_0) = F(f(x - x_0)) = F(T_{x_0}(f)(x)) = A(T_{x_0}(f))(x).$$

For a scaling operator $L_a : L_2(\mathbb{R}^d, \mathbb{C}) \to L_2(\mathbb{R}^d, \mathbb{C})$ where $a > 0$ defined as $L_a(f)(x) = f(xa^{-1})$, the proof is analogous:

$$L_a(A(f))(x) = A(f)(xa^{-1}) = F(f(xa^{-1})) = F(L_a(f)(x)) = A(L_a(f))(x).$$

□

8.2.3 Pooling operations for scale and translation invariance

The output of the operator Φ from Equation (8.3) is a sequence of feature maps which have the same size as the input image. Furthermore, the operator Φ is both scale and translation equivariant. The goal is to use a pooling operation $\rho : L_2(\mathbb{R}^d) \to \mathbb{R}$ that enables the transition from equivariance to invariance (Section 4.2).

For that purpose, a global pooling operator ρ is applied to every feature map from Φ. A global pooling operator is an operator which takes an image or feature map $f \in L_2(\mathbb{R}^d)$ as an input and computes a single summary statistic on the whole image, i.e. $\rho : L_2(\mathbb{R}^d) \to \mathbb{R}$. When applied to an image, this pooling function ρ becomes a global operator over a bounded, discrete domain.

In contrast, one can compute summary statistics on non-overlapping windows on the image, e.g. Max Pooling in Appendix A. These pooling statistics depend on the position in the image or local image structure. Hence, they are called localized or local pooling operators. These

operators only guarantee local invariance depending on the size of the window on which they are calculated. The drawback with local pooling operators is that they are not comparable for images of different sizes since the feature vectors after pooling have different sizes. Hence, we focus on global pooling on bounded discrete domains $D \subset \mathbb{Z}^d$, only.

Standard choices for global pooling operators are average and maximum pooling. Average pooling takes the mean value of the feature map. Hence, it is important that the aspect ratio between the object and the background is fixed. On the other hand, max pooling takes the maximal value of the feature map. Here, the requirement from average pooling does not need to be fulfilled to achieve invariant features because it is based on recording the extreme values rather than the average ones.

Standard choices for global pooling operators are average and maximum pooling. Average pooling takes the mean value of the feature map. To obtain comparable values for different scales, it is important that the ratio between the object and the background is fixed when altering the scale. An alternative is global max pooling which reports the maximal value of the feature map. However, in noisy conditions the extreme values are more sensitive to outliers compared to the mean values. Furthermore, this operator does not guarantee nonexpansiveness of the Riesz feature representation.

8.3 Experiments

Throughout this section, we use a Riesz feature representation with depth 3 based on a discrete rotation group with $M = 4$ angles, if not specified otherwise. In this case, the output of the Riesz feature representation for an image of arbitrary size consists of 85 features. The nonexpansiveness constant C from Section 8.2.2 is set to 1, as justified later experimentally in Section 8.3.2.

8.3.1 MNIST Large Scale

MNIST Large Scale has already been used in Section 6.4 for the Riesz networks. See Figure 6.17 and Figure 6.18 for image examples.

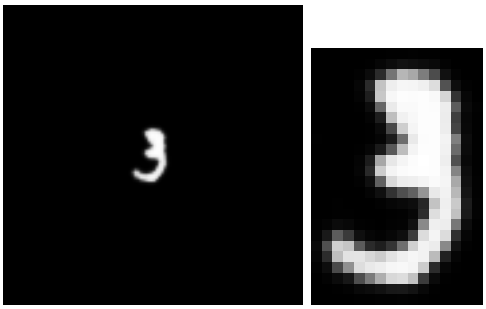


Figure 8.2: Extracting the bounding box (right) from the input image (left). Input image is of size 112×112, bounding box is of size 16×24.

Why scale equivariance is not perfectly suited for this or similar problems?

Remember that scale equivariance implies that the Riesz transform commutes with the scaling operator. In other words, it does not matter in which order we apply Riesz transform and scaling operator to the image. In practice, however, the images under consideration have a bounded

scale	0.5	0.595	0.707	0.841	1	1.189	1.414	1.682
FovAvg 17ch tr1 [99]	98.58	99.05	**99.33**	**99.39**	**99.40**	**99.39**	**99.38**	**99.36**
FovMax 17ch tr1 [99]	**98.71**	**99.07**	99.27	99.34	99.37	99.35	99.36	99.34
RieszNet [3]	96.34	97.59	98.06	98.54	98.58	98.50	98.45	98.40
RieszNet-pad40 [3]	96.34	97.55	98.07	98.47	98.63	98.58	98.53	98.44
ours-MLP	82.78	91.47	94.19	95.23	96.23	95.73	95.79	95.63
ours-SVM	82.36	91.29	93.86	94.83	95.74	95.19	95.13	95.04

scale	2	2.378	2.828	3.364	4	4.757	5.657	6.727	8
FovAvg 17ch tr1 [99]	**99.35**	99.31	99.22	99.12	98.94	98.47	96.20	89.17	71.31
FovMax 17ch tr1	99.33	**99.35**	**99.34**	**99.35**	**99.34**	**99.27**	**97.88**	92.76	79.23
RieszNet-pad [3]	98.39	98.24	98.01	97.51	96.42	93.50	81.58	67.66	51.82
RieszNet-pad40 [3]	98.46	98.39	98.34	98.29	98.16	97.80	96.82	**93.75**	**83.6**
ours-MLP	95.60	95.73	95.79	95.69	95.61	95.64	94.76	88.26	67.92
ours-SVM	95.08	95.02	95.01	94.97	95.05	95.03	94.00	86.19	63.96

Table 8.1: Classification accuracies (in %) for the MNIST Large Scale data set. Several methods trained on the full training set (50,000 images) at scale 1. Best performing method bold. Accuracies for FovAvg 17ch tr1 and FovMax 17ch tr1 are taken from [99].

domain. Hence, mean pooling as discussed above may be sensitive to changes of the size ratio between object and background. Also, problems may arise when an object partially leaves the image when upscaling. This problem is particularly relevant for non-local feature extractors such as the Riesz transform. It is interesting to notice that neural networks might overcome this issue through the training procedure as demonstrated for a trainable Riesz network [3] in Chapter 6. However, Riesz scattering network uses global average pooling as a summary statistics which makes it sensitive to this effect.

To deal with the changing size ratio between the object and the background, one can determine the bounding box around the object of interest. Restriction to the bounding box will preserve the size ratio between object and background. Note that in general, the bounding box computation should be scale equivariant to preserve the same property of the Riesz feature representation. Here, we design a simple scale equivariant four step bounding box algorithm:

1. normalize the image gray values with min-max normalization to be in the interval $[0, 1]$,
2. pad the image with 50 zero pixels on each side,
3. threshold the image with $t = 0.5$, and
4. draw the bounding box on the binary image.

Finally, the bounding box is enlarged to capture black background by elongating the diagonals by 40%. The bounding box is cropped from the image (Figure 8.2) and input to the Riesz feature representation. Afterwards, the SVM classifier is trained on the output from the previous step. The MLP model used for the comparison has roughly $30,000$ trainable parameters (see Table 8.4 "MLP depth 3 angle 4").

Multilayer perceptron (MLP) as classifier

A multilayer perceptron (MLP) is a fully connected feedforward neural network whose layers consist of a fully connected layer (or 1d convolution), batch normalization, and non-linear acti-

scale	0.5	0.595	0.707	0.841	1	1.189	1.414	1.6828
RieszNet train 1000 [3]	**86.94**	**89.07**	**90.98**	**91.54**	**91.64**	**91.93**	**91.61**	**91.42**
ours-MLP train 1000	70.72	80.65	84.38	87.13	89.00	87.21	87.53	87.26
ours-SVM train 1000	71.16	80.28	82.98	84.79	87.49	85.09	84.99	84.81
ours-SVM train 5000	78.02	86.97	89.81	91.20	92.26	91.48	91.31	91.11
ours-SVM train 20000	80.38	89.67	92.20	93.31	94.66	93.91	93.97	93.70
ours-SVM train 50000	82.36	91.29	93.86	94.83	95.74	95.19	95.13	95.04
ours-MLP train 50,000	82.78	91.47	94.19	95.23	96.23	95.73	95.79	95.63

scale	2	2.378	2.828	3.364	4	4.757	5.657	6.727	8
RieszNet train 1000 [3]	**90.93**	**90.24**	**89.32**	**87.97**	**85.78**	82.01	74.84	67.31	56.88
ours-MLP train 1000	87.48	87.28	87.27	87.33	87.27	**86.98**	**86.14**	**78.00**	**59.79**
ours-SVM train 1000	84.74	84.75	84.55	84.50	84.53	84.26	83.71	70.97	48.33
ours-SVM train 5000	91.38	91.18	91.12	91.18	91.14	91.01	90.15	78.23	53.04
ours-SVM train 20000	93.83	93.75	93.71	93.64	93.77	93.59	92.82	84.19	60.69
ours-SVM train 50000	95.08	95.02	95.01	94.97	95.05	95.03	94.00	86.19	63.96
ours-MLP train 50,000	95.60	95.73	95.79	95.69	95.61	95.64	94.76	88.26	67.92

Table 8.2: Classification accuracies (in %) for the MNIST Large Scale data set. Methods trained on a reduced training set comprising $1,000$ images. Best performing method in bold. Training set scale is 1. Note that [99] does not report the accuracies for FovAvg 17ch tr1 and FovMax 17ch tr1 on a training set of size 1,000 images.

vation (ReLU). We explore the MLP as an alternative to the SVM and feed it with the output of the Riesz representation.

Details on training: We apply the following hyperparameters: cross entropy loss, batch size 50, Adam optimizer [119] with initial learning rate 0.001. Number of epochs and learning rate decay are adapted to the size of the training set. For a training set of $1,000$ images, we use 75 epochs and step decay for which the learning rate halves every 10 epochs. For a training set of $50,000$ images, we use 30 epochs and halve the learning rate every 4 epochs.

Comparison with state-of-the-art

The goal of this experiment is to validate scale equivariance properties of the Riesz representation as well as to deduce how many training examples are needed to achieve decent performance. Table 8.1 shows a comparison of our method with state-of-the-art methods: RieszNet [3] from Section 6.4 and CNN on a rescaled version of the images [99]. Details on both methods can be found in Section 6.4.

Generally, accuracies of our method are stable for scales in the range $[0.707, 5.657]$ for both, SVM and MLP. The MLP performs slightly better (around 1%) than the SVM. However, the costs of training and parameter tuning for the MLP are higher. Both competing methods achieve $4-5\%$ higher accuracy than our Riesz representation baseline with SVM. However, this agrees with similar experiments reported in the literature [137] when comparing scattering and deep networks on large datasets.

Accuracies depending on the number of training images are reported in Table 8.2. Here, only a comparison with RieszNet is available. Accuracy with SVM improves significantly as we have more training examples (by 10% in accuracy for 1,000 vs 50,000 training examples) since more training examples generally increase the diversity of handwriting styles for digits. However, performance on the relatively small training set of 1,000 examples (100 training examples per

class) is around 85% for SVM and stable for the scales in the range $[0.707, 5.657]$. When trained on the same training set of size 1,000, MLP is better than SVM (87% vs 85%). Interestingly, MLP outperforms RieszNet for scales larger than 4.757. This could be due to the window cropping procedure.

Ablation study on Riesz representation using MLP classifier

We vary the fineness of discretization of the rotation group and the depth of the Riesz feature representation to understand how to optimally balance Riesz feature representation size and accuracy. The multilayer perceptron with 2 hidden layers each having 128 channels is used as a classifier for every parameter configuration of the Riesz feature representation. We perform this only on a subset of available scales $\{0.5, 1, 2, 4, 8\}$ to reduce the runtimes. Results are shown in Table 8.3. Details on architecture and the number of parameters are given in Table 8.4.

In the chosen architecture, the output of the Riesz feature representation is the first channel. Therefore, the number of parameters of the MLP depends on the parameter configuration of the Riesz feature representation. We use again the Riesz feature representation of depth 3 and with 4-angle rotation group as baseline. Results are reported using the full training set of $50,000$ images.

The increase in depth from 3 to 4 improves the results by not more than 1%, while the number of parameters increases by a factor of 4. This indicates that the new features are non-informative and redundant. Similar effects can be observed when enlarging the finite rotation group from 4 to 8 angles. Simultaneously reducing the depth to 2 yields results slightly worse (around 2%) than those of the baseline while the number of features stays roughly the same: 85 vs 73. These results indicate that for this problem between 70 and 100 features should be optimal.

MLP/scale			0.5	1 (t)	2	4	8
# training images	depth	# angles					
1,000	3	4	70.72	89.00	87.48	87.30	59.79
1,000	4	4	68.40	90.85	90.39	90.29	49.60
50,000	2	8	79.45	94.08	93.64	93.64	67.08
50,000	3	4	82.78	96.23	95.60	95.61	67.92
50,000	3	8	83.64	96.77	96.42	96.48	73.77
50,000	4	4	80.23	96.83	96.48	96.50	64.16

Table 8.3: Results (accuracies in %) of the ablation study for the MLP on MNIST Large Scale. The test set consists of 10,000 images per scale. Trained on scale 1.

	architecture	parameters
MLP depth 2 angle 8	73-128-128-10	27,786
MLP depth 3 angle 4	85-128-128-10	29,322
MLP depth 3 angle 8	585-128-128-10	93,322
MLP depth 4 angle 4	341-128-128-10	62,090

Table 8.4: Details of the applied MLPs with 2 hidden layers. Note that affine batch normalization was used for the two hidden layers in MLP, which increases the number of trainable parameters.

8.3.2 KTH-tips

KTH-tips [147, 148] consists of images of ten classes of textures under varying illumination, orientation, and scale. Texture classes are real materials (sandpaper, crumpled aluminium foil, styrofoam, sponge), fabrics like corduroy, linen, and cotton, and natural structures like brown bread, orange peel, and cracker (Figure 8.3). Each class has 81 images split in 9 scales. We use 40 images for training and the rest for testing. In this dataset, textures fill the image window. Moreover, textures are spatially homogeneous by nature. Thus, no bounding box is needed. A particularly interesting aspect of this dataset is scale variation within every class due to the varying viewing conditions, e.g. due to the changing distance between the camera and the object (Figure 8.4).

As a baseline here, we use Bruna's scattering network [114], implemented using Kymatio [149]. See Appendix G for more details on the baseline. As suggested by the authors, a depth of 2 is used. We use the discrete rotation group with $M = 4$ (as for the Riesz feature representation), i.e. filters are steered for the following angles $\{0, \frac{\pi}{4}, \frac{\pi}{2}, \frac{3\pi}{4}\}$. The maximal number of scales ($J = 7$) is selected to achieve global translation invariance. The scattering network gives 365 features as an output. For both representations, a PCA classifier with the first 20 principal components is used.

PCA classifier

Here, we give details on PCA classification according to [150]. Let $n \in \mathbb{N}$ be the number of training images per class, $C \in \mathbb{N}$ the number of classes, and let $\{I_1, \cdots, I_n\}$ denote the images belonging to class c. Let $E_c(\Phi)$ denote the expected value of the P-dimensional Riesz feature representation for an image from class c. Practically, it is estimated by the mean of $\{\Phi(I_1), \cdots, \Phi(I_n)\}$. The goal of the PCA is to approximate the centered data $\Phi - E_c(\Phi)$ on class c by its projection to a lower (d-) dimensional space V_c for $d << P$. The subspace V_c is then said to be spanned by d principal components. After computing V_c for every class c, the projection errors when projecting a given input image I on each V_c can be calculated. The smaller the projection error, the more likely is the image to belong to that class. Hence, a classification rule can be easily derived: Let P_{V_c} be the projection operator for class c. The define the PCA classifier k_{PCA} for image I by

$$k_{PCA}(I) = \arg\min_c ||\Phi(I) - E_c(\Phi) - P_{V_c}(\Phi(I) - E_c(\Phi))||_2.$$

We used the PCA implementation *sklearn* [151] in Python.

Experimental analysis of nonexpansiveness (Experiment 0)

In Mallat's work [134], nonexpansiveness together with decay of wavelets is a key to proving Lipschitz continuity to small deformations (Appendix G). enerally, missing nonexpansiveness was identified as the main reason why neural networks are sensitive to small deformations. However, in many applications, the output of the feature representations is scaled prior to training the classifier (e.g. SVM), or batch normalization is used. Feature scaling and batch normalization distort the nonexpansiveness. Hence, even if a scattering network is used, one should be careful in classifier design to preserve the nonexpansiveness of the scattering network.

Experimentally, we investigate how sensitive the performance of the Riesz feature representation is to changes in the nonexpansiveness constant C. For $M = 4$, C should be $\frac{1}{4}$ to achieve nonexpansiveness. For $C < \frac{1}{4}$, one has strict inequality for S, i.e. $||S(f) - S(g)|| < ||f - g||$, while for $C > \frac{1}{4}$, there exists a constant $C_1 = 4C > 1$ s.t. $||S(f) - S(g)|| \leq C_1||f - g||$. Hence,

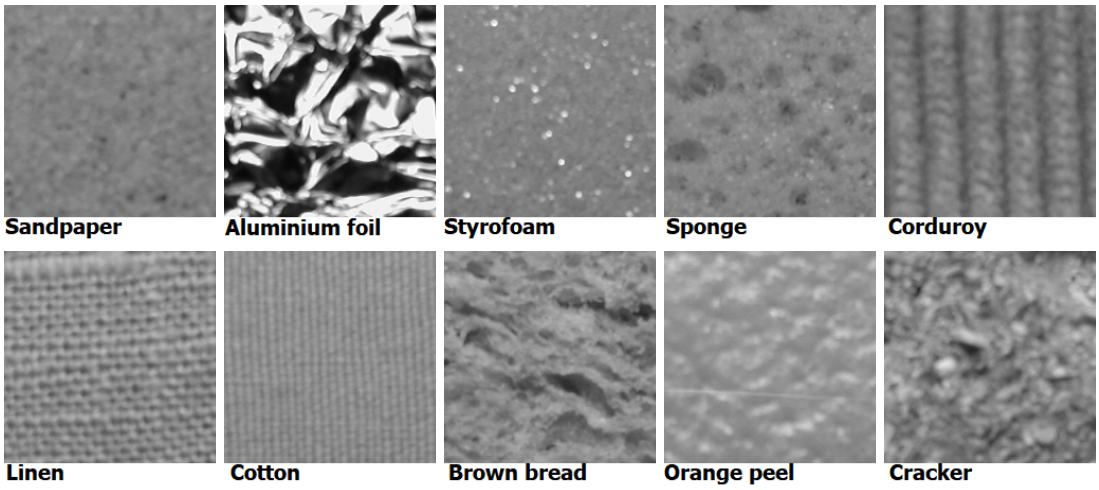


Figure 8.3: Sample classes in KTH-tips dataset.

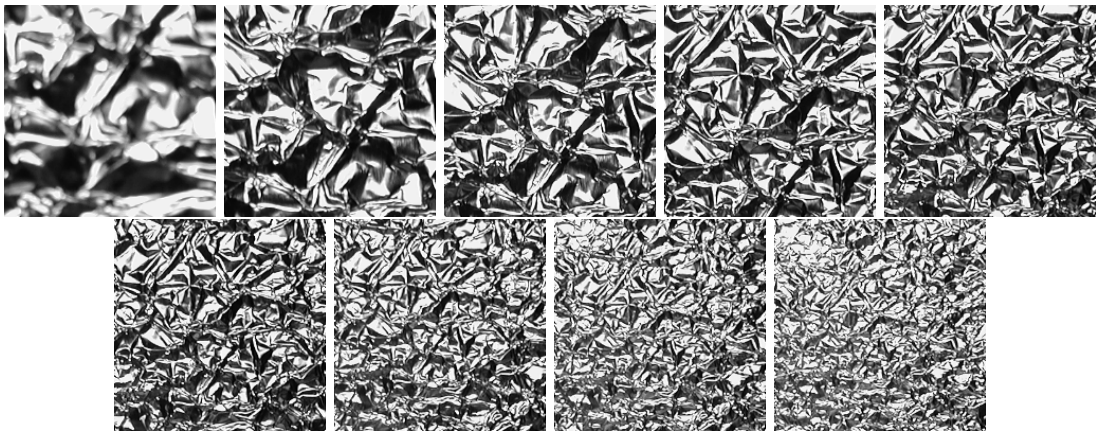


Figure 8.4: Scale variation in KTH-tips dataset for aluminium foil sample.

C	0.125	0.25	0.5	1	2	4
ours	96.34	97.07	96.83	97.32	96.83	96.10

Table 8.5: (Experiment 0): Accuracy of our Riesz representation (in %) for scaling factor C on the randomly split KTH-tips dataset.

for every C, the Riesz feature representation $\Phi(f-g)$ remains bounded by $C_1||f-g||$ for some constant $C_1 > 0$. This is in contrast to feature scaling and batch normalization.

We found that the choice of C did not have a large effect on the classification accuracy on the KTH-tips dataset (Table 8.5). Hence, in all experiments in the paper, we keep C fixed to 1.

	seed 42	seed 21	seed 10	seed 5	seed 0	mean (std)
ours-PCA	97.32	94.39	94.88	95.12	96.34	95.61 (1.196)
Scattering-PCA	97.56	95.61	98.05	96.10	98.05	97.06 (1.133)

Table 8.6: (Experiment 1): Random splitting for 5 seeds: accuracy (in %). Our Riesz representation has depth 3 and uses 4 angles, resulting in 85 features. The scattering network has depth 2 and uses 4 angles (in total 365 features).

Random splitting of the dataset (Experiment 1)

This experiment tests the general performance of the methods on this dataset, i.e. how expressive features of two methods are in characterizing the textures. Following the commonly applied approach, we split the dataset randomly into training and test sets without any regard for orientation, scale, or illumination. We report the accuracy over 5 random splitting with seeds in Table 8.6. The scattering network achieves slightly better results in this setting, but uses 4 times more features. However, random splitting of the dataset does not shed light on how useful or robust the representation is under completely unseen conditions, e.g. scales.

Scale robustness for random splitting of the dataset (Experiment 2)

The goal of this experiment is to test robustness of our representation with respect to rescaling of the image. For this purpose, images are scaled by factors from the range $[0.5, 2]$ using spline interpolation. Scaling imitates small camera movements.

Note that rescaling can destroy valuable texture information due to blurring or interpolation. Hence, performance is expected to worsen as the rescaling factor deviates further from 1. For this experiment, the Riesz feature representation does not require any adjustments since it works on arbitrary image sizes. In contrast, the scattering network requires either cropping (for upscaling) or reflective padding (for downscaling) to the fixed image size of 200×200. To analyze the effect of cropping and padding on the performance of the method, we report the results for the Riesz feature representation both on the same input images as for the scattering network (with cropping/padding) and on the unchanged rescaled images (without cropping/padding).

For fixed seed, accuracy is shown in Table 8.7 where scale 1 refers to the original image without rescaling. Here, we can notice that the Riesz feature representation without padding/cropping is significantly more stable to the larger scale variations than the scattering network. Interestingly, results for the Riesz feature representation with cropping (Table 8.7 top) are significantly better than for the scattering network, while in the case of padding (Table 8.7 bottom) results are comparable to those of the scattering network. This implies that the padding disrupts the textural composition.

scaling factor	1.05	1.1	1.15	1.25	1.5	2
ours PCA	96.34	96.59	96.34	96.34	96.10	96.10
ours PCA (cropping)	95.37	94.63	95.37	95.37	93.41	86.10
scattering PCA	97.56	96.34	95.37	94.15	81.95	65.37
scaling factor	0.5	0.75	0.85	0.9	0.95	1
ours PCA	82.44	95.37	95.85	97.07	96.58	97.32
ours PCA (padding)	67.56	89.02	94.15	94.88	94.39	97.32
scattering PCA	65.12	90.00	96.34	96.10	96.83	97.56

Table 8.7: (Experiment 2): Scale robustness for random splitting of the dataset: controlled scaling of test set. Accuracy in %. The Riesz representation (ours) has depth 3 and uses 4 angles (in total 85 features). The scattering network has depth 2 and uses 4 angles (in total 365 features).

	train-first 40	train-mid 40	train-last 40
ours-PCA	77.07	86.82	88.75
Scattering-PCA	55.61	79.02	62.25

Table 8.8: (Experiment 3): Scale dependent splitting: accuracy (in %). The Riesz representation (ours) has depth 3 and uses 4 angles (in total 85 features). The scattering network has depth 2 and uses 4 angles (in total 365 features)

Scale robustness for scale dependent splitting of the dataset (Experiment 3)

Since scale information is known for this dataset, we can choose different scales for training and test sets and test the scale generalization ability this way. We test the performance of the methods on scales that are outside of the training set scale distribution. This is useful in settings where we are not able to collect all possible scales in the training set. This experiment tests larger scale variations than the previous one. Here, we conduct 3 experiments based on the selection of scales in the training set. The 40 training images are selected from either the smallest, the medium-sized or the largest scales in the dataset.

Scale variation for the aluminium foil class is shown in Figure 8.4. Results on the remainder of the dataset for each training set splitting are shown in Table 8.8. The just described setting turns out to be the hardest due to completely unseen scales in the testing set. The Riesz representation reaches accuracies in the range $[0.77, 0.88]$ which is an improvement over the scattering network with accuracies in the range of $8 - 25\%$ depending on the scenario.

8.4 Discussion

The Riesz feature representation is a hand-crafted feature extractor based on stacking quadrature filters using first and second order Riesz transforms in a hierarchy. This representation creates features that are scale and translation equivariant and treats scale dimension continuously, i.e. it avoids sampling or discretization of the scale dimension.

As a result, the number of features is significantly reduced compared to other representations such as scattering networks. Our representation contains only 85 features, which is a reduction by factor 4 compared to the corresponding scattering representation.

A consequence of scale equivariance is the generalization to unseen scales. We show this empirically on the MNIST Large Scale dataset by training an SVM classifier on the Riesz feature representation on a fixed scale, and testing on unseen scales that differ significantly from the one in the training set. Moreover, our representation proves to be useful for texture classification on

the KTH-tips dataset with a small training set. When including all scales in the training set, the Riesz feature representation yields results comparable to those of the scattering network. However, when splitting training and testing set along scales, the Riesz feature representation turns out to be more robust.

Important benefits of scattering networks are that they avoid the time-consuming training process and parameter tuning as needed for CNNs and work well even for small training sets. Only 40 and 500 training images per class were needed for KTH-tips and MNIST Large Scale, respectively, to achieve accuracies over 0.9. The most important benefit of using the Riesz transform is however that robustness to scale variations is ensured without additional training.

All types of scattering networks including our approach rely on using only the amplitude to design feature representations, i.e. the phase is completely discarded. In one of the first works on scattering networks [114] it was argued that phase can be reconstructed from amplitude information by solving the so-called phase recovery problem [152]. However, phase recovery is a non-convex optimization problem. Hence, the question remains open, how to utilize the phase information in the framework of scattering networks and whether that would result in improved capabilities of this class of methods.

The main challenge in applying the Riesz feature representation to a wider range of problems (e.g. object detection) is to devise a scale equivariant bounding box algorithm. This is required for images with complex scenes which contain several objects belonging to different classes. A scale equivariant bounding box algorithm is hence subject of future work.

Finally, naturally, our representation can be used to create hybrid trainable representations by combining them with building blocks of deep neural networks, see e.g. [141, 139, 140].

Conclusion

This thesis deals with the design of methods for image analysis that satisfy the basic properties of human vision: translation, rotation, and scale invariance. These refer to the analysis of objects in the image independent of their position, orientation, and size relative to the observer or camera. Since it is easier to impose translation invariance than the remaining two, this thesis focuses on rotation (Part I) and scale (Part II) invariance separately.

Another important aspect of the thesis is a practical one. All the methods in the thesis were designed with a clear application in mind as demonstrated throughout the thesis. Engineering materials in combination with computed tomography (CT) offer endless possibilities to analyze oriented structures regardless of their orientation and size. Hence, the mulitdisciplinary aspect between mathematics and engineering science, e.g. civil engineering and materials science is very prominent.

Part I of the thesis deals with orientation independent filtering for various materials in 3d. The main goal is to design computationally affordable but robust methods for 3d. Since orientation space in 3d is harder to finely cover or sample in a reasonable time, we focused on a subset of the orientation space based on the orientation estimation from local image information. Hence, we designed 3d adaptive morphology and local shape operator to denoise lower dimensional oriented structures and distinguish them based on their shape/dimensionality. Later, we extend our method to 3d adaptive line granulometry for quantifying number weighted fiber length and orientation distribution from voxelwise measurements.

The main result of Part II is a scale invariant neural network called Riesz network for the segmentation of cracks in CT images of concrete. The Riesz network uses scale equivariant layers based on the Riesz transform to achieve generalization to the scales far outside of the scale distribution on which it was trained. This is shown practically by training a Riesz network on crack widths 3 and 5 (in pixels) and validated by segmenting cracks of more than 20 pixels width. All this can be achieved without sampling the scale dimension. These abilities make the method very attractive and unique in the field of deep learning. The Riesz transform has other useful properties related to signal decomposition and energy preservation. Hence, it is used to construct the first scale equivariant scattering network called Riesz feature representation. This type of non-trainable network works with very few training examples and achieves superior performance to scale variations than the original scattering network.

This thesis covers a wide range of topics, but many ideas introduced here open up space for further extensions and improvements. For example, orientation and scale invariance have been introduced separately in two different methods. A more general aim would be to design a single method that is both scale and orientation invariant. If additionally, it was translation invariant, it would reflect all the basic properties of human vision. Preliminary work on this in a classical manner interestingly already exists [78] through the combination of second order Riesz transform and Frangi filter, but somehow significance of this work remained sidelined in the general image processing community. The development of alternatives to this remains a goal for future

research. This would be an ideal solution for training deep models for a crack segmentation task in 3d CT images since it would require training data to cover only a few scales and orientations in order to achieve invariance over full scale and orientation dimensions.
Furthermore, our methods in Part I are based on orientation estimation from local image structure. However, the question of the optimal orientation estimator and how to design it was not studied in this thesis, both generally and in the context of our method. A general study on this already exists [7]. An interesting question related to our method is if it is possible to mathematically determine the optimal size of the search cone δ_{max} based on the level of noise that is present in the image. In the case of the Hessian matrix, this is related to the robustness of the calculation of the eigenvector with respect to various types and amounts of noise.
Another aspect is that while theoretical scale equivariance can be shown to hold for Riesz networks in Part II, it is not clear how effective it would be in other applications other than CT image, e.g. large visual databases such ImageNet. This is also due to the lack of datasets (and their quality) on which scale equvariance of the methods can be experimentally evaluated. However, small steps in the right direction are on the horizon [153]. Furthermore, the thesis does not answer the question of whether the Riesz transform is the only suitable or even optimal scale equivariant feature extractor that can be elegantly embedded in deep networks. Nevertheless, according to the author's belief applications of the Riesz transform in this thesis open up many interesting research paths and possibilities that the wider research community could benefit from.

Bibliography

[1] T. Barisin, K. Schladitz, C. Redenbach, M. Godehardt, Adaptive morphological framework for 3d directional filtering, Image Analysis & Stereology 0 (0), https://doi.org/10.5566/ias.2639 (2022).

[2] J. Lienhard, T. Barisin, H. Grimm-Strele, M. Kabel, K. Schladitz, T. Schweiger, Microstructure of 3d printed and molded glass fiber reinforced polypropylene and its influence on the mechanical properties, Strain, Accepted for publication (2023).

[3] T. Barisin, C. Redenbach, K. Schladitz, Riesz networks: scale invariant neural networks in single forward pass, Submitted: https://arxiv.org/pdf/2305.04665.pdf (2023).

[4] T. Barisin, J. Angulo, K. Schladitz, C. Redenbach, Riesz feature representation: scale equivariant scattering network for classification tasks, Submitted: https://arxiv.org/pdf/2307.08467.pdf (2023).

[5] O. Wirjadi, Models and algorithms for image-based analysis of microstructures, Ph.D. thesis, Technische Universität Kaiserslautern, https://kluedo.ub.uni-kl.de/frontdoor/index/index/year/2009/docId/2077 (2009).

[6] K. Schladitz, A. Büter, M. Godehardt, O. Wirjadi, J. Fleckenstein, T. Gerster, U. Hassler, K. Jaschek, M. Maisl, U. Maisl, S. Mohr, U. Netzelmann, T. Potyra, M. Steinhauser, Nondestructive characterization of fiber orientation in reinforced SMC as input for simulation based design, Composite Structures 160 (2017) 195 – 203, http://dx.doi.org/10.1016/j.compstruct.2016.10.019.

[7] O. Wirjadi, K. Schladitz, P. Easwaran, J. Ohser, Estimating fibre direction distributions of reinforced composites from tomographic images, Image Analysis and Stereology 35 (3) (2016) 167–179, http://dx.doi.org/10.5566/ias.1489.

[8] C. Sazak, C. J. Nelson, B. Obara, The multiscale bowler-hat transform for blood vessel enhancement in retinal images, Patter Recogn 88 (2019) 739–750, https://doi.org/10.1016/j.patcog.2018.10.011.

[9] W. T. Freeman, E. H. Adelson, The design and use of steerable filters, IEEE T Patter Anal 13 (9) (1991) 891–906, https://doi.org/10.1109/34.93808. doi:10.1109/34.93808.

[10] P. Soille, H. Talbot, Directional morphological filtering, IEEE T Patter Anal 23 (11) (2001) 1313–1329, https://doi.org/10.1109/34.969120. doi:10.1109/34.969120.

[11] F. Michelet, J.-P. Da Costa, P. Baylou, C. Germain, Local orientation estimation in corrupted images, in: N. Zheng, X. Jiang, X. Lan (Eds.), Lect Notes Comput Sc, Springer Berlin Heidelberg, Berlin, Heidelberg, 2006, pp. 349–358, https://doi.org/10.1007/11821045_37.

[12] K. Sandau, J. Ohser, The chord length transform and the segmentation of crossing fibres, J Microsc-Oxford 226 (1) (2007) 43–53, https://doi.org/10.1111/j.1365-2966.2007.01748.x.

[13] K. Sandberg, M. Brega, Segmentation of thin structures in electron micrographs using orientation fields, J Struct Biol 157 (2) (2007) 403–415, https://doi.org/10.1016/j.jsb.2006.09.007.

[14] K. Robb, O. Wirjadi, K. Schladitz, Fiber orientation estimation from 3d image data: Practical algorithms, visualization, and interpretation, in: Proc. 7th Int. Conf. Hybrid Intelligent Systems, Kaiserslautern, Germany, 2007, pp. 320–325, https://doi.org/10.1109/HIS.2007.26. doi:10.1109/HIS.2007.26.

[15] F. Semeraro, J. C. Ferguson, F. Panerai, R. J. King, N. N. Mansour, Anisotropic analysis of fibrous and woven materials part 1: Estimation of local orientation, Comp Mater Sci 178 (2020) 109631, https://doi.org/10.1016/j.commatsci.2020.109631.

[16] E. Saff, A. Kuijlaars, Distributing many points on a sphere., Mat Intel 19 (1997) 5–11, https://doi.org/10.1007/BF03024331.

[17] H. Altendorf, 3d morphological analysis and modeling of random fiber networks: applied on glass fiber reinforced composites, Ph.D. thesis, Technische Universität Kaiserslautern and Mines ParisTech, http://nbn-resolving.de/urn:nbn:de:hbz:386-kluedo-28323 (2011).

[18] J. Fliege, U. Maier, The distribution of points on the sphere and corresponding cubature formulae, IMA Journal of Numerical Analysis 19 (2) (1999) 317–334, https://doi.org/10.1093/imanum/19.2.317.

[19] R. Lerallut, É. Decencière, F. Meyer, Image filtering using morphological amoebas, Image and Vision Computing 25 (4) (2007) 395–404, lect Notes Comput Sc, https://doi.org/10.1016/j.imavis.2006.04.018.

[20] J. Debayle, J.-C. Pinoli, Spatially adaptive morphological image filtering using intrinsic structuring elements, Image Anal Stereol 24 (3) (2011) 145–158, https://doi.org/10.5566/ias.v24.p145-158.

[21] O. Tankyevych, H. Talbot, P. Dokládal, Curvilinear morpho-hessian filter, in: 2008 I S Biomed Imaging, 2008, pp. 1011–1014, https://doi.org/10.1109/ISBI.2008.4541170.

[22] O. Tankyevych, H. Talbot, P. Dokládal, N. Passat, Direction-adaptive grey-level morphology. application to 3d vascular brain imaging, in: IEEE Image Proc, 2009, pp. 2261–2264, https://doi.org/10.1109/ICIP.2009.5414356.

[23] A. Landström, M. J. Thurley, Adaptive morphology using tensor-based elliptical structuring elements, Pattern Recong Lett 34 (12) (2013) 1416–1422, https://doi.org/10.1016/j.patrec.2013.05.003.

[24] V. Curic, C. Luengo Hendriks, G. Borgefors, Salience adaptive structuring elements, IEEE J Sel Top Signa 6 (2012) 809–819, https://doi.org/10.1109/JSTSP.2012.2207371.

[25] H. Heijmans, M. Buckley, H. Talbot, Path openings and closings, J Math Imaging Vis 22 (2005) 107–119, https://doi.org/10.1007/s10851-005-4885-3.

[26] C. L. Luengo Hendriks, Constrained and dimensionality-independent path openings, IEEE T Image Process 19 (6) (2010) 1587–1595, https://doi.org/10.1109/TIP.2010.2044959.

[27] V. Morard, P. Dokládal, É. Decencière, Parsimonious path openings and closings, IEEE T Image Process 23 (4) (2014) 1543–1555, https://doi.org/10.1109/TIP.2014.2303647.

[28] R. Su, C. Sun, C. Zhang, T. D. Pham, A new method for linear feature and junction enhancement in 2d images based on morphological operation, oriented anisotropic gaussian function and hessian information, Pattern Recogn 47 (10) (2014) 3193–3208, https://doi.org/10.1016/j.patcog.2014.04.024.

[29] P. Dokládal, E. Dokládalova, Grey-scale morphology with spatially-variant rectangles in linear time, in: Lect Notes Comput Sc, 2008, pp. 674–685, https://doi.org/10.1007/978-3-540-88458-3_61.

[30] J. E. Bresenham, Algorithm for computer control of a digital plotter, IBM Systems Journal 4 (1) (1965) 25–30, https://doi.org/10.1147/sj.41.0025.

[31] C. Luengo Hendriks, Structure characterization using mathematical morphology, Ph.D. thesis, https://repository.tudelft.nl/islandora/object/uuid%3Ade60c1ea-b6de-4600-aa8a-885dedb145cb (2004).

[32] D. Eberly, R. Gardner, B. Morse, S. Pizer, C. Scharlach, Ridges for image analysis, J Math Imaging Vis 4 (4) (1994) 353–373, https://doi.org/10.1007/BF01262402.

[33] A. Frangi, W. Niessen, K. Vincken, M. Viergever, Multiscale vessel enhancement filtering, Med. Image Comput. Comput. Assist. Interv. 1496 (2000) 130–137, https://doi.org/10.1007/BFb0056195.

[34] J. Weickert, Coherence-enhancing diffusion filtering, Int. J. Comput. Vis. 31 (2-3) (1999) 111–127, https://doi.org/10.1023/A:1008009714131.

[35] R. Jones, P. Soille, Periodic lines: Definition, cascades, and application to granulometries, Pattern Recognition Letters 17 (10) (1996) 1057–1063, https://doi.org/10.1016/0167-8655(96)00066-9.

[36] P. Dokladal, E. Dokladalova, Grey-scale 1-d dilations with spatially-variant structuring elements in linear time, in: EUSIPCO'08, 2008, pp. 1–5, https://hal-upec-upem.archives-ouvertes.fr/hal-00622464.

[37] P. Dokladal, E. Dokladalova, Computationally efficient, one-pass algorithm for morphological filters, J Vis Commun Image R. 22 (5) (2011) 411–420, https://doi.org/10.1016/j.jvcir.2011.03.005.

[38] S. Perreault, P. Hebert, Median filtering in constant time, IEEE T Image Process 16 (9) (2007) 2389–2394, https://doi.org/10.1109/TIP.2007.902329.

[39] O. Wirjadi, M. Godehardt, K. Schladitz, B. Wagner, A. Rack, M. Gurka, S. Nissle, A. Noll, Characterization of multilayer structures in fiber reinforced polymer employing synchrotron and laboratory X-ray CT, Int J Mater Res 105 (7) (2014) 645–654, https://doi.org/doi:10.3139/146.111082.

[40] D. Dresvyanskiy, T. Karaseva, V. Makogin, S. Mitrofanov, C. Redenbach, E. Spodarev, Detecting anomalies in fibre systems using 3-dimensional image data, Stat Comput 30 (2020) 1–21, https://doi.org/10.1007/s11222-020-09921-1.

[41] J. Sliseris, H. Andrä, M. Kabel, O. Wirjadi, B. Dix, B. Plinke, Estimation of fiber orientation and fiber bundles of MDF, Mater Struct 49, https://doi.org/10.1617/s11527-015-0769-1 (12 2015).

[42] T. Barisin, C. Jung, F. Müsebeck, C. Redenbach, K. Schladitz, Methods for segmenting cracks in 3d images of concrete: A comparison based on semi-synthetic images, Pattern Recognition 129 (2022) 108747, https://doi.org/10.1016/j.patcog.2022.108747.

[43] P. Soille, Morphological Image Analysis: Principles and Applications, Springer, Berlin, Heidelberg, 1999, https://doi.org/10.1007/978-3-662-03939-7.

[44] K. Ehrig, J. Goebbels, D. Meinel, O. Paetsch, S. Prohaska, V. Zobel, Comparison of Crack Detection Methods for Analyzing Damage Processes in Concrete with Computed Tomography, in: Int. Symp. Dig. Ind. Radiol. Comp. Tomogr., 2011, https://www.ndt.net/article/dir2011/papers/p2.pdf.

[45] C. Redenbach, M. Giertzsch, M. Godehardt, K. Schladitz, Modelling a ceramic foam using locally adaptable morphology, J Microsc-Oxford 230 (2008) 396–404, https://doi.org/10.1111/j.1365-2818.2008.01998.x.

[46] C. Redenbach, O. Wirjadi, S. Rief, A. Wiegmann, Modeling of ceramic foams for filtration simulation, Adv Eng Mater 13 (2011) 171 – 177, https://doi.org/10.1002/adem.201000222.

[47] J. Kampf, A.-L. Schlachter, C. Redenbach, A. Liebscher, Segmentation, statistical analysis, and modelling of the wall system in ceramic foams, Mater Charact 99 (2015) 38–46, https://doi.org/10.1016/j.matchar.2014.11.008.

[48] S. Föhst, S. Osterroth, F. Arnold, C. Redenbach, Influence of geometry modifications on the permeability of open-cell foams, AIChE Journal 68 (2) (2021) e17446, https://doi.org/10.1002/aic.17446.

[49] I. Svensson, S. Sjöstedt-de Luna, L. Bondesson, Estimation of wood fibre length distributions from censored data through an EM algorithm, Scandinavian Journal of Statistics 33 (3) (2006) 503–522, https://doi.org/10.1111/j.1467-9469.2006.00501.x.

[50] J. Viguié, P. Latil, L. Orgéas, P. Dumont, S. R. du Roscoat, J.-F. Bloch, C. Marulier, O. Guiraud, Finding fibres and their contacts within 3d images of disordered fibrous media, Composites Science and Technology 89 (2013) 202 – 210, https://doi.org/10.1016/j.compscitech.2013.09.023.

[51] P. Pinter, B. Bertram, K. A. Weidenmann, A novel method for the determination of fiber length distributions from µCT-data, in: Proceedings of 6th Conference on Industrial Computed Tomography, Feb 9-12, 2016, Wels, Austria, 2016, https://www.ndt.net/article/ctc2016/papers/ICT2016_paper_id71.pdf.

[52] M. Tessmann, S. Mohr, S. Gayetskyy, U. Hassler, R. Hanke, G. Greiner, Automatic determination of fiber-length distribution in composite material using 3D CT data, EURASIP Journal on Advances in Signal Processing 2010 (2010) 1–9, https://doi.org/10.1155/2010/545030.

[53] M. Emerson, A. Dahl, K. Conradsen, V. Dahl, Insegt Fibre: a user-friendly software for individual fibre segmentation, in: Proceedings of 22nd International Conference on Composites Materials, 2019, https://core.ac.uk/download/pdf/226961642.pdf.

[54] J. Weissenböck, A. Amirkhanov, W. Li, A. Reh, A. Amirkhanov, E. Gröller, J. Kastner, C. Heinzl, Fiberscout: An interactive tool for exploring and analyzing fiber reinforced polymers, in: 2014 IEEE Pacific Visualization Symposium, 2014, pp. 153–160, https://doi.org10.1109/PacificVis.2014.52.

[55] M. Kronenberger, K. Schladitz, B. Hamann, H. Hagen, Fiber segmentation in crack regions of steel fiber reinforced concrete using principal curvature, Image Analysis & Stereology 37 (2) (2018) 127–137, https://doi.org/10.5566/ias.1914.

[56] M. Kuhlmann, C. Redenbach, Estimation of fibre length distributions from fibre endpoints, Scandinavian Journal of Statistics 42 (4) (2015) 1010–1022, https://doi.org/10.1111/sjos.12148.

[57] J. Niedermeyer, Estimating the fibre length distribution in fibre reinforced polymers, Phd thesis, Technische Universität Kaiserslautern, http://nbn-resolving.de/urn:nbn:de:hbz:386-kluedo-60654 (2020).

[58] A. Senior Bechara, T. Osswald, Measuring fiber length in the core and shell regions of injection molded long fiber-reinforced thermoplastic plaques, Journal of Composites Science 4 (3), https://doi.org/10.3390/jcs4030104 (2020).

[59] G. Matheron, Random sets and integral geometry, John Wiley & Sons, 1974.

[60] C. L. L. Hendriks, L. J. Van Vliet, A rotation-invariant morphology for shape analysis of anisotropic objects and structures, in: Visual Form 2001: 4th International Workshop on Visual Form, IWVF4 Capri, Italy, May 28–30, 2001 Proceedings, Springer, 2001, pp. 378–387, https://doi.org/10.1007/3-540-45129-3_34.

[61] C. L. Hendriks, L. J. van Vliet, Using line segments as structuring elements for sampling-invariant measurements, IEEE transactions on pattern analysis and machine intelligence 27 (11) (2005) 1826–1831, https://doi.org/10.1109/TPAMI.2005.228.

[62] A. Miettinen, C. L. Luengo Hendriks, G. Chinga-Carrasco, E. K. Gamstedt, M. Kataja, A non-destructive X-ray microtomography approach for measuring fibre length in short-fibre composites, Composites Science and Technology 72 (15) (2012) 1901–1908, https://doi.org/10.1016/j.compscitech.2012.08.008.

[63] J. Ohser, K. Schladitz, 3d Images of Materials Structures – Processing and Analysis, Wiley VCH, Weinheim, 2009.

[64] N. Fisher, T. Lewis, B. Embleton, Statistical Analysis of Spherical Data, Cambridge University Press, Cambridge, UK, 1987.

[65] M. Riesz, L'intégrale de Riemann-Liouville et le problème de Cauchy, Acta Mathematica 81 (1949) 1 – 222, https://doi.org/10.1007/BF02395016.

[66] E. M. Stein, Singular integrals and differentiability properties of functions, Vol. 2, Princeton university press, 1970.

[67] A. Schikorra, D. Spector, J. Van Schaftingen, An $\dot{\ell}1$-type estimate for riesz potentials, Revista Matemática Iberoamericana 33 (1) (2017) 291–303, ,https://arxiv.org/pdf/1411.2318.pdf.

[68] Y. Lecun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, Proceedings of the IEEE 86 (11) (1998) 2278–2324, https://doi.org/10.1109/5.726791.

[69] S. Bernstein, J.-L. Bouchot, M. Reinhardt, B. Heise, Generalized Analytic Signals in Image Processing: Comparison, Theory and Applications, 2013, pp. 221–246. doi:10.1007/978-3-0348-0603-9_11.

[70] D. Gabor, Theory of communication. part 1: The analysis of information, Journal of the Institution of Electrical Engineers-part III: radio and communication engineering 93 (26) (1946) 429–441.

[71] M. Felsberg, G. Sommer, The monogenic signal, IEEE Transactions on Signal Processing 49 (12) (2001) 3136–3144, https://doi.org/10.1109/78.969520.

[72] M. Felsberg, Low-level image processing with the structure multivector, Ph.D. thesis, Selbstverlag des Instituts für Informatik, Kiel, https://macau.uni-kiel.de/receive/macau_mods_00001925 (2002).

[73] M. Felsberg, G. Sommer, Scale adaptive filtering derived from the laplace equation, in: Pattern Recognition, 2001, pp. 124–131, https://doi.org/10.1007/3-540-45404-7_17.

[74] M. Felsberg, G. Sommer, The monogenic scale-space: A unifying approach to phase-based image processing in scale-space, Journal of Mathematical Imaging and Vision 21 (2004) 5–26, https://doi.org/10.1023/B:JMIV.0000026554.79537.35.

[75] U. Köthe, M. Felsberg, Riesz-transforms vs. derivatives: on the relationship between the boundary tensor and the energy tensor, Proc. Scale Space Conference (this, Springer) (2005) 179–191, https://doi.org/10.1007/11408031_16.

[76] M. Unser, D. Van De Ville, Wavelet steerability and the higher-order Riesz transform, IEEE Transactions on Image Processing 19 (3) (2010) 636–652, https://doi.org/10.1109/TIP.2009.2038832.

[77] L. Wietzke, G. Sommer, C. Schmaltz, J. Weickert, Differential geometry of monogenic signal representations, in: G. Sommer, R. Klette (Eds.), Robot Vision, Springer Berlin Heidelberg, Berlin, Heidelberg, 2008, pp. 454–465, https://doi.org/10.1007/978-3-540-78157-8_35.

[78] D. Dobrovolskij, J. Persch, K. Schladitz, G. Steidl, Structure detection with second order Riesz transforms, Image Analysis & Stereology 38 (2019) 107, https://doi.org/10.5566/ias.1964.

[79] L. Zhang, L. Zhang, X. Mou, Rfsim: A feature based image quality assessment metric using Riesz transforms, in: 2010 IEEE International Conference on Image Processing, 2010, pp. 321–324, https://doi.org/10.1109/ICIP.2010.5649275.

[80] A. Depeursinge, A. Foncubierta-Rodriguez, D. Van de Ville, H. Müller, Multiscale lung texture signature learning using the Riesz transform, in: N. Ayache, H. Delingette,

P. Golland, K. Mori (Eds.), Medical Image Computing and Computer-Assisted Intervention – MICCAI 2012, Springer Berlin Heidelberg, Berlin, Heidelberg, 2012, pp. 517–524, https://doi.org/10.1007/978-3-642-33454-2_64.

[81] K. Langley, S. J. Anderson, The riesz transform and simultaneous representations of phase, energy and orientation in spatial vision, Vision Research 50 (17) (2010) 1748–1765, https://doi.org/10.1016/j.visres.2010.05.031.

[82] M. Reinhardt, S. Bernstein, B. Heise, Multi-scale orientation estimation using higher order Riesz transforms, International Journal of Wavelets, Multiresolution and Information Processing 20 (03) (2022) 2040007, https://doi.org/10.1142/S021969132040007X.

[83] S. C. Olhede, G. Metikas, The monogenic wavelet transform, IEEE Transactions on Signal Processing 57 (9) (2009) 3426–3441, https://doi.org/10.1109/TSP.2009.2023397.

[84] S. Held, M. Storath, P. Massopust, B. Forster, Steerable wavelet frames based on the riesz transform, IEEE Transactions on Image Processing 19 (3) (2010) 653–667, https://doi.org/10.1109/TIP.2009.2036713.

[85] M. Unser, D. Sage, D. Van De Ville, Multiresolution monogenic signal analysis using the Riesz–Laplace wavelet transform, IEEE transactions on image processing 18 (2009) 2402–18, https://doi.org/10.1109/TIP.2009.2027628.

[86] S. Häuser, B. Heise, G. Steidl, Linearized Riesz transform and quasi-monogenic shearlets, International Journal of Wavelets, Multiresolution and Information Processing 12 (03) (2014) 1450027, https://doi.org/10.1142/S0219691314500271.

[87] R. Joyseeree, J. Otálora Montenegro, H. Müller, A. Depeursinge, Fusing learned representations from riesz filters and deep cnn for lung tissue classification, Medical Image Analysis 56, https://doi.org/10.1016/j.media.2019.06.006 (06 2019).

[88] T. Lindeberg, Feature detection with automatic scale selection, International Journal of Computer Vision 30 (1998) 77–116, https://doi.org/10.1023/A:1008045108935.

[89] J. Weickert, S. Ishikawa, A. Imiya, Linear scale-space has first been proposed in japan, Journal of Mathematical Imaging and Vision 10 (3) (1999) 237–252, https://doi.org/10.1023/A:1008344623873.

[90] A. P. Witkin, Scale-space filtering, in: Readings in Computer Vision, Elsevier, 1987, pp. 329–332, https://doi.org/10.1016/B978-0-08-051581-6.50036-2.

[91] J. J. Koenderink, The structure of images, Biological cybernetics 50 (5) (1984) 363–370, https://doi.org/10.1007/BF00336961.

[92] T. Lindeberg, Scale-space theory: A basic tool for analyzing structures at different scales, Journal of applied statistics 21 (1-2) (1994) 225–270, https://doi.org/10.1080/757582976.

[93] R. Duits, M. Felsberg, L. Florack, B. Platel, α scale spaces on a bounded domain, in: International Conference on Scale-Space Theories in Computer Vision, Springer, 2003, pp. 494–510, https://doi.org/10.1007/3-540-44935-3_34.

[94] R. Duits, L. Florack, J. De Graaf, B. ter Haar Romeny, On the axioms of scale space theory, Journal of Mathematical Imaging and Vision 20 (3) (2004) 267–298, https://doi.org/10.1023/B:JMIV.0000024043.96722.aa.

[95] T. Iijima, Basic theory of pattern observation, Technical Group on Automata and Automatic Control (1959) 3–32.

[96] D. G. Lowe, Object recognition from local scale-invariant features, in: Proceedings of the seventh IEEE international conference on computer vision, Vol. 2, 1999, pp. 1150–1157, https://doi.org/10.1109/ICCV.1999.790410.

[97] T. Lindeberg, Image matching using generalized scale-space interest points, Journal of mathematical Imaging and Vision 52 (2015) 3–36, https://doi.org/10.1007/s10851-014-0541-0.

[98] A. Kanazawa, A. Sharma, D. Jacobs, Locally scale-invariant convolutional neural networks, Deep Learning and Representation Learning Workshop: Neural Information Processing System, https://arxiv.org/pdf/1412.5104.pdf (2014).

[99] Y. Jansson, T. Lindeberg, Scale-invariant scale-channel networks: Deep networks that generalise to previously unseen scales, J Math Imaging Vis 64 (5) (2022) 506–536, https://doi.org/10.1007/s10851-022-01082-2.

[100] C. Jung, F. Müsebeck, T. Barisin, K. Schladitz, C. Redenbach, M. Kiesche, M. Pahn, Towards automatic crack segmentation in 3d concrete images, in: 11th Conference on Industrial Computed Tomography, Wels, Austria (iCT 2022), 2022, https://www.ndt.net/article/ctc2022/papers/ICT2022_paper_id225.pdf.

[101] A. Ess, B. Leibe, K. Schindler, , L. van Gool, A mobile vision system for robust multi-person tracking, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR'08), IEEE Press, 2008, https://doi.org/10.1109/CVPR.2008.4587581.

[102] C. Jung, C. Redenbach, Crack modeling via minimum-weight surfaces in 3d Voronoi diagrams https://arxiv.org/abs/2210.05093 (2022).

[103] D. Marcos, B. Kellenberger, S. Lobry, D. Tuia, Scale equivariance in cnns with vector fields, ICML, https://arxiv.org/abs/1807.11783 (2018).

[104] Z. Cai, Q. Fan, R. S. Feris, N. Vasconcelos, A unified multi-scale deep convolutional neural network for fast object detection, in: B. Leibe, J. Matas, N. Sebe, M. Welling (Eds.), Computer Vision – ECCV 2016, Springer International Publishing, Cham, 2016, pp. 354–370, https://doi.org/10.1007/978-3-319-46493-0_22.

[105] T.-Y. Lin, P. Dollar, R. Girshick, K. He, B. Hariharan, S. Belongie, Feature pyramid networks for object detection, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, https://openaccess.thecvf.com/content_cvpr_2017/papers/Lin_Feature_Pyramid_Networks_CVPR_2017_paper.pdf.

[106] Y. Xu, T. Xiao, J. Zhang, K. Yang, Z. Zhang, Scale-invariant convolutional neural networks, https://arxiv.org/pdf/1411.6369.pdf (11 2014).

[107] M. Jaderberg, K. Simonyan, A. Zisserman, K. Kavukcuoglu, Spatial transformer networks, in: Advances in Neural Information Processing Systems, Vol. 28, 2015, https://proceedings.neurips.cc/paper/2015/file/33ceb07bf4eeb3da587e268d663aba1a-Paper.pdf.

[108] L. Finnveden, Y. Jansson, T. Lindeberg, Understanding when spatial transformer networks do not support invariance, and what to do about it, in: 2020 25th International Conference on Pattern Recognition (ICPR), 2021, pp. 3427–3434, https://doi.org/10.1109/ICPR48806.2021.9412997.

[109] J.-H. Jacobsen, J. Gemert, Z. Lou, A. Smeulders, Structured receptive fields in cnns, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 2610–2619, https://openaccess.thecvf.com/content_cvpr_2016/papers/Jacobsen_Structured_Receptive_Fields_CVPR_2016_paper.pdf.

[110] F. Yu, V. Koltun, Multi-scale context aggregation by dilated convolutions, 2016, https://arxiv.org/abs/1511.07122.

[111] D. Worrall, M. Welling, Deep scale-spaces: Equivariance over scale, in: Advances in Neural Information Processing Systems, Vol. 32, 2019, https://proceedings.neurips.cc/paper/2019/file/f04cd7399b2b0128970efb6d20b5c551-Paper.pdf.

[112] M. Sangalli, S. Blusseau, S. Velasco-Forero, J. Angulo, Scale equivariant neural networks with morphological scale-spaces, in: J. Lindblad, F. Malmberg, N. Sladoje (Eds.), Discrete Geometry and Mathematical Morphology, 2021, pp. 483–495, https://doi.org/10.1007/978-3-030-76657-3_35.

[113] I. Sosnovik, M. Szmaja, A. Smeulders, Scale-equivariant steerable networks, in: International Conference on Learning Representations, 2020, https://doi.org/10.48550/arXiv.1910.11093.

[114] J. Bruna, S. Mallat, Invariant scattering convolution networks, IEEE transactions on pattern analysis and machine intelligence 35 (2013) 1872–1886, https://doi.org/10.1109/TPAMI.2012.230.

[115] L. Sifre, S. Mallat, Rotation, scaling and deformation invariant scattering for texture discrimination, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2013, https://doi.org/10.1109/CVPR.2013.163.

[116] T. Lindeberg, Provably scale-covariant continuous hierarchical networks based on scale-normalized differential expressions coupled in cascade, Journal of Mathematical Imaging and Vision 62 (1) (2020) 120–148, https://doi.org/10.1007/s10851-019-00915-x.

[117] T. Lindeberg, Scale-covariant and scale-invariant gaussian derivative networks, Journal of Mathematical Imaging and Vision 64 (3) (2022) 223–242, https://doi.org/10.1007/s10851-021-01057-9.

[118] S. Ioffe, C. Szegedy, Batch normalization: Accelerating deep network training by reducing internal covariate shift, http://proceedings.mlr.press/v37/ioffe15.pdf (02 2015).

[119] D. Kingma, J. Ba, Adam: A method for stochastic optimization, in: ICLR, 2014, https://arxiv.org/abs/1412.6980.

[120] O. Ronneberger, P. Fischer, T. Brox, U-net: Convolutional networks for biomedical image segmentation, in: Lect. Notes Comput. Sc., 2015, pp. 234–241, https://doi.org/10.1007/978-3-319-24574-4_28.

[121] K. Maryamh, K. Hauch, C. Redenbach, J. Schnell, Influence of specimen size on the fibre geometry and tensile strength of ultra-high-performance fibre-reinforced concrete, Structural Concrete 23 (2) (2022) 1239–1252, https://doi.org/10.1002/suco.202000753.

[122] K. Maryamh, K. Hauch, C. Redenbach, J. Schnell, Influence of production parameters on the fiber geometry and the mechanical behavior of ultra high performance fiber-reinforced concrete, Structural Concrete 22 (1) (2021) 361–375, https://doi.org/10.1002/suco.202000105.

[123] K. Hauch, K. Maryamh, C. Redenbach, J. Schnell, Predicting the tensile behaviour of ultra-high performance fibre-reinforced concrete from single-fibre pull-out tests, Materials 15 (14) (2022) 5085, https://doi.org/10.3390/ma15145085.

[124] C. Jung, A. Nowacka, T. Barisin, D. Meinel, O. Paetsch, S. Grzesiak, M. Salamon, K. Schladitz, C. Redenbach, M. Pahn, 3d imaging and analysis of cracks in loaded concrete samples, in: 12th Conference on Industrial Computed Tomography, Furth, Germany (iCT 2023), 2023, https://www.ndt.net/article/ctc2023/papers/Contribution_124_final.pdf.

[125] F. Schuler, Richtungsanalyse von fasern in beton und charakterisierung von rissquerenden fasern mittels computer-tomografie, Ph.D. thesis, https://kluedo.ub.uni-kl.de/frontdoor/index/index/docId/6204 (2020).

[126] T. Barisin, C. Jung, A. Nowacka, K. Schladitz, C. Redenbach, Crack segmentation in 3d concrete images: perspectives and challenges, in: International Symposium on Non-Destructive Testing in Civil Engineering (NDT-CE 2022), 2022, https://www.ndt.net/article/ndtce2022/paper/60992_manuscript.pdf.

[127] O. Paetsch, D. Baum, K. Ehrig, D. Meinel, S. Prohaska, Automated 3D Crack Detection for Analyzing Damage Processes in Concrete with Computed Tomography, in: 4th Conference on Industrial Computed Tomography (iCT), 2012, https://www.ndt.net/article/ctc2012/papers/339.pdf.

[128] O. Paetsch, Possibilities and Limitations of Automated 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, https://www.ndt.net/article/ctc2019/papers/iCT2019_Full_paper_29.pdf.

[129] Ö. Çiçek, A. Abdulkadir, S. S. Lienkamp, T. Brox, O. Ronneberger, 3D U-Net: Learning dense volumetric segmentation from sparse annotation, in: S. Ourselin, L. Joskowicz, M. R. Sabuncu, G. Unal, W. Wells (Eds.), Medical Image Computing and Computer-Assisted Intervention – MICCAI 2016, 2016, pp. 424–432, https://doi.org/10.1007/978-3-319-46723-8_49.

[130] Master Builder Solutions, MasterFiber 235 SPA Data Sheet: The High-Performance Polypropylene Fiber, Class II EN 14889-2, straßfurt, Germany, 2019.

[131] EN 14651:2005+A1:2007, Test Method for Metallic Fiber Concrete-Measuring the Flexural Tensile Strength (Limit or Proportionality (LOP), Residual), Beuth Verlag GmbH, Berlin, Germany, 2007.

[132] M. Sangalli, S. Blusseau, S. Velasco-Forero, J. Angúlo, Differential invariants for se(2)-equivariant networks, in: 2022 IEEE International Conference on Image Processing (ICIP), IEEE, 2022, pp. 2216–2220, https://doi.org/10.1109/ICIP46576.2022.9897301.

[133] M. Sangalli, S. Blusseau, S. Velasco-Forero, J. Angulo, Moving frame net: Se (3)-equivariant network for volumes, in: NeurIPS 2022 Workshop on Symmetry and Geometry in Neural Representations, https://proceedings.mlr.press/v197/sangalli23a.html.

[134] S. Mallat, Group invariant scattering, Communications on Pure and Applied Mathematics 65 (10) (2012) 1331–1398, https://doi.org/10.1002/cpa.21413.

[135] D. Boukerroui, J. Noble, M. Brady, On the choice of bandpass quadrature filters, Journal of Mathematical Imaging and Vision - JMIV 21 (2004) 53–80, https://doi.org/10.1023/B:JMIV.0000026557.50965.09.

[136] J. Bruna, S. Mallat, Classification with scattering operators, in: CVPR 2011, 2011, pp. 1561–1566, https://doi.org/10.1109/CVPR.2011.5995635.

[137] E. Oyallon, S. Mallat, Deep roto-translation scattering for object classification, in: 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015, pp. 2865–2873, https://doi.org/10.1109/CVPR.2015.7298904.

[138] F. Cotter, N. Kingsbury, Visualizing and improving scattering networks, in: 2017 IEEE 27th International Workshop on Machine Learning for Signal Processing (MLSP), IEEE, 2017, pp. 1–6, https://doi.org/10.1109/MLSP.2017.8168136.

[139] E. Oyallon, E. Belilovsky, S. Zagoruyko, Scaling the scattering transform: Deep hybrid networks, in: Proceedings of the IEEE international conference on computer vision, 2017, pp. 5618–5627, https://openaccess.thecvf.com/content_ICCV_2017/papers/Oyallon_Scaling_the_Scattering_ICCV_2017_paper.pdf.

[140] E. Oyallon, S. Zagoruyko, G. Huang, N. Komodakis, S. Lacoste-Julien, M. Blaschko, E. Belilovsky, Scattering networks for hybrid representation learning, IEEE transactions on pattern analysis and machine intelligence 41 (9) (2018) 2208–2221, https://doi.org/10.1109/TPAMI.2018.2855738.

[141] F. Cotter, N. Kingsbury, A learnable scatternet: Locally invariant convolutional layers, in: 2019 IEEE International Conference on Image Processing (ICIP), IEEE, 2019, pp. 350–354, https://doi.org/10.1109/ICIP.2019.8802977.

[142] S. Gauthier, B. Thérien, L. Alsène-Racicot, M. Chaudhary, I. Rish, E. Belilovsky, M. Eickenberg, G. Wolf, Parametric scattering networks, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022, pp. 5749–5758, https://openaccess.thecvf.com/content/CVPR2022/papers/Gauthier_Parametric_Scattering_Networks_CVPR_2022_paper.pdf.

[143] S. Mallat, Understanding deep convolutional networks, Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences 374 (2065) (2016) 20150203, https://doi.org/10.1098/rsta.2015.0203.

[144] T. Wiatowski, H. Bölcskei, A mathematical theory of deep convolutional neural networks for feature extraction, IEEE Transactions on Information Theory PP, https://doi.org/10.1109/TIT.2017.2776228 (12 2015).

[145] W. H. Chak, N. Saito, Monogenic wavelet scattering network for texture image classification, https://arxiv.org/abs/2202.12491 (2022).

[146] M. Reinhardt, S. Bernstein, J. Richter, Rock classification with features based on higher order riesz transform, Advances in Applied Clifford Algebras 32 (5) (2022) 59, https://doi.org/10.1007/s00006-022-01237-9.

[147] M. Fritz, E. Hayman, B. Caputo, J.-O. Eklundh, The kth-tips database, https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=e00cb4517022420d1bbd03a0e99a96467ce91528 (2004).

[148] E. Hayman, B. Caputo, M. Fritz, J.-O. Eklundh, On the significance of real-world conditions for material classification, in: European conference on computer vision, Springer, 2004, pp. 253–266, https://doi.org/10.1007/978-3-540-24673-2_21.

[149] M. Andreux, T. Angles, G. Exarchakis, R. Leonarduzzi, G. Rochette, L. Thiry, J. Zarka, S. Mallat, J. Andén, E. Belilovsky, et al., Kymatio: Scattering transforms in python., J. Mach. Learn. Res. 21 (60) (2020) 1–6, http://jmlr.org/papers/v21/19-047.html.

[150] J. Bruna, Scattering representations for recognition, Ph.D. thesis, Ecole Polytechnique, https://pastel.archives-ouvertes.fr/file/index/docid/905109/filename/phdmain_final.pdf (2013).

[151] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay, Scikit-learn: Machine learning in Python, Journal of Machine Learning Research 12 (2011) 2825–2830, http://jmlr.org/papers/v12/pedregosa11a.html.

[152] I. Waldspurger, A. d'Aspremont, S. Mallat, Phase recovery, maxcut and complex semidefinite programming, Mathematical Programming 149, https://doi.org/10.1007/s10107-013-0738-9 (06 2012).

[153] T. Altstidl, A. Nguyen, L. Schwinn, F. Köferl, C. Mutschler, B. Eskofier, D. Zanca, Just a matter of scale? reevaluating scale equivariance in convolutional neural networks, https://arxiv.org/abs/2211.10288 (2022).

[154] Y. Bengio, I. Goodfellow, A. Courville, Deep learning, Vol. 1, MIT press Cambridge, MA, USA, 2017.

[155] J. Canny, A computational approach to edge detection, IEEE Transactions on pattern analysis and machine intelligence (6) (1986) 679–698, https://doi.org/10.1109/TPAMI.1986.4767851.

[156] R. Deriche, Using canny's criteria to derive a recursively implemented optimal edge detector, International journal of computer vision 1 (2) (1987) 167–187, https://doi.org/10.1007/BF00123164.

[157] C. Ronse, The phase congruence model for edge detection in two-dimensional pictures: A mathematical study, Rapport de Recherche (1871) (1995).

[158] S. Grzesiak, T. Barisin, K. Schladitz, M. Pahn, Analysis of the bond behaviour of a gfrp rebar in concrete by in-situ 3d imaging test, Materials and Structures, Accepted for publication (2023).

Appendix

A Definition of the basic building blocks in deep learning

The basic idea behind neural networks can be summarized in a quote from [154]: "A motivation for the family of functions defined by multi-layer neural networks is to compose simple transformations in order to obtain highly non-linear ones." This book [154] also covers the basic definitions of these simple building blocks. We here aim to give more rigorous definitions.

Convolution operator on a single channel: Generally, the convolution operator combines two functions: the input function with the weighting function or kernel. For point $x \in \mathbb{R}^d$ in the domain, the weighting function gives weight to $f(y)$ for every point $y \in \mathbb{R}^d$ based on its relative position to x. It is useful for extracting linear features. The convolution operator $* : \mathbb{R}^{\mathbb{R}^d} \times \mathbb{R}^{\mathbb{R}^d} \to \mathbb{R}^{\mathbb{R}^d}$ is defined as

$$(f * g)(x) = \int_{\mathbb{R}^d} f(u)g(x-u)du = \int_{\mathbb{R}^d} f(x-u)g(u)du.$$

Discrete convolution operator on a single channel: Images are discretized versions of reality. Hence, their domain is a subset of $\mathbb{Z}^2$. For that reason, the convolution operator needs to be discretized for $x \in \mathbb{Z}^d$

$$(f * g)(x) = \sum_{k \in \mathbb{Z}^d} f(k)g(x-k) = \sum_{k \in \mathbb{Z}^d} f(x-k)g(k).$$

Machine and deep learning models are based on learning the kernel function g through the optimization algorithm to achieve the optimal feature representation. This step is also known as training

1d convolution: 1d convolution is also known as a fully connected layer or 1×1 convolution and it is calculated as a scalar product operator across multiple channels without any spatial averaging. Neural networks extract several feature maps from a single input image. 1d convolution combines information across channels creating more complex features. Formally, let $n_{input} \in \mathbb{N}$ be a number of input channels, then $\phi_{conv1d} : (\mathbb{R}^{\mathbb{Z}^d})^{n_{input}} \times \mathbb{R}^{n_{input}} \to \mathbb{R}^{\mathbb{Z}^d}$ is defined using a weight vector $w = (w_1, \cdots, w_{n_{input}}) \in \mathbb{R}^{n_{input}}$

$$\phi_{conv1d}(f, w)(x) = \langle f(x, \cdot), w \rangle = w^T f(x, \cdot) = \sum_{k=1}^{n_{input}} f(x, k)w(k).$$

The weight vector w serves as degrees of freedom or trainable parameters which will be inferred during the training process. From the equation from above, the name 1d convolution may be

misleading. However, this operator can be written as the sum of the convolutional operators with a fixed function $g \in \mathbb{R}^{\mathbb{Z}^d}$ across channel dimension, where $g(x) = \mathbb{1}_0(x)$ i.e. $g(0) = 1$ and $g(x) = 0$ if $x \neq 1$:

$$\phi_{conv1d}(f, w)(x) = \langle (f * \mathbb{1}_0)(x, \cdot), w \rangle = w^T \Big((f * \mathbb{1}_0)(x, \cdot) \Big) = \sum_{k=1}^{n_{input}} \Big(w(k) \sum_{y \in \mathbb{Z}^d} f(y, k) \mathbb{1}_0(x - y) \Big).$$

For this reason, this is a special case of the generalization of the discrete convolutional operator from a single channel ($n_{input} = 1$) to multiple channels ($n_{input} > 1$).
Generally, the output of the transformation is not a single channel but rather many channels, i.e. $n_{output} \in \mathbb{N}$ channels. In this case the vector w becomes a matrix $W \in \mathbb{R}^{n_{input} \times n_{output}}$ and the scalar product is replaced with matrix-vector multiplication. Formally, the whole multichannel 1d convolution can be elegantly defined through the operator $\phi^{multi}_{conv1d} : (\mathbb{R}^{\mathbb{Z}^d})^{n_{input}} \times (\mathbb{R}^{n_{input} \times n_{output}}) \to (\mathbb{R}^{\mathbb{Z}^d})^{n_{output}}$

$$\phi^{multi}_{conv1d}(f, W)(x) = W^T \Big((f * \mathbb{1}_0)(x, \cdot) \Big).$$

2d/3d convolution: 2d/3d convolution extends 1d convolutions to the spatial domain $\mathbb{Z}^d$. This is useful for learning kernels for feature extractions. Learned kernels replace hand-crafted features and outperform them. Formally, let $n_{input}, n_{output} \in \mathbb{N}$ be the number of the input and the output channels, respectively, and let $H_l = [-\frac{(l-1)}{2}, \frac{(l-1)}{2}]^d \subset \mathbb{Z}^d$ be a square window of edge length l centered at 0. Then, for point $x \in \mathbb{Z}^d$ the operator $\phi_{conv2d} : (\mathbb{R}^{\mathbb{Z}^d})^{n_{input}} \times (\mathbb{R}^{n_{input} \times n_{output}})^{\mathbb{Z}^d} \to (\mathbb{R}^{\mathbb{Z}^d})^{n_{output}}$ can be defined as a sum of 1d convolutions on the square window $x + H_l = \{x + y | y \in H_l\}$ with edge length $l \in \mathbb{N}$ centered at x.

$$\phi_{conv2d}(f, W)(x) = \sum_{y \in H_l} \phi^{multi}_{conv1d}(f, W(y))(x - y) = \sum_{y \in H_l} W(y)^T \Big((f * \mathbb{1}_0)(x - y, \cdot) \Big).$$

Here, $W \in (\mathbb{R}^{n_{input} \times n_{output}})^{\mathbb{Z}^d}$ gets an additional spatial dimension based on the size of the square window l and consequently the number of parameters increases proportionally to l.
2d/3d convolutions are not well defined at the edge pixels of the images. Therefore, either convolutions are not applied to edge pixels reducing the image size by $l - 1$ or either null or reflective padding can be applied to preserve the original input image dimensions.
2d/3d convolutions are usually defined as local operators on window sizes 3×3 or 5×5 in 2d, and on window size $3 \times 3 \times 3$ or $5 \times 5 \times 5$ in 3d. The reason for this is to extract local features and to keep the number of parameters as low as possible. One can extend the size of windows and learn non-local features by cascading 2d/3d convolutions in sequences. An example of this is shown in Figure 5. Here, five 3×3 convolutions are needed to cover the window of size 11×11 which results in $5 \cdot 3 \cdot 3 = 45$ parameters. If we decide for simple 11×11 convolution, it would contain in total 121 parameters, which is more than two times more than in the first case. By combining 3×3 with max pooling and downsampling one could further reduce the number of 2d convolutions to reach this window size and consequently the number of parameters (see next paragraph). This gives a motivation why very local 3×3 or 5×5 convolutions are used in practice.

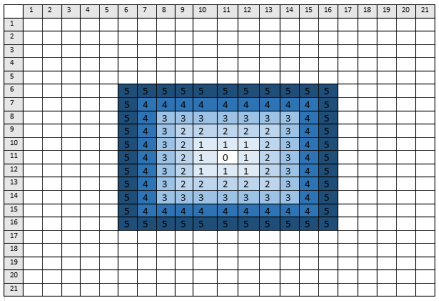


Figure 5: Number of cascaded 3×3 convolutions needed to reach a point on the 21×21 grid from the center point $(11, 11)$.

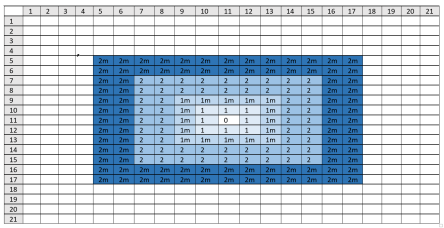


Figure 6: Number of cascaded 3×3 convolutions with 2×2 max pooling needed to reach a point on the 21×21 grid from the center point $(11, 11)$. Number (e.g. 1,2) refers to the number of 2d convolutions, while "m" refers to the max pooling in the same cascade. Convolution 2 (3×3 in the cascade has reach of 5 pixels in the input feature map because of the downsampled version of the feature map after max pooling.

Max Pooling: Similarly as for 2d/3d convolutions, max pooling is usually defined on some local neighbourhood of the point, usually on the square of edge length $l \in \mathbb{N}$. Max pooling $\phi_{maxpool} : \mathbb{N} \times (\mathbb{R})^{\mathbb{Z}^d} \times \mathbb{Z}^d \to (\mathbb{R})^{\mathbb{Z}^d}$ is defined

$$\phi_{maxpool}(l, f, x) = \max_{j=(j_1,\cdots,j_d),|j_1|,\cdots,|j_d|\leq \frac{l-1}{2}} f(x+j).$$

Pooling operators establish invariance to local translations (up to $l \in \mathbb{N}$ pixels) in the feature maps. For this reason, in practice max pooling is applied on non-overlapping windows reducing effectively the size of the feature maps by the factor of local translation invariance l.
Combining 2d convolutions with non-overlapping max pooling and cascading them increases the size of covered windows even more than only cascading 2d convolutions. This further reduces the number of parameters. An example is shown in Figure 6. In this case, only 2 steps of 2d convolution and max pooling are needed to cover the window of size of 12×12. Hence, even fewer parameters are required here: $2 \cdot 3 \cdot 3 = 18$ parameters originating from only two 2d convolutions. This is further parameter reduction from 45 parameters in Figure 6.

Rectifier or rectified linear unit or ReLU: ReLU is a pointwise nonlinear operator. For $a \in \mathbb{R}$ it is defined as function $\phi_{relu} : \mathbb{R} \to \mathbb{R}$

$$\phi_{relu}(a) = \max(a, 0).$$

ReLU can be interpreted as an activation function, i.e. it is activated only for the positive inputs while it is deactivated for the negative ones enabling sparse feature representation.

Sigmoid function: For $a \in \mathbb{R}$, sigmoid function $\phi_{sigmoid} : \mathbb{R} \to \mathbb{R}$

$$\phi_{sigmoid}(a) = \frac{1}{1 + e^{-a}}.$$

It is usually used as a final nonlinearity in two-class classification problems and it can be interpreted as a probability of the input belonging to the first class.

Softmax function: For $a = (a_1, \cdots a_d) \in \mathbb{R}^d$ it is defined as a function $\phi_{softmax} : \mathbb{R}^d \to \mathbb{R}^d$

$$\phi_{softmax}(a) = (\frac{e^{a_1}}{\sum_{j=1}^{d} e^{a_j}}, \cdots, \frac{e^{a_d}}{\sum_{j=1}^{d} e^{a_j}}) \in \mathbb{R}^d.$$

In contrast to ReLU, the softmax function is not a pointwise operator and it can be considered as a probability distribution or scores over a finite set of outcomes. Softmax is usually used as a final nonlinearity in multiclass classification problems to extract a probability per class.

Batch normalization [118]: During training, the training set is usually randomly split into batches, i.e. smaller subsets of the training set on which the back-propagation algorithm is applied. Batch normalization works on the single batch of the training set $\{f_1, \cdots, f_n\}$ and represents a very important regularization technique which contributes to the stability and convergence of the training process. Formally, it can be defined as a pointwise operator $\phi_{batchnorm} : \mathbb{R}^{\mathbb{Z}^d} \to \mathbb{R}^{\mathbb{Z}^d}$

$$\phi_{batchnorm}(f_i)(x) = \frac{f_i(x) - E[f]}{\sqrt{Var[f]}},$$

where $E[f]$ and $Var[f]$ represent the expected value and variance of the feature map set $\{f_1, \cdots, f_n\}$, respectively, and are in practice estimated by the sample mean or variance. Alternatively, it can be more robustly estimated as a weighted average between the previous estimate and the new batch mean or variance with some momentum $mom \in \langle 0, 1 \rangle$. For example, let μ_{k-1} be the mean estimate from the previous step and μ_k^+ the mean from the current batch, $E[f]$ is estimated by μ_k with

$$\mu_k = (1 - mom) \cdot \mu_k + mom \cdot \mu_k^+.$$

There is a possible improvement in the batch normalization by introducing trainable parameters $\beta, \gamma \in \mathbb{R}$ resulting in a similar operator $\phi^+_{batchnorm} : \mathbb{R}^{\mathbb{Z}^d} \to \mathbb{R}^{\mathbb{Z}^d}$

$$\phi^+_{batchnorm}(f_i)(x) = \gamma \frac{f_i(x) - E[f]}{\sqrt{Var[f]}} + \beta.$$

B Additional experiments on scale selection for competing methods related to Riesz network

The largest benefit of the Riesz network is avoiding the sampling of the scale dimension. Here, we give more detailed insight into scale sampling in practice for competing methods: U-net applied on rescaled images and Gaussian derivative networks. We show how segmentation results change as we add additional scales to the output. As we add new scales, cracks that belong (or are close) to the added scales get segmented. However, additional noise gets segmented, too. These noise pixels that are misclassified as cracks originate from two sources: interpolation error and high frequency information characteristic for CT imaging. For simulated data this is shown in Figure 7, Figure 8, Figure 9, Figure 10 and Figure 11. For real cracks this is shown in Figure 15.

The main drawback is that one needs to select the range of scales on which to apply these methods. Since the scale dimension in the images is bounded from above by the size of the view window, when having images of different sizes scale sampling needs to be adjusted or recalibrated. It is not trivial how to achieve this in a general manner. In contrast, the Riesz transform enables simultaneous, continuous, and equal treatment of all scales automatically adapting to the image size.

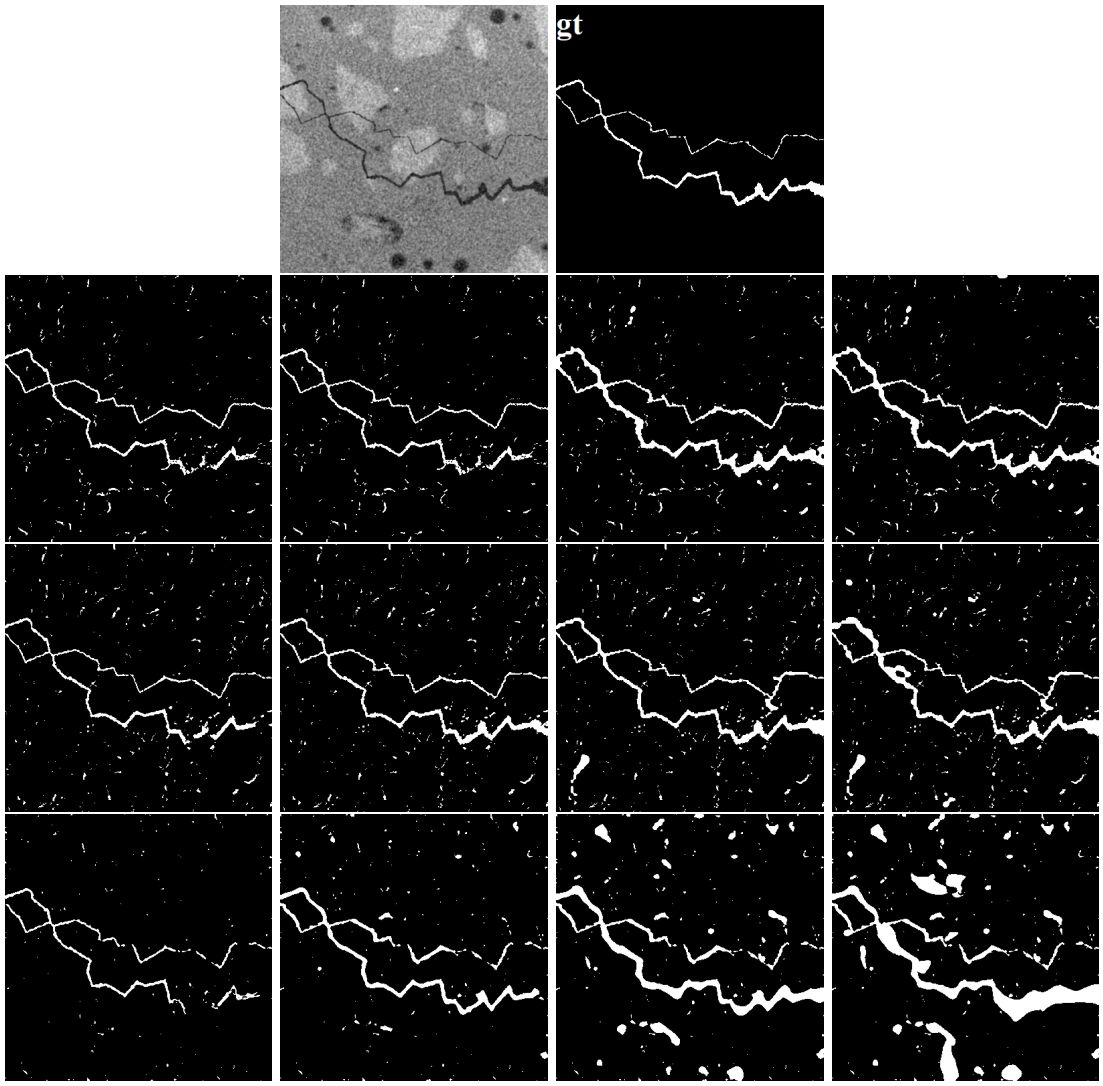


Figure 7: Experiment 2. Cracks with varying width. First row: input image and ground truth image. Second row: U-net applied to several levels of pyramids $\{1, 2, 3, 4\}$ (from left to right). Third row: U-net-mix applied to several levels of pyramids $\{1, 2, 3, 4\}$ (from left to right). Fourth row: Gaussian derivative networks aggregated on growing subsets of scale set $\{1.5, 3, 6, 12\}$ (from left to right). Image size 400×400 pixels.

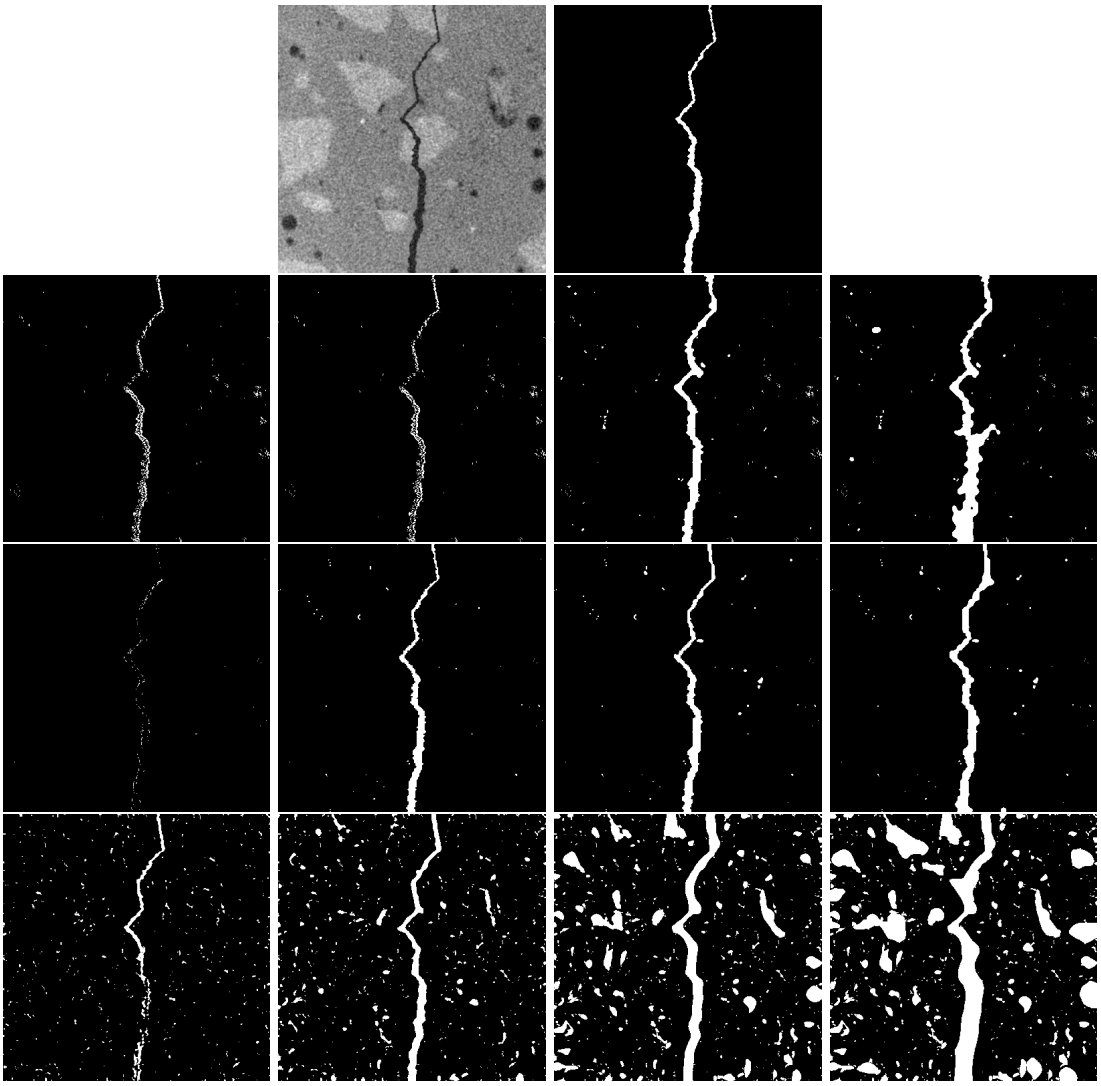


Figure 8: Experiment 2. Cracks with varying width. First row: input image and ground truth image. Second row: U-net applied to several levels of pyramids $\{1, 2, 3, 4\}$ (from left to right). Third row: U-net-mix applied to several levels of pyramids $\{1, 2, 3, 4\}$ (from left to right). Fourth row: Gaussian derivative networks aggregated on growing subsets of scale set $\{1.5, 3, 6, 12\}$ (from left to right). Image size 400×400 pixels.

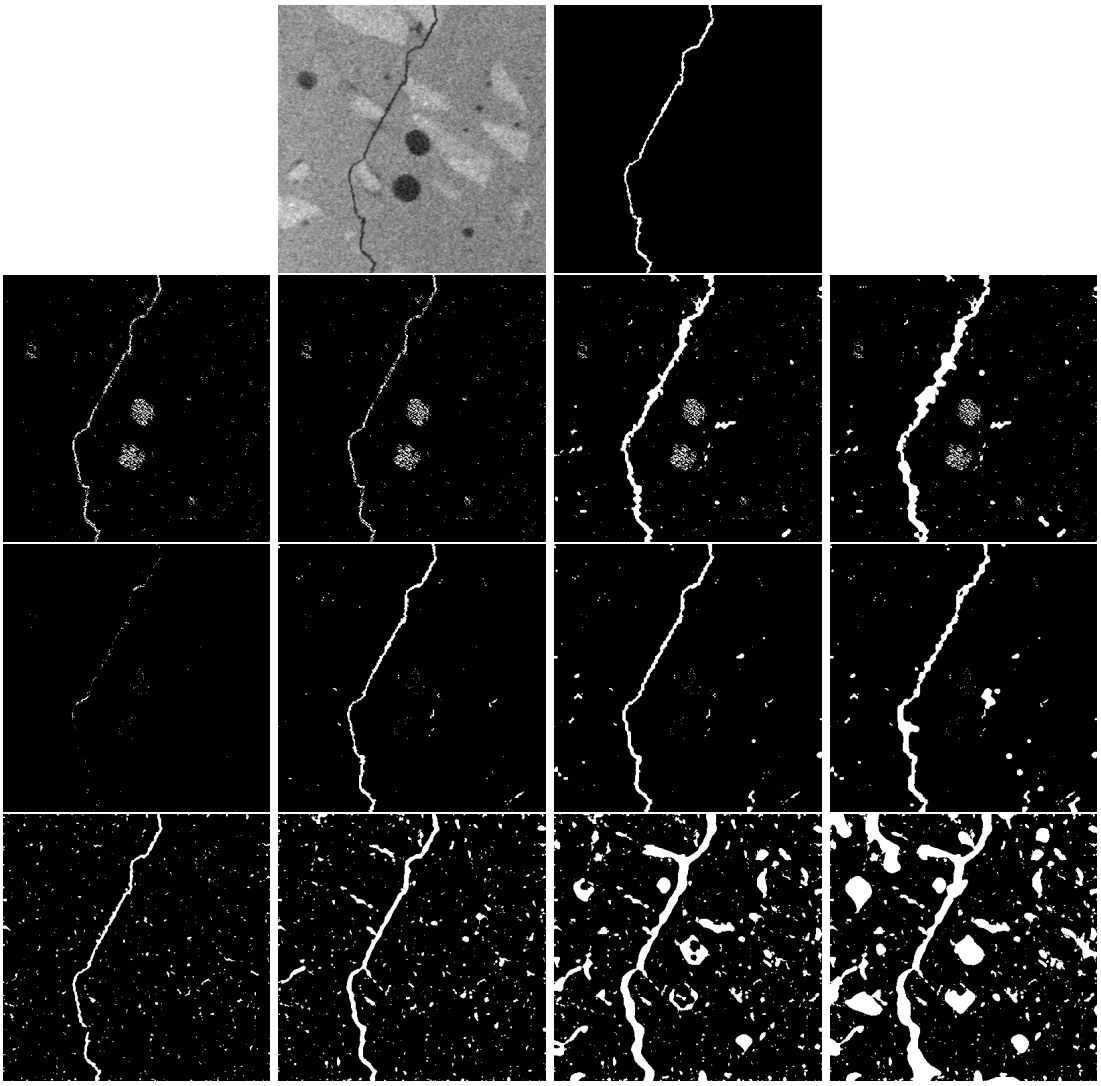


Figure 9: Experiment 2. Cracks with varying width. First row: input image and ground truth image. Second row: U-net applied to several levels of pyramids $\{1, 2, 3, 4\}$ (from left to right). Third row: U-net-mix applied to several levels of pyramids $\{1, 2, 3, 4\}$ (from left to right). Fourth row: Gaussian derivative networks aggregated on growing subsets of scale set $\{1.5, 3, 6, 12\}$ (from left to right). Image size 400×400 pixels.

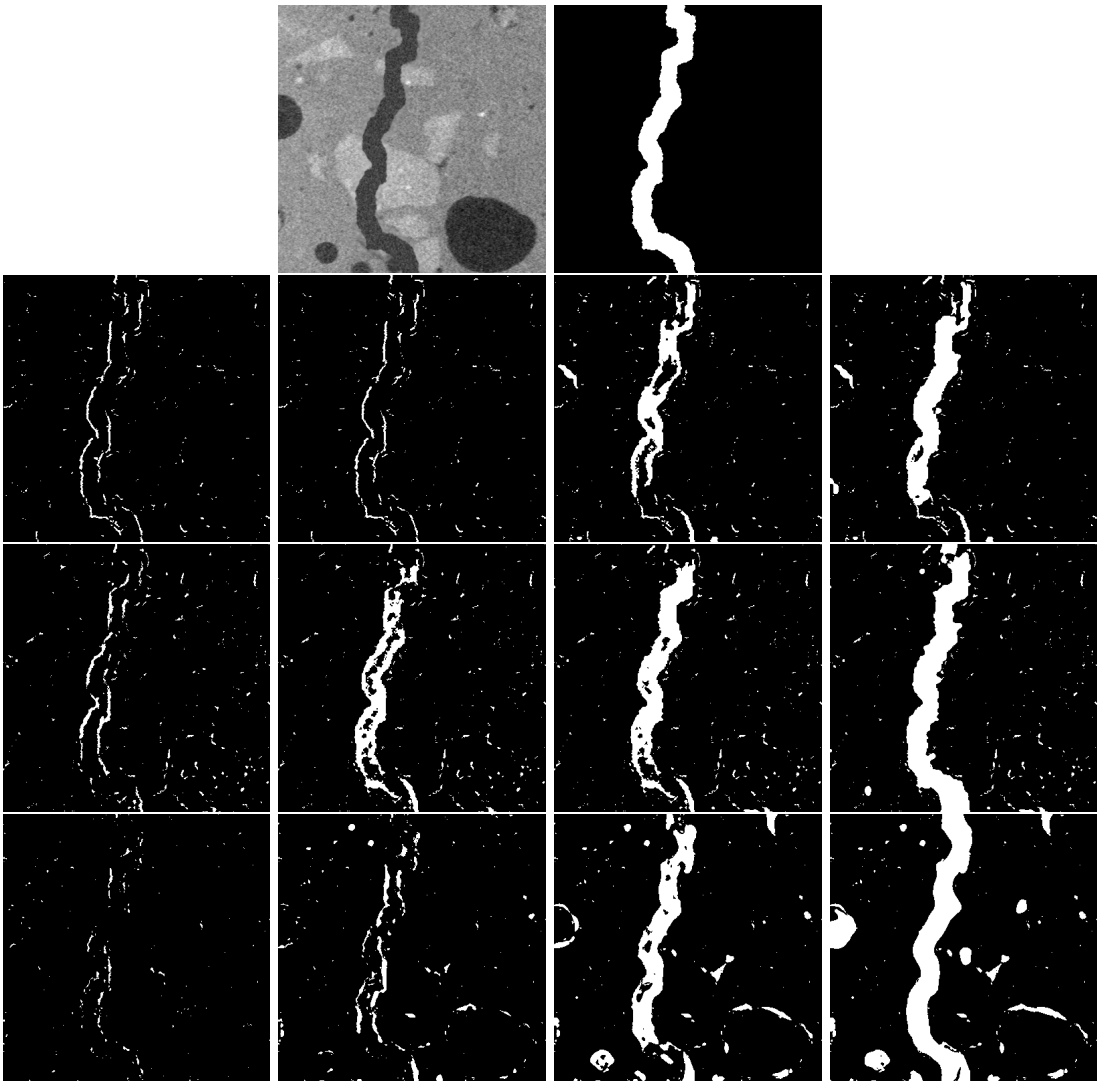


Figure 10: Experiment 2. Cracks with varying width. First row: input image and ground truth image. Second row: U-net applied to several levels of pyramids $\{1, 2, 3, 4\}$ (from left to right). Third row: U-net-mix applied to several levels of pyramids $\{1, 2, 3, 4\}$ (from left to right). Fourth row: Gaussian derivative networks aggregated on growing subsets of scale set $\{1.5, 3, 6, 12\}$ (from left to right). Image size 400×400 pixels.

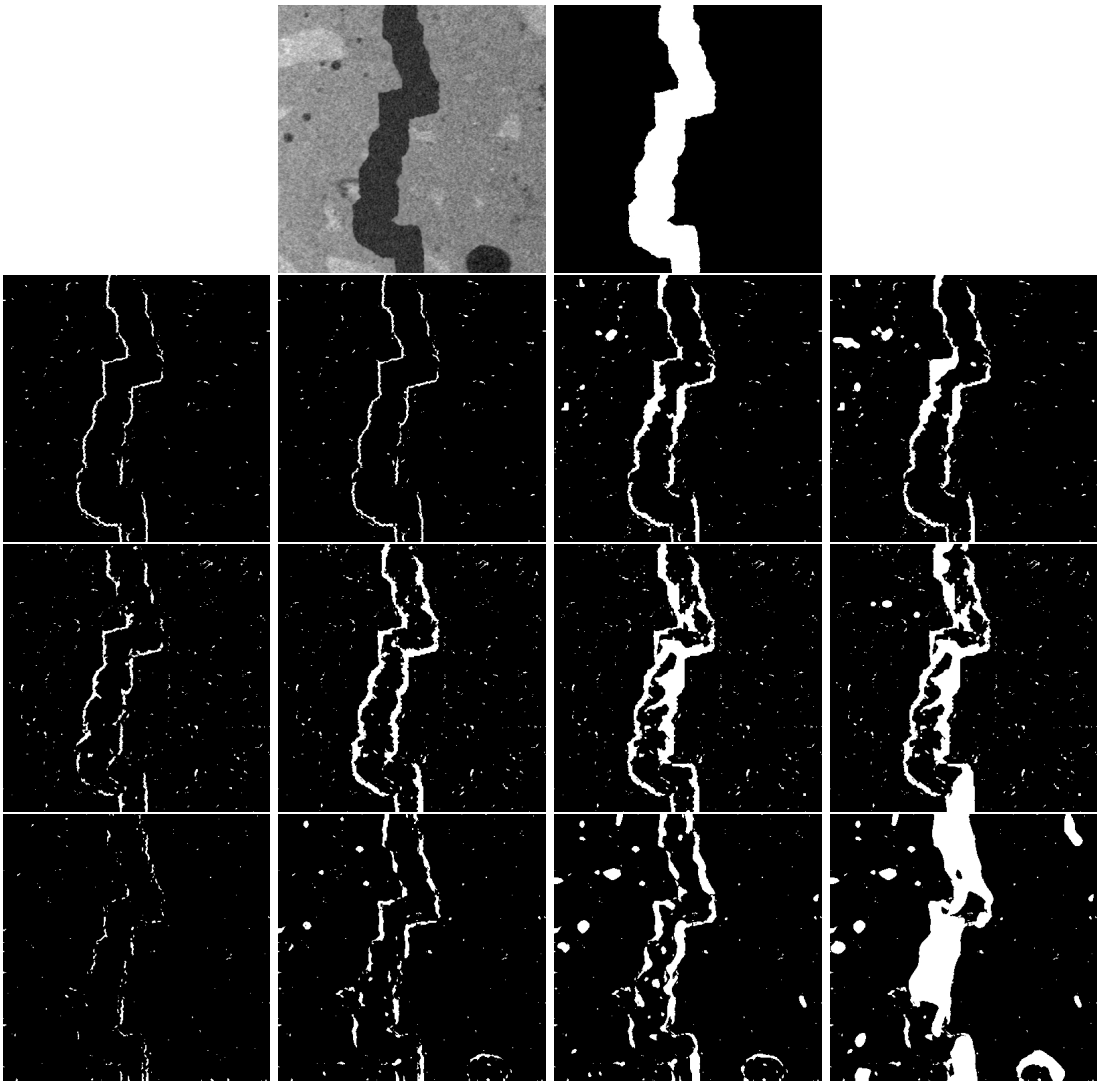


Figure 11: Experiment 2. Cracks with varying width. First row: input image and ground truth image. Second row: U-net applied to several levels of pyramids $\{1, 2, 3, 4\}$ (from left to right). Third row: U-net-mix applied to several levels of pyramids $\{1, 2, 3, 4\}$ (from left to right). Fourth row: Gaussian derivative networks aggregated on growing subsets of scale set $\{1.5, 3, 6, 12\}$ (from left to right). Image size 400×400 pixels.

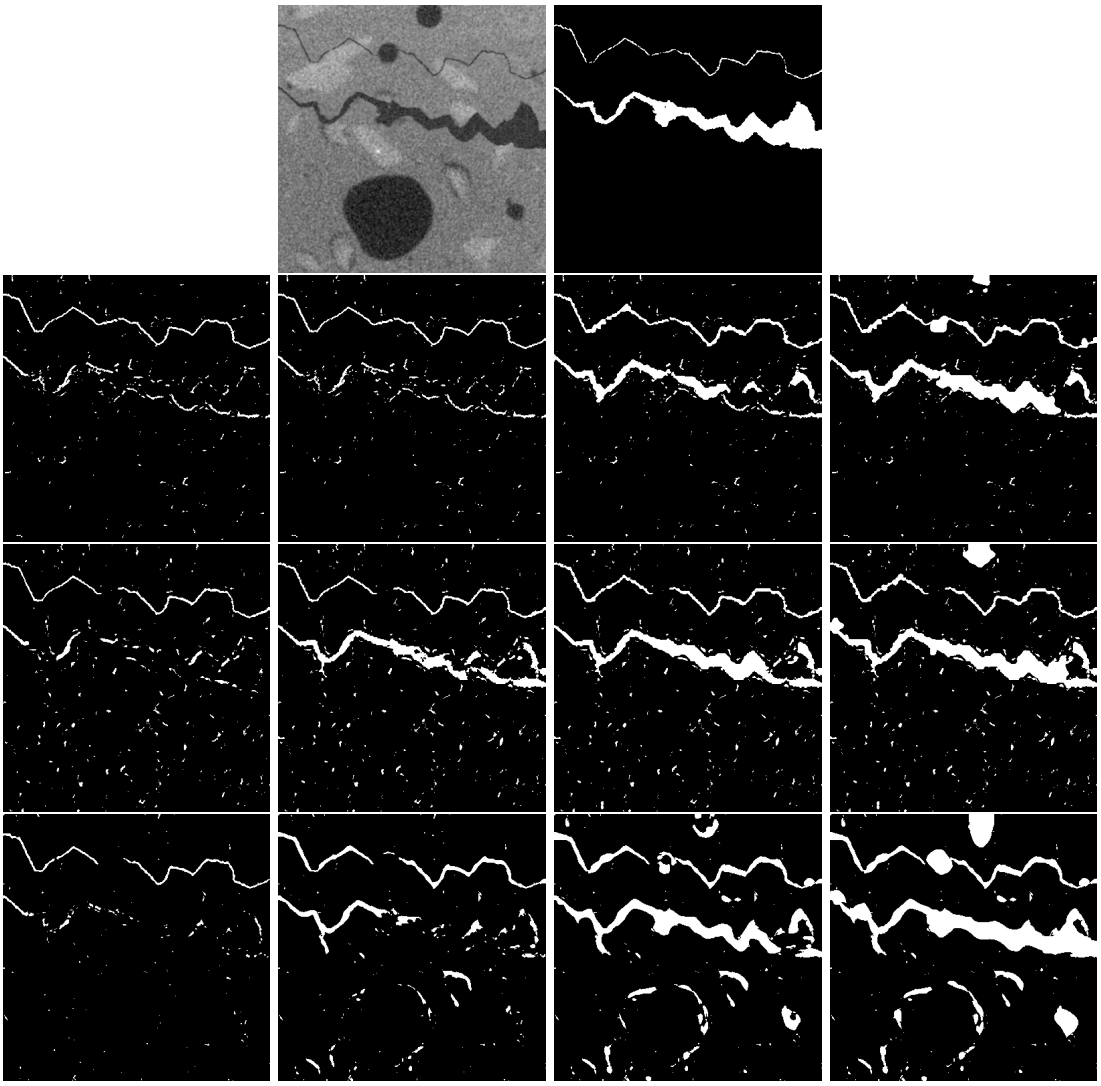


Figure 12: Experiment 2. Cracks with varying width. First row: input image and ground truth image. Second row: U-net applied to several levels of pyramids $\{1, 2, 3, 4\}$ (from left to right). Third row: U-net-mix applied to several levels of pyramids $\{1, 2, 3, 4\}$ (from left to right). Fourth row: Gaussian derivative networks aggregated on growing subsets of scale set $\{1.5, 3, 6, 12\}$ (from left to right). Image size 400×400 pixels.

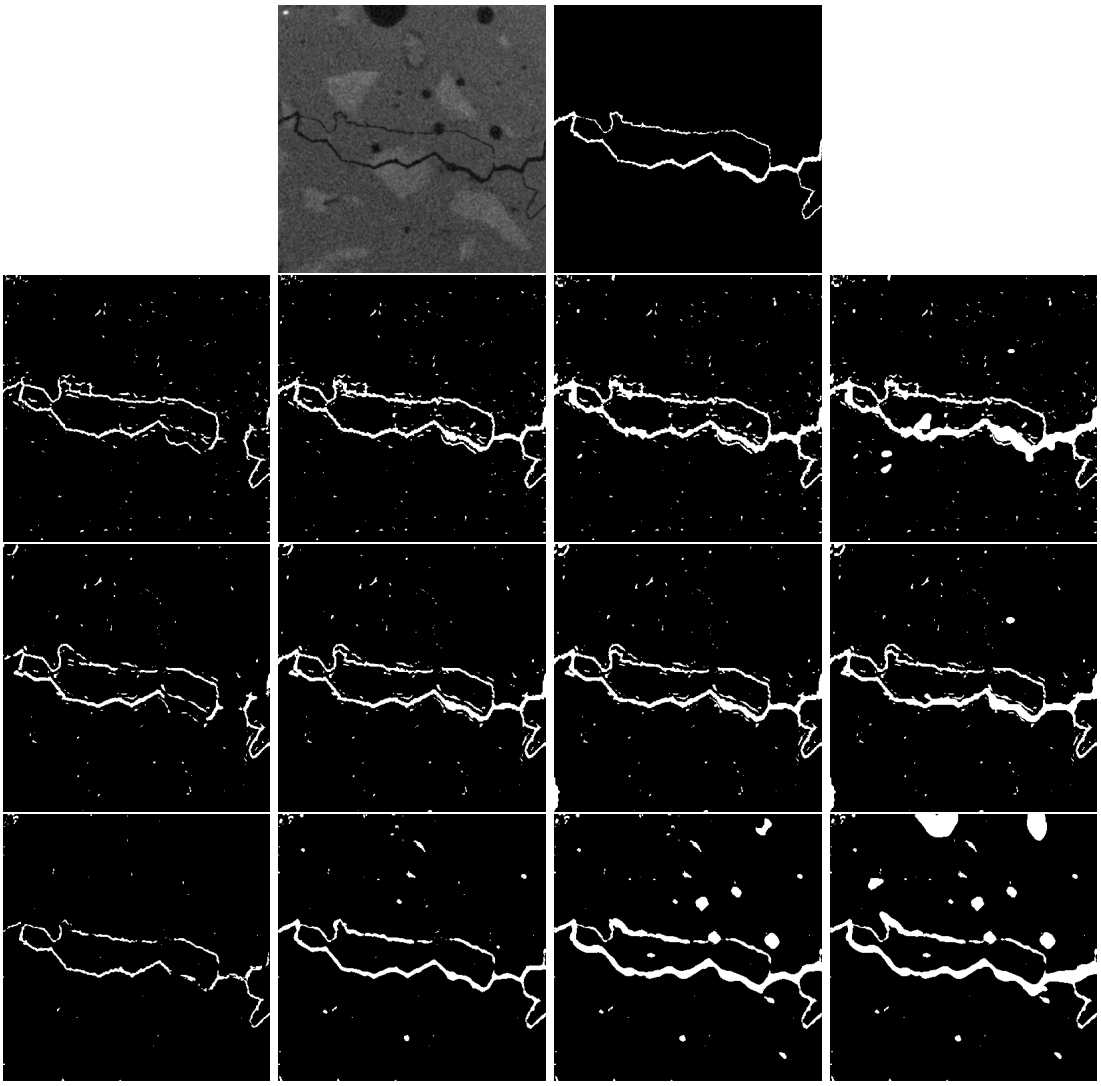


Figure 13: Experiment 2. Cracks with varying width. First row: input image and ground truth image. Second row: U-net applied to several levels of pyramids $\{1, 2, 3, 4\}$ (from left to right). Third row: U-net-mix applied to several levels of pyramids $\{1, 2, 3, 4\}$ (from left to right). Fourth row: Gaussian derivative networks aggregated on growing subsets of scale set $\{1.5, 3, 6, 12\}$ (from left to right). Image size 400×400 pixels.

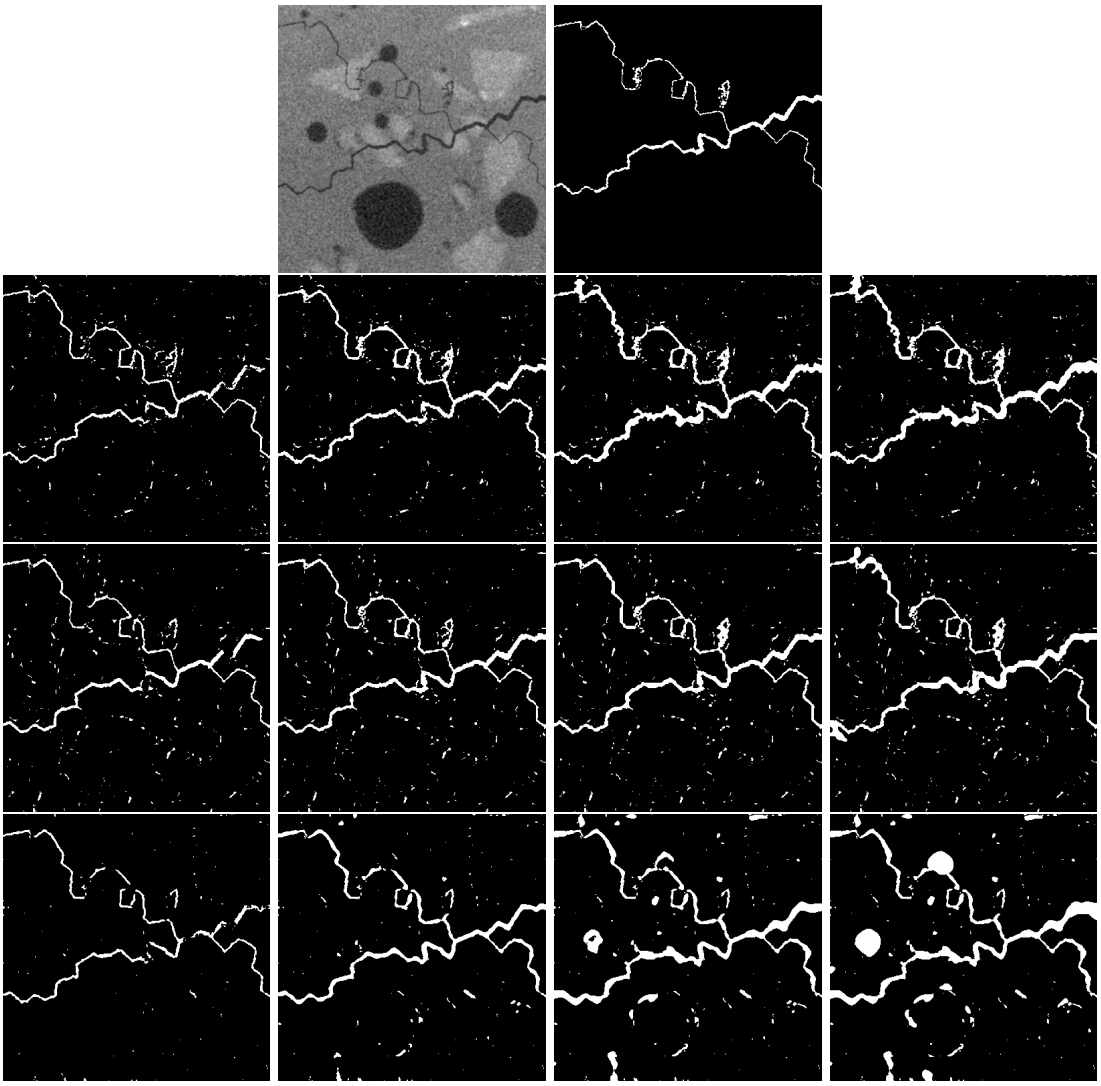


Figure 14: Experiment 2. Cracks with varying width. First row: input image and ground truth image. Second row: U-net applied to several levels of pyramids $\{1, 2, 3, 4\}$ (from left to right). Third row: U-net-mix applied to several levels of pyramids $\{1, 2, 3, 4\}$ (from left to right). Fourth row: Gaussian derivative networks aggregated on growing subsets of scale set $\{1.5, 3, 6, 12\}$ (from left to right). Image size 400×400 pixels.

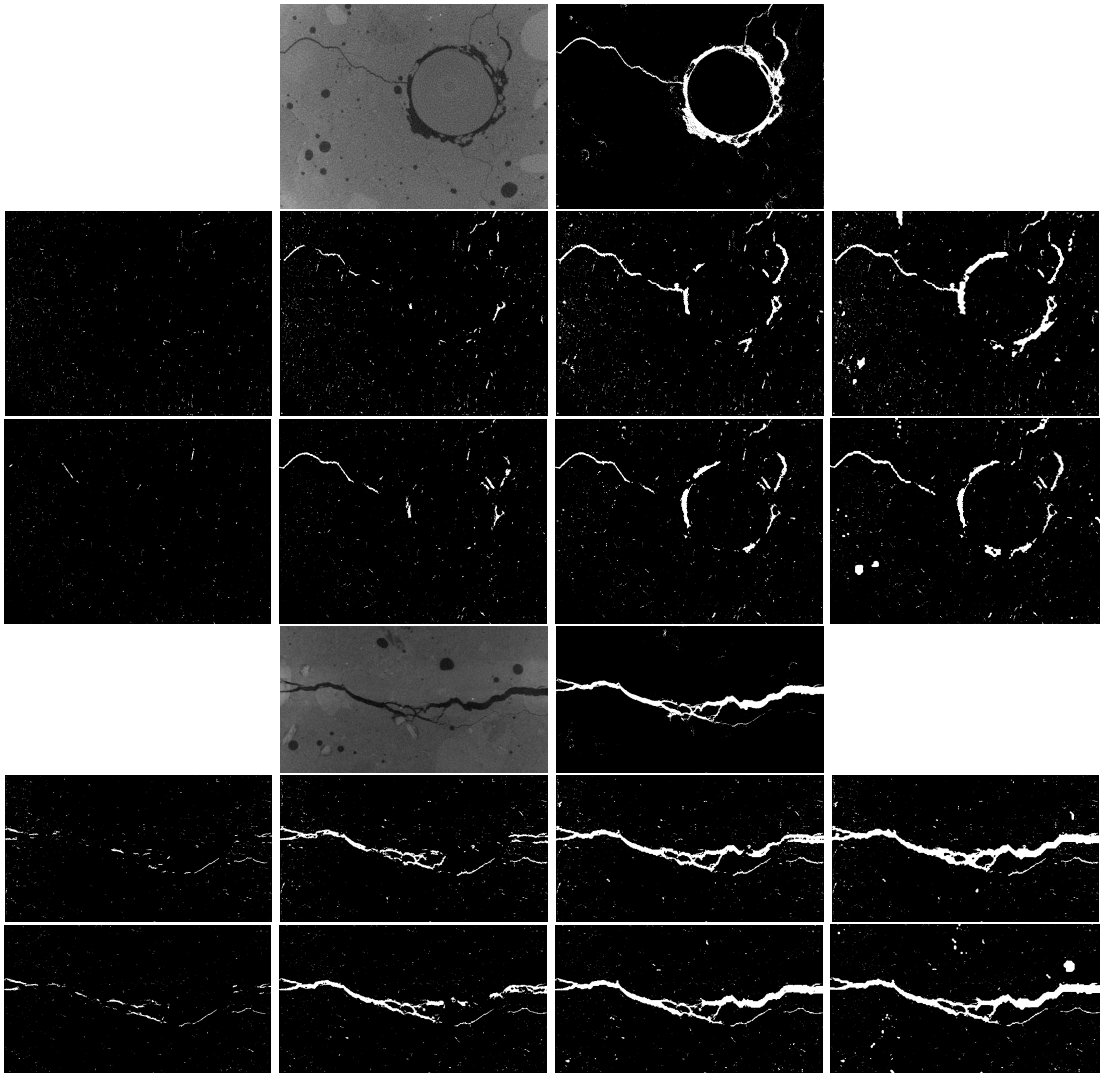


Figure 15: Experiment 3. Real cracks in concrete: slice from input CT image, results of the Riesz network and of U-net and U-net-mix with ranging pyramid levels (from 1 to 4). Image sizes are $832 \times 1\,088$ (1st row) and 544×992 (4th row).

C Details on Hessian-based percolation and 3d U-net

Hessian-based percolation Hessian-based percolation [44] is a two-step algorithm. In the first step, candidate crack voxels are selected from the Frangi filter [33]. Frangi filter calculates the shape statistics from eigenvalues of the Hessian matrix in every voxel of the image. These shape statistics based on eigenvalues can be designed to produce a high response in dark, planar regions in the image. To obtain the candidates, we apply hysteresis thresholding ($t_1 \geq t_2$) to the filter values. This means that the connected component from the foreground voxels for the threshold t_2 is reconstructed if at least one voxel from the foreground voxels for the threshold t_1 belongs to that connected component. The Frangi filter [33] for our use-case is defined as follows. Let $H(p,\sigma) = (h_{i,j})_{i,j=1}^3$ and $h_{i,j}(p) = (I * \frac{\partial}{\partial x_i}\frac{\partial}{\partial x_j} g_\sigma)(p)$ be the Hessian matrix of image I at voxel p, where g_σ is standard Gaussian kernel with $\sigma > 0$. Further let $|\lambda_1(p,\sigma)| \leq |\lambda_2(p,\sigma)| \leq |\lambda_3(p,\sigma)|$ be the eigenvalues of $H(p,\sigma)$. For each voxel p let

$$E(p,\sigma) = \begin{cases} \exp(-Q_A^2(p,\sigma)/\alpha)\exp(-Q_B^2(p,\sigma)/\beta)(1-\exp(-R^2(p,\sigma)/\eta_\sigma)), & \lambda_3(p,\sigma) > 0\,,\ \lambda_2(p,\sigma) \neq 0 \\ \exp(-Q_A^2(p,\sigma)/\alpha)(1-\exp(-R^2(p,\sigma)/\eta_\sigma)), & \lambda_3(p,\sigma) > 0\,,\ \lambda_2(p,\sigma) = 0 \\ 0, & \text{else} \end{cases} \tag{4}$$

with

$$Q_A(p,\sigma) = \frac{|\lambda_2(p,\sigma)|}{|\lambda_3(p,\sigma)|}, \qquad Q_B(p,\sigma) = \frac{|\lambda_1(p,\sigma)|}{\sqrt{|\lambda_2(p,\sigma)||\lambda_3(p,\sigma)|}}, \qquad R(p,\sigma) = \sqrt{\sum_{i=1}^{3} \lambda_i(p,\sigma)^2},$$

and weighting parameters $\alpha, \beta, \eta_\sigma > 0$. It is often recommended [33] to choose $\alpha = \beta = 0.5$, independently of the image content and the choice of σ. The parameter η_σ depends on scale by setting $\eta_\sigma = 2(\max_p R(p,\sigma))^2$. To account for multiple scales, the Frangi filter is defined as

$$F(p) = \max_{\sigma_{min} \leq \sigma \leq \sigma_{max}} E(p,\sigma), \quad p \in I, \tag{5}$$

where $0.5 \leq \sigma_{min} \leq \sigma_{max} \in \mathbb{R}$ are an upper and a lower bound of scale space which is in practice sampled, i.e. $E(p,\sigma)$ is calculated only for the finitely many scales in the interval $[\sigma_{min}, \sigma_{max}]$. The values of $F(p)$ lie in $[0,1)$. The higher $F(p)$, the more likely p belongs to a crack structure. We consider voxel p to be part of the preselection set H_F if $F(p) \geq t_0$ for some threshold $t_0 \in \mathbb{R}$.

In the second step, a region-growing process is started in each preselected voxel. The resulting region is then considered to be part of a crack if its shape matches a fixed shape criterion. Denote the set of candidate voxels by H_F and the input image by I. The iterative region-growing algorithm consists of the following steps:

1. For $p \in H_F$ initialize the percolated region as $P = \{p\}$ and set $t = I(p) + \varepsilon$ for a predefined, real-valued ε.
2. Add all neighbors q of P to P that satisfy $I(q) \leq t$.
3. Set $t = \max\left(\max_{q \in P} I(q), t\right) + \varepsilon$.
4. Repeat 2 and 3 until the region P comes in contact with the boundary of a cubic window of predefined size $(2W+1)^3$, $W \in \mathbb{N}$, centered at p.
5. Compute the shape criterion
$$F_{3D} = |P \cap H_F|/|P| \in [0,1].$$
If $F_{3D} \geq f$ for some predefined $f > 0$, every voxel in P is considered to be part of the crack structure.

6. Repeat 1-5 for every voxel $p \in H_F$. Voxels that have been detected less often than a predefined threshold $\tilde{t} \in \mathbb{N}$ are not considered to be part of the crack structure.

The percolation step serves as a post-processing step which could remove noisy misclassified voxels and improve connectivity in the crack structure.

3d U-net The deep learning neural network which was extensively tested for crack segmentation in 3d CT images is the 3d U-Net [129]. The model consists of a contracting path (encoder), which contracts the input image, and the extracting path (decoder), responsible for expanding the image. The depth of the 3d U-Net is three which means it consists of three convolution blocks in the contracting path (16, 32, and 64 filters), one bottleneck (128 filters), and three convolution blocks in the extracting path (64, 32, and 16 filters).

The convolution block of the contracting path consists of two 3d convolutions of kernel size three, each followed by a batch normalization and a rectified linear unit (ReLU) activation. In the next step we apply a max pooling layer with stride two for downsampling. In the expansive path, we use a 3d transposed convolution of kernel size three for upsampling at the beginning of each step. The transposed convolution is followed by a concatenation with the corresponding cropped feature map from the contracting path.

Due to the memory restrictions, the U-net is applied to image patches of size 64^3. Images are thus tiled into patches of this size. To reduce possible edge effects, patches overlap by 14 voxels. Training data is augmented with rotation, flip and downscaling (zoom 0.5 and 0.25) operations to ensure robustness. In total, the U-net was trained on 3 456 images for 20 epochs with batch size 2. The Adam optimization algorithm [119] was used with a learning rate of 0.001 which is cut in half every 5 epochs.

The cracks observed in the CT data are partially much thicker than five voxels. Hence, scales covered by the U-net are limited by the crack widths in the training set (crack width 1,3, and 5). To take this into account, we use a multi-scale approach. That means, the prediction of the network is computed on the original image as well as on images downscaled by factors 0.5 and 0.25. In this way, crack widths covered by this U-net with a multiscale adjustment range from 1 to $4 \times 5 = 20$ voxels. Predictions for downscaled images are then upscaled to the original size for all images. The final prediction is derived from the voxelwise maximum, thresholded at 0.5.

Parameter configurations Parameter configurations for every dataset used in this thesis are discussed in each publication separately [100, 124, 126].
Specifically, for Section 7.4.1 one should refer to [100]. For Section 7.4.2 details can be found in [126], while [124] discussed the parameter configuration for Section 7.4.3.

D Quality measures for evaluation of segmentation results

To evaluate the goodness of the methods, we compare their outputs with the respective ground truths voxelwise. Outputs and ground truths are binary images. Hence, quality metrics can be defined based on the number *tp* (true positive) of correctly predicted crack voxels, the number *tn* (true negative) of correctly predicted background voxels, the number *fp* (false positive) of falsely predicted crack voxels, and the number *fn* (false negative) of falsely predicted background voxels. Formally, let $X : D \to \{0,1\}$ be the prediction image and $Y : D \to \{0,1\}$ the ground truth image. Then, these metrics are defined

$$tp := tp(X,Y) = \sum_p \mathbb{1}_{X(p)=Y(p)=1}, \quad tn := tn(X,Y) = \sum_p \mathbb{1}_{X(p)=Y(p)=0}$$

$$fp := fp(X,Y) = \sum_p \mathbb{1}_{X(p)=1,Y(p)=0}, \quad fn := fn(X,Y) = \sum_p \mathbb{1}_{X(p)=0,Y(p)=1}$$

Precision (P), recall (R) and F1-score (F1) are defined as

$$P(X,Y) = \frac{tp}{tp+fp}, \quad R(X,Y) = \frac{tp}{tp+fn},$$

$$F1(X,Y) = \frac{2PR}{P+R}.$$

where the F1-score is the weighted average of precision and recall.

Precision provides information on what proportion of voxels classified as positive is indeed positive. Recall measures the fraction of positive voxels that are classified correctly. In general, precision is a good measure when *fp* or oversegmentation should be penalized more while recall puts more weight on *fn* or missing out on segmenting the crack. The F1-score is used when a balance between precision and recall is pursued. In particular, it is a suitable overall measure when dealing with class imbalance. Note that alternative measures, as for instance the accuracy, may be less meaningful due to the high percentage of background voxels.

For segmentation tasks, it is not unusual to introduce a certain voxel tolerance *tol*. That means: A true crack voxel is considered *tp* in the output if its distance to the positives is less or equal *tol*. Otherwise, it is considered *fn*. A predicted crack voxel is only counted as *fp* if its distance to the true crack voxels is larger than *tol*. Tolerance penalizes more misclassifications far from the crack than the ones in the approximate vicinity.

Intersection over Union (IoU) compares the union and intersection of the foregrounds X and Y in the segmented image and the corresponding ground truth, respectively. That is

$$IoU(X,Y) = \frac{|X \cap Y|}{|X \cup Y|}.$$

All these metrics have values in the range $[0,1]$ with values closer to 1 indicating better performance.

E Fourier transform and related results

The Fourier transform decomposes a function in the time domain to frequency components. This can be seen as a switch of bases or representations from time-localized functions to frequency-localized functions. To do this the Fourier transform uses complex sinusoidal functions. The output of the Fourier transform is a complex-valued function. If a certain frequency is not presented in the signal, the Fourier transform at that frequency equals 0. The Fourier transform is an invertible transform, i.e. one can go back from the frequency or Fourier domain to the time domain without any loss of information. The uncertainty principle states the connection between frequency and time representation of the function: functions that are localized in the time domain are spread out across the frequency domain and the other way around.

The Fourier transform is originally derived for 1d signals. However, extension to higher dimensions is analogous. For higher dimension time domain is more often called the spatial domain. Here, we give a definition of Fourier transform $\mathcal{F}$ for a d-dimensional real function $f : \mathbb{R}^d \to \mathbb{R}$, which represents images that are objects of interest in this thesis. Here, $x = (x_1, \cdots, x_d)$ is a point in the spatial domain, while $\omega = (\omega_1, \cdots, \omega_d)$ denotes a point in the frequency domain. Now, the Fourier transform (FT) $\mathcal{F}$ of $f \in L_2(\mathbb{R}^d)$ can be defined as

$$\hat{f}(\omega) := \mathcal{F}(f)(\omega) = \int_{\mathbb{R}^d} f(x) e^{i\langle \omega, x \rangle} dx.$$

Here, $\langle \omega, x \rangle := \sum_{k=1}^{d} x_i \omega_i$ is a standard scalar product in $\mathbb{R}^d$. Inverse Fourier transform $\mathcal{F}^{-1}$ is defined as

$$f(x) = \mathcal{F}^{-1}(\hat{f})(x) = \int_{\mathbb{R}^d} \hat{f}(x) e^{-i\langle \omega, x \rangle} d\omega.$$

Next, we give results based on the Fourier transform that are essential to understanding the content of this thesis. Linearity of the Fourier transform means that for any $\alpha, \beta \in \mathbb{C}$ and $f, g \in L_2(\mathbb{R}^d)$ it holds

$$\mathcal{F}(\alpha f + \beta g)(\omega) = \alpha \mathcal{F}(f)(\omega) + \beta \mathcal{F}(g)(\omega).$$

Linearity combined with the inverse Fourier transform $\mathcal{F}^{-1}$ implies that

$$f = g \iff \mathcal{F}(f) = \mathcal{F}(g).$$

The rescaling operation with real factor $a > 0$ is interesting for us in the context of the Riesz transform and scale equivariance. There exists a useful result on this in the Fourier domain

$$\mathcal{F}(f(a\cdot))(\omega) = \frac{1}{a^d} \mathcal{F}(f)(\frac{\omega}{a}).$$

The result on the Fourier transform with respect to differentiation of $f \in L_2(\mathbb{R}^d)$ with continuous derivatives in the spatial domain is given as

$$\mathcal{F}(\frac{\partial}{\partial x_k} f)(\omega) = i 2\pi \omega_k \mathcal{F}(f)(\omega).$$

The Plancherel theorem is relevant to Chapter 8 and the energy preservation of the Riesz feature representation.

Theorem 5. ***Parseval's formula and Plancherel theorem:*** *For $f, g \in L_2(\mathbb{R}^d)$ square integrable functions, Parseval's formula holds*

$$\langle f, g \rangle_{L_2(\mathbb{R}^d)} = \int_{\mathbb{R}^d} f(x) \overline{g}(x) dx = \int_{\mathbb{R}^d} \hat{f}(\omega) \hat{g}(\omega) d\omega,$$

where $\overline{g}$ denotes the complex conjugate of g. Then Plancherel theorem is only the consequence of Parseval's formula

$$||f||^2_{L_2(\mathbb{R}^d)} = \int_{\mathbb{R}^d} |f(x)|^2 dx = \int_{\mathbb{R}^d} |\hat{f}(\omega)|^2 d\omega.$$

An important result on the Fourier transform and convolution is given in the following theorem.

Theorem 6. ***Convolution theorem:*** *Convolution of two integrable functions in the spatial domain becomes multiplication in the Fourier domain. The reverse holds as well: multiplication of two integrable functions in the spatial domain becomes convolution in the Fourier domain. Formally, for $f, g \in L_2(\mathbb{R}^d)$ integrable functions, the following results hold*

$$\mathcal{F}(f * g)(\omega) = \mathcal{F}(f)(\omega) \mathcal{F}(g)(\omega)$$

and

$$\mathcal{F}(fg)(\omega) = \mathcal{F}(f)(\omega) * \mathcal{F}(g)(\omega).$$

This theorem is a key result for the implementation of the Riesz transform.

In this appendix we described results for functions on the continuous domain $\mathbb{R}^d$. Proofs of the claims in this thesis are also based on the continuous domain. In practice, images are defined on the discrete compact domain. For this purpose, there exists the discrete Fourier transform (DFT). All of these properties have their counterparts in the discrete domain.

F Basics on quadrature filters

Quadrature filters in 1d: Quadrature filters are used to estimate and analyze structural characteristics of the images such as phase, frequency, or amplitude, see [135] for an overview on this topic. Since the estimation of these characteristics is intrisically noisy, different quadrature filters exist. The goal with quadrature filters is to extract localized features. That is why the analytic signal (Section 4.3.1) which is defined on the whole spatial and frequency domain, is not used. Alternatively, the analytic signal can be applied on the band-pass filtered version of the signal which does not alter the phase. This implies that the real part of this type of quadrature filter has to be an even filter, while the imaginary part is an odd filter. To summarize, the basic idea behind quadrature filters is to extract the amplitude and phase of the signal for a limited frequency range using analogous formulas as in Section 4.3.1.
An important example of quadrature filters constructed from the analytic signal is Gabor filters

$$g_{x_0,\omega_0}(x) = g(x - x_0)e^{i\omega_0 x}.$$

The Gabor filter is also known as Short Time Fourier Transform (STFT) or windowed Fourier transform. Here, x_0 refers to the point in the time domain, while ω_0 is a selected frequency. At this time-frequency pair (x_0, ω_0) we aim at measuring local image characteristics. The function g is called window function and it is usually chosen as a Gaussian filter. In that case, the Gabor filter corresponds to the Morlet wavelet from Appendix G up to an additive real constant β. One of the drawbacks is that the Gabor filter has infinite support despite being well-localized[4]. Furthermore, since the cosine component does not integrate to 0, it is not invariant to constant shifts in gray values. Other well-known examples of quadrature filters constructed from the analytic signal are log-Gabor filters, Gaussian derivative filters, or Cauchy filters, etc.

Alternatively, one can observe that the even (odd) part of Gabor and other filters resembles the shape of the second (first) order derivative, respectively. These derivatives are often interpreted as line (edge) detectors. Hence, in the literature [135] there also exist quadrature filters based on the criteria for designing optimal edge and line detectors [155]. One example here is the Deriche filter [156], where the even filter is a line detector, while the odd filter is an edge detector.

Quadrature filters in higher dimensions: First attempts to design quadrature filters in higher dimensions (e.g. 2d or 3d) are related to the directional Hilbert transform[5] [157] $\mathcal{H}_v(f)$ and straightforward extension of the analytic signal (Section 4.3.1) to 2d as complex function $f + i\mathcal{H}_v(f)$. Afterwards, the monogenic signal [71] extended the analytic signal and 1d quadrature filters to higher dimensions in an isotropic manner. The monogenic signal replaces the scalar-valued odd filter $\mathcal{H}_v(f)$ (as in 1d case) with a vector-valued one $(\mathcal{R}_1(f), \mathcal{R}_2(f))$ based on the Riesz transform achieving an isotropic extension.

Relation of the filter from Section 8.2.1 to quadrature filters: Our base filter from Section 8.2.1 can be seen as (anisotropic) higher dimensional quadrature filter [157] corresponding to the alternative characterization of quadrature filters as edge/line detectors. This is a consequence of its differential interpretation (Section 5.2). Furthermore, our base filter is invariant to the constant shifts in the gray values due to zero integral (Lemma 8). However, our filter has infinite support and is non-local due to the property of scale equivariance.

[4] Infinite support is a consequence of $e^{i\omega_0 x} = \sin(\omega_0 x) + i\cos(\omega_0 x)$, i.e. due to the periodicity of sine and cosine functions. Well-localized means that for x s.t. $||x - x_0||$ is large, the Gabor filter at x is approximately 0.
[5] Directional Hilbert transform $\mathcal{H}_v(f)$ is defined in Section 5.2 for $v \in \mathbb{R}^2$, $||v|| = 1$.

From the properties of the Riesz transform, our steered base filter can be used to extract signal characteristics for the signals that are oriented in the same (or close) to the orientation of the steered base filter. Steering base functions enable handling feature extraction for higher dimensional signals without a single preferred or dominant orientation.

G Introduction to scattering networks

Scattering networks are based on applying the wavelet transform and modulus non-linearity in a hierarchy creating a deep feature representation. Features from every level of the hierarchy are used to construct a group invariant representation by a local pooling operation on the group domain. Here, we give an overview of the locally translation invariant scattering network from [114].

First, the Morlet mother wavelet $\psi_M(x) : \mathbb{R}^2 \to \mathbb{C}$ is defined as an oriented bandpass complex function

$$\psi_M(x) = \alpha(e^{i\langle x,\xi\rangle} - \beta)e^{-|x|^2/(2\sigma^2)}.$$

The parameter $\xi \in \mathbb{R}^2, ||\xi||_2 = 1$ defines the orientation of the wavelet, while σ controls its scale. Furthermore, $\beta << 1$ is chosen such that $\int \psi_M(x)dx = 0$. The mother wavelet is rotated by $r \in SO(2)$ and scaled (dilated) by $2^j, j \in \mathbb{Z}$, to create diverse features:

$$\psi_\lambda(x) = 2^{-2j}\psi_M(2^{-j}r^{-1}x),$$

for $\lambda = 2^{-j}r$.

From now on, let $\mathcal{P} = \{2^j r \quad | \quad j \in \{1, \cdots, J\}, r \in G_M\}$ denote the space of all selected scale and rotation transformations[6] of the wavelet ψ_M. Next, non-linearity is needed to construct a (non-trivial) translation invariant representation[7] due to $\int \psi_M(x)dx = 0$, see [114] for details. We select the modulus of the complex number

$$A(y_r + iy_i) = \sqrt{y_r^2 + y_i^2},$$

as it is a pointwise nonexpansive non-linearity operator which preserves signal energy. For an image $f \in L_2(\mathbb{R}^2)$, features at depth 1 for rotation and scale $\lambda = 2^j r \in \mathcal{P}$ are extracted using the **scattering transform** operator $U_\lambda : L_2(\mathbb{R}^2) \to L_2(\mathbb{R}^2)$

$$U_\lambda f = A(f * \psi_\lambda).$$

The output of this scattering transform is called a **scattering coefficient**. To create a hierarchy of scattering coefficients, scattering transforms are applied to the scattering coefficients from the previous step. We refer to these as higher order scattering coefficients. Generally, for depth m we apply the scattering transform m times sequentially. This operator is defined by a sequence $p = (\lambda_1, \cdots, \lambda_m) \in \mathcal{P}^m$ of m scales and rotations via

$$U_p f = U_{\lambda_m} \cdots U_{\lambda_1} f.$$

Operators U_p for $p \in$ transform input image into the feature maps which have the same size as the input image. Hence, similarly as in Section 8.2.3 pooling operations are needed to achieve

[6] Here, J denotes the total number of scales to be considered, while $G_M = \{k\frac{\pi}{M}, k \in \{0, 1, \cdots, M-1\}\}$ is a finite rotation group where $M \in \mathbb{N}$ controls the discretization of the group. Generally, $\mathcal{P}$ is a discrete version of the more general space of scale and rotation transformations $\mathcal{P}^* = \{2^j r \quad | \quad j \in \mathbb{Z}, r \in SO(2)\}$.

[7] For image f, it follows $\int_{\mathbb{R}^2}(f * \psi_M)(x)dx = \int_{\mathbb{R}^2} f(y)\big(\int_{\mathbb{R}^2} \psi_M(x-y)dx\big)dy = 0$. Hence, for every f one gets the trivial translation invariance.

translation invariance. Based on the choice of the pooling operator one can achieve either global or local translation invariance [114].

To calculate (global) translation invariant scattering coefficients, a global pooling operator $\overline{S}_p$ can be applied

$$\overline{S}_p(f) = \int U_p f(x) dx.$$

In some applications, it is beneficial to have features invariant with respect to translations by up to 2^j pixels for $j \in \mathbb{N}$, where an optimal j can be determined through cross-validation. This local translation invariance can be achieved by convolution with a Gaussian kernel g_σ with $\sigma = 2^j$ which yields

$$S_p(f)(x) = (U_p f * g_\sigma)(x).$$

According to scale space theory, the convolution with g_σ removes all information on scales smaller than $\sigma = 2^j$. Hence, the representation can be downsampled by 2^j as local translation invariance up to 2^j pixels is guaranteed.

Finally, scattering coefficients can be written as $W_0(f) = f * g_{2^j}$, $W_1(f) = \big(S_\lambda(f)\big)_{\lambda \in \mathcal{P}}$, and $W_k(f) = \big(S_p(f)\big)_{p \in \mathcal{P}^k}$. Hence, the locally translation invariant scattering representation can be written compactly as

$$\Phi_K(f) = \Big(W_k(f) \quad | \quad k = 0, \cdots, K\Big).$$

The mapping Φ_K is called a **scattering network**. Scattering networks are claimed to be a proxy for convolutional neural networks as they use cascades of the same building blocks: convolutional filters, non-linearities, and pooling operators. The main difference is that convolutional filters in scattering networks are deterministic, i.e. no training procedure is required.

Next, we present two important properties related to signal properties which characterize a scattering network.

Energy preservation and Lipschitz continuity to small deformations: In [114] it was shown that scattering networks induce energy preservation in the sense that

$$||f||^2 = \sum_{k=0}^{\infty} \sum_{p, |p|=k} ||S_p(f)||^2.$$

Let $\tau : \mathbb{R}^2 \to \mathbb{R}^2$ be a small deformation displacement field, i.e., $||\nabla \tau||_\infty \leq 1$. Then for $\Phi_\infty = \cup_{k=0}^{\infty} W_k(f)$, $L_\tau(f)(x) = f(x + \tau(x))$ and under the assumption that g_{2^j}, ψ_M and their first two derivatives have a decay of $O\Big((1 + |x|)^{-(d+2)}\Big)$, Lipschitz continuity to small deformations τ holds. That is

$$||\Phi\big(L_\tau(f)\big) - \Phi(f)|| \leq C||f||\ ||\nabla \tau||_\infty. \tag{6}$$

These claims were proven by Mallat in [134].

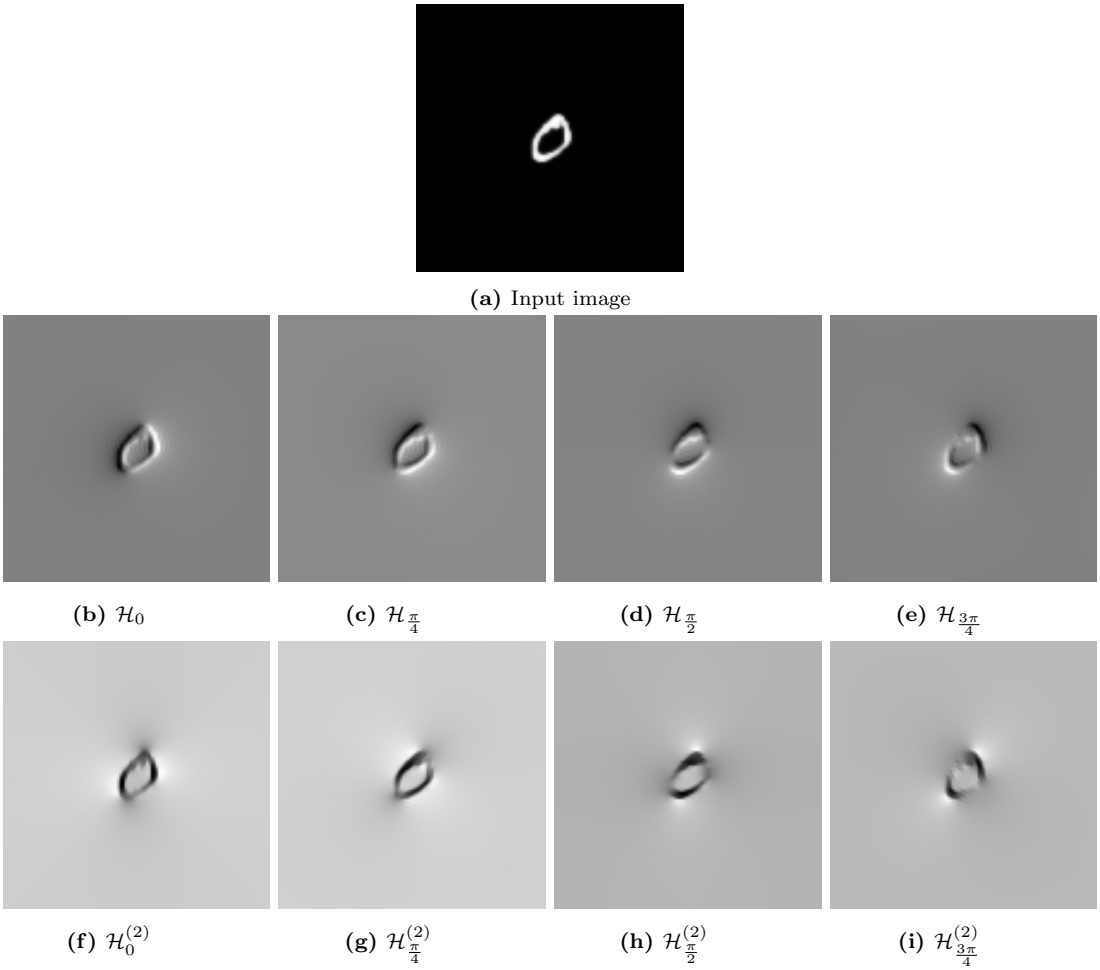
Lipschitz continuity to small deformations for Riesz feature representation:

The Riesz transform does not satisfy Mallat's decay assumption [134]. Instead, we derive Lipschitz continuity from non-expansiveness using a simple trick. In fact, it is enough to apply a smoothing operator which regularizes the representation. Formally, let $\gamma : L^2(\mathbb{R}^d) \to L^2(\mathbb{R}^d)$ be a smoothing operator that is Lipschitz continuous to small deformations, i.e. it satisfies equation (6). Let Φ be an arbitrary representation that is nonexpansive. Then $\Phi \circ \gamma$ is Lipschitz continuous to small deformations:

$$||\Phi(\gamma(f)) - \Phi\big(\gamma(L_\tau(f))\big)|| \leq ||\gamma(f) - \gamma\big(L_\tau(f)\big)|| \leq C||f||\ ||\nabla \tau||_\infty.$$

Hence, a smoothing operator like a convolution with a Gaussian kernel is needed to regularize the input. The smoothing factor σ should however not be too large in order to not destroy important structural information.

H Visualization of feature maps from Riesz representation

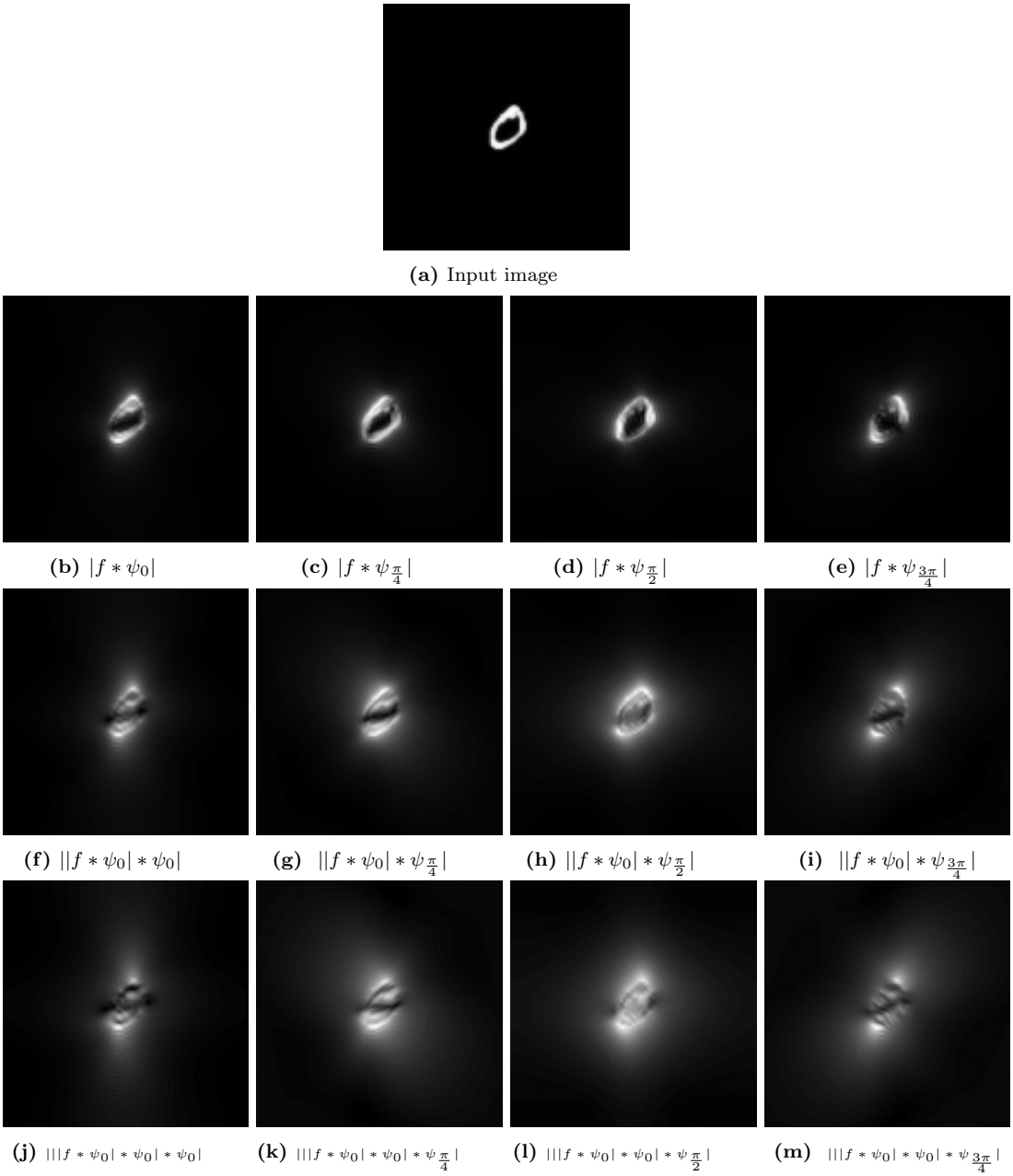


(a) Input image

(b) $\mathcal{H}_0$ **(c)** $\mathcal{H}_{\frac{\pi}{4}}$ **(d)** $\mathcal{H}_{\frac{\pi}{2}}$ **(e)** $\mathcal{H}_{\frac{3\pi}{4}}$

(f) $\mathcal{H}_0^{(2)}$ **(g)** $\mathcal{H}_{\frac{\pi}{4}}^{(2)}$ **(h)** $\mathcal{H}_{\frac{\pi}{2}}^{(2)}$ **(i)** $\mathcal{H}_{\frac{3\pi}{4}}^{(2)}$

Figure 16: Steered first and second order Riesz transforms applied to a digit from the MNIST Large scale dataset. White represents a high positive filter response, gray is close to 0, and black is a negative filter response. Here, we slightly abuse the notation by writing $\mathcal{H}_\phi$ rather than $\mathcal{H}_v$ for $v = (\cos\phi, \sin\phi)$. See Section 8.2.1 for details.



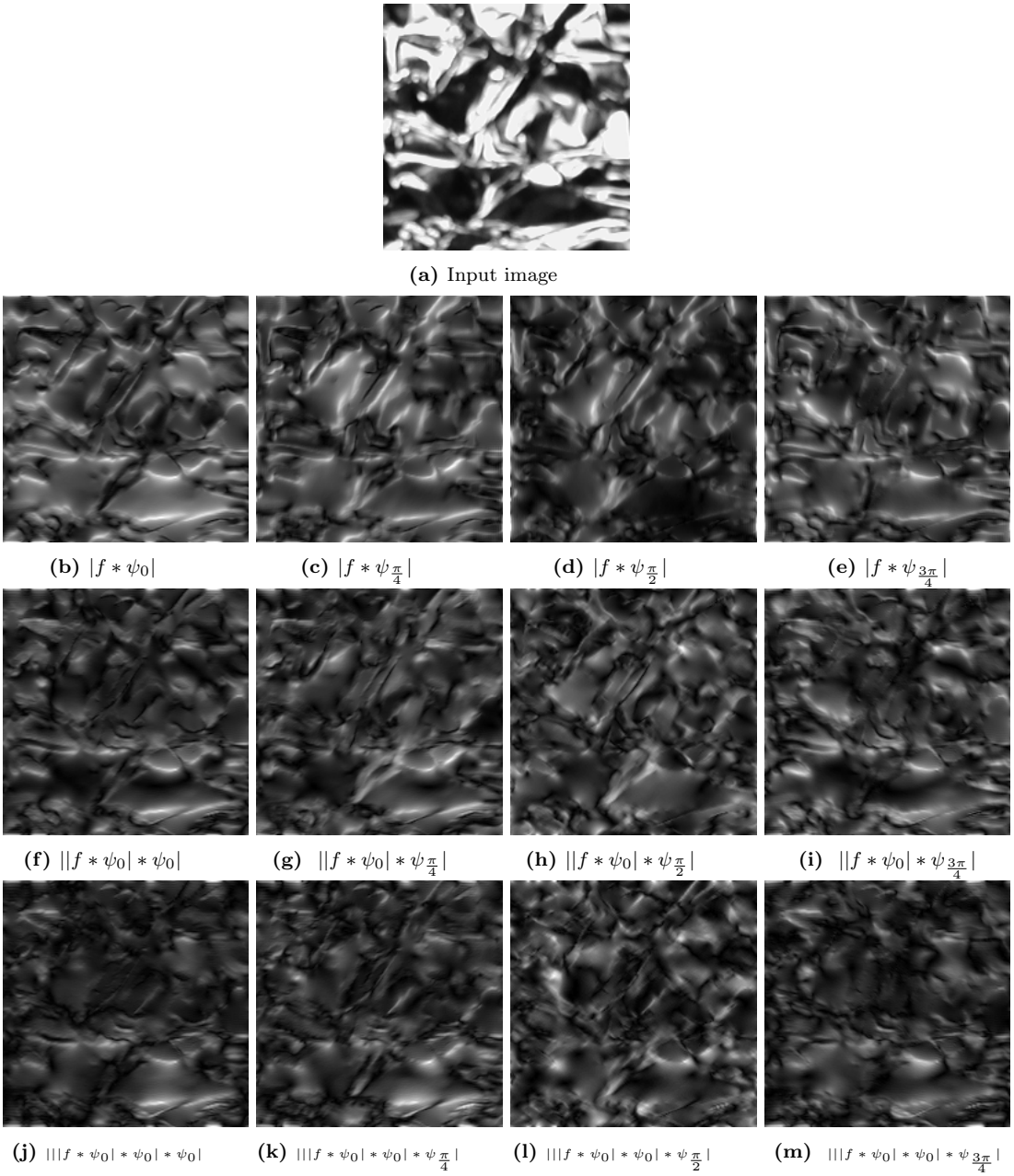
(a) Input image

(b) $|f * \psi_0|$ **(c)** $|f * \psi_{\frac{\pi}{4}}|$ **(d)** $|f * \psi_{\frac{\pi}{2}}|$ **(e)** $|f * \psi_{\frac{3\pi}{4}}|$

(f) $||f * \psi_0| * \psi_0|$ **(g)** $||f * \psi_0| * \psi_{\frac{\pi}{4}}|$ **(h)** $||f * \psi_0| * \psi_{\frac{\pi}{2}}|$ **(i)** $||f * \psi_0| * \psi_{\frac{3\pi}{4}}|$

(j) $|||f * \psi_0| * \psi_0| * \psi_0|$ **(k)** $|||f * \psi_0| * \psi_0| * \psi_{\frac{\pi}{4}}|$ **(l)** $|||f * \psi_0| * \psi_0| * \psi_{\frac{\pi}{2}}|$ **(m)** $|||f * \psi_0| * \psi_0| * \psi_{\frac{3\pi}{4}}|$

Figure 17: Riesz feature representation prior to pooling on a digit from the MNIST Large scale dataset. Top: input image f. Second to fourth row: Riesz representations at depth 1, 2 and 3, respectively. White corresponds to a high positive filter response, while black denotes a filter response close to 0.



(a) Input image

(b) $|f * \psi_0|$ **(c)** $|f * \psi_{\frac{\pi}{4}}|$ **(d)** $|f * \psi_{\frac{\pi}{2}}|$ **(e)** $|f * \psi_{\frac{3\pi}{4}}|$

(f) $||f * \psi_0| * \psi_0|$ **(g)** $||f * \psi_0| * \psi_{\frac{\pi}{4}}|$ **(h)** $||f * \psi_0| * \psi_{\frac{\pi}{2}}|$ **(i)** $||f * \psi_0| * \psi_{\frac{3\pi}{4}}|$

(j) $|||f * \psi_0| * \psi_0| * \psi_0|$ **(k)** $|||f * \psi_0| * \psi_0| * \psi_{\frac{\pi}{4}}|$ **(l)** $|||f * \psi_0| * \psi_0| * \psi_{\frac{\pi}{2}}|$ **(m)** $|||f * \psi_0| * \psi_0| * \psi_{\frac{3\pi}{4}}|$

Figure 18: Riesz feature representation prior to pooling on the aluminum texture from the KTH-tips dataset. Top: input image f. Second to fourth row: Riesz representations at depth 1, 2 and 3, respectively. White corresponds to a high positive filter response, while black denotes a filter response close to 0.

List of publications

Published papers:

- [1] **T. Barisin**, K. Schladitz, C. Redenbach, & M. Godehardt (2022). Adaptive Morphological Framework for 3D Directional Filtering. Image Analysis & Stereology, 41(1). https://doi.org/10.5566/ias.2639
- [100] C. Jung, F. Müsebeck, **T. Barisin**, K. Schladitz, C. Redenbach, M. Kiesche, & M. Pahn (2022). Towards automatic crack segmentation in 3d concrete images. In 11th Conference on Industrial Computed Tomography, Wels, Austria (iCT 2022). https://www.ndt.net/article/ctc2022/papers/ICT2022_paper_id225.pdf
- [42] **T. Barisin**, C. Jung, F. Müsebeck, C. Redenbach, & K. Schladitz (2022). Methods for segmenting cracks in 3d images of concrete: A comparison based on semi-synthetic images. Pattern Recognition, 129, 108747. https://www.sciencedirect.com/science/article/pii/S003132032200228X
- [126] **T. Barisin**, C. Jung, A. Nowacka, K. Schladitz, & C. Redenbach (2022). Crack segmentation in 3d concrete images: perspectives and challenges, NDT-CE 2022 - The International Symposium on Nondestructive Testing in Civil Engineering Zurich, Switzerland, August 16-18, 2022. https://www.ndt.net/article/ndtce2022/paper/60992_manuscript.pdf
- [124] C. Jung, A. Nowacka, **T. Barisin**, D. Meinel, O. Paetsch, S. Grzesiak, M. Salamon, K. Schladitz, C. Redenbach, & M. Pahn (2023.) 3d imaging and analysis of cracks in loaded concrete samples. 12th Conference on Industrial Computed Tomography, Fürth, Germany (iCT 2023). https://www.ndt.net/article/ctc2023/papers/Contribution_124_final.pdf
- [2] J. Lienhard, **T. Barisin**, H. Grimm-Strele, M. Kabel, K. Schladitz, & T. Schweiger (2023). Microstructure of 3D printed and molded glass fiber reinforced polypropylene and its influence on the mechanical properties. Accepted for publication in: Strain. https://doi.org/10.1111/str.12463
- [158] S. Grzesiak, **T. Barisin**, K. Schladitz & M. Pahn (2023). Analysis of the Bond Behaviour of a GFRP Rebar in Concrete by In-situ 3D Imaging Test. Accepted for publication in: Materials and Structures.

Submitted papers:

- [3] **T. Barisin**, K. Schladitz, & C. Redenbach (2023). Riesz network: scale invariant neural network in a single forward pass. https://arxiv.org/pdf/2305.04665.pdf
- [4] **T. Barisin**, J. Angulo, K. Schladitz, & C. Redenbach (2023). Riesz feature representation: scale equivariant scattering network for classification tasks. https://arxiv.org/pdf/2307.08467.pdf
- **T. Barisin**, C. Jung, A. Nowacka, C. Redenbach, & K. Schladitz. (2023) Cracks in concrete.

Relationship between the publications and the chapters of the thesis:

- Chapter 2: based on [1],
- Chapter 3: based on [2],
- Chapter 5: based on [3, 4],
- Chapter 6: based on [3]
- Chapter 7: datasets are from [42, 100, 124, 126],
- Chapter 8: based on [4].

Akademische Lebenslauf

September 2014 - Juli 2017: Zagreb, Kroatien	Bachelorstudium in Mathematik, Universität Zagreb.
September 2017 - Februar 2020: Zagreb, Kroatien	Masterstudium in Mathematischer Statistik, Universität Zagreb.
September - Dezember 2022: Fontainebleau, Frankreich	Forschungsaufenthalt am Zentrum für Mathematische Morphologie (CMM), Mines Paris, PSL Universität. Betreuer: Dr. Jesus Angulo (HDR).
seit März 2020: Kaiserslautern, Deutschland	Promotionsstudium (Dr. rer. nat.) in Mathematik am RPTU Kaiserslautern-Landau und Fraunhofer ITWM. Betreuer: Prof. Dr. Claudia Redenbach.

Academic Curriculum Vitae

September 2014 - July 2017: Zagreb, Croatia	Bachelor's degree (BSc) in Mathematics, University of Zagreb.
September 2017 - February 2020: Zagreb, Croatia	Master's degree (MSc) in Mathematical Statistics, University of Zagreb.
September - December 2022: Fontainebleau, France	Research stay at Center for Mathematical Morphology (CMM), Mines Paris, PSL University. Supervisor: Dr. Jesus Angulo (HDR).
since March 2020: Kaiserslautern, Germany	Doctoral degree (PhD, Dr. rer. nat.) in Mathematics at RPTU Kaiserslautern-Landau and Fraunhofer ITWM. Doctoral advisor: Prof. Dr. Claudia Redenbach.