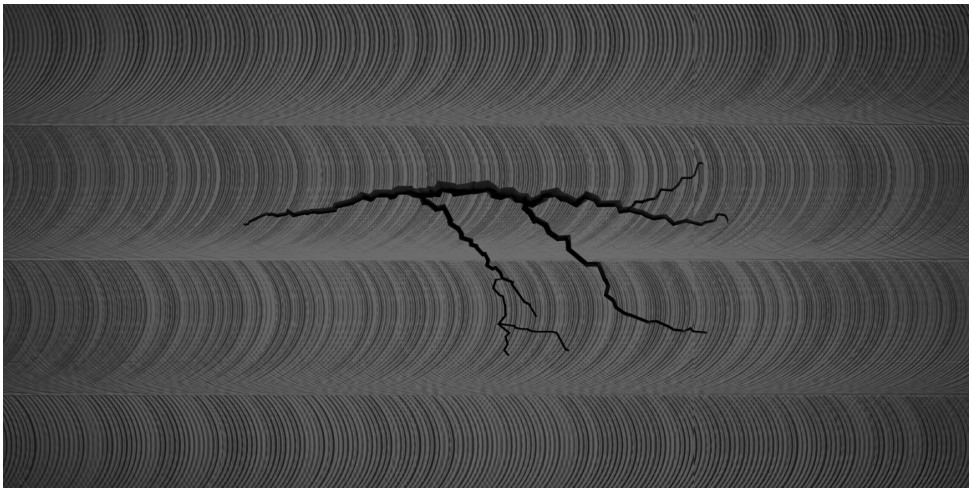
Stochastic Geometry Models for Metallic Surface Textures

Natascha Jeziorski



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

Erstgutachterin:	Prof. Dr. Claudia Redenbach
Zweitgutachterin:	Prof. Dr. Ute Hahn
Datum der Disputation:	12.12.2025

D-386

RPTU Rheinland-Pfälzische Technische Universität Kaiserslautern Landau

Fraunhofer ITWM

Danksagung

Diese Arbeit wurde von dem Bundesministerium für Bildung und Forschung (BMBF) durch das Projekt SynosIs [Fördernummer 01IS21058] und dem Ministerium für Wissenschaft und Gesundheit des Landes Rheinland-Pfalz durch das Projekt eQuality unterstützt. Die Deutsch-Französische Hochschule (DFH) hat den regelmäßigen Austausch mit Nachwuchswissenschaftlerinnen und Nachwuchswissenschaftlern im Rahmen des DFH-Graduiertenkollegs zur Mathematischen Bildverarbeitung ermöglicht.

Zuerst möchte ich mich bei meiner Betreuerin Frau Prof. Dr. Claudia Redenbach bedanken, dass sie mir die Möglichkeit für diese Promotion gegeben hat. Ihre Tür stand jederzeit offen, sei es um Fragen zu diskutieren, Probleme zu lösen oder Zweifel zu beheben. Während meines Bachelor- und Masterstudiums an der TU Kaiserslautern hat sie mein Interesse an der mathematischen Bildverarbeitung und der stochastischen Geometrie geweckt. Beides konnte ich während des Promotionsstudiums weiter vertiefen.

Ein großer Dank geht an die AG Statistik der RPTU Kaiserslautern-Landau unter der Leitung von Frau Prof. Dr. Claudia Redenbach, dass meine Promotion durch das Projekt SynosIs initiiert wurde. Außerdem bedanke ich mich bei der Abteilung Bildverarbeitung des Fraunhofer ITWM unter der Leitung von Herrn Markus Rauhut und Frau Dr. Henrike Stephani, die meine Promotion durch das Projekt eQuality weiter unterstützt hat.

Des Weiteren bedanke ich mich bei allen (ehemaligen) Mitarbeitenden der AG Statistik, die ich während meiner Promotionszeit dort kennenlernen durfte. Bei den täglichen Kaffeepausen hatten wir viele interessante Gespräche, die zu einer lockeren und freundlichen Arbeitsatmosphäre beigetragen haben. Auch möchte ich mich bei allen (ehemaligen) Mitarbeitenden der Abteilung Bildverarbeitung bedanken, vor allem die der Arbeitsgruppe virtuelle Inspektionsplanung, die ihr Wissen in der Computergrafik stets mit mir geteilt haben und mich somit auch für dieses Gebiet faszinieren konnten. Ein großer Dank geht zudem an die Mensa-Gruppe, mit der ich fast täglich die Mittagspause verbracht habe.

Außerdem bedanke ich mich bei allen Personen, die einen Teil dazu beigetragen haben, dass diese Promotion so entstehen konnte – sei es durch das Bereitstellen von Daten, durch Hilfe bei technischen Schwierigkeiten, durch das Teilen ihres Wissens, inspirierender Ideen oder konstruktiver Diskussionen, durch Korrekturlesen von Papern und der Dissertation oder einfach durch wertvolle Gespräche und motivierende Worte.

Darüber hinaus geht ein großer Dank an meine ehemalige Mathematik-Lehrerin Frau Brigitte Lang. Sie hat schon früh meine Begeisterung an der Mathematik erkannt und mich dahingehend bestärkt, ein Studium in diesem Fachgebiet anzustreben. Ihre ermutigenden, aber auch ehrlichen Worte haben vor allem die Anfangszeit des Studiums sehr erleichtert.

Ebenso möchte ich mich bei meinen Freunden bedanken, dass sie immer für mich da sind und mir mit Rat und Tat zur Seite stehen und für die vielen schönen Zeiten und unvergesslichen Momente die wir schon zusammen verbracht haben und sicherlich noch verbringen werden.

Zuletzt geht der größte Dank an Christian und unsere Familien, für ihre Geduld, Unterstützung und Liebe. Sie haben mich immer in meinen Ideen bestärkt, mich zu sämtlichen sportlichen Aktivitäten begleitet, mich in schwierigen Zeiten getröstet und mit mir in glücklichen Momenten gefeiert. Das alles hat mich zu dem Menschen gemacht, der ich heute bin. Dafür bin ich ihnen unendlich dankbar! ♡

Abstract

In industry, automatic visual surface inspection is a common technique for quality control. Machine learning methods provide promising results for automated defect detection. Data in sufficient quantity and quality is needed to train robust models. Real image data, especially of defective surfaces, is commonly not available in the needed amount. For rarely produced objects, not all critical defect types may be included in the data set. In addition, real data has to be annotated manually. Synthetic data offers a suitable alternative. Therefore, the imaging environment including the camera, the light source and the object to be inspected as well as the imaging method have to be simulated. In this thesis we consider the details of the digital twin of the object to be inspected.

We focus on metallic objects which are strongly reflective. Therefore, the visibility of surface defects highly depends on the surface texture. Hence, part one of this thesis deals with the simulation of two different surface textures resulting from the following surface processing methods: sandblasting and linear/spiral milling. Two different models have been developed based on real topography measurements. To model sandblasted surfaces, two steps are needed. To generate textures that cover the entire object surface, small patches are created first using the random phase noise (RPN). Then, those are stitched using an adaptation of the method from Efros and Freeman. Textures of milled surfaces are modeled by a multi-step model, considering the milling pattern, the movement of the milling head and the overlap of neighboring milling paths. Real parameters are included into the model.

Part two of this thesis deals with the simulation of synthetic 3d defect geometries which can then be added to an object geometry. We restrict to common defect types that appear for cast objects, for example cracks, rat tails or lift-offs of the surface top layer. Parametric, defect-specific models are developed. In general, 2d Voronoi tessellations build the basis for the models since they reflect the poly-crystalline micro-structure of metal. The tessellations are used to model defects on different scales and to ensure variability also when using the same parameter configuration due to the underlying random point processes. The diversity of defect geometries within a defect class is obtained by different parameter configurations.

Zusammenfassung

In der Industrie ist die automatische visuelle Oberflächeninspektion ein etabliertes Verfahren zur Qualitätskontrolle. Methoden aus dem maschinellen Lernen liefern vielversprechende Resultate bei der automatischen Defekterkennung. Um robuste Modelle zu trainieren, werden Daten in ausreichender Quantität und Qualität benötigt. Reale Bilddaten liegen häufig nicht in der benötigten Menge vor, das gilt vor allem für Bilder von defekten Oberflächen. Bei Werkteilen mit niedriger Produktionszahl kann es zudem vorkommen, dass nicht alle kritischen Defekttypen in dem Datensatz vorliegen. Zusätzlich müssen reale Daten manuell annotiert werden. Synthetische Daten bieten eine aussichtsreiche Alternative. Hierzu werden die Bildgebungsumgebung, die aus der Kamera, der Lichtquelle und dem zu inspizierenden Objekt besteht, und das Bildgebungsverfahren simuliert. Diese Arbeit befasst sich mit den Details des digitalen Zwillings des zu inspizierenden Objektes.

Wir fokussieren uns auf metallene und somit stark reflektierende Objekte. Die Sichtbarkeit von Oberflächendefekten ist demzufolge stark von der jeweiligen Oberflächentextur abhängig. Der erste Teil dieser Arbeit befasst sich deswegen mit der Simulation von zwei verschiedenen Oberflächentexturen, die aus den folgenden Oberflächenverarbeitungsprozessen resultieren: Sandstrahlen und lineares/spiralförmiges Fräsen. Dazu wurden zwei verschiedene Modelle entwickelt, beruhend auf realen Topografiemessungen. Das Modell für sandgestrahlte Oberflächen besteht aus zwei Schritten. Um eine Oberflächentextur zu erzeugen, welche die komplette Objektoberfläche abdeckt, werden zuerst kleinere Flächen des finalen Texturbildes durch die Methode „random phase noise“ (RPN) generiert. Diese werden anschließend mit einer adaptierten Version der Methode von Efros und Freeman aneinandergefügt. Die Topografien von gefrästen Oberflächen werden mittels eines mehrstufigen Modells erzeugt, welches das Fräsmuster, die Fräskopfbewegung und die Überlappung von Fräspfaden berücksichtigt. Reale Prozessparameter sind in das Modell inkludiert.

Der zweite Teil der Arbeit befasst sich mit der Simulation von synthetischen 3d Defekt-Geometrien, die anschließend in die Objekt-Geometrie eingefügt werden können. Hierbei haben wir uns auf typische Defekttypen beschränkt, die bei gegossenen Werkteilen auftreten können, beispielsweise Risse, Rattenschwänze oder Oberflächenablösungen. Dazu wurden parametrische, defekt-spezifische Modelle entwickelt. Als allgemeine Modell-Grundlage dienen 2d Voronoi-Diagramme, da sie die polykristalline Mikrostruktur von Metall widerspiegeln. Diese werden benutzt, um die Oberflächendefekte auf unterschiedlichen Größenskalen zu modellieren und um die Variabilität auch bei gleicher Parameterkonfiguration wegen der zugrundeliegenden zufälligen Punktprozesse zu gewährleisten. Die Diversität von Defekt-Geometrien innerhalb einer Defektklasse wird durch unterschiedliche Parameterkonfigurationen erzielt.

Contents

1 Introduction **1**

2 Motivation – Synthetic Data for Visual Surface Inspection **3**
2.1 Visual Surface Inspection . . . 3
2.1.1 Challenges in Visual Surface Inspection . . . 5
2.2 Automatic Defect Detection . . . 6
2.2.1 Methods for Automatic Defect Detection . . . 7
2.2.2 Training Data for Machine Learning Approaches . . . 8
2.3 Synthetic Data Generation Using a Virtual Inspection Environment . . . 9
2.3.1 Quality Check of Synthetic Data . . . 14

3 Basics **15**
3.1 Mathematical Fundamentals . . . 15
3.1.1 General Definitions . . . 15
3.1.2 Image Processing . . . 15
3.1.3 Fourier Analysis . . . 17
3.1.4 Filtering . . . 18
3.1.5 Steerable Pyramid . . . 20
3.1.6 Haralick Features . . . 22
3.1.7 Stochastic Geometry Basics: Point Processes . . . 23
3.1.8 Stochastic Geometry Basics: Tessellations . . . 26
3.1.9 Basics from Graph Theory . . . 27
3.1.10 Spline Interpolation . . . 27
3.2 Fundamentals of Computer Graphics . . . 28
3.2.1 3d Object: Shape . . . 28
3.2.2 3d Object: Material . . . 29
3.2.3 3d Object: Texture . . . 30
3.2.4 Light . . . 31
3.2.5 Camera . . . 32
3.2.6 Rendering . . . 33

I Surface Texture Modeling **35**

4 Introduction **37**
4.1 Related Work . . . 38
4.2 Our Contribution . . . 39
4.3 Measurement Data . . . 40

5 Model for Sandblasted Surfaces **43**
5.1 Texture Generation Methods . . . 43
5.1.1 Asymptotic Discrete Spot Noise . . . 44
5.1.2 Random Phase Noise . . . 45
5.1.3 Method of Heeger and Bergen . . . 45

5.1.4 Method of Portilla and Simoncelli . . . 46
5.1.5 Comparison of Methods . . . 46
5.2 Simulating Texture Images of Arbitrary Size and Pixel Spacing . . . 48
5.2.1 Stitching Method by Efros and Freeman . . . 50

6 Model for Milled Surfaces 53
6.1 Sub-Model for Ring Appearance . . . 53
6.1.1 Shape . . . 55
6.1.2 Tilting . . . 56
6.1.3 Noise . . . 57
6.2 Sub-Model for Ring Interaction . . . 58
6.3 Sub-Model for Tool-Path . . . 60
6.3.1 Parallel Pattern . . . 60
6.3.2 Spiral Pattern . . . 61
6.4 Fitting the Model to Data . . . 62
6.5 Model-Adaptations to Simulate Polished Surfaces . . . 66
6.5.1 Appearance of Polishing Rectangles . . . 66
6.5.2 Placement of Polishing Rectangles . . . 67

7 Conclusion 69

8 Dual Data Set of Real and Synthetic Images 71

II Defect Modeling 77

9 Introduction 79

10 Dent Defects 81
10.1 Introduction . . . 81
10.2 Data . . . 81
10.3 Modeling . . . 81
10.3.1 2d Shape . . . 84
10.3.2 Location of Minimum and Maximum . . . 85
10.3.3 3d Shape . . . 85
10.3.4 Sampling . . . 88
10.3.5 Noise . . . 89
10.3.6 Voronoi-Approximation of Defect Surface . . . 89
10.4 Conclusion . . . 90

11 Casting Defects 95
11.1 Introduction . . . 95
11.2 Models for Elongated Defects . . . 96
11.2.1 Details of Individual Modeling Steps . . . 97
11.2.1.1 Computation of a Path . . . 97
11.2.1.2 Dilation of a Path . . . 99
11.2.1.3 Transition from 2d to 3d . . . 100
11.2.1.4 Triangulation of 2d Surface in 3d . . . 101
11.2.2 Defect-Specific Adaptations . . . 103
11.2.2.1 Cold Crack and Hot Tear . . . 103
11.2.2.2 Bulge . . . 105

11.2.2.3 Buckle . . . 105
11.2.2.4 Coat Lift . . . 107
11.2.2.5 Cold Shut and Rat Tail . . . 108
11.3 Model for Scab Defects . . . 109
11.3.1 Voronoi Tessellation in the Dilated Window . . . 110
11.3.2 Refinement of Cell Borders . . . 111
11.3.3 Triangulation of Fine Cells . . . 111
11.3.4 Assignment of Height Values . . . 111
11.3.4.1 Elevation of Outer Areas $H_{\text{elev}}(\cdot)$ . . . 112
11.3.4.2 Texture Height $H_{\text{tex}}(\cdot)$ . . . 112
11.3.5 Computation of 3d Mesh . . . 113
11.4 Conclusion . . . 114

12 Adaptations and Expansions of Certain Casting Defect Models 115
12.1 Model Adaptation to Other Point Process Types for Generating the Voronoi Tessellation . . . 115
12.2 Alternative to Model Buckle Defects . . . 117
12.3 Modeling Progressive Defecting by Growing 3d Mesh Objects . . . 120
12.3.1 Related Work . . . 120
12.3.2 Modeling 3d Crack Objects at Different Time Steps . . . 120
12.3.3 Modeling 3d Buckle Objects at Different Time Steps . . . 123

13 Conclusion 125

Scientific career 139
List of publications . . . 139
Curriculum vitae . . . 141

1 Introduction

Visual surface inspection is a common method in industry used for quality control of manufactured objects. A typical system consists of a camera and a light source to scan the object surface. Automatic defect detection is then performed on the acquired images. Nowadays, such methods are often based on machine learning approaches which are trained on data pairs consisting of an acquired image and its labeled version. However, real data is often not available in sufficient quantity and quality to train robust defect detection algorithms. Synthetic data provides a promising alternative. Here, we use rule-based methods to create realistic data resembling the images acquired in the real inspection environment. The applied methods take into account simulations of the camera, the light source, the object to be inspected and the image acquisition process. Having realistic digital twins of manufactured parts with and without defects is crucial. This includes an appropriate representation of their geometry, surface topography and possible defects. The surface topography in µm-scale defines the surface texture and is thus important as it influences the visibility of defects.

In this thesis we deal with stochastic modeling of surface textures and different types of surface defects, for example dents, cracks and delaminations.

The surface texture usually results from the applied surface finishing method. Therefore, in Part I of this thesis, we introduce texture models of two different processing methods based on topography measurements, namely sandblasting in Chapter 5 and milling in Chapter 6. As the resulting surface patterns differ significantly, separate modeling approaches for the two cases are used. Sandblasted surfaces are modeled by a combination of data-based texture synthesis methods from computer graphics and rely thus entirely on the measurements. In contrast, textures of milled surfaces are obtained by a parametric, multi-step model that includes all process-related parameters known from the machine settings. This work was supported by the SynosIs project.

In Part II of this thesis we consider simulation models for various types of surface defects, supported by the projects SynosIs and eQuality. First, models for dent defects are investigated in Chapter 10, taking into account the shape of the associated defecting tool. Then, we propose models for different casting defects in Chapter 11. These models are based on Voronoi tessellations, separations of space into disjoint cells, to include the polycrystalline micro-structure of metal on a coarse scale. In Chapter 12, different adaptations and expansions of the casting defect models are presented, for example the simulation of an object that is exposed to progressive defecting such as cracking. Although the proposed models are motivated by defect geometries for cast metal objects, they can be transferred to other manufacturing processes.

2 Motivation – Synthetic Data for Visual Surface Inspection

Before starting with the main topic of this thesis, namely stochastic models for surface textures and surface defects, we want to present the area of application of the proposed methods to motivate their importance.

2.1 Visual Surface Inspection

Every industrially produced object needs to be reviewed during its manufacturing process to ensure product-specific quality standards. Sometimes more than one quality check throughout the production process is necessary. Particularly, safety-critical components have to be controlled intensively to guarantee and report their intactness after every production step. Defective parts are thus sorted out in an early stage which also reduces material wastage, especially if the scrap can be recycled. However, some defects can also be fixed by intermediate post-processing such that the product can pass the next manufacturing steps. In general, the demands on inspection systems are robustness, reliability, objectivity and speed.

Non-destructive testing (NDT) methods are preferred inspection techniques since they do not alter the product's functionality while examining. Examples are magnetic particle inspection (MPI), ultrasonic testing, radiography, computed tomography (CT) and visual inspection. The decision for the most suitable NDT method depends on the circumstances and the product areas to be inspected. If needed, different NDT methods can be combined. Radiography or CT are chosen if the interior of the product should be analyzed. The other three methods are favored to check the product surface. Especially MPI and ultra sonic testing can additionally inspect sub-surface areas. However, all methods except visual inspection require a special test equipment or a pre-treatment of the product. In this thesis we mainly focus on visual inspection systems. Note that, although we focus on product quality, visual inspection can also be used for the maintenance of manufacturing machines.

Product quality is a broad term and has to be defined specifically for every object. Usually, certain standards and tolerances are determined. Common defect classes are for example material defects (corrosion, enclosed air bubbles), surface defects (dents, scratches) and dimensional deviations from the optimal product geometry. Typical defect types that can be found by visual inspection affect the object shape and surface. In this thesis we restrict to surface defects.

Traditionally, visual inspection is performed manually by human eye. Inspection experts are trained to detect the production-specific defects and anomalies. However, manual inspection can be slow because humans need recovery breaks to maintain the attentiveness. To prevent the inspection process from pausing, more qualified manpower is needed which leads to higher costs. Moreover, the process is subjective since the decision whether a product is defective depends on the individual perception and assessment of the worker. Also, there is a size limit of detectable defects even when using magnification.

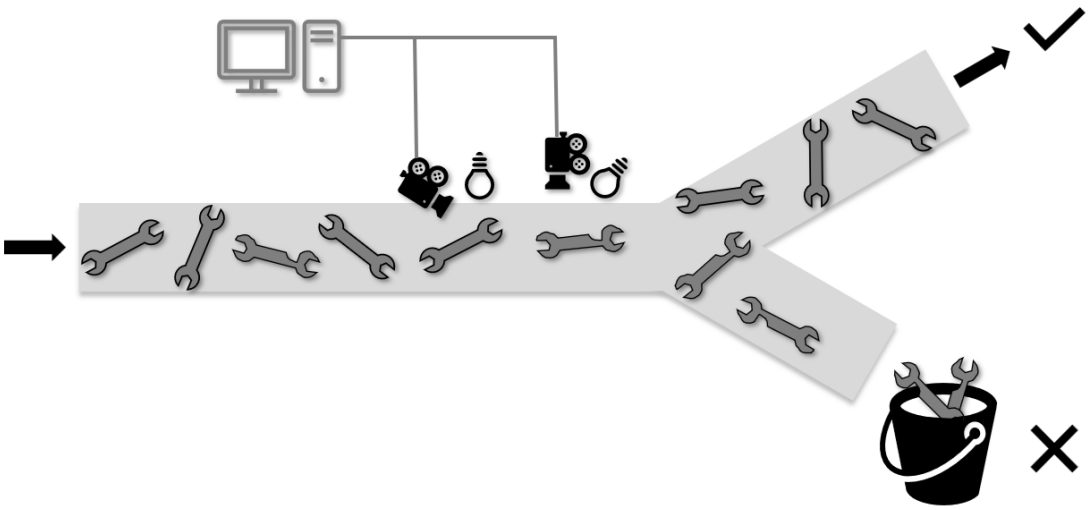
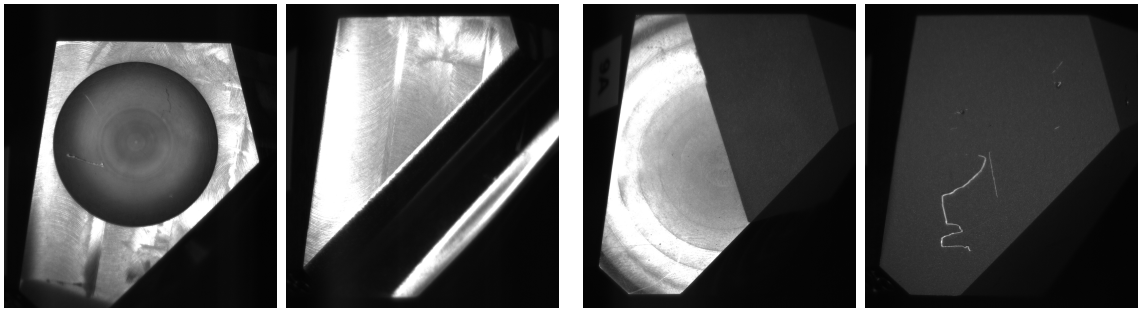


Figure 2.1: Schematic drawing of automatic visual surface inspection for objects lying on a conveyor belt.

To avoid the dependency on human perception, camera systems can be used. Then, the minimum detectable defect size relies on certain camera properties, namely its focal length, its distance to the object to be inspected and the resolution of its sensor. The acquired images must be evaluated, which can be done manually or automatically by image processing algorithms. Fully automatic defect detection is objective and reduces manpower, which leads to a decrease of costs and increase of speed. Methods for automatic defect detection are discussed in more detail in Section 2.2.

Besides the camera, the acquisition hardware includes an illumination element which typically consists of a suitable light source. Both components have to be placed in relation to the object to acquire optimal images. Therefore, several product characteristics influencing the surface light reflection have to be taken into account, for example its geometry, material and surface topography. To examine a specified inspection area, a single camera-light-position is usually insufficient. Thus, several compatible positions have to be investigated such that their united coverage area includes the whole inspection area. Once the product-specific inspection positions are determined, they stay fixed. A more in-depth description of the challenges in visual surface inspection is provided in Section 2.1.1.

In industry, there are two common procedures to pass an object through all specified camera-light-positions: conveyor belt or robot arm. For the first option, the products lie on a conveyor belt transporting them through an inspection line that consists of several fixed camera-light-combinations placed at their defined positions. A schematic drawing of that procedure is given in Figure 2.1. Here, the products are moving while the camera-light-combinations are fixed. Using a robot arm usually reverses that interaction, meaning that the product position is fixed while a robot arm with a camera-light-combination moves around the object. Its motion is prescribed so that images are taken from all specified positions. To ensure the exactly determined relations between the product and the camera-light-combination in both procedures, an unchanging object positioning and a reliable calibration of the robot arm or the conveyor belt are crucial.



(a) Different object geometries with parallel milled surfaces: milled hole (left), milled notch (right).

(b) Different surface processing methods: mix of spiral milling and sandblasting (left), sandblasting (right).

Figure 2.2: Images of metallic objects of varying geometry and differently processed surfaces. All objects contain various surface defects. All images are taken in the same acquisition environment using a ring light and a single-channel grayscale camera. Objects were created by Fraunhofer IOF and image acquisition was performed by Juraj Fulir from Fraunhofer ITWM.

2.1.1 Challenges in Visual Surface Inspection

To obtain high quality images during acquisition, the parameters and placements of both camera and illumination have to be chosen depending on certain product features. These are the object material, the object geometry and the object surface topography on different scales.

The object *material* determines the amount of light that is absorbed or reflected. For example, metals are more reflective than plastic materials. This affects the choice of the optimal exposure time and light intensity such that the object surface details are visible in the acquired images. Moreover, the material determines the object color which influences the decision for an appropriate camera. If surface discoloration is a possible defect type, then a multi-channel camera, e.g. red, green, blue (RGB), could be favored since it takes multi-color images. Otherwise, a single-channel camera acquiring monochromatic grayscale images could be sufficient.

The way light is reflected depends on the surface topography and affects thus the amount of visible area in the acquired data. To describe the surface topography, we distinguish between three different scales: macro, meso and micro scale [35]. The macro scale is defined by the *object geometry* and large geometrical features such as defects and chamfered edges. Figure 2.2 shows how different defect shapes influence the light reflectivity and hence their visibility. The impact of different object geometries is presented in Figure 2.2a. It is obvious that certain areas are not adequately represented and can therefore not be inspected satisfactorily. Micro scale structures are so small that they are not separately distinguishable by the imaging sensor but are contributing to the light reflection. Usually, those structures are formed by the micro-structure of the material which determines the distribution of reflected light. Finally, all remaining surface structures of intermediate size define the meso scale. They are much smaller than macro scale but larger than micro scale features. As discussed by Dorsey et al. [24], meso scale structures are at least larger than the wavelength of visible light. *Surface processing methods* are often responsible for meso scale surface structures since they change the surface topography in µm-scale. The rougher the resulting surface structure, the wider the light scattering which leads to a darker appearance of the object surface in the acquired image. Figure 2.2b shows the

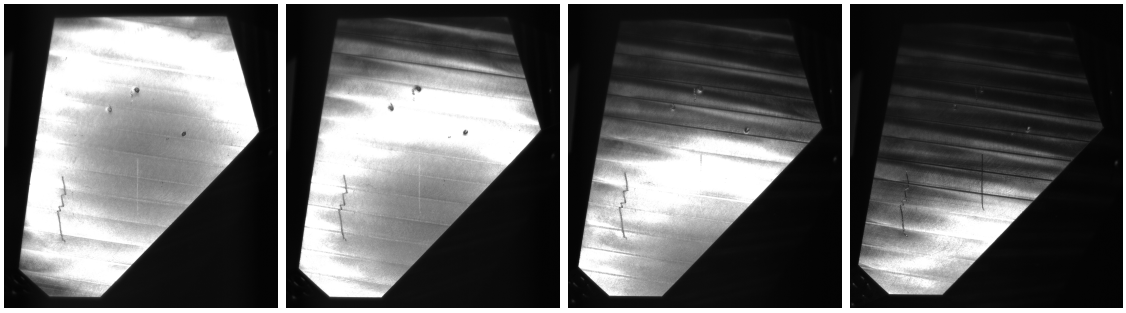


Figure 2.3: The visibility of defects depends on the viewpoint. Images of the same object but taken at different angles. The object was created by Fraunhofer IOF and image acquisition was performed by Juraj Fulir from Fraunhofer ITWM.

influence of two different surface finishings.

As the appearance of the object surface in the acquisition images is mainly determined by the object material, the object geometry and the surface processing methods, those factors also influence the visibility of surface defects which is crucial for their detection. In addition, the shape of surface defects affects the local light reflection as well. Changing the position of the camera-light-combination by varying the angle to the object surface affects the appearance and thus the visibility of surface defects. See Figure 2.3 for an example.

From now, we call a position of a camera-light-combination a viewpoint. Various viewpoints are needed to guarantee the completeness of the acquisition data, that is the coverage of the inspection area and the visibility of all detectable surface defects. To increase speed, as few viewpoints as possible should be used without loosing the coverage of the area. However, with increasing complexity of the object geometry, the specification of optimal viewpoints gets more difficult. Here, manual visual inspection is advantageous because the experts can either rotate the product manually or change their relative position to the object instinctively. Now, human intuition to obtain the optimal observation positions has to be digitized. Selecting digital viewpoints manually is cumbersome and time-consuming. Thus, an automatic optimal viewpoint specification solves that problem as investigated by Gospodnetić at al. [41]. When using a robot arm as inspection basis, the optimal order of the viewpoints has to be determined to optimize the arm's movement. After designing the inspection plan, that is, the placement and temporal order of viewpoints, it stays fixed and is repeated constantly for every part.

2.2 Automatic Defect Detection

In the manufacturing industry, the automation of inspection procedures progresses constantly. Besides the automatic data acquisition, their automatic processing in terms of defect detection is required. Both together form an automated inspection system. Note that the development of automatic defect detection algorithms is an ongoing research topic. However, the drawback of fully automated inspection systems is their specialization to one individual application. For every product and each inspection step, a product-specific inspection environment and a defect detection approach have to be developed. It is challenging, if even possible, to generalize those customized procedures to make them practicable for other applications. Here, manual inspection is again advantageous because experts can often adapt their knowledge quickly to other situations, for example other

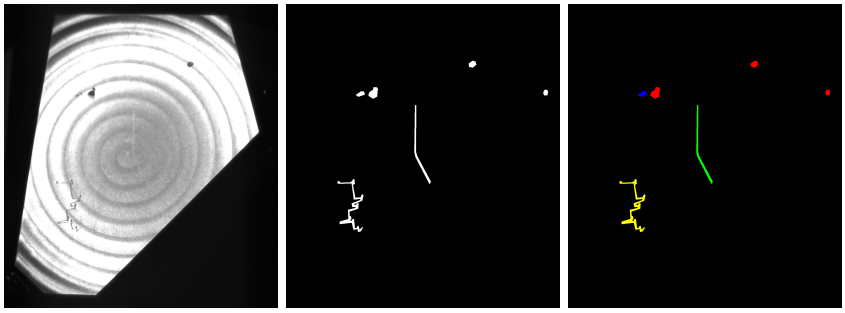


Figure 2.4: Examples for binary (center) and multi-class labeling (right) of an acquired image (left). The object was created by Fraunhofer IOF. Image acquisition and labeling was performed by Juraj Fulir from Fraunhofer ITWM.

object geometries, different materials or varying defect types. But, when automatic inspection systems work reliably, they are superior to manual inspection in terms of speed, objectivity and consistency.

When developing automatic defect detection methods, two steps have to be performed: localizing and classifying the possible defects. In the first step, all structures similar to the relevant defect types have to be found. Then, those selected regions have to be classified to the appropriate defect type if more than one type is considered. Moreover, it has to be decided whether the detected region marks a truly defective area on the product. Other commonly occurring material structures sometimes lead to similar appearances in the acquired data, for example small cracks and lenticular air pores in 3d CT images of concrete or specular bumps originating from the surface topography and dents in visual inspection of metal surfaces. In addition, prescribed defect-specific tolerances have to be taken into account during the decision making. For example, anomalies that are too small with regard to the given defect standards are not classified to be erroneous but can be found in the defect localization step. Thus, defect detection is specified to the product characteristics and the defect types to be detected.

2.2.1 Methods for Automatic Defect Detection

The decision making of defect detection experts is based on the extensive training they ran through and their experience gathered over time. That expertise has now to be transferred to automatic systems. We restrict to NDT methods that generate image data. Different approaches for defect localization exist: traditional image processing and machine vision methods. For defect classification machine learning approaches are commonly applied.

Note that labeled image data is needed when considering learning-based approaches. That is, the data set contains image pairs consisting of the actual image and its labeled version. Labeling refers to marking pixels or the area that define a defect. The marking process is also called annotation. We distinguish between binary and multi-class labels, see Figure 2.4 for examples. Binary labeling just differentiates between defective and defect-free and is thus sufficient to train defect localization methods. In contrast, multi-class labeling contains different labels marking different defect types and is needed for defect classification algorithms. Typically, annotation is done manually which requires the knowledge of domain experts and is extremely cumbersome and time-consuming. Especially for 3d image data, manual annotation is almost impossible.

Typical traditional image processing methods for extracting possibly defective regions are thresholding, edge detection, filtering and morphological operations. Xie [119] dis-

cusses various traditional image processing methods for detecting texture and discoloration defects on varying materials. Luo et al. [68] published a detailed overview of different methods to recognize surface defects on flat steel products in visual image data. Their summary includes classical and machine learning approaches.

Traditional methods have to be manually designed and customized to the present application by identifying characteristic features for the critical defect types. Adapting those systems to changes in the inspection setup is costly since they always have to be redesigned. Supervised learning methods offer a promising alternative as they use a general model which is adapted to specific input data via its training. Thus, it is adjustable to other situations by replacing or adapting the input. Also, it removes human biases from designing the defect detection method. Note that input data in sufficient quality and quantity is required to train reliable models. Thus, the adaptation of supervised learning methods to different circumstances is feasible just if enough valuable annotated data is available.

Independent of the best localization technique, a classification of the critical areas is required. Examples of two-step models for defect detection based on learning methods are given in [67, 100]. A detailed overview of segmentation approaches and classification algorithms is provided by Tang et al. [101].

To summarize, learning-based defect detection methods are in focus of recent research and gain more and more interest. Also, when considering other NDT methods, automatic data processing is widespread. Examples are automatic defect detection in welded objects using radiography [97,106], crack segmentation in concrete using computed tomography [5, 58], crack detection using magnetic particle inspection [121] or defect detection in visual systems [15, 53]. However, their reliability is yet to be proven, making the industry skeptic towards their integration into production lines. The quality and robustness of learning-based approaches relies on the quality of the provided input data set in terms of completeness and correct labeling.

2.2.2 Training Data for Machine Learning Approaches

Both automated inspection methods using machine learning approaches and people performing manual inspection need to be trained to detect production-relevant defects. Thus, example data of defective objects acquired by the currently used inspection setups are required. For training of inspection workers, it is enough to have a specification of standards and a few examples, whereas for machine learning models, one needs thousands of labeled examples. This is problematic in applications where real data is not available in the quantity and diversity needed for reliable defect detection. Several publicly available data sets for some specific applications exist, e.g. GDXray [74] and RIAWELC [104] for radiographic images of welded objects or KolektorSDD2 [15], Severstal [3], CSEM-MISD [53] and CSDD [70] for photographic images of metallic objects. The annotation therein was done manually.

The main drawback of the available data sets is the restriction to specific applications. The data depends on the setup, the imaging process of the utilized NDT method and the object to be inspected, in particular, the object geometry, the material and production-specific defects. Thus, it is difficult to use these data sets to train robust machine learning algorithms for other use cases. Each application needs individual training data coming from the product-specific inspection environment. In addition, it is impossible to account for the large variety and variability of possible defects as manufacturers try to prevent defect formation during production. In particular, safety-related defects occur rarely as they are avoided best possible.

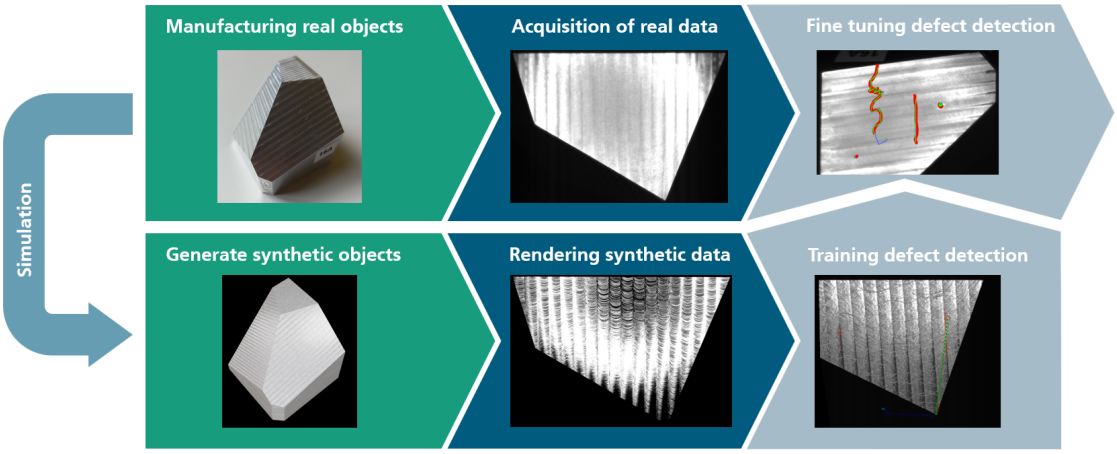


Figure 2.5: Explanation how synthetic data can be used for automatic defect detection [35]. Real objects were created by Fraunhofer IOF. Image acquisition and defect detection was performed by Juraj Fulir from Fraunhofer ITWM. Synthetic images were created by Lovro Bosnar from Fraunhofer ITWM.

Synthetic data that resemble the appearance of inspection data can overcome this problem [78]. AI-based techniques are of recent interest for synthetic image data generation [54, 114, 122]. While generative methods increase the quantity of the data set, the diversity of the synthetic data is restricted to the diversity of the training data. Moreover, learning-based approaches are known for introducing hallucinations [90], which may have negative impact on the subsequent training. In addition, synthetic images generated using generative methods are not always directly annotated and require subsequent labeling.

Another method of increasing the amount of image data showing defective areas is to insert defects into images of defect-free samples. Haselmann and Gruber [48] present a fake sample generator that adds easy fabric defects to visual image data. For the detection of cracks in concrete using micro-computed tomography (µCT) imaging, a data base of semi-synthetic images is available [60]. The images are obtained by simulating crack geometries using a stochastic model and then inserting them into real µCT images of concrete.

An alternative approach for the generation of synthetic data is to use rule-based methods, meaning to physically simulate the inspection environment using techniques from computer graphics [18, 89, 105]. Advantages of this procedure are the automatic pixel-perfect annotation of the synthetic data and fast data generation in sufficient quantity. Approaches for CT images of cast aluminum objects [32, 99] or 3d-woven composites [31] and visual surface inspection of metal surfaces [4, 9, 21, 33, 35, 62] exist. In [33, 35], promising results are obtained by training machine learning models on synthetic data followed by fine tuning with a small real data batch to overcome the domain gap. This improves model performance if just little real data is available. For an illustration of the procedure see Figure 2.5. Details of that synthetic data generation approach for visual surface inspection are given in Section 2.3.

2.3 Synthetic Data Generation Using a Virtual Inspection Environment

The following synthetic data generation approach is based on developing a digital twin of the visual inspection environment, a so-called virtual inspection environment. That

includes physical simulations of all components contained in the real inspection setup: camera, light source and object to be inspected, see Section 3.2 for more details. Especially all critical product characteristics discussed in Section 2.1.1 have to be treated with great accuracy when modeling the digital twin of the object. The pipeline to generate photo-realistic synthetic data via a virtual inspection environment subdivides into the following steps [14, 35, 42] which are also depicted in Figure 2.6.

1. **Input data for product specification**
   First, the digital twin of the object is generated. The object geometry is usually given as 3d CAD-file (Computer-Aided Design). If this is not the case, 3d surface scans may offer an alternative [88]. Usually, a post-processing of the virtual objects is needed since small geometrical features are often not provided by the CAD format but influence the reflectivity of the object surface, for example bevel cuts or details of screw holes. For further processing, the final object geometry is stored as a triangulated surface mesh. Moreover, material characteristics which determine the object's appearance have to be given, namely its color, the shape of light scattering and its degree of reflectance.

2. **Defecting**
   The defecting process is subdivided into three steps [12]. First, the defect position on the object surface has to be determined followed by creating the synthetic defect geometry. Then, the defect shapes are geometrically imprinted into the object geometry, meaning the mesh is restructured such that the newly introduced geometrical macro features are included. Real defect parameters are taken into account when developing the defect models such that the synthetic defect geometries are generated with regard to true parameter ranges and tolerances. We use stochastic modeling to ensure the generation of random defect shapes for a specific defect type but restricted to the given parameter limitations. Part II of this thesis considers various models for different types of surface defects.

3. **Surface texture**
   The surface topography has to be included because it mainly influences the surface-light interaction. This will be modeled by surface textures which are functions specifying the small-scale changes of a surface [82]. The texture models depend on the surface processing method and thus include the machining-specific features and parameters. Here, we use stochastic modeling to include variations within the surface textures to increase realism of the simulated surface topographies. Texture models for differently machined metallic object surfaces are introduced in Part I of this thesis.

4. **Creating 3d scene**
   After finalizing the digital twin of the object to be inspected, the camera and light source have to be considered to complete the virtual inspection environment, see Section 3.2.4 and Section 3.2.5 for more details. They are modeled with respect to the real acquisition parameters. For illumination that means for example that the light geometry and intensity are included. For the camera, the focal length, the exposure time and the resolution of the sensor are important. Moreover, the positions of camera and light source equal those from the real acquisition plan.

5. **Rendering**
   To generate physically correct synthetic images, the imaging process has to be simulated, that is, rendering the virtual 3d scene into a 2d image. It is determined by

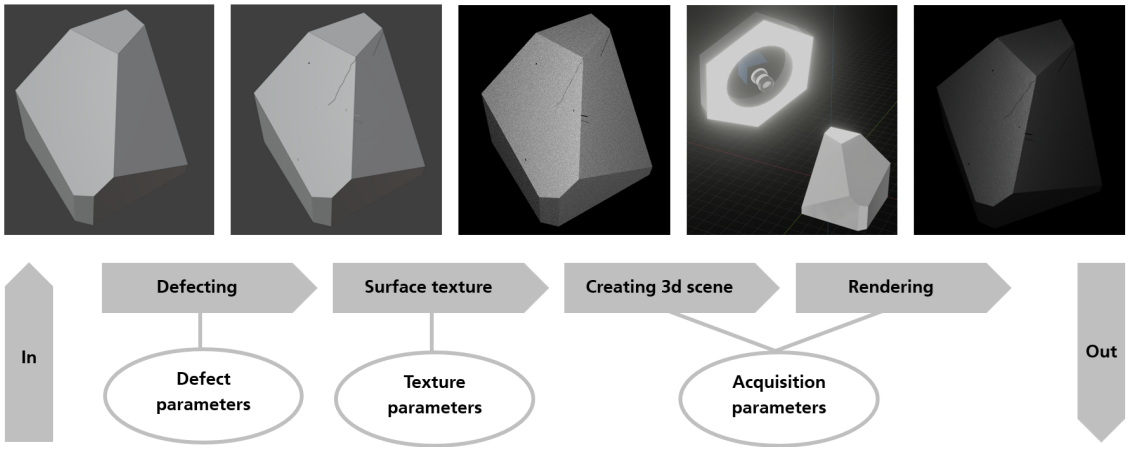


Figure 2.6: Pipeline for synthetic data generation in a virtual inspection environment.

the interaction of the object surface with light rays to identify which rays hit which sensor pixels in which intensity. This process of virtually taking images is called photo-realistic rendering and will be introduced in Section 3.2.6.

6. **Output**
   The output of the virtual inspection environment are synthetic photo-realistic images rendered using the synthetic object and the synthetic acquisition setup. Those synthetic images resemble the real images taken in the real acquisition environment. Their resolution is equal. The quality of the object appearance in the synthetic images depends on the degree of details considered in the individual models.

To conclude, the previously presented procedure to generate synthetic images is fully physically based since the virtual inspection environment resembles the real acquisition environment and photo-realistic rendering simulates the real imaging process. The physical simulations ensure the quality of the synthetic data and their parametrization enables to control the complete synthetic data generation pipeline. Moreover, data diversity is guaranteed by using different parameter configurations when generating the surface textures and surface defects. Data variability is assured by sampling the individual parameter values from suitable random distributions. When considering, for example, the generation of synthetic cracks, defects of short and long lengths can be produced by defining two separate intervals, that are two possible parameter configurations. The values for every realization may then be sampled from a uniform distribution defined on the associated interval. Realistic images can be created by using real situation-dependent parameter restrictions. Furthermore, the defect masks for the synthetic images are computed parallel to the data generation. Since the exact location of defects is known, automatic pixel-perfect annotation and labeling are ensured.

Although the quality of the synthetic data is achieved by physical simulations, its similarity to real data has to be proven. Figure 2.7 and Figure 2.8 show examples of real acquisition images compared to synthetic images generated with the previously presented virtual inspection environment. When comparing them visually, differences can be identified pointing out the present domain gap. The more complex the surface structure, the more differ the images. Also, real surfaces appear more reflective. Besides the data similarity its task similarity is important, meaning, how synthetic data impacts the model performance [35]. A deeper description on how to prove synthetic image data quality is given in Section 2.3.1.

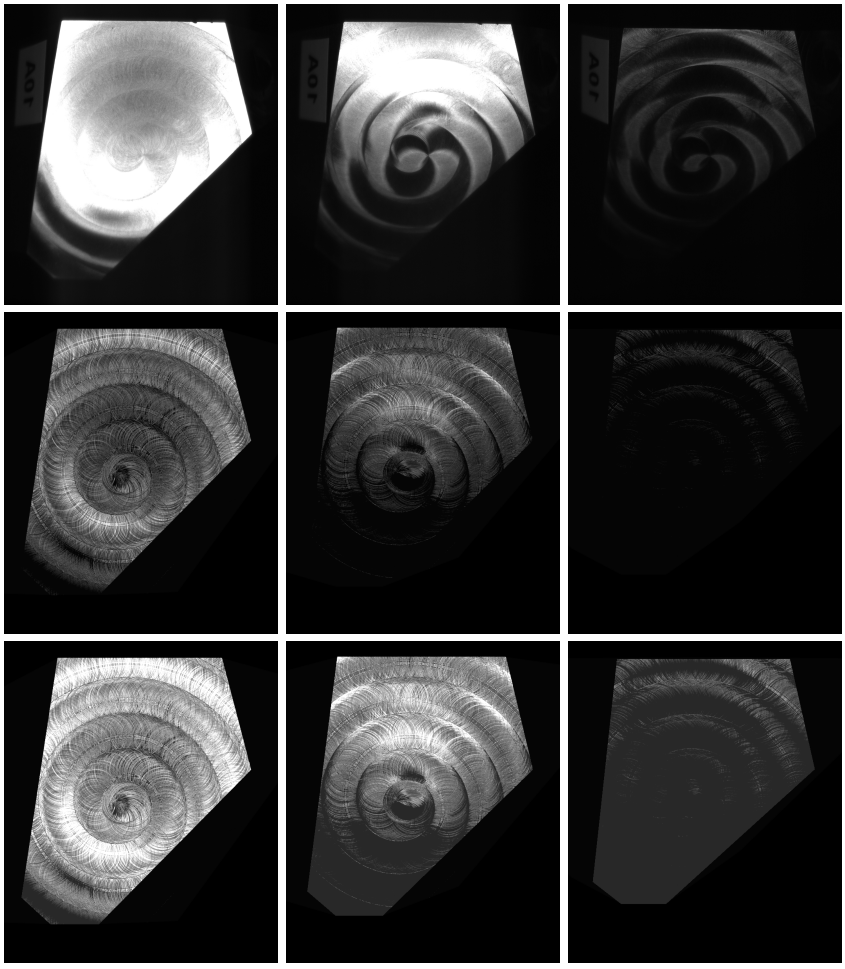


Figure 2.7: Comparison of real (top row), synthetic (middle row) and post-processed synthetic images (bottom row). Images taken from the same defect-free object but under changing viewing angles. As post-processing just histogram adaptation was applied. Rendering the synthetic images was performed by Lovro Bosnar and acquisition of real images and post-processing of the synthetic images by Juraj Fulir, both from Fraunhofer ITWM. Real objects were created by Fraunhofer IOF.

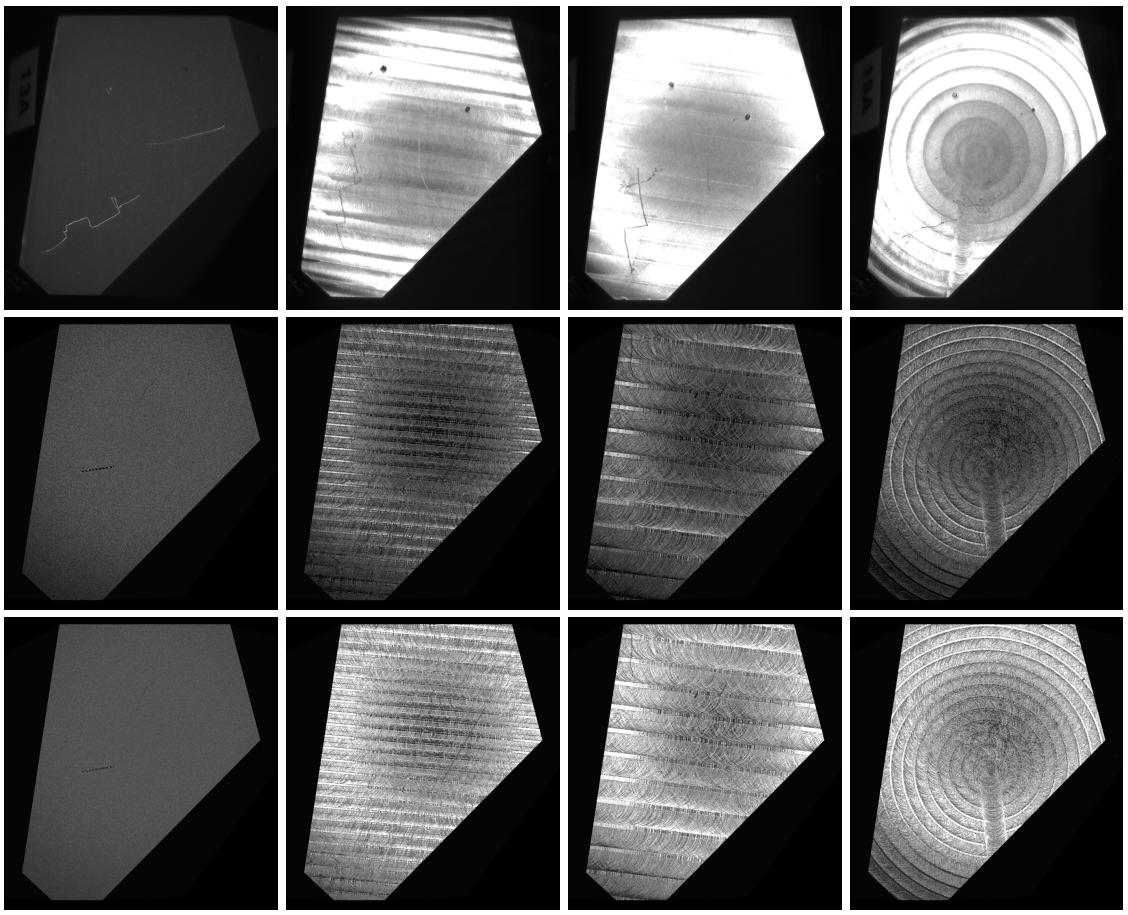


Figure 2.8: Comparison of real (top row), synthetic (middle row) and post-processed synthetic images (bottom row) of different surface textures of objects having defects. As post-processing just histogram adaptation was applied. Rendering the synthetic images was performed by Lovro Bosnar and acquisition of real images and post-processing of the synthetic images by Juraj Fulir, both from Fraunhofer ITWM. Real objects were created by Fraunhofer IOF.

The presented pipeline is specified to virtual inspection, but can be adapted to any other NDT method if the essential requirements are provided, especially a simulation of the data acquisition process. Moreover, since the whole virtual inspection environment consists of different single components and models, they can be exchanged easily. Thus, the adaptation to other circumstances and setups is straightforward if a simulation of the newly required element exists already. Therefore, the virtual inspection environment can also be used for planning optimal inspection setups and to test different hardware components, for example cameras from different suppliers, if the camera-specific properties are provided. Hence, this data generation pipeline may help to simplify and speed up investigating the best inspection setup for a given product-specific situation.

2.3.1 Quality Check of Synthetic Data

The validation of the synthetic image data set can be done either via domain similarity or via task similarity as discussed by Fulir et al. [35]. Methods belonging to the first group measure the similarity of synthetic and real images by comparing the distributions of different features, e.g. the pixel intensities. This can either be done by pixel- or pattern-wise metrics [46, 69, 112]. In [35], different metrics are considered to compare the similarity of the provided dual data set, see the example images in Figure 2.7 and Figure 2.8. As already observed, there are visible differences between real and synthetic data. These were subsequently reduced by post-processing the synthetic images, for example via histogram adaptation (exemplary results in Figure 2.7 and Figure 2.8), improvement of the surface-light interaction in the synthetic data by a multiplication factor and addition of blurring. However, in [35] it is shown that the domain gap is still present even after post-processing. Note that this domain gap is smaller for less complex textures (Figure 2.8 left).

The task similarity compares the model performance, that is, how well a model trained on synthetic data generalizes to real data. Fulir et al. [33] have shown how synthetic data generated in a virtual inspection environment can improve learning-based defect detection for visual inspection, especially if just little real data is available. They achieve the best results when training deep learning models with synthetic data, followed by fine-tuning with real data to overcome the domain gap. The effectiveness of their approach is proven by comparing the results to other models trained on other available real industrial metal data sets.

3 Basics

3.1 Mathematical Fundamentals

3.1.1 General Definitions

Definition 1
In 2d, the transformation ρ from Cartesian to polar coordinates and its inverse ρ^{-1} are defined via

$$\begin{aligned}\rho : \mathbb{R}^2 \to \mathbb{R}_{\geqslant 0} \times (-\pi, \pi] : (x, y) &\mapsto \left(\sqrt{x^2 + y^2}, \operatorname{atan2}(y, x)\right), \\ \rho^{-1} : \mathbb{R}_{\geqslant 0} \times (-\pi, \pi] \to \mathbb{R}^2 : (r, \varphi) &\mapsto (r\cos(\varphi), r\sin(\varphi)).\end{aligned}$$

Definition 2
Rounding to the nearest integer is denoted by $\lfloor \cdot \rceil$.

3.1.2 Image Processing

This section introduces the basic definitions and notations from image processing. It is mainly based on [40, 98].

Definition 3
A *homogeneous lattice* of dimension d is a set $\mathbb{L}^d \subset \mathbb{R}^d$ defined by

$$\mathbb{L}^d = \left\{ x \in \mathbb{R}^d : x = \sum_{i=1}^{d} \lambda_i u_i,\ \lambda_i \in \mathbb{Z} \right\}$$

with $\{u_i\}_{i=1,\dots,d}$ forming a basis of $\mathbb{R}^d$.

Definition 4
An *image* of dimension d is defined as mapping

$$\mathcal{I} : \mathbb{L}^d \cap W \to S$$

with $W \subset \mathbb{R}^d$ a compact observation window and S its co-domain. We restrict to $d = 2$ and choose the scaled canonical basis $u_i = \nu_i e_i$ for constant lattice spacing $\nu_i \in \mathbb{R}_{>0}$. An element $p \in \mathbb{L}^2 \cap W$ is called *pixel* with side lengths ν_i. We choose equal pixel side lengths $\nu_1 = \nu_2 = \nu$.

Moreover, we consider images of *size* $M \times N$ for $M, N \in \mathbb{N}$ and take $\lambda_1 = 0, \dots, M-1$ and $\lambda_2 = 0, \dots, N-1$. That means, the real size of the imaged area is $\nu M \times \nu N$. Then, the domain can be replaced by $\Omega = \{0, \dots, M-1\} \times \{0, \dots, N-1\}$ and a pixel can be uniquely denoted as $p = (m, n) \in \Omega$. Depending on the choice of the co-domain, different image types are defined.

- $\mathcal{I}$ is called a *gray value image* (or *grayscale image*) if $S \subset \mathbb{R}$. When considering *discrete gray value images*, it holds that $S = \{0, 1, \dots, 2^k - 1\}$. Typical choices for k are $k = 1$, then $\mathcal{I}$ is a *binary image*, and $k = 8$. A *continuous gray value image*

is defined by choosing an interval $S \subset \mathbb{R}$. Usually, $S = [0, 1]$ is taken. In general, 0 indicates black and the highest gray value stands for white.
A *height image* (or *height map*) is a continuous image but with a different interpretation of the co-domain. Instead of gray values, height values are assigned to each pixel such that the set $\{(p, \mathcal{I}(p)) : p \in \Omega\}$ can be interpreted as a 2d surface in $\mathbb{R}^3$.

- Typical image types when S is of higher dimension are color images. For example, if $\mathcal{I}$ is an RGB-image, then $S \subset \mathbb{R}^3$ consists of three color channels, namely red, green and blue. The classification into discrete and continuous images is the same as for grayscale images.
  For rendering, we need *normal images* (or *normal maps*) which are similar to color images, so $S \subset \mathbb{R}^3$, but with a different interpretation. Each pixel is assigned its local unit normal vector in $\mathbb{R}^3$. Assume a height image is given, the local unit normal vector is computed by considering the tangent plane at the pixel's location.

Definition 5
The *neighborhood* $\mathcal{N}_{s,r}(p)$ of size $s \in \mathbb{R}_{>0}$ of a point $p \in \Omega \subset \mathbb{Z}^d$ associated to a given norm $\|\cdot\|_r$ for $r \in \mathbb{N}_{>0} \cup \{\infty\}$ is defined by

$$\mathcal{N}_{s,r}(p) = \left\{ q \in \mathbb{Z}^d \backslash \{p\} : \|p - q\|_r \leqslant s \right\}.$$

In 2d, common examples are the

- *4-connected neighborhood* $\mathcal{N}_4(p) = \mathcal{N}_{1,1}(p)$ (top, right, bottom, left)
- *8-connected neighborhood* $\mathcal{N}_8(p) = \mathcal{N}_{1,\infty}(p)$ (4-connected neighborhood plus diagonal lattice points)

Definition 6
Let $\mathcal{I}$ be an image. *Down-sampling* an image refers to the reduction of the image size while keeping the real size of the imaged area. Thus, ν increases while M and N decrease.

A common method to do so is nearest neighbor interpolation. It picks the value of the closest pixel in $\mathcal{I}$.

Definition 7
Let $f : \mathbb{R}^2 \to S$ be a function and $\mathbb{L}^2$ a homogeneous lattice. An image $f_{\mathbb{L}^2}$ can be obtained by *sampling* from f, that means evaluating f at the given lattice points

$$f_{\mathbb{L}^2}(p) = f(p) \text{ for } p \in \mathbb{L}^2.$$

Definition 8
Let $\mathcal{I}$ be an image of size $M \times N$. *Padding* an image with a value v means creating a new image $\tilde{\mathcal{I}}$ of increased size $\tilde{M} \times \tilde{N}$ with $\tilde{M} > M$ and $\tilde{N} > N$ where $\mathcal{I}$ is placed in the center and the remaining pixels get value v. Assuming $M, N, \tilde{M}, \tilde{N}$ to be odd, that means for $p = (m, n)$

$$\tilde{\mathcal{I}}(p) = \begin{cases} \mathcal{I}\left(m - \frac{\tilde{M}-M}{2}, n - \frac{\tilde{N}-N}{2}\right) & \text{if } 0 \leqslant m - \frac{\tilde{M}-M}{2} < M,\ 0 \leqslant n - \frac{\tilde{N}-N}{2} < N \\ v & \text{else.} \end{cases}$$

If it holds $v = 0$, then the associated padding is called *zero padding*.

3.1.3 Fourier Analysis

This section introduces the main definitions of Fourier analysis based on [40].

Definition 9
For $T \in \mathbb{R}$, let $f \in L_2(\mathbb{R})$ be a T-periodic function, that means $f(x+T) = f(x)$ for all $x \in \mathbb{R}$. Then, its *Fourier series* reads

$$f(x) = \sum_{k \in \mathbb{Z}} c_k(f) \exp\left(\frac{2\pi i k x}{T}\right), x \in \mathbb{R}$$

with *Fourier coefficients*

$$c_k(f) = \int_{-\frac{T}{2}}^{\frac{T}{2}} f(x) \exp\left(-\frac{2\pi i k x}{T}\right) dx.$$

Remark 1
When representing the Fourier coefficients in polar coordinates, meaning $c_k(f) = r_k \exp(i\varphi_k)$, then the coefficients can be interpreted as follows:

- $\{r_k\}_{k \in \mathbb{Z}}$ with $r_k = |c_k(f)|$ is called the *modulus* of f,
- $\{\varphi_k\}_{k \in \mathbb{Z}}$ with $\varphi_k = \operatorname{atan2}(\operatorname{Im}(c_k(f)), \operatorname{Re}(c_k(f)))$ is called the *phase* of f.

Definition 10
For $T_1, \ldots, T_d \in \mathbb{R}$, Definition 9 can be generalized for a $(T_1, \ldots, T_d)$-periodic function $f \in L_2(\mathbb{R}^d)$ by

$$f(x) = \sum_{k \in \mathbb{Z}^d} c_k(f) \exp\left(2\pi i \left\langle \frac{k}{T}, x \right\rangle\right)$$
$$\text{with } c_k(f) = \int_{-\frac{T_1}{2}}^{\frac{T_1}{2}} \cdots \int_{-\frac{T_d}{2}}^{\frac{T_d}{2}} f(x) \exp\left(-2\pi i \left\langle \frac{k}{T}, x \right\rangle\right) dx$$

for $\frac{k}{T} = \left(\frac{k_1}{T_1}, \ldots, \frac{k_d}{T_d}\right)$.

This definition can be transferred to non-periodic functions by using the continuous Fourier transform.

Definition 11
Let $f \in L_1(\mathbb{R}^d)$ be an integrable function. Then, the *continuous Fourier transform* $\mathcal{F}(f)$ of f is defined via

$$\mathcal{F}(f)(\omega) = \hat{f}(\omega) = \int_{\mathbb{R}^d} f(x) \exp\left(-2\pi i \langle x, \omega \rangle\right) dx, \ \omega \in \mathbb{R}^d.$$

Here, $\langle \cdot, \cdot \rangle$ is the standard scalar product. If it holds that $\hat{f} \in L_1(\mathbb{R}^d)$, then the inverse $\mathcal{F}^{-1}(\hat{f})$ exists and reads

$$\mathcal{F}^{-1}(\hat{f})(x) = \int_{\mathbb{R}^d} \hat{f}(\omega) \exp\left(2\pi i \langle x, \omega \rangle\right) d\omega, \ x \in \mathbb{R}^d.$$

It holds $f(x) = \mathcal{F}^{-1}(\hat{f})(x)$ for almost all $x \in \mathbb{R}^d$. If f is additionally continuous, this equality is pointwise true for all $x \in \mathbb{R}^d$.

The Fourier transform can be generalized to discrete signals by considering them as discretized periodic functions.

Definition 12
Let $F : J \to \mathbb{R}$ with $J = \{0, \dots, n_1\} \times \cdots \times \{0, \dots, n_d\}$ for $n_1, \dots, n_d \in \mathbb{N}$ be a d-dimensional discrete signal. Then, its *discrete Fourier transform* (DFT) $\mathcal{F}(F) : J \to \mathbb{C}$ and the corresponding inverse read

$$\mathcal{F}(F)_k = \hat{F}_k = \sum_{j \in J} F_j \exp\left(-2\pi i \left\langle \frac{j}{n}, k \right\rangle\right), \, k \in J,$$
$$\mathcal{F}^{-1}(\hat{F})_j = F_j = \frac{1}{n_1 \cdots n_d} \sum_{k \in J} \hat{F}_k \exp\left(2\pi i \left\langle \frac{j}{n}, k \right\rangle\right), \, j \in J$$

with $\frac{j}{n} = \left(\frac{j_1}{n_1}, \dots, \frac{j_d}{n_d}\right)$.

Remark 2
Images can be interpreted as 2-dimensional discrete signals with $d = 2$, $n_1 = M - 1$ and $n_2 = N - 1$, that is $J = \Omega$. To define a valid DFT, we assume periodic images. Thus, the domain of the associated DFT which is called *Fourier domain*, can also be chosen as

$$\widehat{\Omega} = \left\{-\frac{M-1}{2}, \dots, \frac{M-1}{2}\right\} \times \left\{-\frac{N-1}{2}, \dots, \frac{N-1}{2}\right\}$$

for M, N being odd. The definitions of modulus and phase of an image $\mathcal{I}$ transfer to its coefficients $\hat{\mathcal{I}}_k$ of the DFT.

Remark 3
Let $\mathcal{I}$ be an image and assume M and N to be odd. The sample auto-correlation $C_{\mathcal{I}} \in \mathbb{R}^{M \times N}$ of $\mathcal{I}$ is defined for $\delta \in \left\{-\frac{M-1}{2}, \dots, \frac{M-1}{2}\right\} \times \left\{-\frac{N-1}{2}, \dots, \frac{N-1}{2}\right\}$ via

$$C_{\mathcal{I}}(\delta) = \frac{1}{|\Omega_\delta|} \sum_{p \in \Omega_\delta} (\mathcal{I}(p) - \mu)(\mathcal{I}(p + \delta) - \mu)$$

with $\Omega_\delta = \{p \in \Omega : p + \delta \in \Omega\}$, μ the arithmetic mean of $\mathcal{I}$ and $|\cdot|$ the cardinality of a set.

The Wiener–Khinchin theorem states that the sample auto-correlation can be computed by using the DFT, namely it is the inverse Fourier transformation of the squared Fourier modulus. That is

$$C_{\mathcal{I}} = \mathcal{F}^{-1}\left(|\mathcal{F}(\mathcal{I})|^2\right).$$

3.1.4 Filtering

Here, the filtering, especially of images, is introduced based on [40], also providing some important examples.

Definition 13
The continuous convolution of $f \in L_1\left(\mathbb{R}^2\right)$ and $g \in L_1\left(\mathbb{R}^2\right)$,

$$(f * g)(x) = \int_{\mathbb{R}^2} f(y) g(x - y) dy, \, x \in \mathbb{R}^2,$$

is called *linear smoothing filter* of f with *filter kernel* g.

Definition 14
Let $\mathbb{L}^2$ be a homogeneous lattice and $f_{\mathbb{L}^2}$ and $g_{\mathbb{L}^2}$ be the sampling of measurable functions f and g, respectively. Then, the *discrete convolution* of $f_{\mathbb{L}^2}$ and $g_{\mathbb{L}^2}$ is defined by

$$f_{\mathbb{L}^2} * g_{\mathbb{L}^2}(p) = \nu^2 \sum_{q \in \mathbb{L}^2} f_{\mathbb{L}^2}(q) g_{\mathbb{L}^2}(p-q),$$

where the multiplication factor is the volume of the unit cell $[-\nu/2, \nu/2]^2$ of $\mathbb{L}^2$.

In case g has bounded support, meaning that $g_{\mathbb{L}^2}$ can be represented as a rectangular array, it is called *filter mask*.

Example 1
The filter kernel of the *mean filter* has bounded support and is defined for a neighborhood $S = \left\{-\frac{s-1}{2}, \dots, \frac{s-1}{2}\right\}^2$ of size $s \times s$ with $s \in \mathbb{N}$ being odd via its filter mask

$$g_{\mathbb{L}^2} = \mathrm{mean}_S = \frac{1}{s^2} \mathbb{1}_S.$$

Remark 4
Note that an image $\mathcal{I}$ can be seen as a sampling from a measurable function. Thus, a linear filter can be applied to images by discrete convolution.

Remark 5
Discrete filtering equals component-wise multiplication in the Fourier domain which speeds up its computation, meaning $f_{\mathbb{L}^2} * g_{\mathbb{L}^2} = \mathcal{F}^{-1}\left(\mathcal{F}(f_{\mathbb{L}^2}) \cdot \mathcal{F}(g_{\mathbb{L}^2})\right)$.

Example 2
In the following we will introduce the low pass filter, the high pass filter and band-pass filters for $Q \in \mathbb{N}$ orientations. As those filters extract certain areas in the Fourier spectrum, the corresponding filter kernels l, h and b_q for $q = 0, \dots, Q-1$ are defined in the Fourier domain, that are $\hat{l}$, $\hat{h}$ and $\hat{b}_q$. To apply those filters to discrete images, the kernels are sampled from appropriate functions L, H and B_q defined in the continuous frequency domain $[-\pi, \pi]^2$. Those functions are defined in polar coordinates. The following definitions are based on [108, 109].

The functions determining the low and high pass filter kernel are given by

$$L : \mathbb{R}_{\geqslant 0} \times (-\pi, \pi] \to [0,1] : (r, \varphi) \mapsto \begin{cases} 1 & \text{if } r \leqslant \frac{\pi}{4} \\ \cos\left(\frac{\pi}{2} \log_2\left(\frac{4r}{\pi}\right)\right) & \text{if } \frac{\pi}{4} \leqslant r \leqslant \frac{\pi}{2} \\ 0 & \text{if } r \geqslant \frac{\pi}{2}, \end{cases}$$

$$H : \mathbb{R}_{\geqslant 0} \times (-\pi, \pi] \to [0,1] : (r, \varphi) \mapsto \begin{cases} 0 & \text{if } r \leqslant \frac{\pi}{4} \\ \cos\left(\frac{\pi}{2} \log_2\left(\frac{2r}{\pi}\right)\right) & \text{if } \frac{\pi}{4} \leqslant r \leqslant \frac{\pi}{2} \\ 1 & \text{if } r \geqslant \frac{\pi}{2}. \end{cases}$$

It holds $L(r, \varphi) = L(r)$ and $H(r, \varphi) = H(r)$ since both functions are rotation-invariant. The oriented band-pass filter kernels are obtained by using $B_q(r, \varphi) = H(r) G_q(\varphi)$ with cone-shaped filter kernels $G_q(\varphi) = G_q(r, \varphi)$ defined via

$$G_q : \mathbb{R}_{\geqslant 0} \times (-\pi, \pi] \to [0,1] : (r, \varphi) \mapsto \begin{cases} \alpha_Q \cos\left(\varphi - \frac{\pi q}{Q}\right)^{Q-1} & \text{if } \left|\varphi - \frac{\pi q}{Q}\right| \leqslant \frac{\pi}{2} \\ \alpha_Q \cos\left(\varphi - \frac{\pi (q-Q)}{Q}\right)^{Q-1} & \text{if } \left|\varphi - \frac{\pi (q-Q)}{Q}\right| \leqslant \frac{\pi}{2} \\ 0 & \text{else} \end{cases}$$

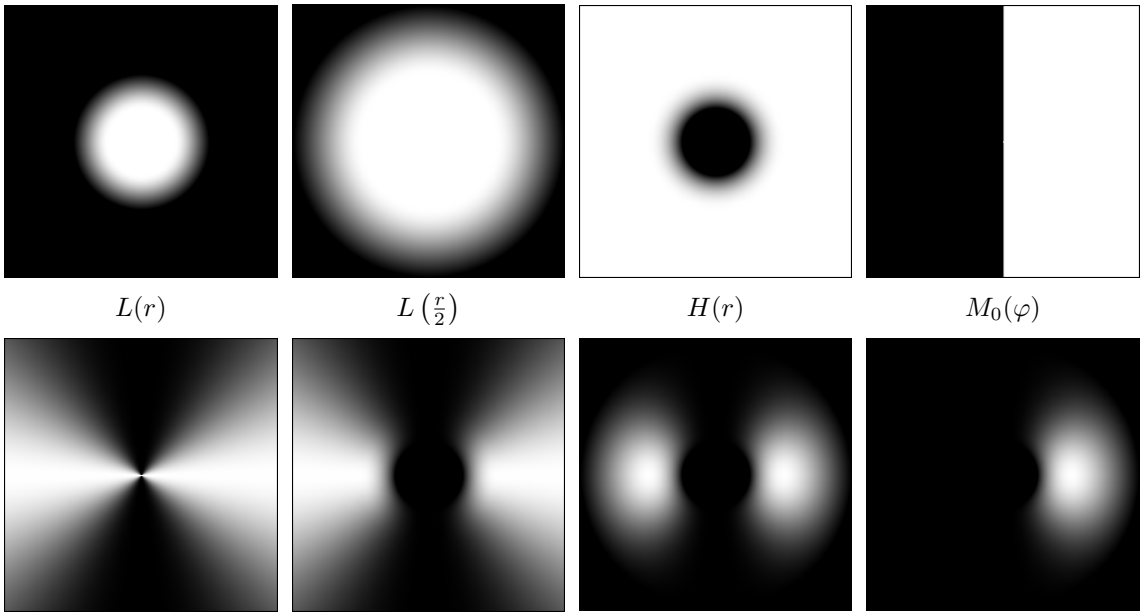


Figure 3.1: Overview of different filters in the Fourier domain using $Q = 4$.

with

$$\alpha_Q = 2^{Q-1}\frac{(Q-1)!}{\sqrt{Q\,(2\,(Q-1))!}}.$$

Then, for $f \in \{l, h, b_q\}$ the DFT $\hat{f} \in \left\{\hat{l}, \hat{h}, \hat{b}_q\right\}$ is obtained by sampling from $F \in \{L, H, B_q\}$ in the transformed discrete Fourier domain $\hat{\Omega}$. That is

$$\hat{f}(m,n) = F \circ \rho\left(\frac{2\pi m}{M}, \frac{2\pi n}{N}\right) \text{ for } (m,n) \in \widehat{\Omega}.$$

Note that ρ is the transformation into polar coordinates introduced in Definition 1. See Figure 3.1 for an overview of the filters.

Definition 15
The *Hilbert filters* M_q, $q = 0, \ldots, Q-1$, for $Q \in \mathbb{N}$ orientations are defined in [108] by

$$M_q : \mathbb{R}_{\geqslant 0} \times (-\pi, \pi] \to [0,2] : (r,\varphi) \mapsto \begin{cases} 2 & \text{if } \left|\varphi - \frac{\pi q}{Q}\right| < \frac{\pi}{2} \\ 1 & \text{if } \left|\varphi - \frac{\pi q}{Q}\right| = \frac{\pi}{2} \\ 0 & \text{if } \left|\varphi - \frac{\pi q}{Q}\right| > \frac{\pi}{2}, \end{cases}$$

see also Figure 3.1.

3.1.5 Steerable Pyramid

Simoncelli and others [30, 94, 95] introduced the steerable pyramid which is a linear image decomposition. That is, an image is linearly decomposed into multiple scaled and oriented sub-bands. Therefore, let $\mathcal{I}$ be an image of size $M \times N$ and resolution ν. Down-sampling by factor two is applied to obtain sub-bands of different size and resolution levels. In total, P levels are considered. Thus, the image side lengths must be a multiple of 2^P.

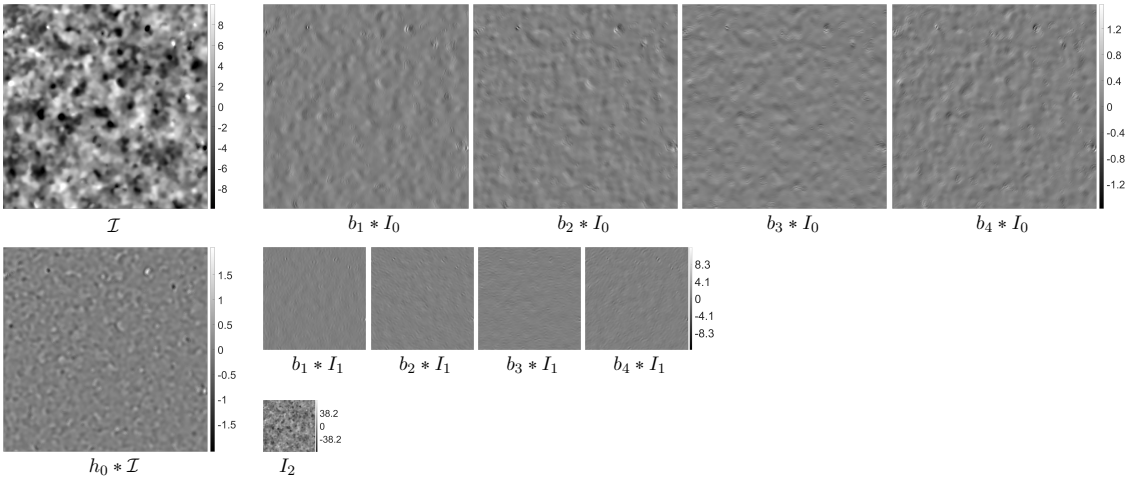


Figure 3.2: Steerable pyramid $\mathcal{P}_{2,4}(\mathcal{I})$ using $P = 2$ scales and $Q = 4$ orientations.

Definition 16
Down-sampling $\downarrow_2$ is done in the Fourier domain by cropping out the central part of the image, meaning

$$\downarrow_2 (\mathcal{I}) = \mathcal{F}^{-1} \left(\mathcal{F}(\mathcal{I})(\Omega_{\downarrow_2}) \right) \text{ with } \Omega_{\downarrow_2} = \left\{ \frac{1}{4}M, \ldots, \frac{3}{4}M \right\} \times \left\{ \frac{1}{4}N, \ldots, \frac{3}{4}N \right\}.$$

Definition 17
Let P be the number of scales and Q be the number of orientations. To perform the multi-scale transform, first split the image into its high and low frequency part. Then, oriented symmetric band-pass filters are applied to the low frequency image followed by down-sampling by factor two. That procedure is repeated P times considering Q orientations each. In total, the pyramid consist of $PQ + 2$ images of different sizes and resolutions.

To compute the pyramid, the filters obtained by the functions

$$L_0(r) = L\left(\frac{r}{2}\right), H_0(r) = H\left(\frac{r}{2}\right), H(r), B_q(r, \varphi)$$

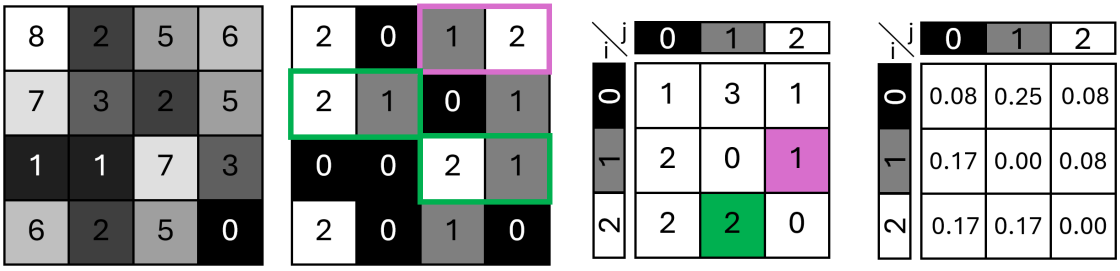
are needed, see Example 2 for their definitions. Then, the *real steerable pyramid* $\mathcal{P}_{P,Q}(\mathcal{I})$ is defined by

- high frequency residual $h_0 * \mathcal{I}$ of size $M \times N$ and resolution ν,
- oriented high frequency bands $b_q * I_p$ of size $\frac{M}{2^p} \times \frac{N}{2^p}$ and resolution $(p+1)\nu$ for $p = 0, \ldots, P-1$ and $q = 0, \ldots, Q-1$ using the down-sampled low frequency residuals

$$I_p = \downarrow_2 (l * I_{p-1}) \text{ for } p = 1, \ldots, P-1 \text{ with } I_0 = l_0 * \mathcal{I},$$

- low frequency residual $I_P = \downarrow_2 (l * I_{P-1})$ of size $\frac{M}{2^P} \times \frac{N}{2^P}$ and resolution $(P+1)\nu$.

The convolutions for the filters are computed by multiplication in the Fourier domain. Since numerical artifacts cause complex-valued sub-bands having a negligible imaginary part, just their real parts are used. An example is given in Figure 3.2.



Discrete gray value image $\mathcal{I}$ having values in $\{0, \dots, 8\}$.

Quantized image $\mathcal{Q}$ using $B = 3$ bins.

Non-normalized GLCM $X_\delta(\mathcal{Q})$ for $\delta = (1, 0)$.

Normalized GLCM $P_\delta(\mathcal{Q})$ for $\delta = (1, 0)$.

Figure 3.3: Illustration of the steps to compute the Haralick features similar to [66].

Remark 6
The steerable image decomposition is a pseudo-invertible operator [94] which allows to reconstruct the image from the pyramid. To reconstruct the image, all operations from the image decomposition are performed in reversed order. However, image up-sampling by factor two is needed instead of image down-sampling. That is performed using zero-padding in the Fourier domain.

Definition 18
In the real steerable pyramid the phase information contained in the oriented sub-bands is ignored. That information is included in the *complex steerable pyramid* by applying the respective Hilbert filters to the oriented band-pass filters, thus $m_q * b_q$, to obtain complex-valued filters [108], see Figure 3.1 for the corresponding filter kernel. The real part of each filter coincides with the corresponding real-valued filter used in the real-valued steerable pyramid.

3.1.6 Haralick Features

In image analysis, Haralick features are common texture descriptors capturing local texture variations [46]. They are computed from the gray-level co-occurrence matrix (GLCM) which counts the frequencies of gray value pairs for a given distance vector [46, 66]. To limit the size of the matrix, the gray values are binned to a selected number of bins.

Definition 19
Let $\mathcal{I} : \Omega \to [a, b]$ for $a, b \in \mathbb{R}$ with $a < b$ be a continuous image. Then, the following steps are needed to compute the Haralick features of $\mathcal{I}$, see also Figure 3.3.

1. *Quantize* the image to obtain an image $\mathcal{Q}$ having as number of gray values the desired number $B \in \mathbb{N}$ of bins. Therefore, apply a quantization function

$$q : [a, b]^{M \times N} \to \{0, \dots, B-1\}^{M \times N}$$

to $\mathcal{I}$, that is $\mathcal{Q} = q(\mathcal{I})$. We use linear quantization, meaning that $[a, b]$ is divided into B equidistant bins.

2. Compute the *non-normalized GLCM* $X_\delta(\mathcal{Q}) \in \mathbb{N}^{B \times B}$ of $\mathcal{Q}$ using a given distance vector $\delta = (\delta_x, \delta_y) \in \mathbb{Z}^2$. Hence, pixel $(i, j) \in \{0, \dots, B-1\}^2$ in $X_\delta(\mathcal{Q})$ is assigned the number of times the gray values i and j occur as neighbors in $\mathcal{Q}$ defined via δ, that is

$$X_\delta(\mathcal{Q})(i, j) = \left|\{q \in \Omega : \mathcal{Q}(q) = i,\ \mathcal{Q}(q + \delta) = j\}\right|.$$

3. Normalize $X_\delta(\mathcal{Q})$ to obtain the *normalized GLCM* $P_\delta(\mathcal{Q}) \in \mathbb{Q}^{B\times B}$ by

$$P_\delta(\mathcal{Q})(i,j) = \frac{X_\delta(i,j)}{\sum_{i=0}^{B-1}\sum_{j=0}^{B-1} X_\delta(i,j)},\ (i,j) \in \{0,\dots,B-1\}^2.$$

Thus, $P_\delta(\mathcal{Q})(i,j)$ estimates the probability that the combination $\mathcal{Q}(p) = i$ and $\mathcal{Q}(p+\delta) = j$ occurs.

4. Compute the *Haralick features* using the normalized GLCM. Examples are energy, homogeneity, contrast and correlation defined via

$$\begin{aligned}
\textit{Energy}:&\ \sum_{i=0}^{B-1}\sum_{i=0}^{B-1} P_\delta(\mathcal{Q})(i,j)^2\\
\textit{Homogeneity}:&\ \sum_{i=0}^{B-1}\sum_{i=0}^{B-1} \frac{P_\delta(\mathcal{Q})(i,j)}{1+(i-j)^2}\\
\textit{Contrast}:&\ \sum_{i=0}^{B-1}\sum_{i=0}^{B-1} (i-j)^2 P_\delta(\mathcal{Q})(i,j)\\
\textit{Correlation}:&\ \sum_{i=0}^{B-1}\sum_{i=0}^{B-1} \left(\frac{i-\mu_x}{\sigma_x}\right)\left(\frac{j-\mu_y}{\sigma_y}\right) P_\delta(\mathcal{Q})(i,j)\\
&\text{with } \mu_x = \sum_{i=0}^{B-1} iP_x(i),\ \sigma_x^2 = \sum_{i=0}^{B-1}(i-\mu_x)^2 P_x(i)\\
&\text{for } P_x(i) = \sum_{j=0}^{B-1} P_\delta(\mathcal{Q})(i,j)\\
&\text{and } \mu_y,\ \sigma_y \text{ defined similarly.}
\end{aligned}$$

The first three features indicate whether the texture is uniform or heterogeneous in the neighborhood determined by δ. In doing so, homogeneity gives higher weights to similar gray values whereas contrast favors gray values differing significantly from each other. Correlation describes how strong the dependency of the image structure is in the given neighborhood, with low values indicating unstructured textures.
See [46, 66] for a detailed overview of all commonly used Haralick features.

Remark 7
Löfstedt et al. [66] adapted the original Haralick features to be gray-level invariant, that is, their results become asymptotically invariant to the number of bins B used for the quantization.

3.1.7 Stochastic Geometry Basics: Point Processes

In the following, point processes and different examples thereof will be introduced based on [17,91]. Let E be a locally compact space with countable base and $\mathcal{B}$ the Borel σ-algebra of E.

Definition 20
Let $\mathcal{C}$ be the system of compact subsets of E.

- A measure μ on $(E,\mathcal{B})$ is called *locally finite*, if it holds that $\mu(C) < \infty$ for all $C \in \mathcal{C}$. The set of all locally finite measures on $(E,\mathcal{B})$ is denoted by $M(E)$. Let $\mathcal{M}$ be the

smallest σ-algebra on $M(E)$ such that the mapping $\mu \to \mu(B)$ is measurable for all $B \in \mathcal{B}$.

- A measure $\mu \in M(E)$ is called *counting measure* on E, if $\mu(B) \in \mathbb{N}_0 \cup \{\infty\}$ for all $B \in \mathcal{B}$. The set of all locally finite counting measures on E is denoted by $N(E)$. Let $\mathcal{N} = \{M \cap N(E) : M \in \mathcal{M}\}$ be the trace of $\mathcal{M}$ on $N(E)$.
- A counting measure $\mu \in N(E)$ is called *simple* if $\mu(\{x\}) \leqslant 1$ for all $x \in E$.

Example 3
The countable or finite sum of Dirac measures $\varphi(B) = \sum_{k=1}^{n} \delta_{x_k}(B)$ for $n \in \mathbb{N} \cup \{\infty\}$ and $\varphi(B) = 0$ for $n = 0$ is a counting measure.

Definition 21
Let $(\Omega, \mathcal{A}, \mathbb{P})$ be a probability space.

- A *random measure* on E is a measurable mapping $\Psi : (\Omega, \mathcal{A}, \mathbb{P}) \to (M(E), \mathcal{M})$.
- A *point process* is a measurable mapping $\Phi : (\Omega, \mathcal{A}, \mathbb{P}) \to (N(E), \mathcal{N})$.
- A point process is called *simple* if its realizations are almost surely simple.
- The *distribution* of Φ is defined by the image measure $\mathbb{P}_\Phi = \Phi(\mathbb{P})$.

Thus, a point process can be interpreted as randomly choosing an element in $N(E)$ with respect to a distribution $\mathbb{P}_\Phi$. A simple point process can be interpreted as countable or finite collection of distinct points, that is $\varphi = \{x_1, \ldots, x_n\}$ for $n \in \mathbb{N}_0 \cup \{\infty\}$.

Definition 22
The *intensity measure* of a point process Φ is defined as a mapping

$$\Lambda : \mathcal{B} \to [0, \infty] : B \mapsto \mathbb{E}(\Phi(B)).$$

For a simple point process, the intensity measure of B equals the expected number of points falling in B.

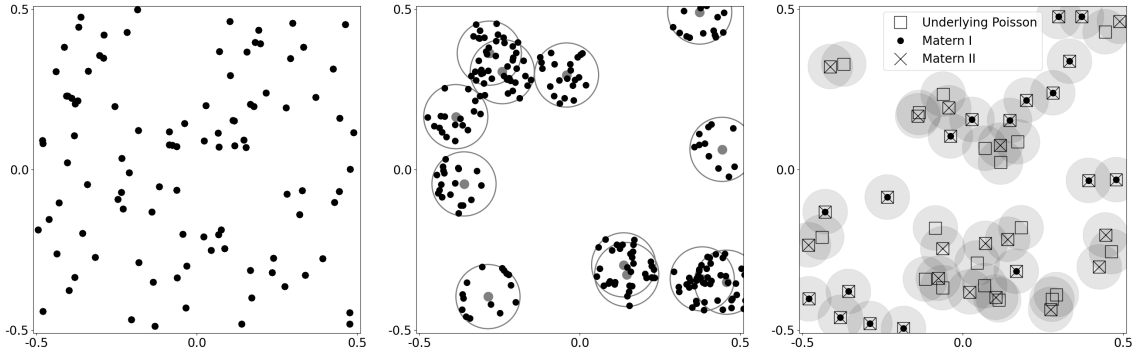
Let $E = \mathbb{R}^d$ and Φ be a stationary point process, that is $\Phi + x \overset{d}{=} \Phi$ for all $x \in \mathbb{R}^d$. Then, the intensity measure is translation invariant. If $\Lambda(\cdot)$ is locally finite, it can be expressed by $\Lambda(B) = \lambda\nu(B)$ with $\lambda \in [0, \infty)$ the *intensity* of Φ and $\nu(B)$ the Lebesgue measure of B.

Example 4
A point process Φ on E with the properties

1. Φ is simple,
2. the random variables $\Phi(B_1), \ldots, \Phi(B_n)$ are independent for pairwise disjoint sets $B_k \in \mathcal{B}$, $k = 1, \ldots, n$,
3. the random variable $\Phi(B)$ is Poisson-distributed for all $B \in \mathcal{B}$ with expectation $\Lambda(B)$, that is $\Phi(B) \sim \mathcal{P}(\Lambda(B))$,

is called a *Poisson (point) process*.

To simulate a stationary Poisson process with intensity λ in a compact observation window W, two steps are needed. First, determine the number of points $n \in \mathbb{N}_0$ by sampling from $\mathcal{P}(\lambda\nu(W))$. Then, sample n points from the uniform distribution on W. See Figure 3.4 for a realization of a Poisson process.



Poisson point process using $\lambda = 100$.

Matérn cluster point process using $\lambda_{\text{parent}} = 9$, $\lambda_{\text{child}} = 20$ and $r = 0.1$. The parent point process with the corresponding circles is marked in gray.

Matérn hardcore point process using $\lambda = 50$ and $r = 0.07$. The marks indicate the points corresponding to the process type. Circles (gray) of radius r are centered in the points of the Poisson process.

Figure 3.4: Realizations of different types of point processes in $W = [-0.5, 0.5]^2$.

Example 5
Another example similar to a Poisson process is a *binomial point process*. Therefore, a predefined number of points is sampled from the uniform distribution on a compact observation window W.

Example 6
Other examples for point processes are the Matérn cluster point process and the Matérn hardcore point process. The first type generates points clustered in groups. In contrast, the points in a hardcore process keep a certain distance to each other. Since we will only need these point processes later as examples, we do not define them precisely, but rather describe them only through their simulation. The simulation of both point processes are based on an underlying Poisson process. We restrict to stationary point processes. See Figure 3.4 for exemplary realizations of both types of point processes and descriptions for their simulation.

- For simulating the *Matérn cluster point process* in a compact observation window W, first a parent process is needed to define the location of the clusters. Therefore, let $\varphi_{\text{parent}} = \{x_1, \dots, x_{n_{\text{parent}}}\}$ be the realization of a stationary Poisson process with intensity λ_{parent} in W. Then, the points defining the Matérn cluster point process are obtained by Poisson processes $\varphi_{\text{child}}(x_k)$, the child processes. A cluster consists of $n_{\text{child}_k} \sim \mathcal{P}(\lambda_{\text{child}})$ points which are sampled from the uniform distribution defined on the ball $\mathcal{S}_r(x_k)$ with radius $r > 0$ and centered in $x_k \in \varphi_{\text{parent}}$ for $k = 1, \dots, n_{\text{parent}}$. Finally, the resulting Matérn cluster point process reads $\varphi_{\text{Matérn cluster}} = \bigcup_{k=1}^{n_{\text{parent}}} \varphi_{\text{child}}(x_k)$.
- To obtain a *Matérn hardcore point process*, first a stationary Poisson process with intensity λ in W is generated. Let $\varphi = \{x_1, \dots, x_n\}$ be a realization thereof. Then, a thinning of the points follows such that the desired distance $r > 0$ between points is met. Therefore, two different types exist defining Matérn hardcore point processes of type I and II. For the first type, remove all points in φ that have distance less or

equal r to each other. As this thinning process is very strict, processes of type II are obtained by deleting just one point of a pair that violates the distance constraint. For this purpose, all points in φ are assigned an age in the beginning and the younger one is removed.

3.1.8 Stochastic Geometry Basics: Tessellations

In general, a tessellation subdivides space into disjoint cells. Examples of tessellation classes are hyperplane, T- or Voronoi tessellations. The latter is a common approach to model the micro-structure of poly-crystalline materials. The following definitions are based on [17, 79].

Definition 23
For $S \subseteq \mathbb{R}^d$, let $\mathscr{S} = \{S_1, S_2, \dots\}$ be a countable set of compact subsets $S_i \subseteq S$, $i = 1, 2, \dots$ with non-empty interior. Then, $\mathscr{S}$ is called a *tessellation* of S if the following properties are fulfilled:

1. $(S_i \backslash \partial S_i) \cap (S_j \backslash \partial S_j) = \varnothing$ for all $i, j = 1, 2, \dots$ with $i \neq j$,
2. $\bigcup_{i=1}^{\infty} S_i = S$,
3. $\#\{S_i : S_i \cap B \neq \varnothing\} < \infty$ for all bounded Borel sets B ($\mathscr{S}$ is locally finite).

The sets S_i are called the *cells* or *regions* of the tessellation.

Remark 8
If $d = 2$ and all regions are triangles, then $\mathscr{S}$ is called a *triangulation*. In $d = 3$, a 2d surface can be approximated by triangles denoted as *surface triangulation*, see Figure 3.6 for an example. A tessellation in $d = 3$ where the regions are tetrahedrons is called *tetrahedrization*.

Remark 9
Let $\mathscr{P} = \{p_1, \dots, p_n\}$ be a locally finite set of points in $\mathbb{R}^d$ and $\mathrm{dist}(\cdot)$ a distance measure in $\mathbb{R}^d$ to define

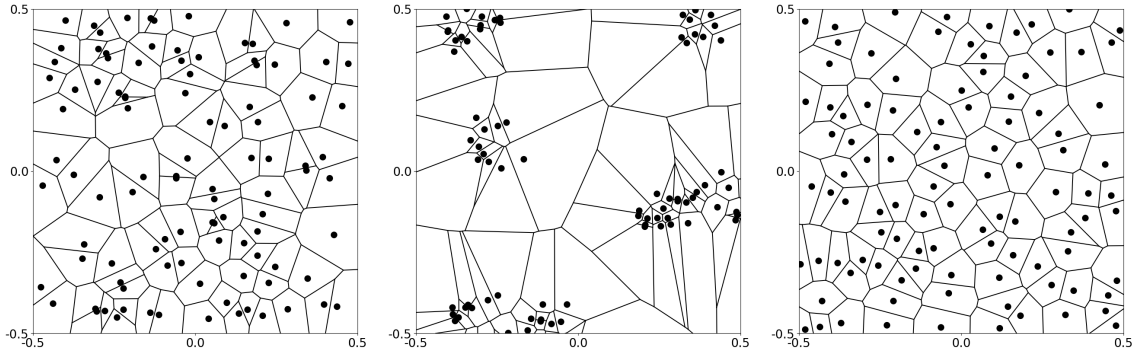
$$C(p_i, \mathscr{P}) = \left\{ x \in \mathbb{R}^d : \mathrm{dist}(x, p_i) \leqslant \mathrm{dist}(x, p_j) \text{ for all } p_j \in \mathscr{P} \right\}.$$

Then, a tessellation in $\mathbb{R}^d$ can be defined by $\mathscr{S}(\mathscr{P}) = \{C(p_i, \mathscr{P}) : p_i \in \mathscr{P}\}$. A point p_i is called *generator point* of *cell* $C(p_i, \mathscr{P})$ and $\mathscr{P}$ the *set of generator points*. If the set of generator points is obvious, we denote a cell by $C_i = C(p_i, \mathscr{P})$. A point is thus assigned the cell to whose generator it is closest to with respect to the chosen distance measure.

Example 7
Let $\mathscr{P} = \{p_1, \dots, p_n\}$, $2 \leqslant n < \infty$, be a set of distinct points and $\mathrm{dist}(\cdot) = \|\cdot\|_2$ the Euclidean distance. Then, the corresponding tessellation $\mathscr{S}(\mathscr{P})$ is called *Voronoi tessellation*. See Figure 3.5 for an overview of Voronoi tessellations generated by using realizations of different point process types as generator points.

Remark 10
As most applications consider bounded regions $W \subseteq \mathbb{R}^d$, we restrict to the bounded tessellation $\{C_1 \cap W, \dots, C_n \cap W\}$.

Remark 11
Tessellations have several lower-dimensional sub-structures. For $d = 2$, vertices and edges form the tessellation's 0- and 1-dimensional faces. For $d = 3$, the 2-dimensional faces are the cell's facets.



Poisson point process using $\lambda = 100$.

Matérn cluster point process using $\lambda_{\text{parent}} = 5$, $\lambda_{\text{child}} = 15$ and $r = 0.1$.

Matérn hardcore point process of type II using $\lambda = 130$ and $r = 0.045$.

Figure 3.5: Voronoi tessellations using realizations of different types of point processes in $W = [-0.5, 0.5]^2$. All point processes have approximately 100 points.

3.1.9 Basics from Graph Theory

Let $G = (V \times A)$ be a directed graph with V the set of vertices v_i, $i = 1, \ldots, n$, and $A \subseteq V \times V$ the set of directed arcs a. The start vertex of an arc a is denoted by α and its end vertex by ω. A *path* P in G is defined as a finite sequence of arcs $P = (a_0, a_1, \ldots, a_k)$, $k \geqslant 0$, with $\omega_i = \alpha_{i+1}$ for $i = 0, \ldots, k-1$. Alternatively, a path can be defined by the sequence of the visited vertices, that is $P = (w_0, \ldots, w_{k+1})$ for $w_i = \alpha_i$, $i = 0, \ldots, k$ and $w_{k+1} = \omega_k$.

Let $c : A \to \mathbb{R}_{>0}$ be a weight function assigning each arc a positive value. The *weight of a path* is then defined by $c(P) = \sum_{i=0}^{k} c(a_i)$.

Given two vertices $v, v' \in V$, the *shortest path* with $\alpha_0 = v$ and $\omega_k = v'$ is defined by minimizing the weight of all paths connecting v and v'. Famous algorithms for solving the shortest path problem are Dijkstra's [22], Bellman and Ford's [8] or the A* search algorithm [47].

3.1.10 Spline Interpolation

The following definitions are based on [84, 92].

Definition 24
For $n \in \mathbb{N}$, let $(x_0, y_0), \ldots, (x_n, y_n) \in \mathbb{R}^2$ with $x_0 < x_1 < \cdots < x_n$ be the control points. We consider a function $S : [x_0, x_n] \to \mathbb{R}$ that passes through these points, that is $S(x_i) = y_i$ for $i = 0, \ldots, n$.

- S is called *spline*, if it is a piece-wise polynomial function in the sense that $s_i = S\big|_{[x_i, x_{i+1}]}$ is a polynomial on $[x_i, x_{i+1}]$ for $i = 0, \ldots, n-1$.
- S is called a *cubic spline* if it is a spline with piece-wise cubic, twice continuously differentiable polynomials fulfilling $s_i'(x_{i+1}) = s_{i+1}'(x_{i+1})$ and $s_i''(x_{i+1}) = s_{i+1}''(x_{i+1})$ for all $i = 0, \ldots, n-2$. The resulting spline is thus also twice continuously differentiable.
- S is called a *natural cubic spline* if it is a cubic spline and if its second derivative is additionally zero at the curve's end vertices, that is $S''(x_0) = s_0''(x_0) = 0$ and $S''(x_n) = s_{n-1}''(x_n) = 0$.

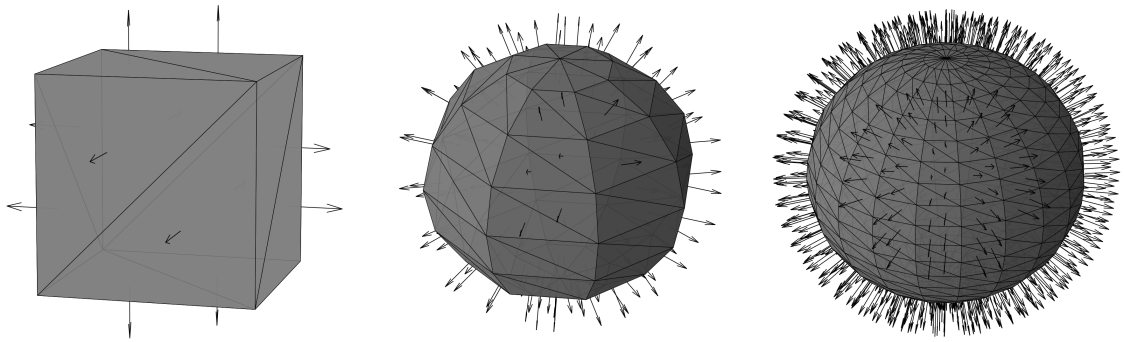


Figure 3.6: Triangulated surfaces of a cube and a sphere geometry. For the latter, different degrees of approximation are used. Normal vectors of triangles are added that represent the global surface normals of the geometry.

3.2 Fundamentals of Computer Graphics

This section deals with the basics of image synthesis using computer graphic techniques. As we focus on modeling surface defects and the surface meso scale geometry, this section serves as a short introduction on how to apply the outcomes of the proposed modeling methods and how the synthetic data generation using rendering is performed. First, a 3d scene is needed consisting of 3d objects, camera and light sources. Regarding the 3d objects we introduce their shape, texture and material separately to follow the classification into macro, meso and micro scale geometry as proposed in Section 2.1.1. Then, rendering describes the synthetic imaging process of the virtual camera. That is, generating a 2d image out of a 3d scene observed from the camera position. The definitions and descriptions from the following sections are based on [11, 51, 82].

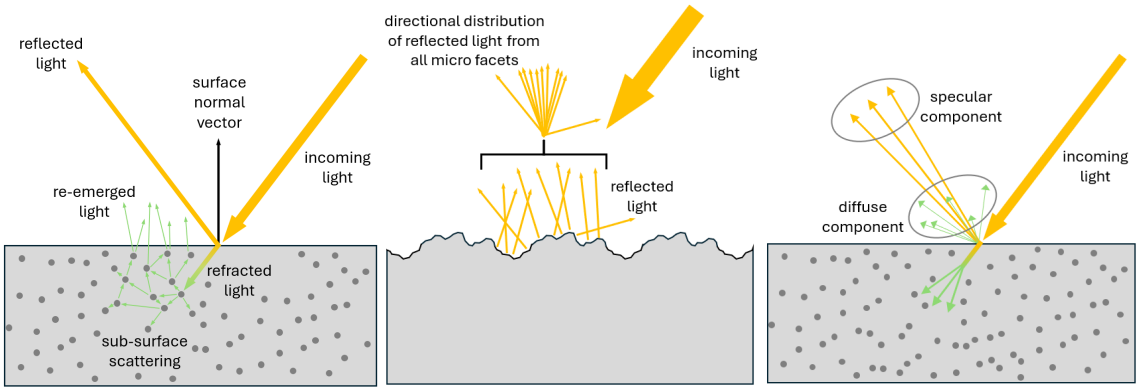
3.2.1 3d Object: Shape

The shape of an object is described by its macro scale geometry. Especially the object placement in relation to camera and illumination as well as the object surface normals are crucial for rendering, both are provided by the object geometry. Surface normals determine the behavior of light rays that hit the object surface, see Section 3.2.2 for more details.

Different methods to represent the object shape exist, e.g. polygon meshes and parametric surface representations or point clouds, see [11] for a more detailed overview. An appropriate representation is chosen dependent on the present application.

Since we focus on surfaces of rigid metal parts, thus the internal object structure is irrelevant, we use methods to represent the object geometry by its surface. For this purpose, polygon meshes are commonly used. That is, the object surface is provided as a set of polygons. A polygon is defined via a set of ordered vertices. Note that neighboring polygons share vertices. Thus, a polygon mesh consist of a set of the vertices and a set of polygons indicating the required vertices. Triangles are a typical choice for polygons because they allow efficient rendering of surfaces. We also restrict to object shape representations given by a triangulated surface mesh. Surface normal vectors for every triangle are thus provided directly, which are called global normal vectors from now on. Examples can be seen in Figure 3.6.

Engineers commonly use the computer-aided design (CAD) software to develop and design suitable object shapes. As we focus on industrial metal parts, they are usually



(a) Split of incoming light into reflection and refraction with following sub-surface scattering.

(b) Application of micro-facet theory to determine the directional distribution of reflected light.

(c) Specular and diffuse components used to define a BRDF.

Figure 3.7: Schematic drawings of light-surface interaction. Images are recreated based on [51].

provided in file formats that are supported by the CAD software. If the chosen data type does not represent the object as surface mesh, conversion to a suitable file format is required.

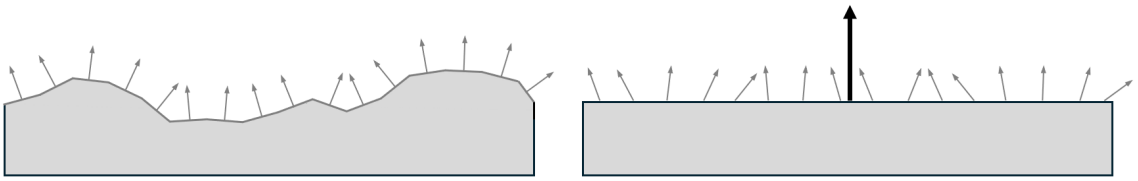
Considering smooth surfaces, the main drawback of polygon meshes is the piece-wise approximation of the surface by flat snippets. The coarser the representation, the more the discretization effect is visible as shown in Figure 3.6. However, using a fine approximation decreases speed and increases cost of the rendering process. Phong [83] for example provides a solution by interpolating the normals across polygons during rendering such that the surface appears smoother while the initial coarse approximation stays unchanged. Since in this work we consider objects having planar faces, smoothing the polygons is not necessary.

3.2.2 3d Object: Material

When considering the synthesis of images, the interaction of the 3d object with light is crucial, determined by the material. Assuming a perfectly flat object surface, the incoming light is split in exactly two directions defining *reflection* and *refraction*, see Figure 3.7a. The distribution of light assigned to each direction is material-dependent. The direction of reflection is independent of the material but is determined by the law of reflection using the surface normal vector and the direction of incoming light.

Refraction depends on the material. It splits up into *light absorption* and *light scattering*. Light absorption decreases the amount of light while its direction is not modified, the light converts into thermal energy. This determines the material color as light is a composition of different colors and just certain parts are absorbed. Metals for example absorb the total amount of refracted light. Light scattering is characterized by splitting up light into multiple directions while keeping the overall amount of light constant. For non-metal materials, refracted light scatters under the surface, called *sub-surface scattering*, including partial absorption in each bounce. Then, the remaining light may re-emerge from the surface but weakened, in different directions and at different locations depending on the scattering paths.

Now, perfectly flat surfaces are impossible due to the atomic structure of a material's



Real surface topography represented as piecewise flat patches with corresponding local surface normals.

Mapping of local surface normals to macro scale geometry using texture. Global surface normal is represented in black.

Figure 3.8: Schematic drawing of local perturbation of global surface normal using texture.

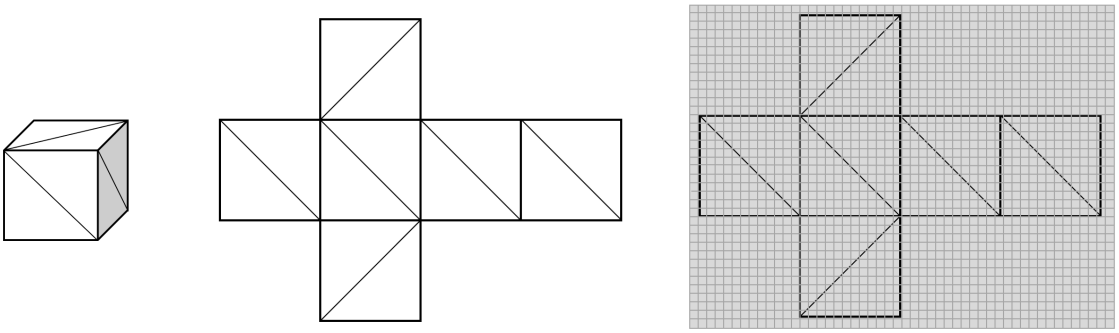
components. In computer graphics, optically flat surfaces are considered perfectly flat, although they are perfectly flat just within the range of wavelength of visible light, so in the scale of hundreds of nanometers. However, most surfaces have irregularities which are too small to be observable but are much larger than the wavelength of visible light. To model their behavior of light reflection and refraction, they can be treated as collection of many tiny optically flat but differently oriented facets approximating the object surface on micro scale, called micro-facet theory [102]. The result is then the aggregation of outcomes of all contributing micro-facets, see Figure 3.7b. A surface appears rougher the more the local micro-facet orientations differ from the global surface normal. The variance in the distribution of micro-facet normals is usually controlled by a parameter called *roughness*.

To model the light response especially of non-optically flat surfaces, *Bidirectional Reflectance Distribution Functions* (BRDF) are commonly used. A BRDF is a parametric function depending on the directions of incoming light. It gives the relative distribution of incoming light into directly reflected light and light re-emerged after sub-surface scattering. Typically, both components are considered separately by a specular and a diffuse term, respectively. See Figure 3.7c for an illustration.

3.2.3 3d Object: Texture

BRDFs are used to model the interaction of light with an object surface. So far, the entire object surface appears uniform which does not properly reflect the properties of real-world materials. Usually, the appearance is characterized by spatial variation, e.g. in color, roughness or the amount of reflectance, resulting from local, small changes in the material composition. To increase realism of the simulated object surfaces, spatial variation can be integrated by modifying the input parameters of the BRDF over the object surface. Therefore, textures are used which assign each surface point its individual parameters.

Moreover, textures are also used to introduce meso scale variations of the object surface geometry. Now, instead of adapting the parameters that control the BRDF, the global surface normal vectors given by the surface triangulation are perturbed, see Figure 3.8. Thus, small, local geometrical details are included by pretending to change the surface geometry without actually altering the macro scale object geometry itself. This is an efficient procedure to add small scale variations since modifying a triangulated surface to achieve such a level of detail results in complex meshes which lead to long rendering times and high demands on the memory. However, changing the local macro scale geometry by textures does not produce shadows during rendering which reduces realism of the synthetic images, especially for bigger geometrical modifications. In this work we use textures to model the meso scale surface geometry formed by different surface finishing methods of



Triangulated 3d geometry.

Unwrapped 3d geometry into 2d plane using UV-mapping.

Projecting UV-mapping onto texture image represented in gray.

Figure 3.9: Illustration of UV-mapping of a triangulated cube geometry.

metal objects, see Part I.

Note that both types of textures may be applied similarly to benefit from both adapting the BRDF parameters and perturbing the surface normals.

There are mainly two different texturing methods distinguished: procedural and image texturing. *Procedural texturing* uses mathematical functions to assign values to any arbitrary surface point. Thus, infinite resolution allows a huge level of detail. Moreover, textures can be varied in real-time directly during rendering. *Image texturing* is defined by 2d texture images containing the texture information which are then mapped onto the object surface. Therefore, a unique assignment of every surface point to a certain image pixel is required. A common approach is UV-mapping that unwraps the 3d object surface mesh into a planar 2d patch, see Figure 3.9, which can then be projected to the texture image. Note that large texture images may be needed dependent on the image's resolution and surface area.

In this work we focus on models determining the meso scale geometry that have texture images as outputs. To apply texture images that perturb the surface normals we use normal maps as described in Definition 4.

Independent of the utilized texturing method, aliasing can happen. It appears when using high frequency procedural textures or if too many texture pixels affect one single camera sensor pixel. For image texturing, it can be solved using the Nyquist sampling theorem. That is, either increasing the samples per pixel during rendering as described in Section 3.2.6 or reducing the number of pixels in the texture image. Moreover, the resolution of the texture image has to be chosen big enough such that the discretization given by the image grid is not visible.

3.2.4 Light

Illumination is necessary to make objects visible in the 3d scene. For image synthesis using computer graphic techniques, light is assumed to travel along rays from a light source into the 3d scene. Therefore, light is defined as color of particular intensity emitted from a certain location to a specific direction. Light originates from a light source, an object emitting light that is created by transforming energy from other sources. Many different types of light sources are distinguished in computer graphic applications. Here, we just want to highlight area and environmental light. Area light sources, e.g. ring lights, have a specified size, shape and direction of light emission. They represent a certain light source in an inspection setup. In contrast, environmental light, e.g. ambient light, simulates

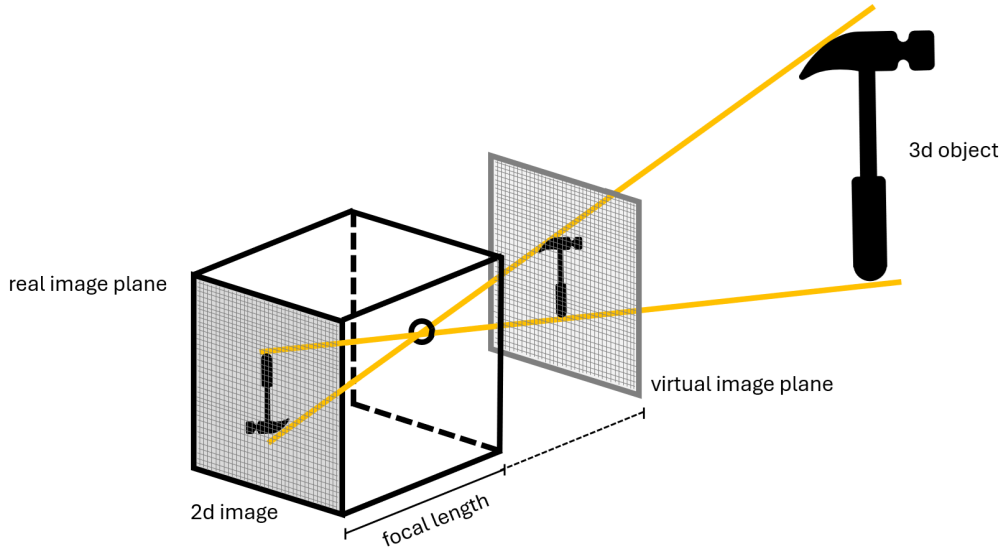


Figure 3.10: Schematic drawing of the components and the operating principle of a pinhole camera including its simulation using a virtual image plane.

the indirect diffuse light present in a 3d scene. As we consider a darkened inspection environment, we just need a simulation of the utilized area light and the environmental light can be neglected.

3.2.5 Camera

To enable realistic image synthesis, real-world camera concepts are used to model the performance of synthetic cameras. Here, we limit ourselves to pinhole cameras as they are the simplest camera type but still sufficient for our application.

In real-world applications, a pinhole camera is a light-proof box having an aperture at one side, a small hole to allow light to enter the box and a digital image sensor at the opposite inside of the box, see Figure 3.10. Light rays enter the box via the aperture and hit subsequently the digital image sensor. The exposure time determines the time period where light can enter the box and is usually controlled by opening and closing a shutter. Long exposure times allow more light to enter the camera resulting in brighter images but moving objects appear blurred. The size of the recorded area is determined by the focal length which coincides with the length of the box. A small focal length allows a larger recorded area and vice versa. Moreover, the bigger the diameter of the aperture, the more rays from differing directions influence the outcome of one image pixel which results in blurred images. In contrast, small apertures increase the image sharpness but reduce the amount of incoming light. To enable sharp images also when using large apertures, lenses are placed in front of the aperture resulting in more complex optical systems. Then, the aperture size controls the depth of field, the distance between real-world objects that appear sharp in the resulting image.

For modeling a virtual pinhole camera, the aperture size is assumed to be infinitely small such that every point in the recorded area maps to a unique point on the image sensor. That is, the aperture is considered a 3d point called eye. Then, a virtual image plane located in front of the camera can be generated replacing the real image sensor, see Figure 3.10. Thus, the virtual pinhole camera is completely defined by the position and

orientation of the eye, the resolution and size of the virtual image plane equal those of the real image senor and the distance between eye and virtual image plane coincides with the focal length. The main drawback of the synthetic pinhole camera is that it ignores the depth of field by assuming the aperture to be a point. Thus, every recorded object appears sharp independent of the distance to the camera which is not possible in real-world scenarios. Therefore, more advanced synthetic camera models including different lens systems exist. Also, optical aberrations such as global blurring or distortion produced by lenses are not taken into account. Those effects can be included by post-processing the rendered image.

3.2.6 Rendering

Rendering describes the process of virtually generating a 2d image out of a 3d scene observed from a specific point. Therefore, the light rays emitted from the provided light sources that subsequently hit the virtual image plane are crucial. Following the real-world imaging process to model rendering by tracking the light flow is called forward tracing. That is, the paths of all possible light rays emitted from every present light source have to be simulated including scattering between different objects in the scene and splitting of rays caused by rough surfaces. As just a little amount of rays enter the camera, it is inefficient to simulate all existing rays. Therefore, the usual approach is backward tracing, meaning tracking the essential rays that hit the virtual image plane but in opposite direction. As this approach takes the physical behavior of light rays into account, photo-realistic images result. Backward rendering subdivides into two steps: ray tracing to identify the crucial surface point defining the output of the corresponding sensor pixel and shading to determine its resulting color.

First, *ray tracing* solves the visibility problem. That is, finding the intersection point of a camera ray with any other object in the 3d scene closest to the eye. A camera ray is starting at the eye and passing through a certain pixel in the virtual image plane. Ray tracing is performed for every pixel. Using just one camera ray per pixel may cause the aliasing effect as described in Section 3.2.3. Therefore, multiple camera rays per pixel are usually used, specified by the number of samples per pixel (spp). Those camera rays are passing through different points within the same pixel determined by sampling from the area the pixel covers. Then, the resulting color values of every camera ray are combined to better approximate the real-world appearance. Although ray tracing is implemented efficiently for 3d objects represented as surface triangulations, increasing the number of triangles leads to a significant increase of intersection calculations. Thus, the run-time of the ray tracing algorithm highly depends on the complexity of the objects in the 3d scene.

Shading defines the computation of color for the surface points identified by ray tracing, called shading points. That color is computed by aggregating the intensities of all outgoing light rays emitted at that surface point whose direction coincides with the reversed direction of the considered camera ray. Those outgoing light rays emerge from incoming light rays of different directions, origins and strengths. The color and intensity of every outgoing light is determined by evaluating the BRDF at that surface point depending on the global surface normal, the incoming and outgoing light direction and the parameters provided by the texture at the shading point. The directions, colors and intensities of contributing incoming light rays are computed using light transport, see Figure 3.11. Different types of incoming light are distinguished. Direct light emerges from a light source which is not hidden by any other object in the scene. Indirect light is emitted from other objects by reflection or sub-surface scattering. For simulating indirect light, the procedure of ray tracing and shading has to be repeated iteratively for every object hit by the light

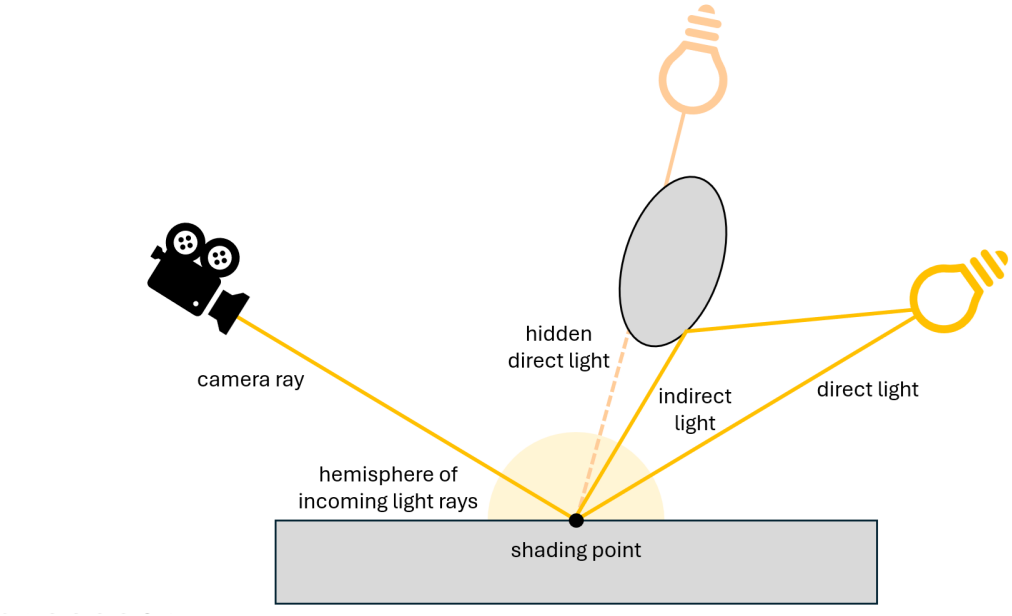


Figure 3.11: Schematic drawing of shading. Two representatives of possible incoming light ray directions are considered.

ray. For this purpose, path tracing introduced by Kajiya [61] is a common algorithm. It is a Monte-Carlo ray tracing based method that integrates all possible incoming light directions in the hemisphere over the shading point attenuated by the BRDF. The number of light ray bounces has to be chosen and influences the quality of the synthesized images. Less bounces decrease realism as much indirect light is ignored which results in too dark shadows. However, run-time grows while increasing the number of bounces.

Part I

Surface Texture Modeling

4 Introduction

Part I of this thesis deals with modeling the surface meso scale geometry as it strongly influences the light reflection, see Section 2.1.1 for more details. In computer graphics, textures are a common approach to model the variations of the surface meso scale geometry during rendering which is introduced in Section 3.2.3. We concentrate on texture synthesis methods for the surface topography of metallic objects with planar surfaces. The surface topography of industrial products is determined by the surface finishing methods used during their manufacturing. Here, we focus on two methods resulting in completely different surface patterns, namely sandblasting and milling. See Figure 4.1 for test objects treated by those machining methods.

In sandblasting, a strong air-jet mixed with an abrasive such as sand is used to mat a smooth surface, to smooth a rough surface or to remove contaminants such as rust or dirt. The process results in a non-periodic homogeneous random pattern. The process parameters are the pressure of the air-jet, the stand-off distance (distance between the tool and the surface), the grain size distribution, the blasting duration and the blasting angle.

Contrary, milling, more precisely face-milling, creates periodically arranged deterministic ring-shaped structures on the surface. It is a well-known machining method used to form work pieces in high precision. A milling tool equipped with an end mill moves over the surface following a prescribed path. While moving, the milling head rotates and its blades leave ring-shaped scratches. Process parameters include the diameter of the milling head, the length of the blades, the axial depth of cut, the radial depth of cut, the feed rate, the rotation speed and others [20], see Figure 4.2 for a description. Both the tool-path and the milling head diameter are deterministic while the variation within the rings is random.

In this part of the thesis, stochastic texture models based on topography measurements of such surfaces are presented. As the surface patterns differ significantly, separate modeling approaches for the two cases are used. Sandblasted surfaces are modeled by a combination of data-based texture synthesis methods that rely entirely on the measurements.

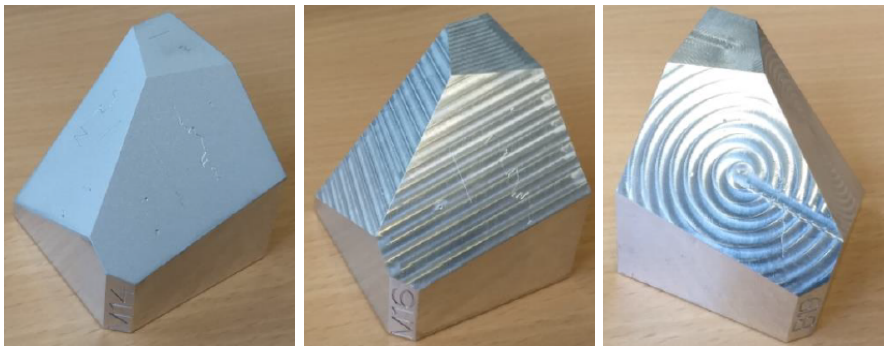
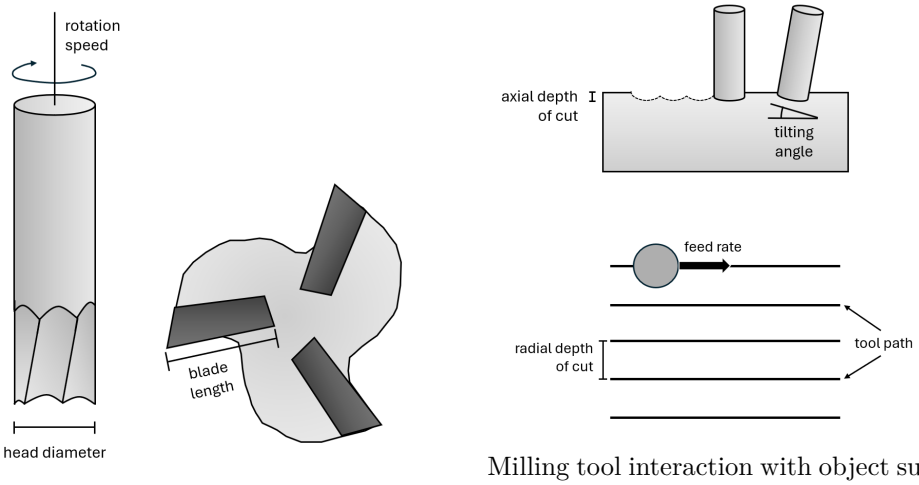


Figure 4.1: Test objects made of aluminum having a base about 5 cm in size and differently machined surfaces: sandblasted (left), parallel milled (middle) and spiral milled (right). The objects were produced by Fraunhofer IOF.



Exemplary milling tool.

Milling tool interaction with object surface (top) and milling path (bottom).

Figure 4.2: Schematic drawing of a milling tool and illustration of the milling procedure including descriptions of the milling parameters.

In contrast, the model for milled surfaces is procedural and includes all process-related parameters known from the machine settings.

First, Section 4.1 gives an overview of existing work for modeling sandblasted and milled surfaces. Since model fitting is based on topography measurements of such machined surfaces, the corresponding data is introduced in Section 4.3. Chapter 5 describes the use of phenomenological, data-based methods to synthesize texture images of sandblasted surfaces. A stochastic, procedural texture synthesis model is developed for milled surfaces which is introduced in Chapter 6. Chapter 7 serves as conclusion of the modeling approaches. The synthetic data generated in a virtual inspection environment using the synthetic texture images is provided in Chapter 8.

This part was published in [56] and partially published in [35].

4.1 Related Work

Various different approaches for the synthesis of the surface texture exist. *Process-based* approaches generate textures by simulating the physical process of the surface treatment in high precision. Contrary, *phenomenological* approaches create new surface textures based on a given realization thereof. In general, those simulation methods can be classified into statistics-based, patch rearrangement and machine learning methods, see [86] for an overview. *Statistics-based* methods generate random output textures that mimic selected statistics of the input. *Patch rearrangement* methods take given realizations in form of images as input, copy and randomly rearrange smaller patches of the image to generate a new texture image. *Machine learning* methods use generative models to generate output images resembling the input texture [38, 107, 113, 117, 118].

In computer graphics, texture synthesis methods are often divided into procedural and data-based. The first group includes parameterized methods that define new textures on a continuous domain. Considering machined surfaces, process parameters can be directly included as model parameters. However, they are restricted and specialized to a specific surface processing method. Contrary, for data-based models, the whole information to describe a pattern is delivered by an image. Since less assumptions on the imaged textures

are required, data-based models are typically more widely applicable than procedural methods.

For sandblasting, Oranli et al. [80] developed a process-based method describing the surface topography to assess the resulting surface roughness virtually. Varying air pressures and stand-off distances are considered. Aluminum particles serve as grains to sandblast polymeric objects. The model describes the surface deformation realistically when differently shaped grains hit the surface. The particle-specific velocity is computed and taken into account. Yu et al. [120] simulate the sandblasting process on wind turbine blades to estimate the best parameter choices. The grain shape is assumed to be spherical while different grain radii, air pressures and stand-off distances are considered.

A class of parametric models that can be used for stochastic simulation of sandblasted surfaces are Gaussian random fields. However, little is known about the relation of process and model parameters. We decided to use data-based methods as they are applicable to random stationary patterns with short-range correlations and are thus suitable for sandblasted surfaces. Details on the models are discussed in Chapter 5.

In the literature, there exist several process-based models for milling. Felhő et al. [28] consider various shapes of inserts which approximate the blades. Theoretical line profile roughness values of simulated surfaces are compared to measurements from identically machined real surfaces made of steel. This research was extended to analyze the influence of the feed rate, the speed at which the tool moves over the surface [29]. Besides the line profile roughness values, the surface profile roughness values are also compared. However, modeling is restricted to a small central part of the ring-shaped milling pattern. Kundrák et al. [64] investigated the influence of different inserts and feed rates, comparing two more lateral positions and identifying differences to the central part in the height values. The re-cutting effect resulting from multiple passages of the tool over the same spot on the surface is not taken into account in the surface models. Hadad and Ramezani [44] included the re-cutting effect by giving an in-depth description of the cutting edge path.

Moreover, Tosello et al. [103] summarize models to simulate surface topographies produced by different machining methods such as milling, grinding and laser machining to enable process optimization such that the resulting surfaces fulfill the requested functional properties. For milling, they mostly restrict to ball end milling and present various, mainly process-based methods.

Bosnar et al. [13] developed a stochastic model for circular, parallel and knurling texture synthesis without taking an explicit machining method into account. Parts of that model can be taken as basis for stochastic models for other milling types. To the best of our knowledge, there is no literature in which milled surface textures were considered as examples of data-based models. Since most of these methods reproduce spatially homogeneous structures they are less suitable for milled surfaces because of their ring-shaped components.

4.2 Our Contribution

In this part we introduce stochastic texture models for sandblasted and milled surfaces based on topography measurements of such surfaces. Their outputs are texture images of arbitrary size to cover the entire area of arbitrarily large surfaces. Separate modeling approaches are used for both cases because their surface patterns differ significantly.

A novel phenomenological approach to model sandblasted surfaces is proposed relying on a combination of existing texture generation methods. As the impact of individual machining parameters to the characteristic features of the resulting surface topography

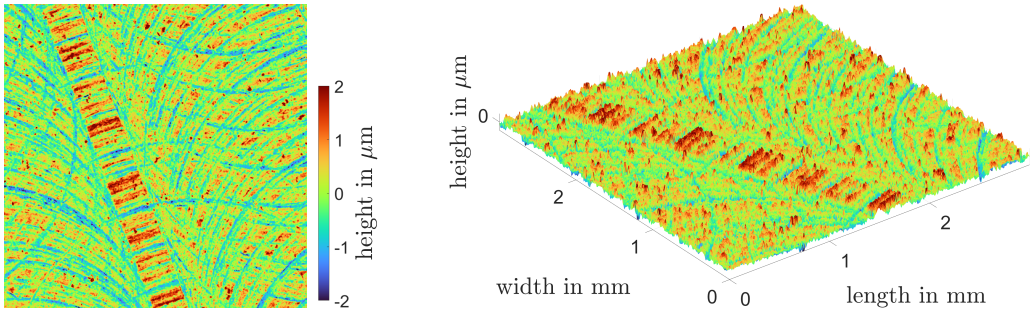


Figure 4.3: Measurement of a milled surface using a focus-variation microscope, provided by Fraunhofer IOF. 2d height image (left) and its representation as a surface in $\mathbb{R}^3$ (right). Imaged region is 3 mm × 3 mm.

is not straightforward, we use data-based methods. Thus, the impact of production-specific parameters is provided by the topography measurements. The presented procedure subdivides into two steps. First, patches of the measurement are used as input of statistics-based methods to generate texture images of the same size and resolution. Then, those newly generated texture patches serve as input for patch rearrangement methods to obtain arbitrarily large texture images.

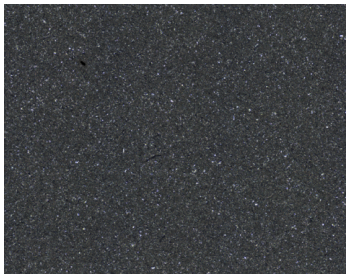
For milled surfaces, a novel process-based texture model is proposed as the typical surface pattern is directly determined by the machining procedure and its process-specific parameters. To integrate those process-related parameters, the proposed texture model is parameterized. Values of real world parameters known from the machine settings can directly be utilized, the remaining are estimated from the topography measurements. Moreover, the proposed model is procedural and thus given by a function defined in a continuous domain. To obtain discrete texture images, this function is evaluated on the image grid pixels. We present a multi-step model to individually integrate the impact of different machining components on the manufacturing process, which also affect the texture on different length scales.

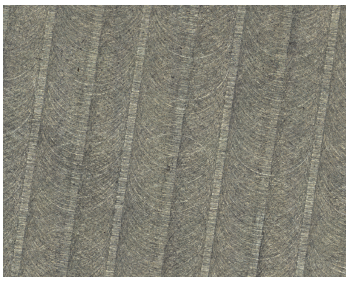
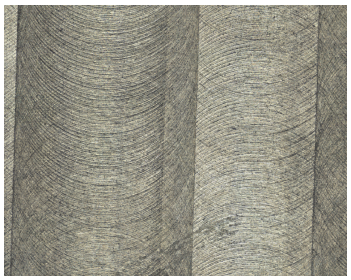
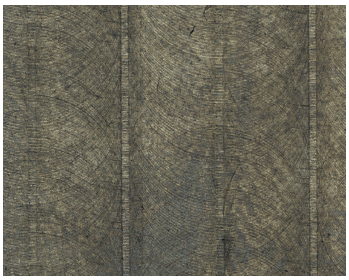
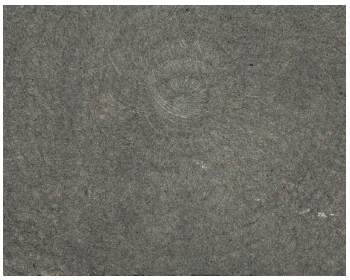
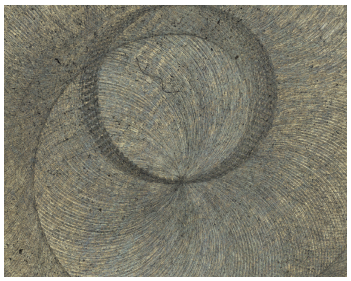
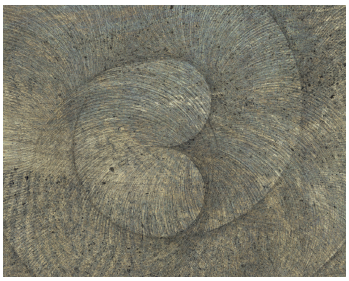
4.3 Measurement Data

The models for sandblasted and milled textures are developed based on measurements of the surface topography of machined work pieces with planar surfaces. Both machining methods were performed using several different sets of processing parameters. For sandblasting, we modify only the pressure. The stand-off distance was not considered since the samples are treated manually and thus it cannot be guaranteed that the distance will be kept constant. The grain size distribution was not known. For milling, there are three parameters. The tool-path defines whether parallel or spiral milling is applied. Then, the milling head diameter and the radial depth of cut (a_e [20]) are chosen. The latter defines the lateral step size of the milling tool proportionally to the milling head diameter. Table 4.1 gives an overview of all available samples.

The measurements were performed using the focus-variation microscope Bruker alicona InfiniteFocus G4 with a vertical resolution of 420 nm. Data are provided as 2d height images $\mathcal{H}$ of size $M \times N$ and homogeneous pixel spacing ν. In our measurements, we have $\nu = 1.75$ µm and image sizes varying between 4301×3175 and 4440×3288. See Figure 4.3 for an example. Measurement errors were edited by replacing the height values of the affected area using the mean heights of the surrounding pixels.

Table 4.1: Overview of the resulting surfaces of both manufacturing processes with the parameters used in each case. Optical 3d images obtained by focus-variation microscopy provided by Fraunhofer IOF. The imaged region for sandblasting is 7.3 mm × 5.7 mm and for milling 14.6 mm × 11.5 mm.

Sandblasting	
pressure = 2.5 bar	**pressure** = 6 bar
	

path / **head ⌀**		**Milling**		
		$\mathbf{a_e}$ = 0.2	$\mathbf{a_e}$ = 0.5	$\mathbf{a_e}$ = 0.8
parallel	4 mm			
	8 mm			
spiral	4 mm			
	8 mm			

5 Model for Sandblasted Surfaces

Sandblasting produces a homogeneous surface topography that can be interpreted as a stationary random field. Almost no deterministic components exist. Depending on the air-jet's pressure, the resulting patterns differ in depth and their degree of roughness, see Figure 5.1. We conjecture that the distance between the tool and the surface as well as the grain size distribution of the sand also influence the textures [7, 80].

Fitting Gaussian random fields to the measurements did not result in visually similar texture images. Additionally, it is not obvious how to define a parametric model that takes process-induced parameters into account. Thus, we use data-based texture generating methods that take the measured topography images as input and generate similar outputs. Here, we limit ourselves to statistics-based and patch rearrangement methods, as these are comprehensible and provide sufficiently good results. Although many statistical constraints are fitted within the presented statistics-based methods, their number is still small compared to the amount of features in deep learning methods for texture generation. In particular, the statistics-based method of Portilla and Simoncelli is still considered as state-of-the-art and used to compare to recent learning-based methods.

Statistics-based methods are used to generate texture images of the same size and pixel spacing as the measurements. Several methods are summarized and compared in Section 5.1. Then, patch rearrangement methods are used to produce images of arbitrary size. This procedure is detailed in Section 5.2.

5.1 Texture Generation Methods

Four different statistical data-based texture generating methods are considered: the asymptotic discrete spot noise (ADSN) [37,65], the random phase noise (RPN) [37] and the methods by Heeger and Bergen (HB) [49, 109] and by Portilla and Simoncelli (PS) [85, 108]. All methods are divided into two steps. First, certain statistics of the input image are

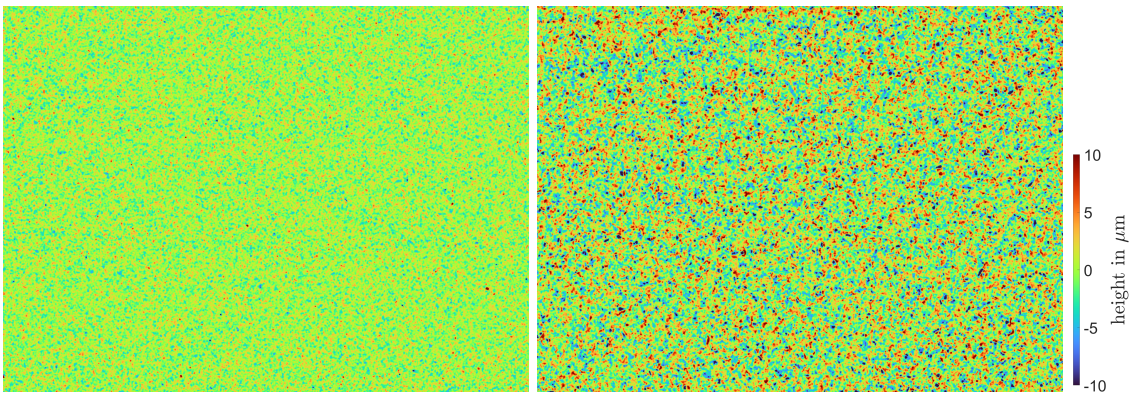


Figure 5.1: Measurements of sandblasted surfaces using an air-jet with pressure 2.5 bar (left) and 6 bar (right) obtained by focus-variation. Data is provided by Fraunhofer IOF. The imaged region is 5.6 mm × 7.7 mm.

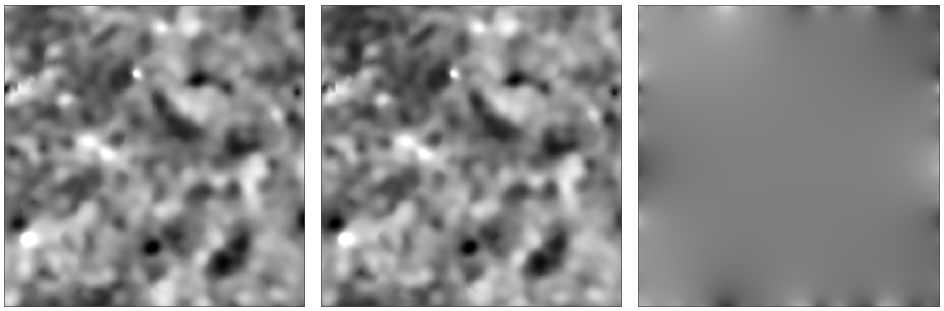


Figure 5.2: Decomposition into periodic and smooth component as proposed by Moisan [75], from left to right: $\mathcal{I}$, $\mathcal{I}_{\text{per}}$, $\mathcal{I}_{\text{smooth}}$.

computed. Then, the output image is calculated by first creating a Gaussian white noise image $\mathcal{G}$ with $\mathcal{G}(p) \sim \mathcal{N}(0,1)$ i.i.d. for $p \in \Omega$ and then adjusting it so that the desired statistics match. Here, let $\mathcal{I}$ be the input image of size $M \times N$ defined on the grid $\Omega = \{0, \dots, M-1\} \times \{0, \dots, N-1\}$. Since all methods are based on the Fourier transform, we assume $\mathcal{I}$ to have periodic boundaries to avoid artifacts. That is, the transitions of the left and right and also of the top and bottom edges are smooth. Moisan [75] proposed a method to obtain periodic boundary conditions by decomposing an image $\mathcal{I}$ into a periodic component $\mathcal{I}_{\text{per}}$ and a smooth component $\mathcal{I}_{\text{smooth}}$. The periodic component is obtained as the unique solution of the Poisson problem

$$\begin{cases} \Delta \mathcal{I}_{\text{per}} = \Delta_i \mathcal{I} \\ \text{mean}\left(\mathcal{I}_{\text{per}}\right) = \text{mean}\left(\mathcal{I}\right) \end{cases}$$

where Δ is the discrete periodic Laplacian and Δ_i the discrete Laplacian in the interior of Ω. Here, the interior is understood to be the entire domain excluding the outermost pixels. In general, the operators are defined for $p \in \Omega$ via

$$\Delta \mathcal{I}(p) = 4\,\mathcal{I}(p) - \sum_{q \in \mathcal{N}_4(p)} \mathcal{I}(q \bmod \Omega)$$

$$\Delta_i \mathcal{I}(p) = |\mathcal{N}_4(p) \cap \Omega|\,\mathcal{I}(p) - \sum_{q \in \mathcal{N}_4(p) \cap \Omega} \mathcal{I}(q)$$

with $(q_1, q_2) \bmod \Omega = (q_1 \bmod M, q_2 \bmod N)$ for $\cdot \bmod \cdot$ the modulo operator and $|\cdot|$ the number of elements. Thus, $\mathcal{I}$ and $\mathcal{I}_{\text{per}}$ coincide in the interior but have varying boundaries. Figure 5.2 shows the decomposition obtained by Moisan's method and Figure 5.3 proves the gained periodic boundary conditions.

5.1.1 Asymptotic Discrete Spot Noise

Galerne et al. [37] introduced the ADSN as the limit of the normalized discrete spot noise (DSN). The DSN of order n associated with input image $\mathcal{I}$ is given as sum of random shifts $\mathcal{I}_k(p) = \mathcal{I}(p - p_k)$ of $\mathcal{I}$ for p_k independent and identically uniformly distributed in Ω, namely $DSN_n(\mathcal{I}) = \sum_{k=1}^{n} \mathcal{I}_k$. By the central limit theorem, the DSN asymptotically follows a Gaussian distribution

$$\sqrt{n}\left(\frac{1}{n} DSN_n(\mathcal{I}) - \hat{\mu}_{\mathcal{I}}\right) \xrightarrow{d} ADSN\left(\mathcal{I}\right) \sim \mathcal{N}\left(0, \widehat{C}_{\mathcal{I}}\right),$$

where $\hat{\mu}_{\mathcal{I}}$ is the arithmetic mean and $\widehat{C}_{\mathcal{I}}$ the covariance matrix of $\mathcal{I}$. The ADSN can thus be simulated by sampling from the multivariate normal distribution $\mathcal{N}\left(\hat{\mu}_{\mathcal{I}}, \widehat{C}_{\mathcal{I}}\right)$.

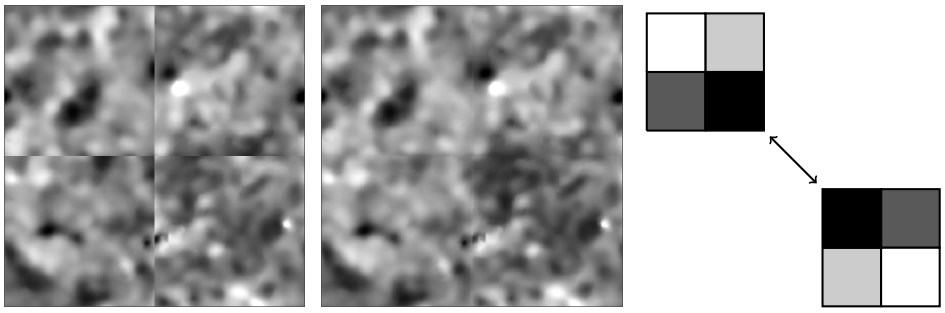


Figure 5.3: Swapped quadrants of images $\mathcal{I}$ (left) and $\mathcal{I}_{\text{per}}$ (middle) of Figure 5.2 using the procedure illustrated on the right.

The ADSN is a non-iterative and parameter-free method. Figure 5.4 shows an exemplary output. The Fourier modulus of a newly generated texture image is the Fourier modulus of the input image multiplied with Rayleigh noise [37]. Thus, minor differences are visible when comparing their sample auto-correlations, see Figure 5.5b.

5.1.2 Random Phase Noise

The random phase noise associated with an image $\mathcal{I}$ generates new texture images having the same Fourier modulus as $\mathcal{I}$ but a random phase, see [36,37,52] for more details. This method is also non-iterative and parameter-free.

Assume M and N to be even. Then, the RPN is defined in the Fourier domain $\widehat{\Omega}$ by $\widehat{RPN}(\mathcal{I})(\xi) = \widehat{\mathcal{I}}(\xi) \cdot \exp(\mathrm{i}\theta(\xi))$ for $\xi \in \widehat{\Omega}$. Here, θ is a uniform random phase which is defined as a function $\theta : \widehat{\Omega} \to (-\pi, \pi]$ fulfilling the following properties

1. $\theta(-\xi) = -\theta(\xi)$ for all $\xi \in \widehat{\Omega}$
2. $\theta(\xi) \sim \mathcal{U}((-\pi, \pi])$ for $\xi \in \widehat{\Omega} \backslash \left\{(0,0), \left(0, \frac{N}{2}\right), \left(\frac{M}{2}, 0\right), \left(\frac{M}{2}, \frac{N}{2}\right)\right\}$
3. $\theta(\xi) \sim \mathcal{U}(\{0, \pi\})$ for $\xi \in \left\{(0,0), \left(0, \frac{N}{2}\right), \left(\frac{M}{2}, 0\right), \left(\frac{M}{2}, \frac{N}{2}\right)\right\}$
4. For any subset $\widehat{\omega} \subset \widehat{\Omega}$ not containing symmetric points ($\xi \in \widehat{\omega} \Rightarrow -\xi \notin \widehat{\omega}$), the random variables $\{\theta(\xi) : \xi \in \widehat{\omega}\}$ are independent.

A Gaussian white noise image has a uniform random phase. In spite of Assumption 3. above, we assume additionally $\theta((0,0)) = 0$ such that $RPN(\mathcal{I})$ maintains the sample mean of $\mathcal{I}$. Note that the method is also valid for odd M or N. See Figure 5.4 for a realization of the RPN. Since $\mathcal{I}$ and $RPN(\mathcal{I})$ have the same Fourier modulus, their sample auto-correlations coincide as can be seen in Figure 5.5b.

5.1.3 Method of Heeger and Bergen

The following method was primarily introduced by Heeger and Bergen [49]. Briand et al. [109] published a more detailed version including an in-depth description of the method and possible extensions. In addition, an implementation is provided.

The HB-method iteratively modifies an initial image $HB_0 = \mathcal{G}$. Consider K iterations resulting in updated texture images HB_k, $k = 1, \ldots, K$. In each iteration, the histogram of HB_{k-1} is first matched to that of I. Then, the image is decomposed by using the steerable pyramid of P levels and Q orientations as introduced by Simoncelli and others [94,95], see Definition 17. Histogram matching is applied on all sub-bands to adapt the corresponding

image pyramid to that of $\mathcal{I}$. Finally, the updated texture image HB_k is reconstructed from the adapted image pyramid.

The histogram matching algorithm first sorts the gray values of both images and thus gives each pixel a rank. If ties occur, the rank is assigned according to the scanning order [109]. Then, each pixel in the image to be matched is assigned the gray value of that pixel in the reference image having the same rank.

The output of the HB-method is denoted by $HB_K^{P,Q}(\mathcal{I})$. Briand et al. [109] recommended $P = Q = 4$ and showed that convergence is fast such that $K = 5$ is sufficient. Additionally, they provided MATLAB-code which we used for the computation. Figure 5.4 shows that edge effects occur in form of elongated structures. In order to remove that artifact, the output's boundary region can be cropped.

5.1.4 Method of Portilla and Simoncelli

The method of Portilla and Simoncelli [85] is an extension of the HB-method. There are two major differences to the HB-method. First, the complex steerable pyramid is taken, see Definition 18. Second, instead of using image-wise histogram matching, several summary statistics are matched to those estimated from the input image. An algorithm for doing so is given in [85, 108]. A recommendation for the choice of relevant summary statistics is based on experimental evaluation done by Portilla and Simoncelli [85]. Marginal statistics (e.g. mean, variance, skewness) and auto-correlations on the low-frequency bands at each scale and auto- and pairwise cross-correlations of the modulus of all oriented sub-bands are considered as relevant. The necessity of the selected statistics is confirmed when showing the effect of removing some of them [85, 108].

The output of the iterative PS-method is denoted by $PS_K^{P,Q,S}(\mathcal{I})$. Compared to the HB-method, just one additional parameter is needed: the adjustment of auto-correlations is limited to its central area of size $S \times S$ for odd $S \in \mathbb{N}$. Vacher and Briand [108] discuss the influence of the parameter choices. In particular, they recommend to choose $S = 7$. Further, they provide code which we used for the computations. Figure 5.4 shows a realization of the PS-method.

5.1.5 Comparison of Methods

The input image is assumed to have periodic boundary conditions which is usually not the case for arbitrary images. We will use $\mathcal{I}_{\text{per}}$ as input image.

Figure 5.4 summarizes all texture generating methods introduced previously. Visually, they all resemble the texture in the input image while being pixel-wise different. The distributions of height values are similar as well, see Figure 5.5c. For further texture validation, we consider the Haralick features which are texture descriptors capturing local texture variations, see Section 3.1.6 for a more detailed description of the Haralick features. For computations we used the MATLAB-implementation provided by Löfstedt et al. [66]. Figure 5.5a shows convincing results for correlation, energy, homogeneity and contrast.

Information about spatial relations is given by the auto-correlation in Figure 5.5b. It is noticeable that the auto-correlation of the RPN resembles the auto-correlation of the input image since both have the same Fourier modulus. The result of the PS-method has a similar auto-correlation as well. However, the auto-correlations of the other two methods show that smaller structures are erroneously present in their output images. The different methods show similar results in the Haralick features. In terms of auto-correlation, RPN and PS are the best methods. Of those, RPN has a substantially faster run-time, see Figure 5.5d. Therefore, from now on we will use the RPN as texture generation method.

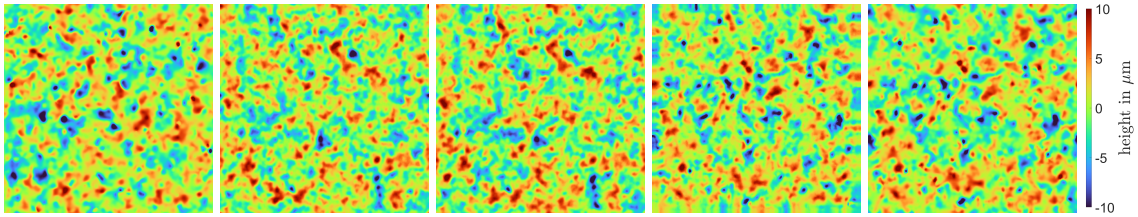
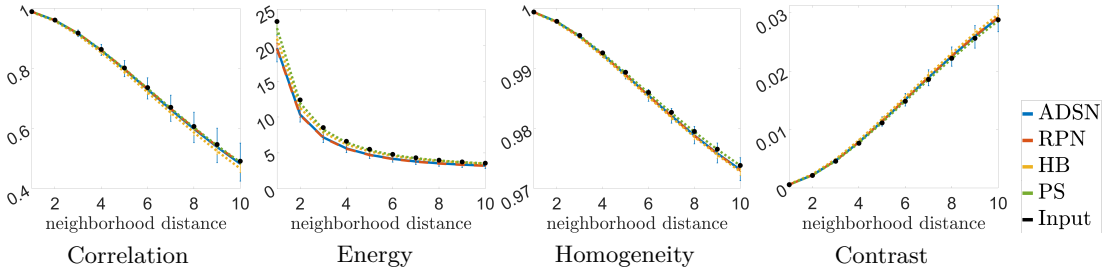
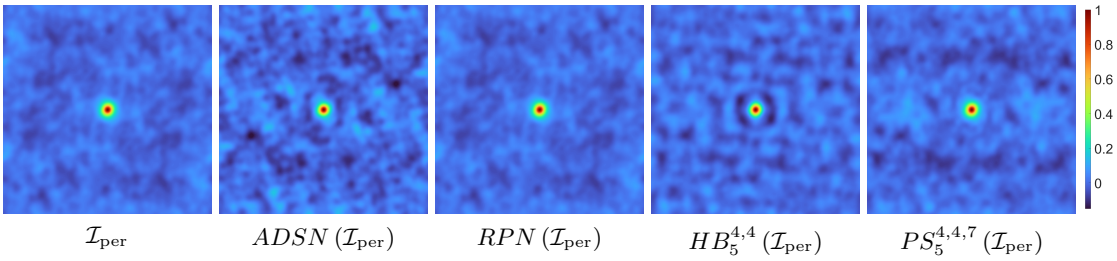


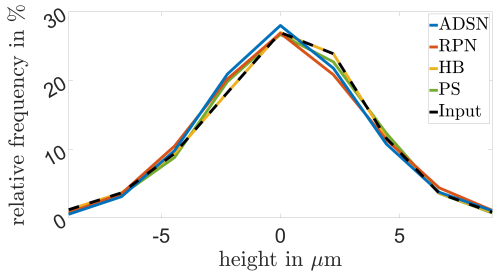
Figure 5.4: Output of texture generation methods using the same initial Gaussian white noise image. Image size is 512 × 512 and pixel spacing 1.75 µm which corresponds to an imaged region of 0.9 mm × 0.9 mm.



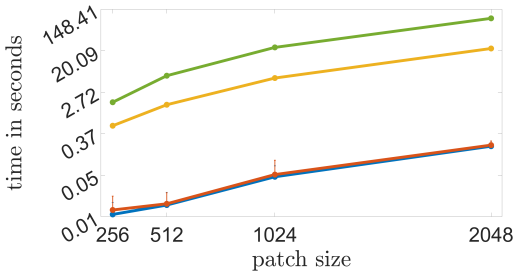
(a) Haralick texture features using the scaled 8-connected neighborhood. Lines represent means of 100 realizations of images of size 512 × 512. Dots represent the measurement. Error bars indicate minimal and maximal values. 64 gray-levels and the 8-connected neighborhood of varying distance are used to compute the GLCM.



(b) Auto-correlation matrices of images given in Figure 5.4.



(c) Height value distribution.



(d) Mean run-time as means of 100 repetitions of MATLAB-code of methods in logarithmic scale dependent on the patch size. Error bars indicate minimal and maximal values. Colors as in Figure 5.5c.

Figure 5.5: Comparison of texture generation methods. The run-time measurements were performed on a HP Z240 Tower Workstation equipped with Intel Core i7-3700 CPU at 3.40GHz × 8.

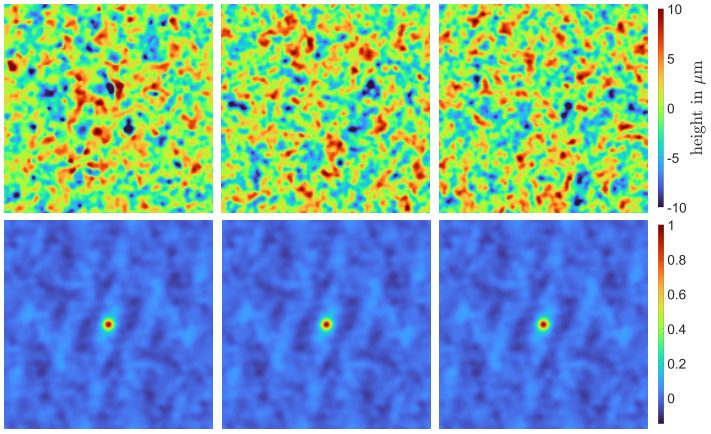


Figure 5.6: Adjustment of pixel spacing: Comparison between down-sampled reference image (left) and adjustments of pixel spacing either by down-sampling the synthetic texture image (center) or the input image (right). RPN is used for texture generation. Top: the textures' height image, bottom: their auto-correlations. Factor three is used for down-sampling, so image size reduces to 171×171 and pixel spacing is 5.25 µm. The covered area is $0.9\,\text{mm} \times 0.9\,\text{mm}$.

5.2 Simulating Texture Images of Arbitrary Size and Pixel Spacing

The methods presented in Section 5.1 generate images that maintain the image size and pixel spacing of the input image. Here, we discuss how to generate texture images $\mathcal{T}$ of arbitrary size $M_\mathcal{T} \times N_\mathcal{T}$ and pixel spacing $\nu_\mathcal{T} \geqslant \nu_\mathcal{I}$ larger than the input image. The case of higher resolved texture images ($\nu_\mathcal{T} < \nu_\mathcal{I}$) is not considered as the input $\mathcal{I}$ does not contain the required image information.

There are two possibilities to create down-sampled texture images. Either a texture image of the same pixel spacing as the input is generated and then down-sampled afterwards or the synthetic texture image is generated directly from a down-sampled version of the input. Results of both options when using nearest neighbor interpolation for down-sampling are shown in Figure 5.6. Visual comparison does not reveal obvious differences to the down-sampled input image. Also, the related auto-correlation matrices look similar. The main advantage of down-sampling prior to texture synthesis is the drastically reduced computation time. Thus, this procedure is used from now on.

Now, consider the adjustment to generate texture images of arbitrary size. A reduction of the image size is achieved by simple cropping of the input image or the output. Note that HB and PS require input image sizes to be powers of two.

When attempting to increase the output image size compared to the input, the adjustment becomes more difficult. Under certain conditions on $M_\mathcal{T}$ and $N_\mathcal{T}$, HB and PS can produce output images that are larger than the input image [108,109]. When using ADSN and RPN, the input image is extended to match the desired output size [37]. That is, take an image of the desired size $M_\mathcal{T} \times N_\mathcal{T}$ with the input image's sample mean $\hat{\mu}_\mathcal{I}$ as constant padding value and place the normalized initial image $\sqrt{M_\mathcal{T} N_\mathcal{T}}/\sqrt{M_\mathcal{I} N_\mathcal{I}}\,(\mathcal{I} - \hat{\mu}_\mathcal{I}) + \hat{\mu}_\mathcal{I}$ in its center to maintain the sample mean and variance of $\mathcal{I}$, see Figure 5.7 top row. The border of the texture image in the central part is multiplied with a sigmoid function to obtain a smooth transition to the padding. The enlarged image is then used as input image.

Using an extended input image may result in artifacts. For example, the auto-correlation

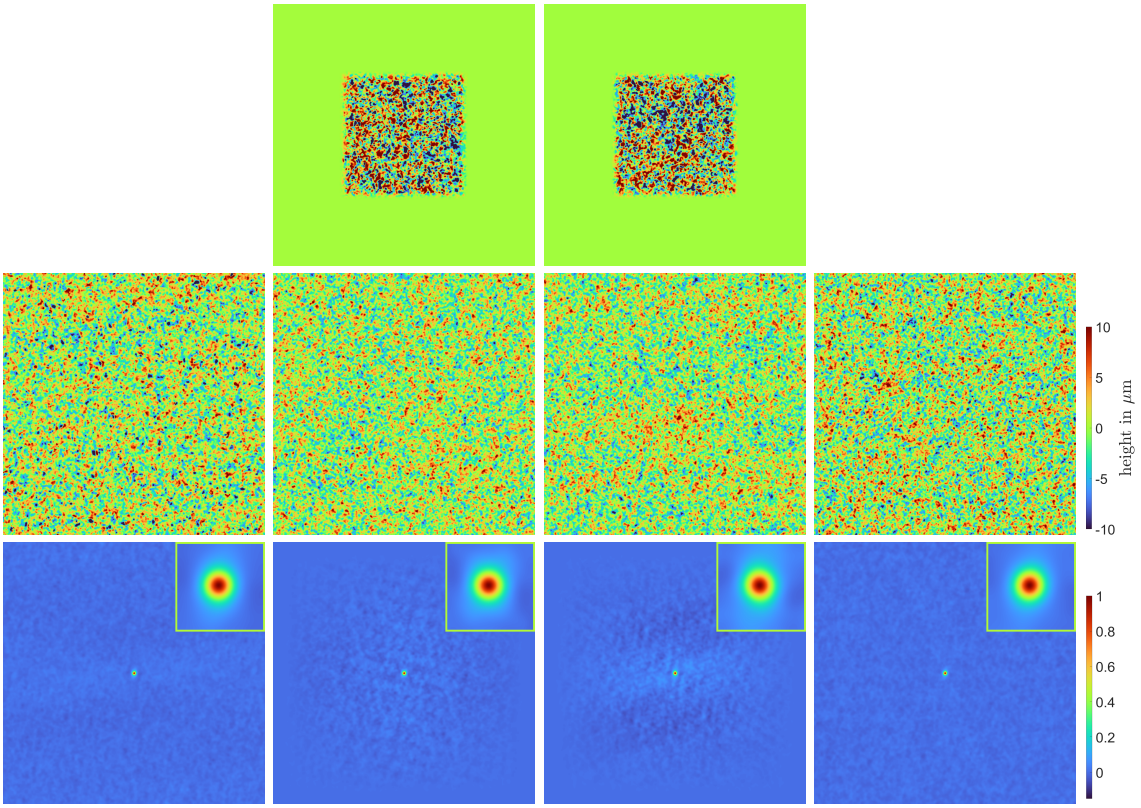


Figure 5.7: Comparison between methods to obtain texture images larger than the input image. Height images in the middle row: reference image (left), two realizations of the RPN (center) and a texture image generated using EF-stitching (right). EF-stitching is applied for patches of size 256×256 and overlap width 128. Input images for the RPN in the top row: extensions of different normalized cut-outs of size 342×342 of the reference image. Corresponding autocorrelations with zoom-in of the central part in the bottom row. Images are of size 684×684 and have pixel spacing 5.25 µm such that they cover an area of size $3.6\,\text{mm} \times 3.6\,\text{mm}$.

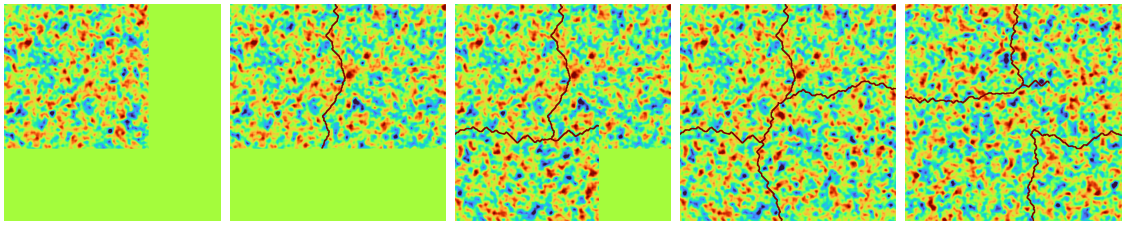
of a synthetically generated texture image using a patch of the given measurement as input (Figure 5.7 center right) indicates an anisotropic behavior along the x-axis which is not observed in the reference measurement (Figure 5.7 left). Especially when the input image is much smaller than the desired output ($M_{\mathcal{I}}, N_{\mathcal{I}} \ll M_{\mathcal{T}}, N_{\mathcal{T}}$), the resulting pattern looses randomness since structures are repeated periodically.

This effect is avoided when simulating smaller texture patches first which are then stitched together (EF-stitching). The procedure is explained in Section 5.2.1 and is considered the preferred method for generating larger texture images.

In summary, the procedure to generate texture images of arbitrary size and pixel spacing is as follows:

1. **Adaptation of pixel spacing:** Down-sample the input image by a factor $\nu_{\mathcal{T}}/\nu_{\mathcal{M}}$.

2. **Adaptation of image size:** If the down-sampled input image has at least the size of $\mathcal{T}$, crop the image and apply a texture generation method. Otherwise, take patches of the down-sampled input image at random positions as input for a texture generation method and stitch those simulations using EF-stitching. Note that the



Start | Horizontal alignment | Vertical alignment | L-shaped alignment | Disconnected paths

Figure 5.8: Illustration of original EF-stitching [25,87] using patches of size 512×512 and overlap width $o = 256$. The minimal paths are marked.

run-time of the texture generation method depends on the chosen patch size, see Figure 5.5d.

5.2.1 Stitching Method by Efros and Freeman

EF-stitching [25] is a non-parametric phenomenological texture generation method based on stitching together small patches taken from the input image. It is an extension of the method of Efros and Leung [26], in which a new image is synthesized pixel by pixel. Raad and Galerne [87] have taken up Efros and Freeman's method and published an in-depth description of it. Additionally, they describe a possible partial parallelization of the computation.

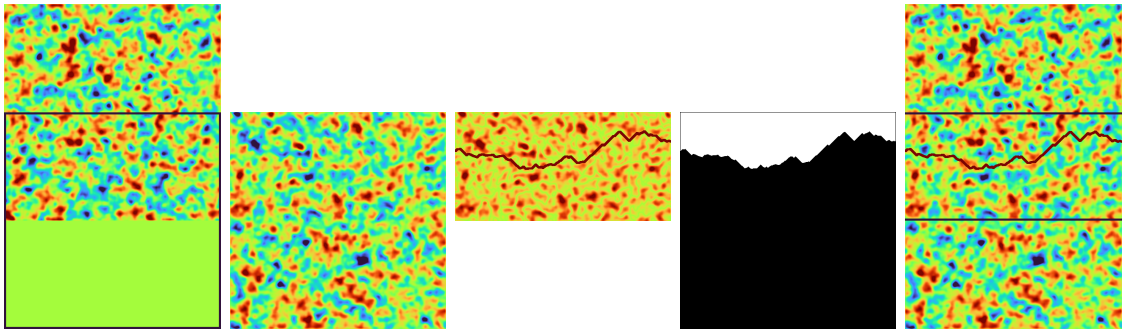
When applying EF-stitching, patches taken from the input image are placed equidistantly in the output image, taking into account an overlap region. Within this region, a cutting edge for stitching is determined as a path where pixel values in the images are most similar. To smooth the transition of both patches, the mean filter of size 3×3 is applied to the seam pixels. The method inserts new patches in raster scan order (from top left to bottom right). We use a modification of that method since newly generated patches are taken instead of cut-outs of the input image.

Assume quadratic patches $P_{i,j}$ of size $M_P \times M_P$ for $i,j = 1,\dots,n$ and an overlap region of width $1 < o < M_P$. The resulting patchwork image $\mathcal{I}$ is of size $M_\mathcal{I} \times M_\mathcal{I}$ with $M_\mathcal{I} = (M_P - o) \cdot n + o$. A new patch $P_{i,j}$ is inserted into $\mathcal{I}$ by updating the corresponding cut-out $\mathcal{I}_{i,j}$. A mask $A_{i,j} \in \{0,1\}^{M_P \times M_P}$ decides whether the existing part or the new patch is taken. That is, $\mathcal{I}_{i,j}$ is updated to $A_{i,j} \cdot \mathcal{I}_{i,j} + (1 - A_{i,j}) \cdot P_{i,j}$. Depending on the alignment of both patches, we distinguish between horizontal, vertical and L-shaped stitching, see Figure 5.8. The corresponding masks are defined as follows

$$A_{i,j} = \begin{cases} \mathbf{0} & \text{if } i = 1,\ j = 1 \\ \text{horizontal}(P_{i,j}, \mathcal{I}_{i,j}) & \text{if } i > 1,\ j = 1 \\ \text{vertical}(P_{i,j}, \mathcal{I}_{i,j}) & \text{if } i = 1,\ j > 1 \\ \text{L-shaped}(P_{i,j}, \mathcal{I}_{i,j}) & \text{if } i > 1,\ j > 1. \end{cases}$$

Given the position of the patch, the type of overlap region $\Omega^\bullet$, $\bullet \in \{h, v, L\}$ is determined. For horizontal alignment, a vertically elongated rectangle serves as overlap region $\Omega^h = \{1,\dots,o\} \times \{1,\dots,M_P\}$ while for vertical alignment we get $\Omega^v = \left(\Omega^h\right)^T$. If the patch's alignment is none of both, the overlap region $\Omega^L = \Omega^h \cup \Omega^v$ is L-shaped.

To determine the mask, we compute the 8-connected path within the corresponding overlap region minimizing the sum of point-wise squared differences $e(p) = (\mathcal{I}_{i,j}(p) - P_{i,j}(p))^2$ for $p \in \Omega^\bullet$. It separates the patches where they are most similar.



Position of $\mathcal{I}_{1,2}$ within $\mathcal{I}$ | $P_{1,2}$ | $e\left(\Omega^v\right)$ with ζ^h | $A_{1,2}$ | Stitching based on ζ^h

Figure 5.9: Illustration of EF-stitching for vertical alignment using patches of size 512×512 and overlap width $o = 256$.

In case of vertical alignment, the overlap region is divided by a horizontal path $\zeta^h = \left(\zeta_1^h, \ldots, \zeta_{M_P}^h\right)$. The path is oriented from right to left such that for successive path elements $\left(\zeta_k^h\right)_1 - \left(\zeta_{k-1}^h\right)_1 = -1$ for $k = 2, \ldots, M_P$. The mask is then computed by

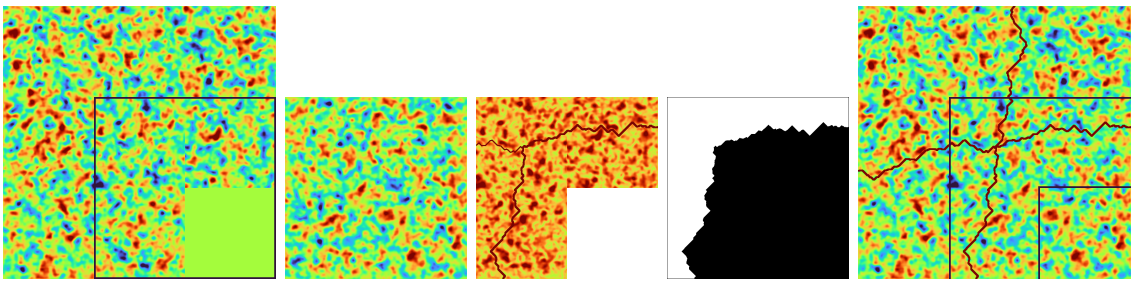
$$\text{vertical}(P_{i,j}, \mathcal{I}_{i,j})(p) = \begin{cases} 0 & \text{if } p_2 \geqslant \left(\zeta_{p_1}^h\right)_2 \\ 1 & \text{else} \end{cases}$$

for $p \in \{1, \ldots, M_P\}^2$. Figure 5.9 illustrates the stitching procedure for vertical alignment. The formulation for horizontal alignment is analogous.

An L-shaped path ζ^L is a combination of a horizontal and a vertical path connected at a point located in $\{1, \ldots, o\}^2$. Simply searching for the minimal L-shaped path in the overlap region as described in [87] may result in a path being disconnected to the paths constructed in the previous stitching steps, see Figure 5.8. This causes discontinuities in the stitched image. To avoid this problem, we elongate the existing horizontal path $\zeta^{L,h}$ from the previous stitching step optimally and find the minimal vertical path $\zeta^{L,v}$ connected to that elongation. The mask for L-shaped overlap is defined by

$$\text{L-shaped}\left(\mathcal{P}_{i,j}, \mathcal{I}_{i,j}\right)(p) = \begin{cases} 0 & \text{if } p_1 \geqslant \left(\zeta_{p_2}^{L,v}\right)_1 \text{ and } p_2 \geqslant \left(\zeta_{p_1}^{L,h}\right)_2 \\ 1 & \text{else} \end{cases}$$

for $p \in \{1, \ldots, M_P\}^2$. Figure 5.10 illustrates the adapted EF-stitching for L-shaped alignment.



Position of $\mathcal{I}_{2,2}$ within $\mathcal{I}$ | $P_{2,2}$ | $e\left(\Omega^L\right)$ with ζ^L and the old path | $A_{2,2}$ | Stitching based on ζ^L

Figure 5.10: Illustration of adapted EF-stitching for L-shaped alignment using patches of size 512×512 and overlap width $o = 256$.

6 Model for Milled Surfaces

Milling produces significantly different textures than sandblasting, compare the measurements given in Figure 4.3 and Figure 5.1. In particular, face-milled surfaces are characterized by a non-stationary pattern of periodic ring-shaped indentations. These structures cannot be reproduced by the methods introduced in Section 5.1 since they are only suitable for stationary textures with short-range correlations [37, 86, 108]. Thus, a separate model to generate textures of milled surfaces is needed. Additional measurements of milled surfaces using all milling parameter configurations to be investigated are shown in Table 6.1.

The characteristic ring-shaped indentations are formed by the milling tool when following its prescribed path along the material surface. During this motion, the cutting edge of the milling head follows a cycloidal path. In our model, this path will be simplified to a pattern of successive rings, see Figure 6.1.

The milling model is introduced as a parameterized function $f : \mathbb{R}^2 \to \mathbb{R}$ on a continuous domain and is thus procedural. It is then evaluated at given image grid points to obtain the texture image $\mathcal{T}$. A model containing $n \in \mathbb{N}$ rings is given by $f(x) = f_n(R_1, \ldots, R_n)(x)$. The modeling procedure consists of three steps describing the appearance, interaction and location of the rings as follows:

1. the appearance of an individual ring is described by the function $R_k : \mathbb{R}^2 \to \mathbb{R}$,
2. the interaction between n rings is described by the mapping $f_n : \left(\mathbb{R}^2 \to \mathbb{R}\right)^n \to \left(\mathbb{R}^2 \to \mathbb{R}\right) : (R_1, \ldots, R_n) \mapsto f_n(R_1, \ldots, R_n)$,
3. the tool-path providing the position and order of the rings by defining their center points $c_k \in \mathbb{R}^2$.

These sub-models are explained in more detail in Section 6.1, Section 6.2 and Section 6.3 below. The milling process is controlled by several parameters both for the milling head, e.g. the head diameter and the blade length, and for the tool-path such as the feed rate. The known real-world parameters are integrated into the model and the measurements are used to estimate the remaining unknown parameters. The adaptation of the model to given data is described in Section 6.4.

6.1 Sub-Model for Ring Appearance

A ring $R_k : \mathbb{R}^2 \to \mathbb{R}$ approximates the cut of one cutting edge during a complete rotation of the milling head. The shapes of indentation resulting from the cutting process and of possi-

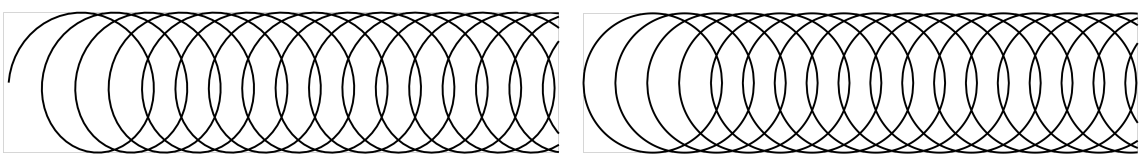
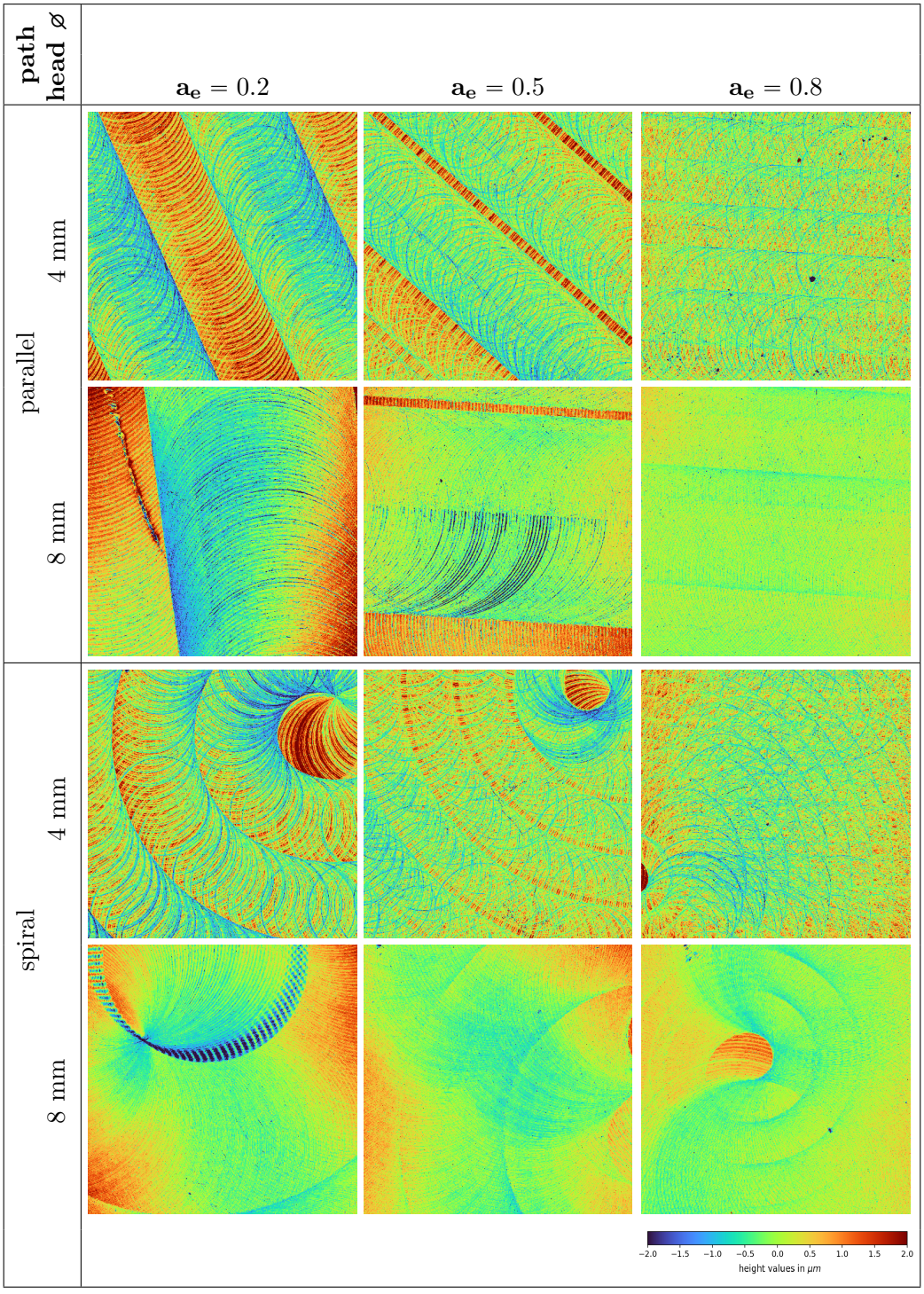


Figure 6.1: Path of one cutting edge (left) and its approximation with rings (right).

Table 6.1: Overview of 2d height images of milled surfaces using the parameters introduced in Section 4.3. Measurements are performed by Fraunhofer IOF using white light interferometry. The imaged region is 10 mm × 10 mm and the resolution is $\nu = 2.21$ µm.



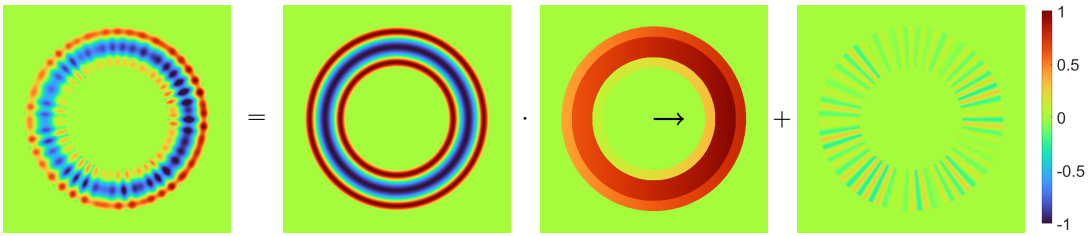


Figure 6.2: Illustration of the composition of the sub-model for the ring appearance.

ble material deposition are captured by the shape function $S_k : \mathbb{R}^2 \to [-1, 1]$. In practice, the milling tool is often tilted forwards in direction of the tool-path to prevent re-cutting [44, 63]. This circumstance is modeled by the tilting function $T_k : \mathbb{R}^2 \to [-1, 1]$ which is multiplied with the shape. Moreover, irregularities can occur during the milling process due to vibrations or resistances in the material. These are added as noise $N_k : \mathbb{R}^2 \to [-1, 1]$ to the tilted ring. Thus, the sub-model for a ring reads $R_k = S_k \cdot T_k + N_k$, see Figure 6.2.

6.1.1 Shape

To define a ring, the following spatial components are needed: the center point $c_k \in \mathbb{R}^2$, the radius $r \in \mathbb{R}_{>0}$ and the width w_k^-. The location of the center point depends on the tool-path, described in Section 6.3. The radius is directly given by the diameter d of the milling tool as $r = d/2$. The width is modeled by a random variable $w_k^- \sim \mathcal{N}(\mu_{w^-}, \sigma_{w^-})$ whose mean $\mu_{w^-} \in (0, r)$ is given by the length of the cutting edge. The set of points belonging to the ring is denoted by $P_k^- = \{x \in \mathbb{R}^2 : r - w_k^- \leqslant \|x - c_k\|_2 \leqslant r\}$. In the following, we define several indentation shapes which are illustrated in Figure 6.3.

1. **Indicator function**
   This shape function is constant on the set of points lying in the ring. Thus, there are no height differences within the ring. We define this shape function by

$$S_k^{\text{Indicator}}(x) = \begin{cases} -1 & \text{if } x \in P_k^- \\ 0 & \text{else.} \end{cases}$$

2. **Cosine function**
   The indicator shape function is extended to have a smooth transition and to resemble the shape of the cutting edge. The function is modeled using the cosine function and is defined based on the normalized distance

$$D_k^- : \mathbb{R}^2 \to [-1, 1] : x \mapsto \left(\|x - c_k\|_2 - r + \frac{w_k^-}{2} \right) \cdot \frac{2}{w_k^-} \cdot \mathbb{1}_{P_k^-}(x)$$

   between points in the ring and the ring's central circle. The resulting shape function is

$$S_k^{\text{Cosine}}(x) = \begin{cases} -\cos\left(\frac{\pi}{2} \cdot D_k^-(x)\right) & \text{if } x \in P_k^- \\ 0 & \text{else.} \end{cases}$$

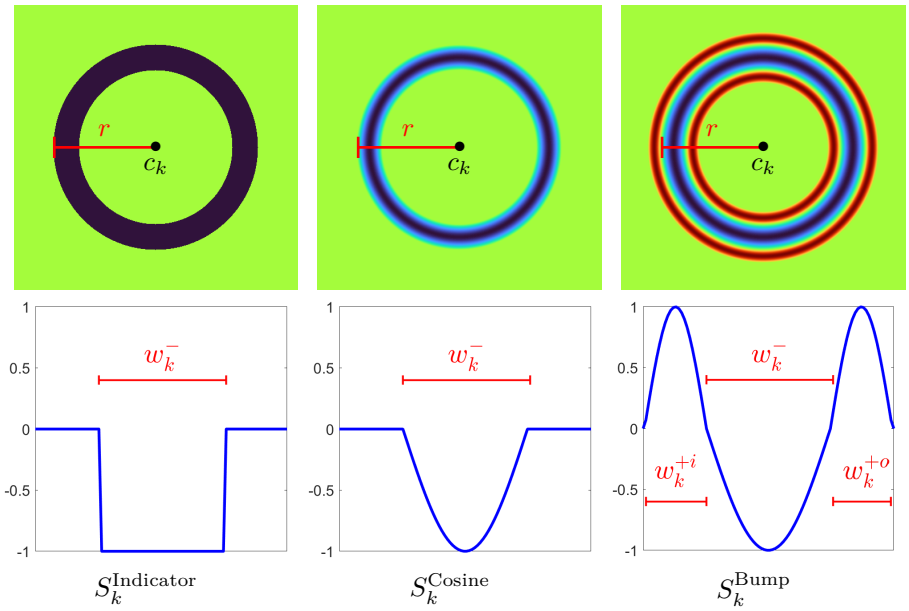


Figure 6.3: Illustration of shape functions and their parameters: height images (top) and 1d-intersections thereof (bottom).

3. **Bump function**
   The cosine shape function is extended to include accumulations outside the indentation caused by material deposition. These are modeled by positive parts of a cosine function on two rings outside and inside P_k^-. The inner and outer ring have widths $w_k^{+i} \sim \mathcal{N}(\mu_{w^{+i}}, \sigma_{w^{+i}})$ and $w_k^{+o} \sim \mathcal{N}(\mu_{w^{+o}}, \sigma_{w^{+o}})$. Denote the set of points lying in the inner and outer ring by

$$P_k^{+i} = \left\{x \in \mathbb{R}^2 : r - w_k^- - w_k^{+i} \leqslant \|x - c_k\|_2 \leqslant r - w_k^-\right\},$$
$$P_k^{+o} = \left\{x \in \mathbb{R}^2 : r \leqslant \|x - c_k\|_2 \leqslant r + w_k^{+o}\right\}.$$

   The distance functions $D_k^{+i}(x)$ and $D_k^{+o}(x)$ are defined in analogy to $D_k^-(x)$. The complete shape function then reads

$$S_k^{\text{Bump}}(x) = \begin{cases} \cos\left(\frac{\pi}{2} \cdot D_k^{+i}(x)\right) & \text{if } x \in P_k^{+i} \\ -\cos\left(\frac{\pi}{2} \cdot D_k^-(x)\right) & \text{if } x \in P_k^- \\ \cos\left(\frac{\pi}{2} \cdot D_k^{+o}(x)\right) & \text{if } x \in P_k^{+o} \\ 0 & \text{else.} \end{cases}$$

6.1.2 Tilting

So far, milling shapes were assumed to be aligned to the milled surface. Tilting of the milling head introduces a depth gradient within the rings with increasing depth towards the front, i.e., in motion direction of the milling tool. For a ring R_k, this direction is given by the angle $\theta_k \in (-\pi, \pi]$ between the planar tool-path at point c_k and the x-axis. It can directly be computed by

$$\theta_k = \arccos\left(\frac{(c_{k+1})_1 - (c_k)_1}{\|c_{k+1} - c_k\|_2}\right).$$

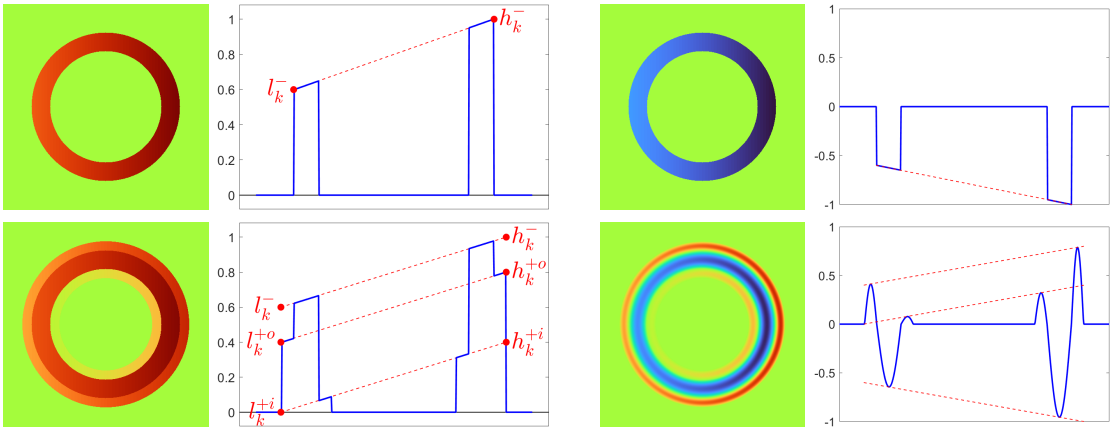


Figure 6.4: Illustration of tilting with $\theta_k = 0$, applied to indicator (top) and bump shape (bottom). Height images and their 1d-intersections.

Tilting is introduced by location dependent scaling of the shape function S_k. The slope of the tilting plane is defined by the highest and the lowest scaling factors $h_k^- \sim \mathcal{N}(\mu_{h^-}, \sigma_{h^-})$ and $l_k^- \sim \mathcal{N}(\mu_{l^-}, \sigma_{l^-})$, respectively, which are applied to the fore- and rearmost point of the indentation ring. The plane of contact between milling head and surface thus becomes

$$\text{plane}_k^-(x) = \frac{h_k^- - l_k^-}{2r} \left(\cos(\theta_k) \cdot (x_1 - (c_k)_1) + \sin(\theta_k) \cdot (x_2 - (c_k)_2)\right) + \frac{h_k^- + l_k^-}{2} \tag{6.1}$$

modeling the gradient of the indentation's height values, see Figure 6.4 for an illustration. Tilting is applied separately to indentation and accumulations by

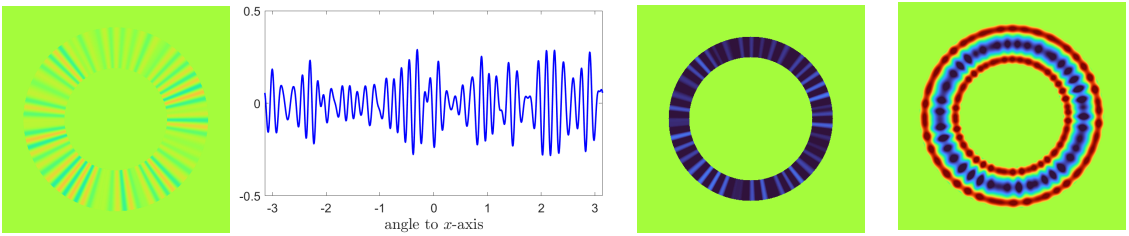
$$T_k(x) = \left(\text{plane}_k^- \cdot \mathbb{1}_{P_k^-} + \text{plane}_k^{+o} \cdot \mathbb{1}_{P_k^{+o}} + \text{plane}_k^{+i} \cdot \mathbb{1}_{P_k^{+i}}\right)(x).$$

The planes for the accumulation are defined similarly to Equation (6.1) with the corresponding highest $h_k^\bullet \sim \mathcal{N}(\mu_{h^\bullet}, \sigma_{h^\bullet})$ and lowest scaling factors $l_k^\bullet \sim \mathcal{N}(\mu_{l^\bullet}, \sigma_{l^\bullet})$ for $\bullet \in \{+i, +o\}$. To keep the model general, the highest and lowest scaling factors can be chosen independently. However, as real processing parameters serve the tilting angle φ between the x-y-plane and the z-axis and the indentation depth l in the center of the milling tool from which we obtain μ_{l^-} and μ_{h^-} by $l \pm r \cdot \sin(\varphi)$. None of the remaining tilting parameters can explicitly be derived from the machining parameters, but the planes are usually chosen to be parallel. The outer accumulation needs higher scaling factors than the inner since the tool's presence prevents material from being deposited on the inside.

6.1.3 Noise

As can be seen in Figure 4.3, the height values along a ring are not constant. We assume that the fluctuations follow a periodic pattern induced by oscillations of the blade as well as a transition between cutting and material deposition. Hence, the noise is modeled as a sum of $\lambda_k \sim \mathcal{P}(\lambda)$ sine curves, having frequencies $\tau_{k,j} \sim \mathcal{P}(\tau)$ and random shifts $\xi_{k,j} \sim \mathcal{U}((-\pi, \pi])$ for $j = 1, \ldots, \lambda_k$. Thus, the noise is given by

$$N_k(x) = \left(\frac{1}{\lambda_k} \sum_{j=1}^{\lambda_k} \tau_{k,j} \cdot \sin\left(\text{arctan2}\left(x_2 - (c_k)_2, x_1 - (c_k)_1\right) + \xi_{k,j}\right)\right) \cdot \mathbb{1}_{P_k}(x)$$



N_k. Height image and its circular 1d-intersection. $S_k^{\text{Indicator}} + N_k$. $S_k^{\text{Bump}} + N_k$.

Figure 6.5: Illustration of noise using $\lambda = \tau = 50$.

and illustrated in Figure 6.5. Here, P_k is defined as set of points to be considered, dependent on the used shape function. For those having an indentation only, we have $P_k = P_k^-$ and for those with additional accumulations we get $P_k = P_k^- \cup P_k^{+i} \cup P_k^{+o}$.

6.2 Sub-Model for Ring Interaction

The interaction between $k \in \mathbb{N}$ rings is defined as a mapping $f_k(R_1, \ldots, R_k)$. Assume $f_0(R_0)(x) = 0$ for all $x \in \mathbb{R}^2$ and consider the following functions which can be compared in Figure 6.6.

1. **Minimum**
   Since material is removed during the milling process, the deepest indentation at any point should be decisive. Note that the rings' order is not maintained when using the minimum. This interaction function then reads

   $$f_k^{\text{Min}}(R_1, \ldots, R_k)(x) = \min(R_1(x), \ldots, R_k(x)).$$

2. **Latest**
   Here, we take into account that milling is a consecutive process. Hence, newly introduced rings overwrite older ones which leads to a dominant appearance of their rear parts. We define this interaction by

   $$f_k^{\text{Latest}}(R_1, \ldots, R_k)(x) = \begin{cases} R_k(x) & \text{if } x \in P_k \\ f_{k-1}^{\text{Latest}}(R_1, \ldots, R_{k-1})(x) & \text{else.} \end{cases}$$

3. **Convex combination**
   In the previous approaches, the value in each point is determined by a single ring. A more realistic assumption might be that the height value is obtained by a superposition of rings overlapping at that point. To this end, we use a convex combination of the rings with most weight on the last rings, see Figure 6.7. Since the surface is changed more by deeper cuts those should be weighted more. Thus, we use a tilted plane $A_k : \mathbb{R}^2 \to \mathbb{R}$ instead of a global parameter. It is defined similarly to Equation (6.1) but with $l_k = a_k \sim \mathcal{U}([a^{\min}, a^{\max}])$ and $h_k = b_k \sim \mathcal{U}([b^{\min}, b^{\max}])$ for $a^{\min}, a^{\max}, b^{\min}, b^{\max} \in [0, 1]$ fulfilling $a^{\min} \leqslant a^{\max}$ and $b^{\min} \leqslant b^{\max}$. The resulting interaction function is

   $$f_k^{\text{Convex}}(R_1, \ldots, R_k)(x) = \left(A_k \cdot R_k + (1 - A_k) \cdot f_{k-1}^{\text{Convex}}(R_1, \ldots, R_{k-1})\right)(x).$$

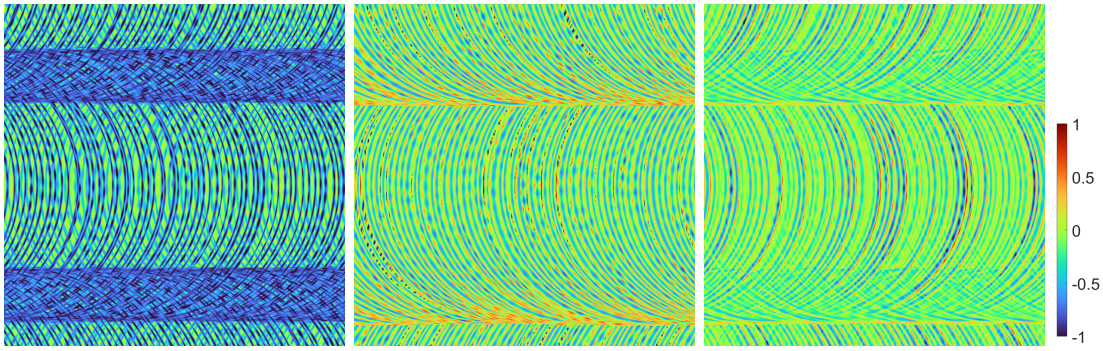


Figure 6.6: Illustration of interaction functions using bump shape. Imaged region is 5 mm × 5 mm.

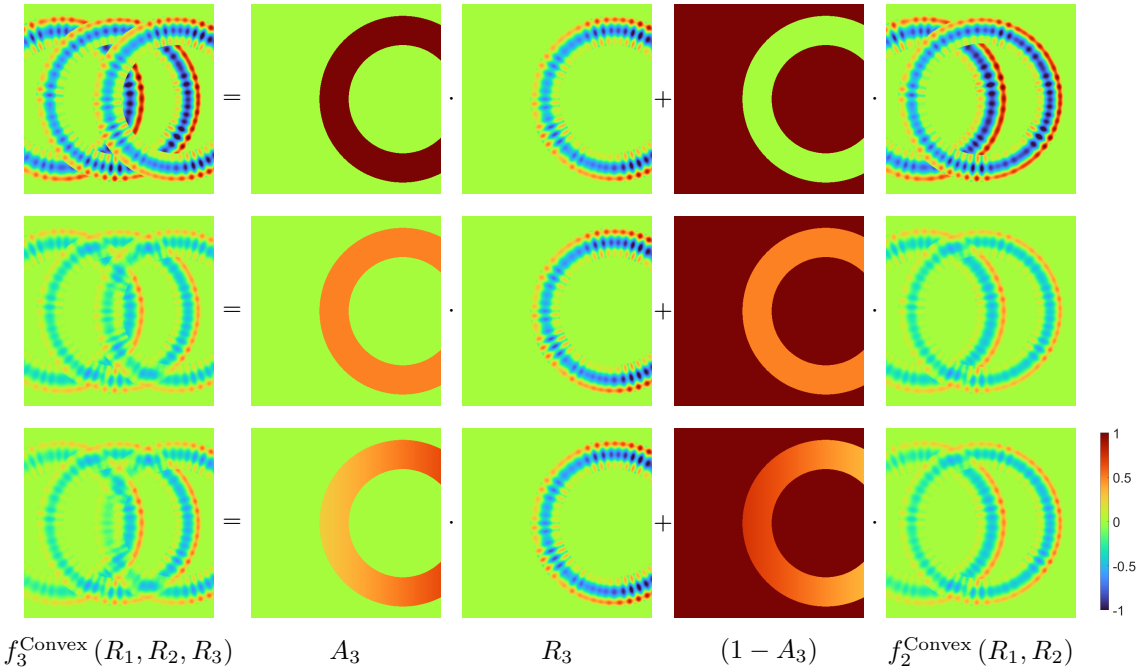
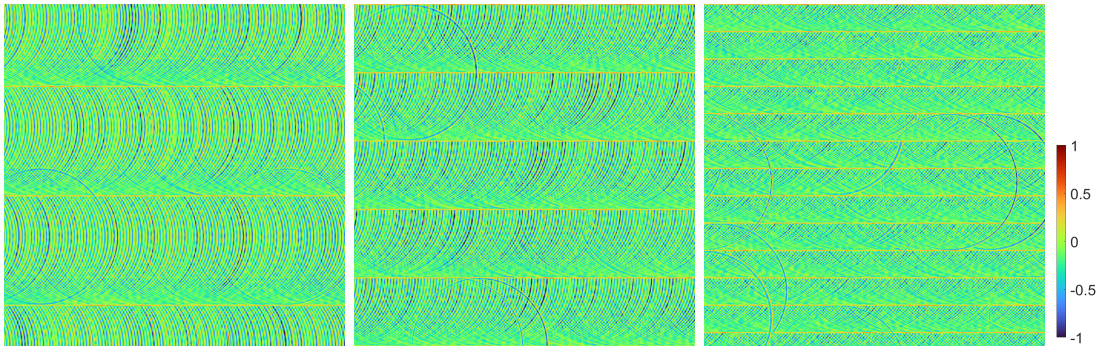


Figure 6.7: Illustration of the convex combination as interaction function using tilting directions $\theta_k = 0$ for $k = 1, 2, 3$. Different weights are used, from top to bottom: $A_k(P_k) = 1$ (equals f^{Latest}), a global weight $A_k(P_k) = const$ and a tilted plane.



$\alpha = 0.2$, $d = 4$ mm, $\delta = 0.09$ mm.

$\alpha = 0.5$, $d = 4$ mm, $\delta = 0.09$ mm.

$\alpha = 0.8$, $d = 4$ mm, $\delta = 0.09$ mm.

$\alpha = 0.2$, $d = 8$ mm, $\delta = 0.09$ mm.

$\alpha = 0.2$, $d = 4$ mm, $\delta = 0.15$ mm.

$\alpha = 0.2$, $d = 4$ mm, $\delta = 0.25$ mm.

Figure 6.8: Influence of parameters for the tool-path. Imaged region is 10 mm × 10 mm.

6.3 Sub-Model for Tool-Path

In this section, a parametric model is formulated for both the parallel and the spiral tool-path. Ring center points $c_{(k)}$ are modeled by random displacements of points $\bar{c}_{(k)}$ on the tool-path. That is, $c_{(k)} \sim \mathcal{N}\left(\bar{c}_{(k)}, \Sigma_c\right)$. To account for irregularities in the visibility of rings, the deterministic ring order is changed by a certain amount $\epsilon \in [0, 1]$. To do so, choose $\mathcal{E} \subset \{1, \ldots, n\}$ with $|\mathcal{E}| = \lceil \epsilon \cdot n \rceil$ randomly. The order of rings with index in $\mathcal{E}$ is then permuted randomly to obtain c_k.

We denote the distance between neighboring tool-paths by $\rho = (1-\alpha) \cdot d \in (0, d)$, where $\alpha \in (0, 1)$ specifies the amount of ring overlap. It depends on the radial depth of cut via $\alpha = 1 - a_e$. Remember that the radial cutting depth defines the distance the tool steps over in relation to the ring diameter. As the blades do not reach the edge of the tool, α is reduced here by taking $\alpha - \gamma$ with $\gamma \in [0, \alpha)$. Moreover, the distance between successive center points along the tool-path is $\delta \in \mathbb{R}_{>0}$, which depends on the tool's feed rate and the rotational speed of the milling head. All parameters are process-induced and directly modifiable in the machine setting. Figure 6.8 illustrates the influence of those parameters.

6.3.1 Parallel Pattern

For parallel milling, the center points $\bar{c}_{(i,j)}$ are located on a parallel system of lines with distance ρ. Here, index j is the index of the line while index i represents a set of equidistant points at distance δ along each line. Depending on the angle $\beta \in (-\pi, \pi]$ between the lines

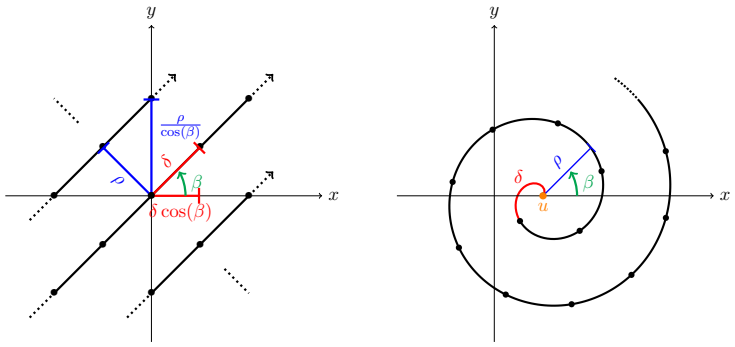
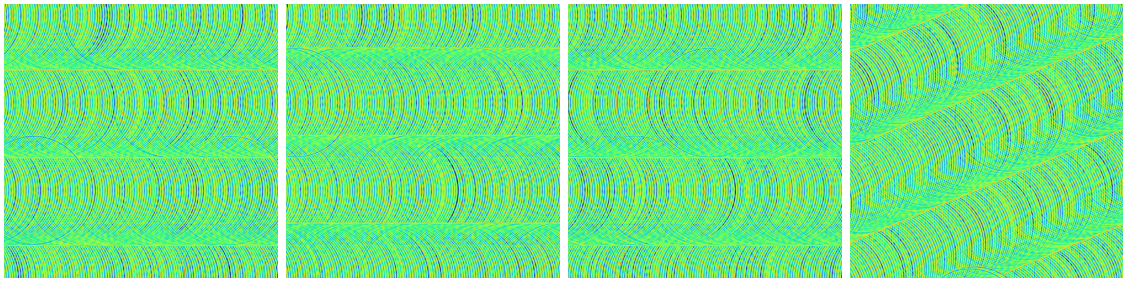


Figure 6.9: Illustration of parallel (left) and spiral tool-path (right), their parameters and preliminary center points lying thereon.



Milling in the same direction, bottom to top.

Milling in the same direction, top to bottom.

Milling in alternating directions.

Milling for $\beta = \pi/8$.

Figure 6.10: Several parallel milled patterns. Imaged region is 10 mm × 10 mm.

and the x-axis, we get the set of center points as $C^{\text{parallel}} = \left\{\bar{c}_{(i,j)}\right\}_{i,j\in\mathbb{Z}}$ for

$$\bar{c}_{(i,j)} = \begin{cases} \left(x_i, \tan(\beta)x_i + \frac{j\rho}{\cos(\beta)}\right),\ x_i = i\delta\cos(\beta) & \text{if } \beta \in (-\pi,\pi]\setminus\left\{-\frac{\pi}{2},\frac{\pi}{2}\right\} \\ (j\rho, i\delta) & \text{if } \beta \in \left\{-\frac{\pi}{2},\frac{\pi}{2}\right\} \end{cases}$$

as illustrated in Figure 6.9.

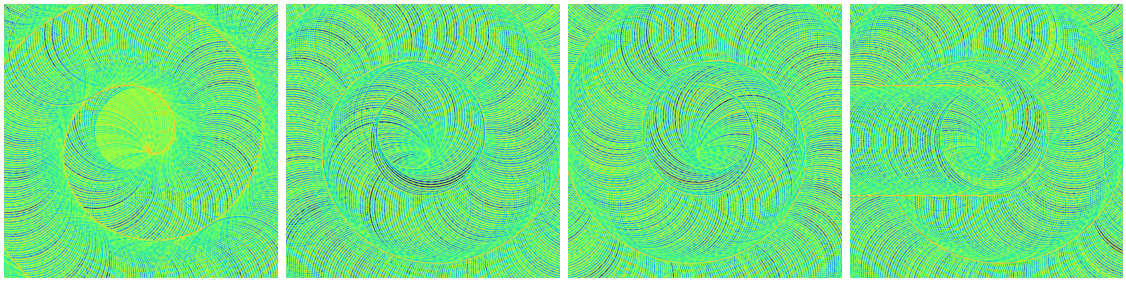
The indices k required for the interaction function are derived from the temporal order of milling the lines. We start by the first visible point on the line that is milled first. Then, we visit all indices i on this line following the milling direction. After all visible rings on the line have been captured, we proceed to the line that is milled next. Figure 6.10 shows patterns that are produced by different orderings. Finally, to obtain $\left\{\bar{c}_{(k)}\right\}_{k\in\mathbb{N}}$, restrict the computation of $\bar{c}_{(i,j)}$ such that the resulting rings are visible in the image.

6.3.2 Spiral Pattern

The spiral tool-path is modeled using the Archimedean spiral which is defined such that the radius grows proportionally to the angle. Its parametric representation in polar coordinates is

$$\text{spiral} : \varphi \mapsto \begin{pmatrix} o\cdot a\varphi\cdot\cos(\varphi+\beta) \\ a\varphi\cdot\sin(\varphi+\beta) \end{pmatrix} + u, \quad \varphi \geqslant 0.$$

Here, $u \in \mathbb{R}^2$ is the origin of the spiral, $\beta \in (-\pi,\pi]$ the angle to the x-axis at the origin and $o \in \{-1,1\}$ its orientation. If $o = 1$, then the spiral turns counter-clockwise. The



Outward milling. | Inward milling. | Counter-clockwise inward milling. | Inward milling with exit line.

Figure 6.11: Several spiral milled patterns. Imaged region is 10 mm × 10 mm.

distance of neighboring arcs is given by $\|\text{spiral}(\varphi) - \text{spiral}(\varphi + 2\pi)\|_2 = 2\pi a$. To achieve a distance of ρ, we have to choose $a = \frac{\rho}{2\pi}$. Figure 6.9 shows an exemplary spiral tool-path explaining all required parameters. For simplicity, we only consider $o = 1$, $\beta = 0$ and $u = (0, 0)$ in the following. According to our definition, the tool-path starts at the spiral's origin and moves to the outside (outward milling). Alternatively, the tool can also move in the opposite direction (inward milling). In this case, the tool can produce an exit line when it is not lifted from the surface before removing it. Exemplary textures of spiral milled surfaces are shown in Figure 6.11.

The center points are given by $C^{\text{spiral}} = \left\{\bar{c}_{(k)} = \text{spiral}\left(\varphi_{(k)}\right)\right\}_{k\in\mathbb{N}}$ for suitable $\varphi_{(k)}$. To ensure that consecutive center points have equal distance δ, the angles $\varphi_{(k)}$ have to be chosen such that they fulfill $L\left(\varphi_{(k)}\right) = k \cdot \delta$, where $L(\varphi) = \frac{a}{2}\left(\varphi\sqrt{1+\varphi^2} + \ln\left(\varphi + \sqrt{1+\varphi^2}\right)\right)$ is the spiral's arc length. Equidistant $\varphi_{(k)}$ are not possible since $L(\cdot)$ is not linear, see Figure 6.12. Moreover, no analytic solution of $L\left(\varphi_{(k)}\right) - k \cdot \delta = 0$ exists. Therefore, we use Newton's method

$$\varphi_{(k,0)} = \varphi_{(k-1)}, \qquad \varphi_{(k,j+1)} = \varphi_{(k,j)} - \frac{L\left(\varphi_{(k,j)}\right) - k \cdot \delta}{L'\left(\varphi_{(k,j)}\right)},\ j \in \mathbb{N},$$

with $\varphi_{(0)} = 0$. The derivative of $L(\cdot)$ is given by

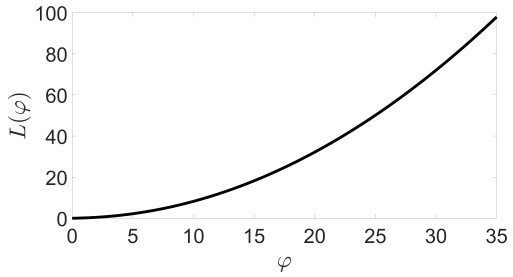
$$L'(\varphi) = \frac{a}{2}\left(\frac{2\varphi^2+1}{\sqrt{1+\varphi^2}} + \frac{1}{\sqrt{1+\varphi^2}}\right) = a\sqrt{1+\varphi^2}.$$

Hence, approximate $\varphi_{(k)} \approx \varphi_{(k,I)}$ for $I \in \mathbb{N}$. We decided for $I = 1$ because negligible mistakes are just visible for small k, see Figure 6.12. Hence, use

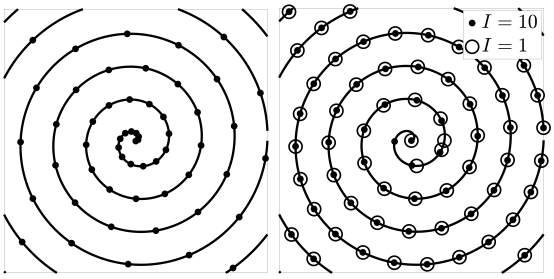
$$\varphi_{(0)} = 0, \qquad \varphi_{(k+1)} = \varphi_{(k)} - \frac{L\left(\varphi_{(k)}\right) - k \cdot \delta}{L'\left(\varphi_{(k)}\right)}.$$

6.4 Fitting the Model to Data

The modular multi-step model for generating texture images of milled surfaces has components for modeling the shape of an individual ring and the interaction between rings which are compared in Table 6.2. It shows the necessity of randomizing the ring structures and of all components for the sub-model of the rings' appearance. We chose the bump shape and the convex combination as interaction function since these two variants reflect

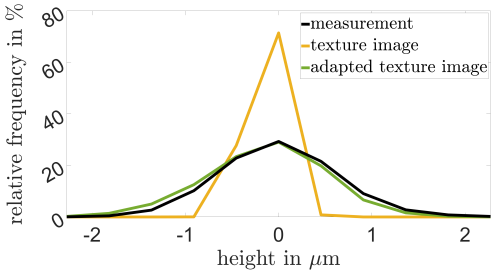


(a) Arc length $L(\varphi)$.

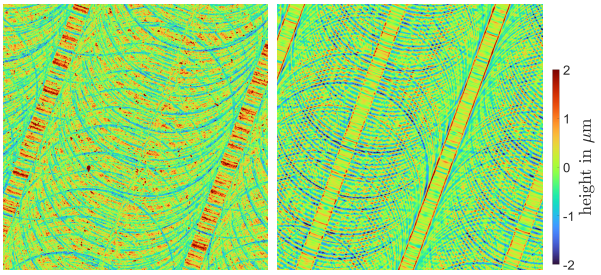


(b) Comparison of equidistant $\varphi_{(k)}$ (left) and those computed using Newton's method (right) with different I.

Figure 6.12: Explanation to use Newton's method for the computation of $\varphi_{(k)}$ considering $a = \frac{1}{2\pi}$.



(a) Height value distribution of the measurement compared to the simulation with and without height value adaptation.



(b) Height images of the measurement (left) and the adapted simulation (right).

Figure 6.13: Comparison of measurement obtained by focus-variation microscope and simulation of parallel milling with milling head diameter 4 mm and ring overlap 0.46 ($\alpha = 0.5$, $\gamma = 0.04$). Imaged region is 5.5 mm × 5.5 mm.

the corresponding processes most realistically. Although this is the most complex choice, most of the parameters are related to the milling process. Some parameters are real-world parameters and can thus be taken immediately from those of the machining process. The remaining parameters are estimated by visual comparison of the resulting patterns and the measurements. All parameters are summarized in Table 6.3.

Due to the choice of the convex combination as interaction function, the height values of a simulated texture image $\mathcal{T}$ may not resemble those of the corresponding measurement $\mathcal{M}$, see Figure 6.13. Hence, we adapt the height values of the simulated image such that their sample mean $\hat{\mu}_{\mathcal{T}}$ and sample variance $\hat{\sigma}_{\mathcal{T}}$ coincide with the sample mean $\hat{\mu}_{\mathcal{M}}$ and the sample variance $\hat{\sigma}_{\mathcal{M}}$ of the measurement.

The computation time of the texture synthesis method grows non-linearly with increasing image size, see Figure 6.14. It depends on both the pixel spacing and the number of rings. The number of rings increases with the size of the overlap. Thus, run-time for large high-resolution images can get critical. Parallelization is possible by precomputing and storing the required parameters for each ring. Then, smaller sub-images can be computed simultaneously and merged afterwards.

Table 6.2: Comparison of interaction functions using indicator and bump shape. The model's improvements are added successively: variability in the ring width ($\sigma_{w\bullet}, (\Sigma_c)_{ii} \neq 0$), integration of tilting and noise and change of the rings' ordering. Imaged region is 5 mm × 5 mm.

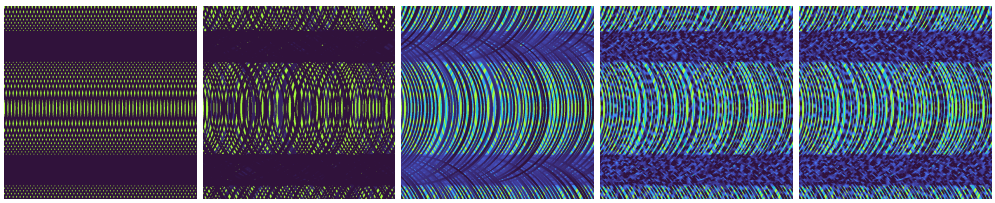
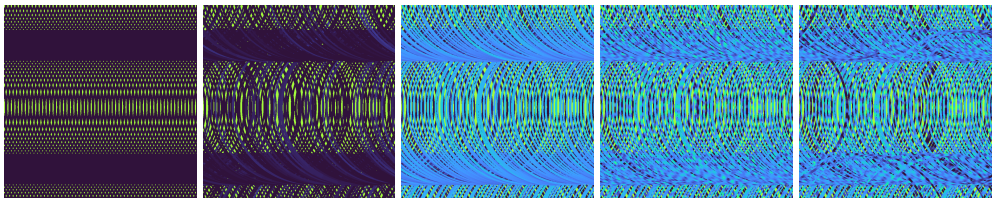
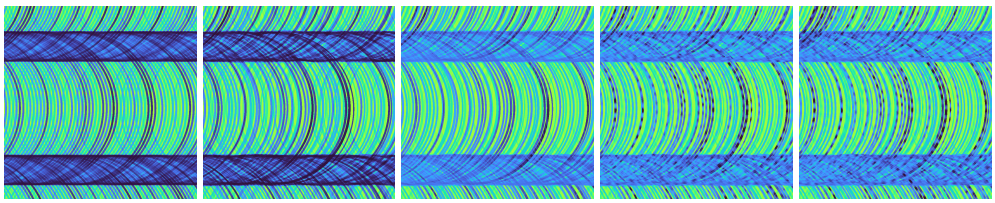
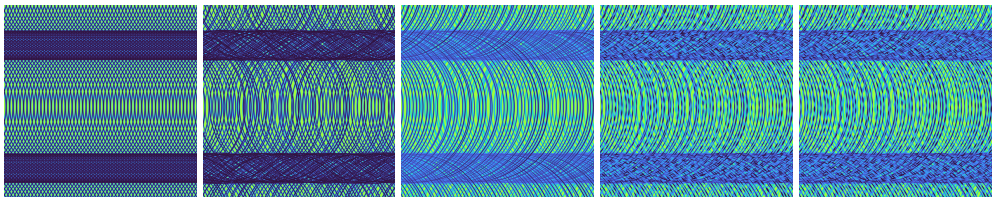
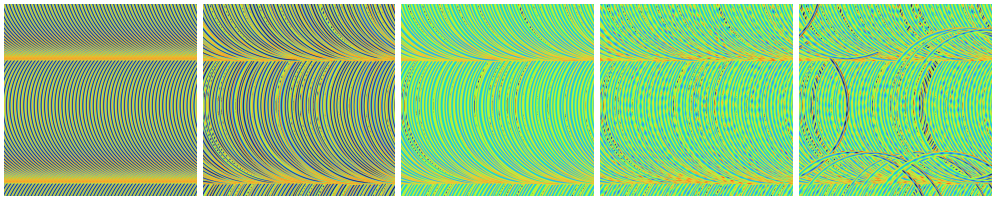
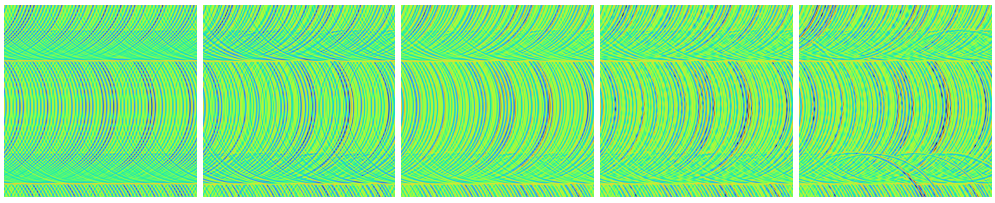
$\sigma \neq 0$	✗	✓	✓	✓	✓
Tilting	✗	✗	✓	✓	✓
Noise	✗	✗	✗	✓	✓
$\epsilon \neq 0$	✗	✗	✗	✗	✓
Indicator shape, Minimum					
Indicator shape, Latest					
Indicator shape, Convex combination					
Bump shape, Minimum					
Bump shape, Latest					
Bump shape, Convex combination					

Table 6.3: Overview of parameters needed for the model to simulate milled surfaces. Directly known process parameters are highlighted.

		notation	definition	relation to process	distribution	parameters
appearance	shape	d	diameter of ring	**diameter of milling head**		$d \in \mathbb{R}_{>0}$
		w_k^-	width of indentation	**width of cutting edge**	$\mathcal{N}(\mu_{w^-}, \sigma_{w^-})$	$\mu_{w^-} \in (0, d/2)$ $\sigma_{w^-} \in \mathbb{R}_{>0}$
		w_k^{+i}	width of inner accumulation	depends on edges' sharpness	$\mathcal{N}(\mu_{w^{+i}}, \sigma_{w^{+i}})$	$\mu_{w^{+i}} \in [0, d/2 - \mu_{w^-})$ $\sigma_{w^{+i}} \in \mathbb{R}_{>0}$
		w_k^{+o}	width of outer accumulation	depends on edges' sharpness	$\mathcal{N}(\mu_{w^{+o}}, \sigma_{w^{+o}})$	$\mu_{w^{+o}} \in \mathbb{R}_{\geqslant 0}$ $\sigma_{w^{+o}} \in \mathbb{R}_{>0}$
	tilting	θ_k	tilting direction	depends on tool-path		$\theta_k \in (-\pi, \pi]$
		l_k^-, h_k^-	minimal/maximal scaling of indentation depth	**cutting depth with tilting**	$\mathcal{N}(\mu_{\bullet^-}, \sigma_{\bullet^-})$	$\mu_{\bullet^-} \in \mathbb{R}_{\geqslant 0}$ $\sigma_{\bullet^-} \in \mathbb{R}_{>0}$
		l_k^{+i}, h_k^{+i}	minimal/maximal scaling of inner accumulation height	depends on edges' sharpness	$\mathcal{N}(\mu_{\bullet^{+i}}, \sigma_{\bullet^{+i}})$	$\mu_{\bullet^{+i}} \in \mathbb{R}_{\geqslant 0}$ $\sigma_{\bullet^{+i}} \in \mathbb{R}_{>0}$
		l_k^{+o}, h_k^{+o}	minimal/maximal scaling of outer accumulation height	depends on edges' sharpness	$\mathcal{N}(\mu_{\bullet^{+o}}, \sigma_{\bullet^{+o}})$	$\mu_{\bullet^{+o}} \in \mathbb{R}_{\geqslant 0}$ $\sigma_{\bullet^{+o}} \in \mathbb{R}_{>0}$
	noise	λ_k	number of sine curves		$\mathcal{P}(\lambda)$	$\lambda \in \mathbb{N}$
		τ_{k_j}	frequency of sine curves		$\mathcal{P}(\tau)$	$\tau \in \mathbb{N}$
		ξ_{k_j}	shift of sine curves		$\mathcal{U}((-\pi, \pi])$	
interaction		a_k	front value for convex combination		$\mathcal{U}([a^{\min}, a^{\max}])$	$a^{\min} \in [0, 1]$ $a^{\max} \in [a^{\min}, 1]$
		b_k	rear value for convex combination		$\mathcal{U}([b^{\min}, b^{\max}])$	$b^{\min} \in [0, 1]$ $b^{\max} \in [b^{\min}, 1]$
tool-path		c_k	ring center points	determined by tool-path	$\mathcal{N}(\bar{c}_k, \Sigma_c)$	$\Sigma_c \in \mathbb{R}^{2\times 2}_{>0}$
		α	defines ρ (distance between neighboring tool-paths)	**amount of overlap of neighboring tool-paths**		$\alpha \in (0, 1)$
		γ	increase distance between neighboring tool paths	distance between blade and outer edge of tool		$\gamma \in [0, \alpha)$
		δ	distance between center points	**depends on feed rate and tool rotational speed**		$\delta \in \mathbb{R}_{>0}$
		ϵ	amount of rings with changed order			$\epsilon \in [0, 1]$
		β	angle to x-axis			$\beta \in (-\pi, \pi]$
		u	spiral origin			$u \in \mathbb{R}^2$
		o	spiral orientation	**(counter-)clockwise**		$o \in \{-1, 1\}$

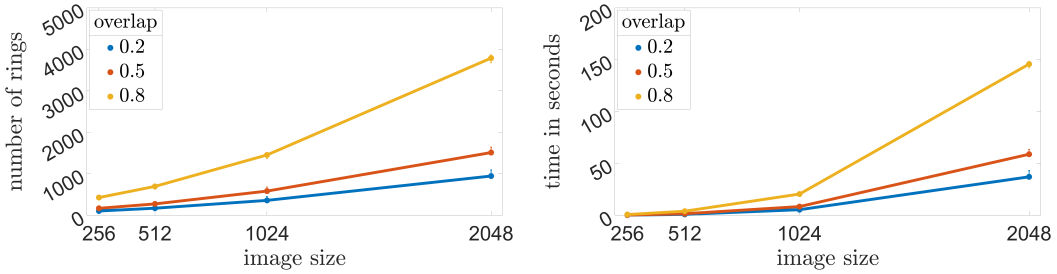
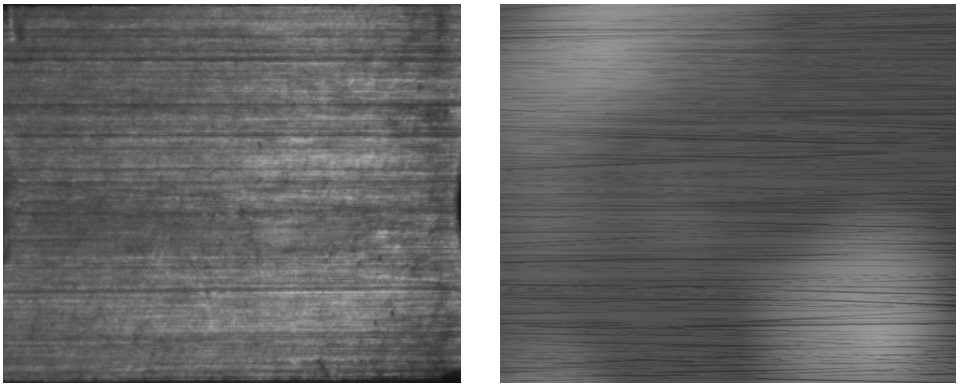


Figure 6.14: Mean computation time (left) and mean number of relevant rings (right) for milling pattern using pixel spacing $\nu = 6.1$ µm, milling tool diameter $d = 4$ mm and several choices for the overlap. Means of 100 realizations. Error bars indicate minimal and maximal values. Run-times of the MATLAB-implementations were measured on a HP Z240 Tower Workstation equipped with Intel Core i7-3700 CPU at 3.40GHz × 8.



(a) Real acquired image taken as crop from the dual data set of a clutch published by Fulir et al. [33].

(b) Synthetic image rendered using Blender [1]. Procedural color texture is similar as proposed in [71] for simulating surface impurities.

Figure 6.15: Real and synthetic image of a polished surface.

6.5 Model-Adaptations to Simulate Polished Surfaces

Polishing is a common method for post-processing the top layer of a machined product to obtain smooth surfaces. Different polishing methods exist, for example chemical, ultrasonic or mechanical polishing. We restrict to the latter. Therefore, abrasives like sandpaper are used to remove material from the top surface layer. The process can be performed automatically or by hand. Figure 6.15a shows an example of a polished metal surface. The texture is characterized by elongated line-shaped indentations resulting from the material removal of individual abrasives during the polishing movement.

To model the surface texture of polished objects, the milling model can be adapted. Instead of rings, we use elongated rectangles as basic shape representing the line-shaped indentations. The exact adaptations are explained in Section 6.5.1. Then, also the placement of the rectangles differs form the arrangement of milling rings as explained in Section 6.5.2. Just the interaction of several rectangles stays unchanged compared to the model for milled surfaces.

An example of the resulting synthetic texture is shown in the rendering provided in Figure 6.15b. Note that an appropriate texture to define the color variations and topography changes on smaller scale must be chosen to obtain more realistic simulations. This was not further investigated in this work.

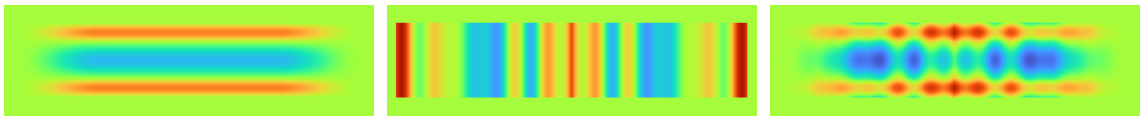
6.5.1 Appearance of Polishing Rectangles

Similar to the milling rings, the appearance of an individual polishing rectangle is defined by $R_k = S_k T_k + N_k$ with S_k the shape, T_k the scaling and N_k the noise. The shape of the polishing rectangle is defined by the length l_k and width $w_k = w_k^- + 2w_k^+$. The latter consists of the indentation width w_k^- and the width w_k^+ for the accumulation occurring at both sides. We choose $l_k \gg w_k$ to obtain elongated, thin rectangles. Then, the shape functions are defined equally as for the milling rings, see Figure 6.16a. The scaling function is used to adapt the height values and to obtain smooth transitions at the short sides of the rectangle. Finally, noise is added similarly as for the milling rings, see Figure 6.16b.



S_k T_k $S_k T_k$

(a) Scaling of the cosine shape applied on a rectangle.



$S_k T_k$ N_k $R_k = S_k T_k + N_k$

(b) Application of noise to obtain final element.

Figure 6.16: Illustration of the components that model the appearance of a polishing rectangle.

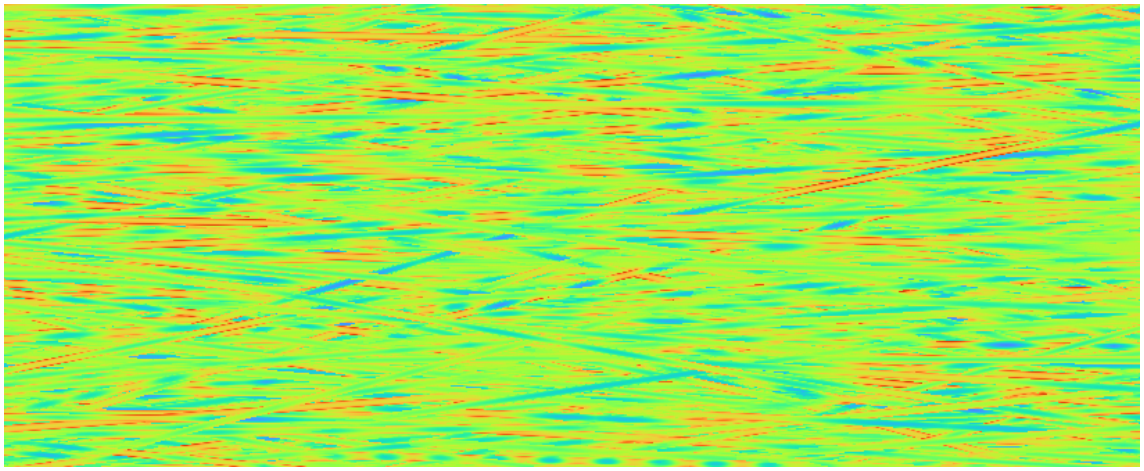


Figure 6.17: 2d height image of a simulated polished surface texture.

6.5.2 Placement of Polishing Rectangles

For milling, the ring center points were determined by following the pre-defined milling path. As polishing results in a less structured texture pattern, the rectangles are randomly placed. Therefore, the center of mass is used as center point whose location is sampled from a uniform distribution defined on the window that covers the object surface. Note that center points outside the window must be chosen as well to create uniform patterns also at the window boundaries. Moreover, the rectangles can be rotated by a random but restricted angle to increase realism especially of manually polished surfaces. See Figure 6.17 for an example of the resulting height image.

7 Conclusion

We developed texture synthesis models for sandblasted and milled surfaces. A common model was not feasible, as the surface patterns of the two methods have no common properties. The models are based on topography measurements of real surfaces.

To create the sandblasted test objects, only the pressure could be reliably controlled and thus varied to obtain surfaces treated by two different parameter choices. Hence, it is difficult to develop a procedural model that only refers to that parameter. Therefore, we use a combination of existing data-based texture models that rely entirely on the measurement. Sandblasted texture images are created in two steps. First, image patches are generated and stitched afterwards. The second step can be neglected if the output texture is to be smaller than the measurement.

We examined and compared several methods for the patch generation step, namely the ADSN, the RPN, the method of Heeger and Bergen and the method of Portilla and Simoncelli. For our purpose, RPN turned out to be the best method, as it has a short computation time and similar output texture images compared to the input measurement. In particular, the height value distribution and the auto-correlation resemble those of the measurement.

To achieve a coarser pixel spacing than the measurement, the input is down-sampled beforehand using nearest neighbor interpolation. The case that the output texture should have a higher resolution than the measurement was not investigated. Images larger than the input can be calculated directly by padding the image before applying the RPN. However, the resulting texture image is highly dependent on the given input patch. This can be avoided by computing smaller patches with inputs taken from random positions and stitching them using the method of Efros and Freeman.

Texture images of arbitrary size and with a pixel spacing coarser or equal to that of the measurement can be generated. For future research, a parametric texture synthesis model would be beneficial that is independent of input images. Ideally, the model parameters should rely on the real machining parameters. Moreover, we aim to develop a procedural method which is not feasible with the current model.

The texture synthesis model for milled surfaces is parameterized including the knowledge of process parameters. A wide variety of texture images can be generated simply by using different parameter settings. Assuming the individual parameter values to be sampled from suitable random distributions ensures texture variability also when using the same parameter ranges.

The model for milled surfaces is procedural since it is defined as function on a continuous domain. Texture images can be obtained by evaluating the function at given grid points. For efficient use of the model in rendering, fast approaches for computing the height value at a given surface point have to be developed. This could avoid the intermediate image-generating step. The crucial point is a fast searching algorithm to find all rings affecting one surface point.

8 Dual Data Set of Real and Synthetic Images

The models proposed in the previous sections offer the possibility to simulate the surface meso scale geometry of sandblasted and milled objects. Thus, synthetic visual data rendered in a virtual inspection environment can be generated using the resulting normal maps as texture. In the SynosIs project we published an annotated dual data set consisting of real and synthetic images of non-defective and defective sandblasted and milled objects [34, 35]. The defects considered therein are dents and scratches. Figure 8.2, Figure 8.3 and Figure 8.4 compare real and synthetic data of non-defective objects for the modeled surface processing methods.

For creating the real test objects, three main planar surfaces are processed by sandblasting or milling using the parameter configurations described in Section 4.3. In total, ten different object configurations were designed, where each of them is produced three times, once without defects and twice having defects. We consider dent and scratch defects resulting from different tools applied with varying pressures. This results in ten defect-free and 20 defective objects. All three main surfaces were imaged from three different viewpoints, see Figure 8.1, planned by using the software V-POI [43]. The real acquisition setup consists of a matrix grayscale camera which is placed in the middle of a ring light. Both camera and light have the same orientation and are placed at a robot arm such that they are moved together when changing the viewpoint. The acquisition environment is darkened using a black curtain to remove the surrounding illumination. In total, the real data set contains 270 images, 90 of defect-free and 180 of defective surfaces. Annotation was performed manually using the software LabelMe [110].

For the synthetic data set, each of the ten different object configurations is simulated 60 times, 30 defective and non-defective each. We follow the synthetic data generation pipeline described in Section 2.3. Synthetic dent and scratch defects are applied using

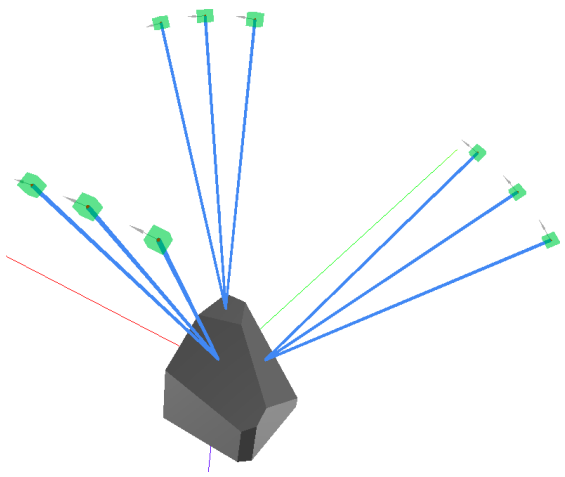
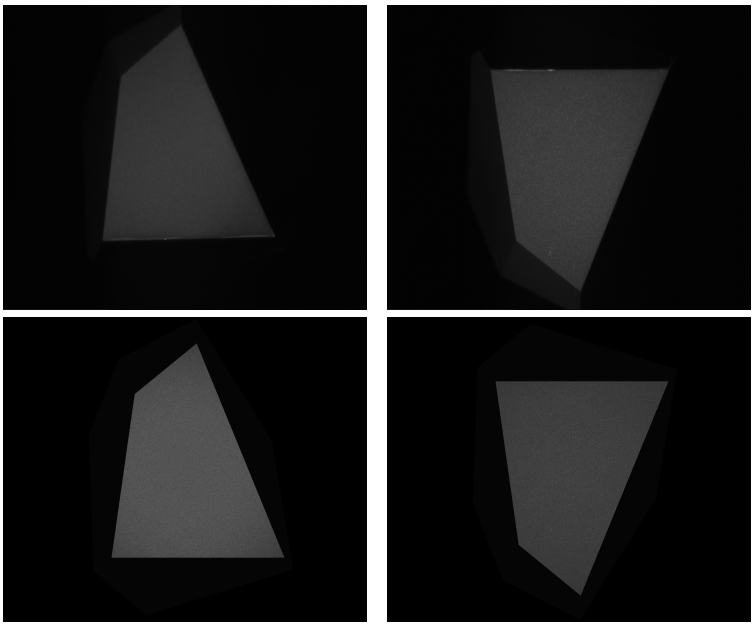


Figure 8.1: Visualization of viewpoints used for inspection similar to Figure 19 in [35].

Table 8.1: Texture parameter ranges used to sample for random texture generation. Default values are highlighted. The values for the standard deviations for sampling from normal distributions were chosen such that, according to the 3σ rule, the sampled values fall into prescribed ranges. Therefore, set conf = 3.

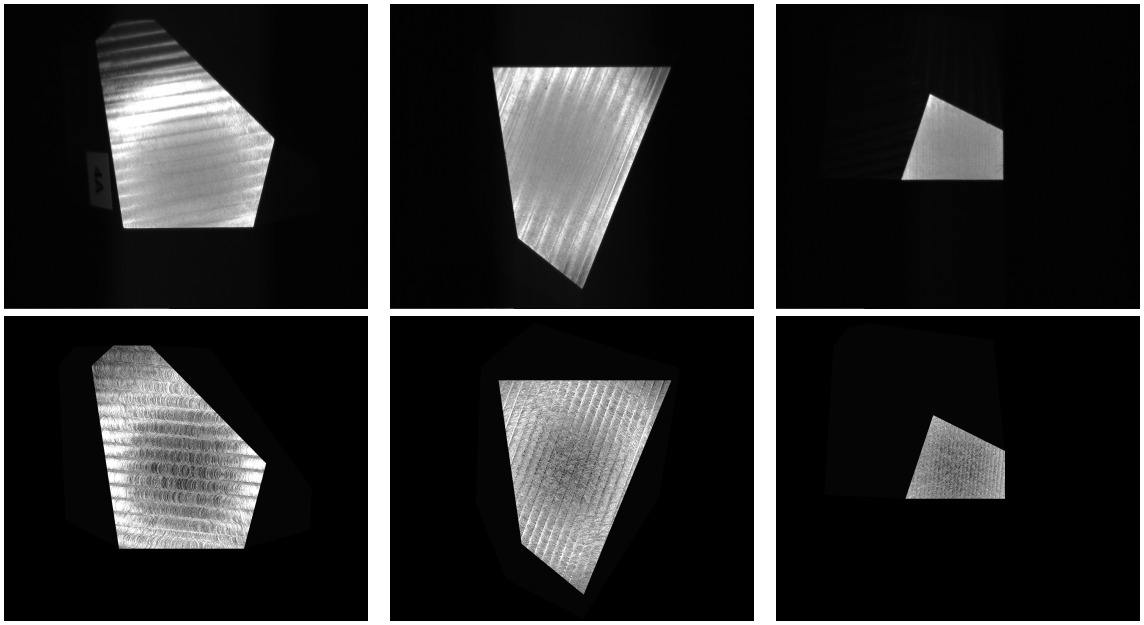
Parameter	Set of values
Ring center points noise (Σ_c)	$\{\mathbf{0.01}, 0.05, 0.1, 0.2, 0.4, 0.6, 0.8\} \cdot \delta/\text{conf}$
Ring distance (δ)	$\{0.7, 0.75, 0.8, 0.85, \mathbf{0.9}, 0.95, 1.0, 1.05, 1.1\}$
Ring flip probability (ϵ)	$\{\mathbf{0.01}, 0.05, 0.1, 0.2, 0.3, 0.4, 0.5\}$
Ring width noise (σ_{ω^-})	$\{0.015, 0.02, \mathbf{0.025}, 0.03, 0.035\} \cdot 1/\text{conf}$
Ring depth noise ($\sigma_{l^-}, \sigma_{h^-}$)	$\{0.4, 0.6, \mathbf{0.8}, 1.0, 1.2\} \cdot 1/\text{conf}$
Number of sine curves (λ)	$\{30, \mathbf{50}, 70\}$

the methods introduced by Bosnar et al. [12]. Textures for sandblasted and milled surfaces are generated by the methods presented in Chapter 5 and Chapter 6, respectively. The parameters to generate the texture images for milled surfaces are sampled from the corresponding random distributions as summarized in Table 6.3 using the parameter values given in Table 8.1. Texture images are rotated by an angle randomly sampled from $[-15, 15]^\circ$ and translated randomly by 1, 3 or 5 pixels before mapping them to the object surface. The virtual inspection environment is structured similarly to the real acquisition setup. We use a virtual pinhole camera and a torus-like light geometry with diffuse emission. The same viewpoints as for the acquisition are taken resulting in 5400 synthetic images in the data set, 1800 images of defect-free and 3600 of defective surfaces. Rendering is performed using the APPLESEED rendering engine [6]. For shading we consider 256 samples per pixel and eight light bounces. Noise is added to the synthetic images to include light scattering caused by air particles and to take small-sized impurities on the real camera into account. The synthetic images are post-processed using histogram adaptation according to the histograms of the associated real images, see [35] for more details. Real and synthetic images are both of size 1224×1025.



2.5 bar. 6 bar.

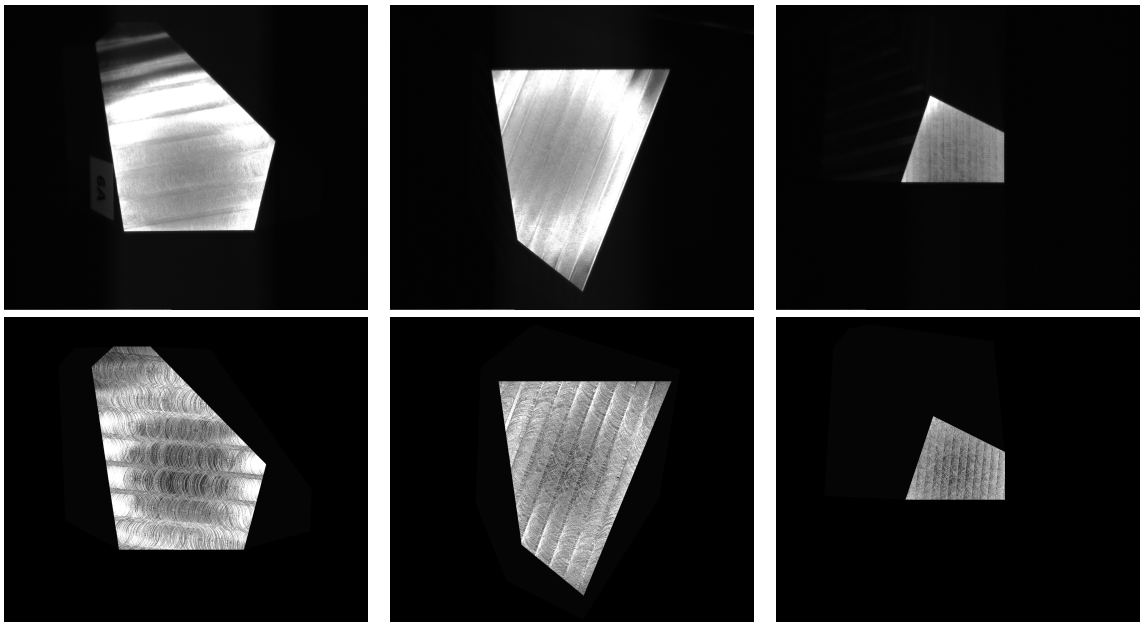
Figure 8.2: Real acquisition image (top) and synthetic image rendered in a virtual inspection environment with subsequent histogram adaptation (bottom) of a non-defective sandblasted object using different pressures. Perpendicular view is shown. Real images were acquired by Juraj Fulir and synthetic images were rendered by Lovro Bosnar, both from Fraunhofer ITWM, using texture images generated as described in Chapter 5.



Head diameter 4 mm, radial depth of cut 0.8.

Head diameter 4 mm, radial depth of cut 0.5.

Head diameter 4 mm, radial depth of cut 0.2.

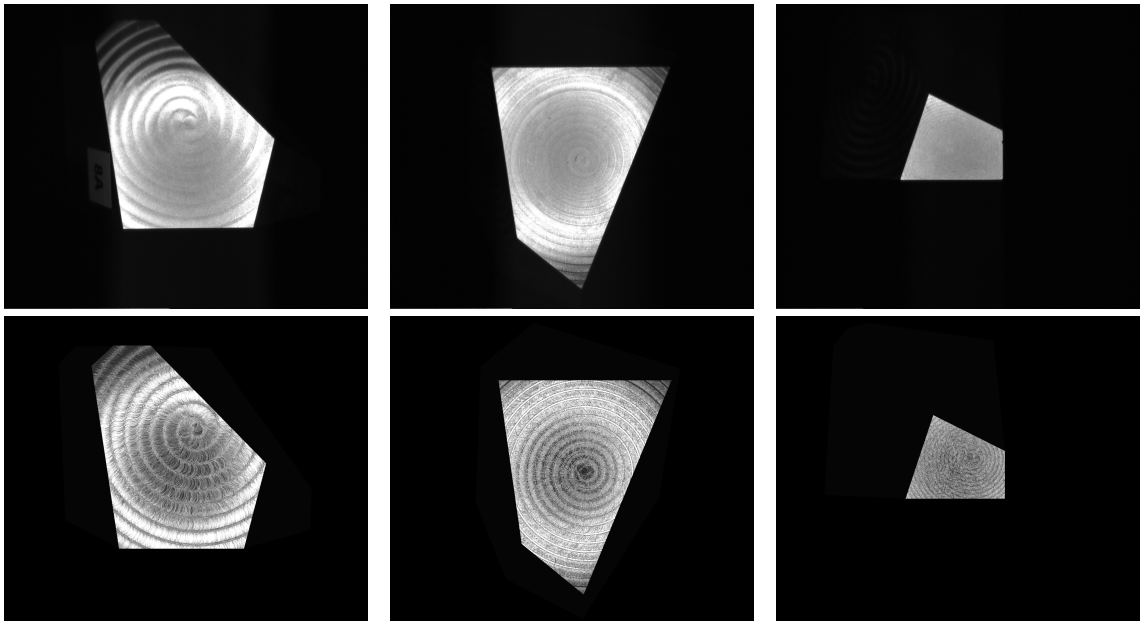


Head diameter 8 mm, radial depth of cut 0.8.

Head diameter 8 mm, radial depth of cut 0.5.

Head diameter 8 mm, radial depth of cut 0.2.

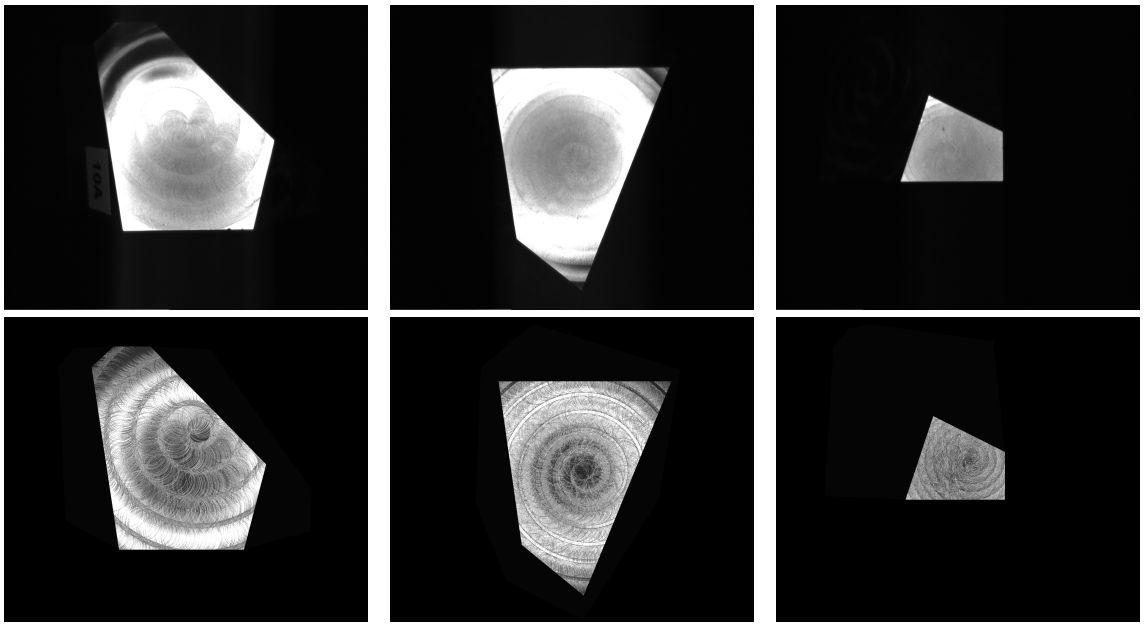
Figure 8.3: Real acquisition image (top) and synthetic image rendered in a virtual inspection environment with subsequent histogram adaptation (bottom) of a non-defective parallel milled object using different parameter configurations. Perpendicular view is shown. Real images were acquired by Juraj Fulir and synthetic images were rendered by Lovro Bosnar, both from Fraunhofer ITWM, using texture images generated as described in Chapter 6.



Head diameter 4 mm,
radial depth of cut 0.8.

Head diameter 4 mm,
radial depth of cut 0.5.

Head diameter 4 mm,
radial depth of cut 0.2.



Head diameter 8 mm,
radial depth of cut 0.8.

Head diameter 8 mm,
radial depth of cut 0.5.

Head diameter 8 mm,
radial depth of cut 0.2.

Figure 8.4: Real acquisition image (top) and synthetic image rendered in a virtual inspection environment with subsequent histogram adaptation (bottom) of a non-defective spiral milled object using different parameter configurations. Perpendicular view is shown. Real images were acquired by Juraj Fulir and synthetic images were rendered by Lovro Bosnar, both from Fraunhofer ITWM, using texture images generated as described in Chapter 6.

Part II

Defect Modeling

9 Introduction

Part II of this thesis deals with modeling the geometry of various types of surface defects to enable synthetic data generation of defective parts in a virtual test environment as discussed in Section 2.2.2. This approach requires models to generate and place defect geometries. In this thesis, placement methods are not considered. For a reference on this topic see [12]. We propose parametric models for simulating geometries of different surface defect types. Choosing appropriate ranges from which individual parameters are randomly sampled values ensures the variability of the resulting defect geometries. As a testing method, we will use visual surface inspection, but the procedure for synthetic data generation can easily be applied in combination with any other Monte Carlo simulation, e.g. for simulating X-ray imaging.

Defect modeling approaches based on computer graphic techniques do usually not take the physical material properties and the physically correct forming process into account, but rather model realistically appearing defects in an artistic way. Examples are provided by Bosch [10] for modeling surface grooves and scratches and by Bosnar and Gospodnetić [12] for surface dents and scratches. Merillou et al. [73] proposed a model for scratch simulation integrating real-world measurements. Moreover, Fan et al. [27] provide an approach for modeling brittle material fracturing of the object. Therefore, deformations of the discretization of the 3d object are computed, which is commonly given by tetrahedrons. Instead of adding individual defects, the whole object is usually destroyed and broken down into parts. Hu et al. [116] published a survey of modeling techniques for cuts in deformable bodies which are commonly applied for simulating surgery scenarios. They distinguish between mesh-based, finite element and mesh-free simulation methods.

Further simulations of particular defect shapes based on the underlying material structure and characteristics have been provided in the literature. For instance, Jung and Redenbach [59] proposed a model for 3d crack simulation in concrete based on 3d Voronoi tessellations. Also, Ghazvinian et al. [39] use Voronoi tessellations to simulate brittle rock damaging. Dogahe et al. [23] published models to simulate fatigue cracking in stents which have a granular material structure using the finite element method.

Here, we restrict on metal objects and their typical defects. In real-life scenarios, surface defects are often created by external defecting tools resulting for example in dents or scratches. Also, surface defects can occur due to internal factors such as crack formation caused by internal tensile stress. Moreover, problems in the production of a part can result in surface defects too, for example incomplete casting. Our defect models take into account the formation process of the associated real defect type. In Chapter 10 we consider dent defects created after surface finishing is applied. Chapter 11 deals with various casting defects, for example cracks, rat tails and delaminations.

10 Dent Defects

10.1 Introduction

Dents are common defect types in almost every production of rigid products as their creation does not necessarily depend on a certain manufacturing process but can happen in many ways. Often, external tools that hit an object are responsible for dent formation. This causes an indentation in the object surfaces whose shape resembles the negative of the associated tool. Excessive material is then either compressed or relocated, mostly to the surrounding area.

In the following, we propose a parametric, multi-step model for simulating dent defects. Different shapes to describe simple tool geometries are included, based on real-world observations. Further, we assume material relocation modeled by accumulations around the indentation. The defect geometry is first determined as a function in a continuous domain from which we sample afterwards to generate the desired surface mesh. Therefore, different approximation methods are presented.

The proposed model provides thus many interchangeable components to generate various defect geometries through their diverse combinations. Also, parametrization and random sampling from defined parameter ranges offers the possibility to generate a variable set of dent defects.

10.2 Data

For modeling dent defects, real measurements from defective objects produced for the SynosIs project exist. See Section 4.3 and Chapter 8 for detailed descriptions of the test objects. Different dent shapes are investigated, created by three different tools applied with varying force. The measurements were performed using the focus-variation microscope Bruker alicona InfiniteFocus G4 with a vertical resolution of 420 nm. Two sets of measurements taken from different test objects are available. Once, dent defects were produced on a sandblasted metal plate. For the corresponding measurements we have $\nu = 1.75$ µm, see Figure 10.1 for examples of the different tool shapes. The second data set consists of defects in the actual test bodies, see Figure 10.2. Those images have a resolution of $\nu = 0.44$ µm.

10.3 Modeling

In the following, we propose a parametric multi-step model to simulate dent defects. The outputs are centered in $(0, 0, 0)$ but can arbitrarily be translated and rotated to enable proper imprinting into the object geometry. Usually, dent defects have a positive and a negative component which will be added and subtracted, respectively, from the object geometry. We denote the negative part as indentation, as it simulates the result of the defecting tool hitting the object surface. Then, material is pushed to the outside which we call lateral material accumulation. The model subdivides into five steps, see Figure 10.3, which will be intensively discussed in the subsequent sections, namely

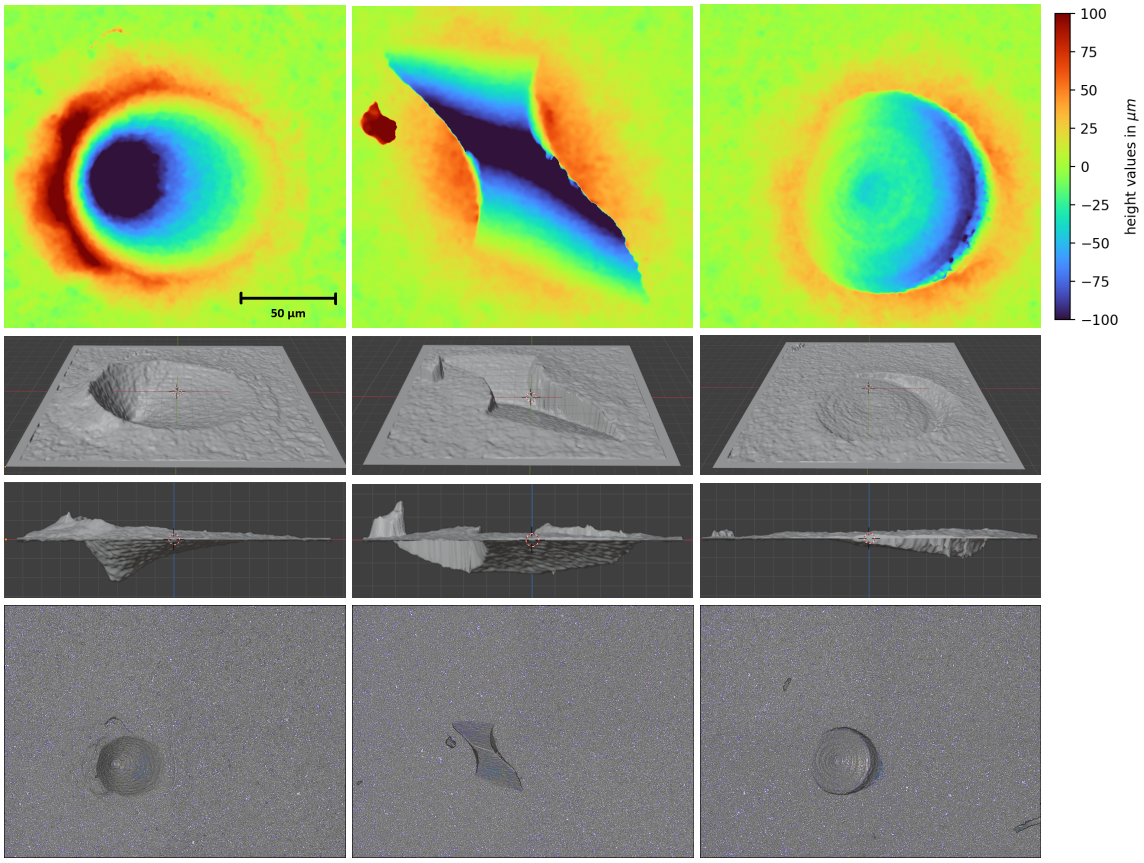


Figure 10.1: Dent defects on a metal plate caused by different tools but with the same force. From top to bottom: 2d height images provided by Fraunhofer IOF using a focus-variation microscope, 3d renderings of the surfaces using Blender [1] (top and side view), optical 3d images of a larger surface area provided by Fraunhofer IOF. The resolution of the images is the same but the size of the imaged areas differs.

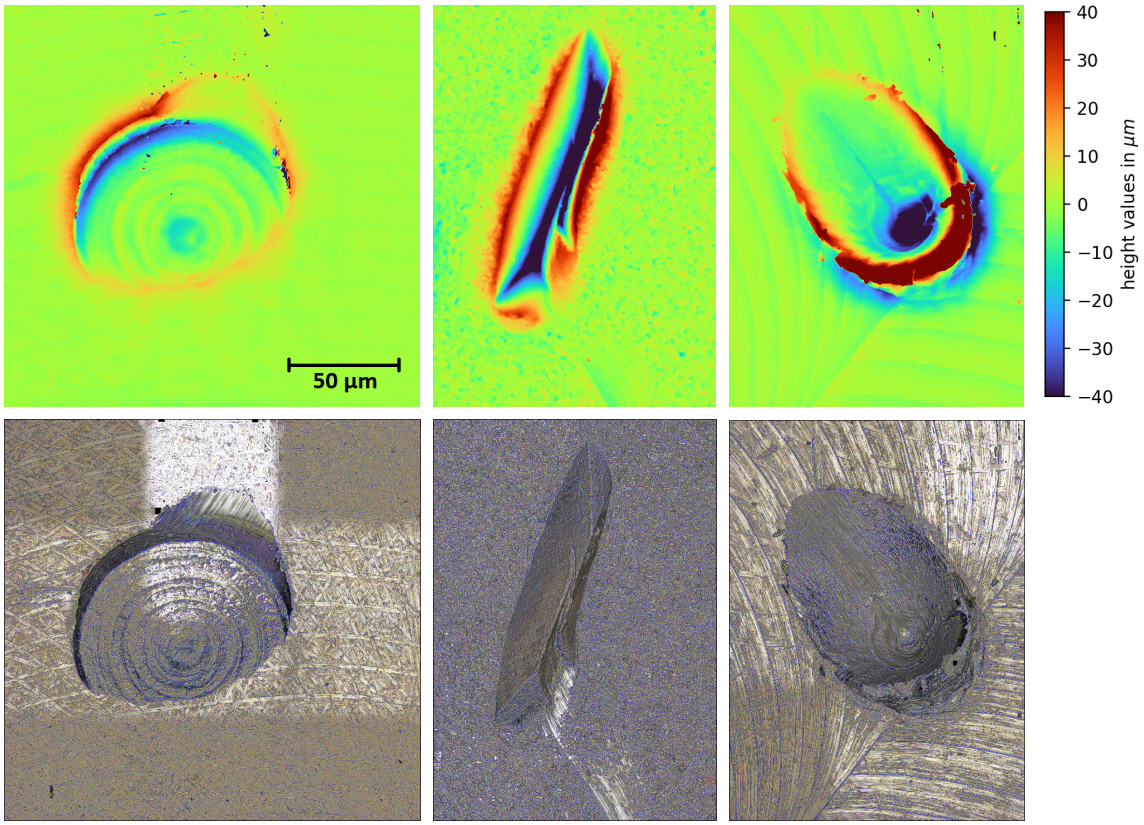
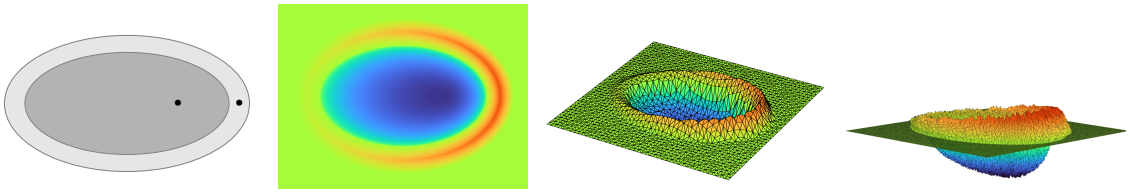


Figure 10.2: Dent defects on test objects caused by different defecting tools and different forces. Images are provided by Fraunhofer IOF using a focus-variation microscope. All images have the same resolution but the size of the imaged region varies. The top row shows 2d height images and the bottom row contains the corresponding optical 3d images.



Basic 2d shape with indentation minimum and accumulation maximum.

3d shape represented as 2d height image.

Triangulated 3d shape using image grid points as vertices.

Additional noise to perturb the triangulation vertices.

Figure 10.3: Overview of modeling pipeline for dent defects.

1. definition of the basic 2d shape,
2. positioning of the indentation minimum and the accumulation maximum,
3. determination of the 3d behavior of the surface,
4. sampling from the 3d shape to compute the surface triangulation,
5. addition of noise to the triangulation vertices.

10.3.1 2d Shape

As can be seen in the Figures shown in Section 10.2, the 2d shape of the dent's indentation depends on the geometry of the defecting tool. Moreover, as the tool hits the object surface, the material has to evade and is thus pushed to the sides. This results in accumulations around the indentation.

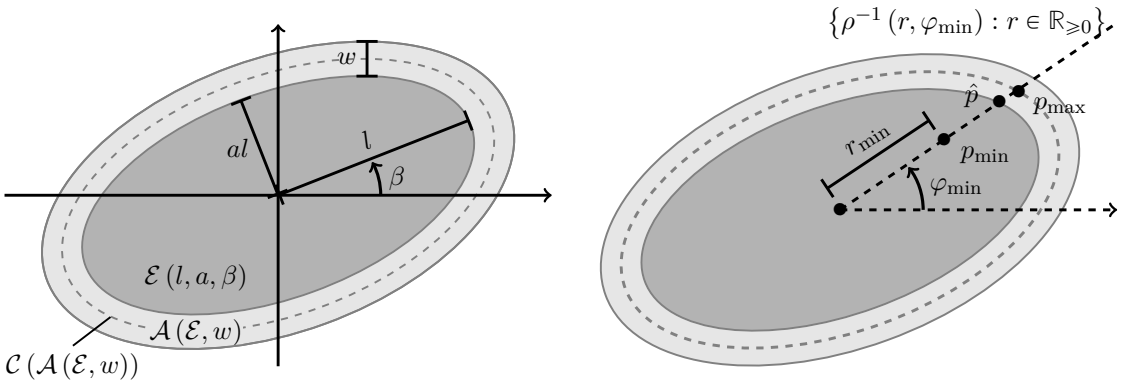
We will model the 2d form of the indentation using two basic shapes, namely an ellipse or a rectangle. Both are defined using the half length $l \in \mathbb{R}_{>0}$ of the main axis and its ratio $a \in (0, 1]$ to the half length of the minor axis. First, the set of points defining the ellipse $\mathcal{E}(l, a)$ and the rectangle $\mathcal{R}(l, a)$ centered in $(0, 0)$ and aligned with the x-axis are given by

$$\mathcal{E}(l, a) = \left\{ (x, y) \in \mathbb{R}^2 : \left\| \left(\frac{x}{l}, \frac{y}{al} \right) \right\|_2 \leqslant 1 \right\},$$
$$\mathcal{R}(l, a) = \left\{ (x, y) \in \mathbb{R}^2 : \left\| \left(\frac{x}{l}, \frac{y}{al} \right) \right\|_\infty \leqslant 1 \right\}.$$

Second, both shapes can be rotated around the center point by an angle $\beta \in [0, \pi)$. This gives an extra parameter resulting in $\mathcal{E}(l, a, \beta)$ and $\mathcal{R}(l, a, \beta)$. See Figure 10.4a for an illustrated explanation of all parameters. For general definitions we use $\mathcal{G} \in \{\mathcal{E}(l, a, \beta), \mathcal{R}(l, a, \beta)\}$.

The lateral accumulation is modeled by dilating the indentation shape by a certain width $w \in \mathbb{R}_{\geqslant 0}$ according to the associated norm. The resulting ring-shaped area is denoted by $\mathcal{A}(\mathcal{G}, w)$ for $\mathcal{G}$ the basic shape of the indentation. For abbreviation we will use $\mathcal{A}$ if the parameters are irrelevant. Later, the center line $\mathcal{C}(\mathcal{A}(\mathcal{G}, w))$ of the accumulation area is needed. For simplicity assume $\beta = 0$. Then, the center line is given by

$$\mathcal{C}(\mathcal{A}(\mathcal{E}, w)) = \left\{ (x, y) \in \mathbb{R}^2 : \left\| \left(\frac{x}{(l + w/2)}, \frac{y}{(al + w/2)} \right) \right\|_2 = 1 \right\},$$
$$\mathcal{C}(\mathcal{A}(\mathcal{R}, w)) = \left\{ (x, y) \in \mathbb{R}^2 : \left\| \left(\frac{x}{(l + w/2)}, \frac{y}{(al + w/2)} \right) \right\|_\infty = 1 \right\}.$$



(a) Illustration of the parameters of the 2d shape given by an ellipse.

(b) Illustration of the definitions of the indentation minimum and the accumulation maximum.

Figure 10.4: Schematic drawings of the 2d ellipse shape and the location of the indentation minimum and the accumulation maximum.

10.3.2 Location of Minimum and Maximum

To compute the 3d shape of a dent defect, first the locations of its indentation minimum and accumulation maximum need to be determined. The minimum $p_{\min} \in \mathcal{G}$ is not necessarily set as the center of the 2d shape but has to be located in the basic shape $\mathcal{G}$. We specify its position in polar coordinates given by the angle $\varphi_{\min} \in [0, 2\pi)$ to the x-axis and the distance $r_{\min} \in \mathbb{R}_{\geqslant 0}$ to $(0,0)$. As $r_{\min}$ should be a general parameter defined independently of the size of $\mathcal{G}$, it is given by $r_{\min} = a_{\min}\|\hat{p}\|$ for $a_{\min} \in [0,1)$. Here, $\hat{p}$ is the intersection of the 2d shape's border $\partial\mathcal{G}$ with the line segment starting in $(0,0)$ and growing in direction $\varphi_{\min}$, that is $\{\rho^{-1}(r,\varphi_{\min}) : r \in \mathbb{R}_{\geqslant 0}\}$ with ρ^{-1} the transformation from polar to Cartesian coordinates as introduced in Definition 1. Thus, the minimum is shifted from $(0,0)$ to $\hat{p}$ relative to the distance $\|\hat{p}\|$.

The location of the accumulation maximum $p_{\max} = \rho^{-1}(r_{\max}, \varphi_{\max}) \in \mathcal{A}(\mathcal{G}, w)$ is also given in polar coordinates. However, it is sufficient to provide its direction $\varphi_{\max}$ only as we assume $p_{\max}$ to be located on the center line, see Figure 10.4b. We commonly use $\varphi = \varphi_{\min} = \varphi_{\max}$.

10.3.3 3d Shape

The 3d shape depends on the tool geometry and the way it hits the object surface. It is modeled as 2d surface in 3d.

The basic defect geometry is modeled by a function $\mathcal{S} : \mathbb{R}^2 \to [-1,1]$ with $\mathcal{S}(\mathcal{G}) \leqslant 0$, $\mathcal{S}(\mathcal{A}) \geqslant 0$ and $\mathcal{S}(x) = 0$ for $x \in \mathbb{R}^2 \setminus (\mathcal{G} \cup \mathcal{A})$. The actual heights are then obtained by scaling $\mathcal{S}$ by $\mathcal{H} : \mathbb{R}^2 \to \mathbb{R}$, that is $\mathcal{S}(x)\mathcal{H}(x)$.

To take into account the diversity of real-world defecting tools, we consider four different shape functions which are illustrated in Figure 10.5. Note that different functions can be chosen for indentation and accumulation.

1. **Indicator shape**
   The simplest 3d shape is obtained by just projecting the 2d shape. Here, no locations of the indentation minimum and accumulation maximum are needed. The shape

function reads

$$\mathcal{S}_{\text{indicator}}(x) = \begin{cases} -1 & \text{if } x \in \mathcal{G} \\ 1 & \text{if } x \in \mathcal{A} \\ 0 & \text{else.} \end{cases}$$

2. **Linear shape**
   The linear shape of the indentation assumes a linear decline in direction of the indentation minimum. For the accumulation, the height values increase linearly to the center ring of the dilated area. To define the shape function, we first need functions $\mathcal{D}_{\text{indent}}$ and $\mathcal{D}_{\text{accum}}$ dependent on the distance to $p_{\min}$ and $\mathcal{C}(\mathcal{A})$, respectively. For $x \in \mathbb{R}^2$ they are defined via

$$\mathcal{D}_{\text{indent}}(x) = \frac{\|x - p_{\min}\|}{\|\hat{x} - p_{\min}\|} \text{ for } \hat{x} = \partial\mathcal{G} \cap \{\lambda(x - p_{\min}), \lambda \geqslant 0\},$$
$$\mathcal{D}_{\text{accum}}(x) = \frac{\|x - \tilde{x}\|}{w/2} \text{ for } \tilde{x} = \perp_{\mathcal{C}(\mathcal{A})}(x).$$

   Here, $\perp_X(x)$ is the orthogonal projection of a point x on a set $X \subset \mathbb{R}^2$. It holds $\mathcal{D}_{\text{indent}}(p_{\min}) = \mathcal{D}_{\text{accum}}(\mathcal{C}(\mathcal{A})) = 0$ and $\mathcal{D}_{\text{indent}}(\partial\mathcal{G}) = \mathcal{D}_{\text{accum}}(\partial\mathcal{A}) = 1$. Then, the shape function is given as

$$\mathcal{S}_{\text{linear}}(x) = \begin{cases} \mathcal{D}_{\text{indent}}(x) - 1 & \text{if } x \in \mathcal{G} \\ 1 - \mathcal{D}_{\text{accum}}(x) & \text{if } x \in \mathcal{A} \\ 0 & \text{else.} \end{cases}$$

3. **Cosine shape**
   To obtain shapes with smooth minimum and maxima, a cosine shape instead of a linear behavior can be used. The function also relies on $\mathcal{D}_{\text{indent}}$ and $\mathcal{D}_{\text{accum}}$ to evaluate the cosine properly. Then, the resulting shape function reads

$$\mathcal{S}_{\text{cosine}}(x) = \begin{cases} -\cos\left(\frac{1}{2}\pi\,\mathcal{D}_{\text{indent}}(x)\right) & \text{if } x \in \mathcal{G} \\ \cos\left(\frac{1}{2}\pi\,\mathcal{D}_{\text{accum}}(x)\right) & \text{if } x \in \mathcal{A} \\ 0 & \text{else.} \end{cases}$$

4. **Half-sphere shape**
   A shape based on a half-sphere can be applied to smooth also the transition at the border between $\mathcal{G}$ and $\mathcal{A}$. Moreover, the indentation is broader than for the linear and cosine shape. The corresponding shape function is defined by

$$\mathcal{S}_{\text{half-sphere}}(x) = \begin{cases} -\sqrt{\left(1 - (\mathcal{D}_{\text{indent}}(x))^2\right)} & \text{if } x \in \mathcal{G} \\ \sqrt{\left(1 - (\mathcal{D}_{\text{accum}}(x))^2\right)} & \text{if } x \in \mathcal{A} \\ 0 & \text{else.} \end{cases}$$

To scale the shape function, first the corresponding values $h_{\min} \in \mathbb{R}_{>0}$ for the indentation minimum and $h_{\max}(\varphi_{\max}) \in \mathbb{R}_{\geqslant 0}$ for the accumulation maximum are required. The accumulation heights may decrease in direction $\varphi_{\max} + \pi$. To include this behavior let

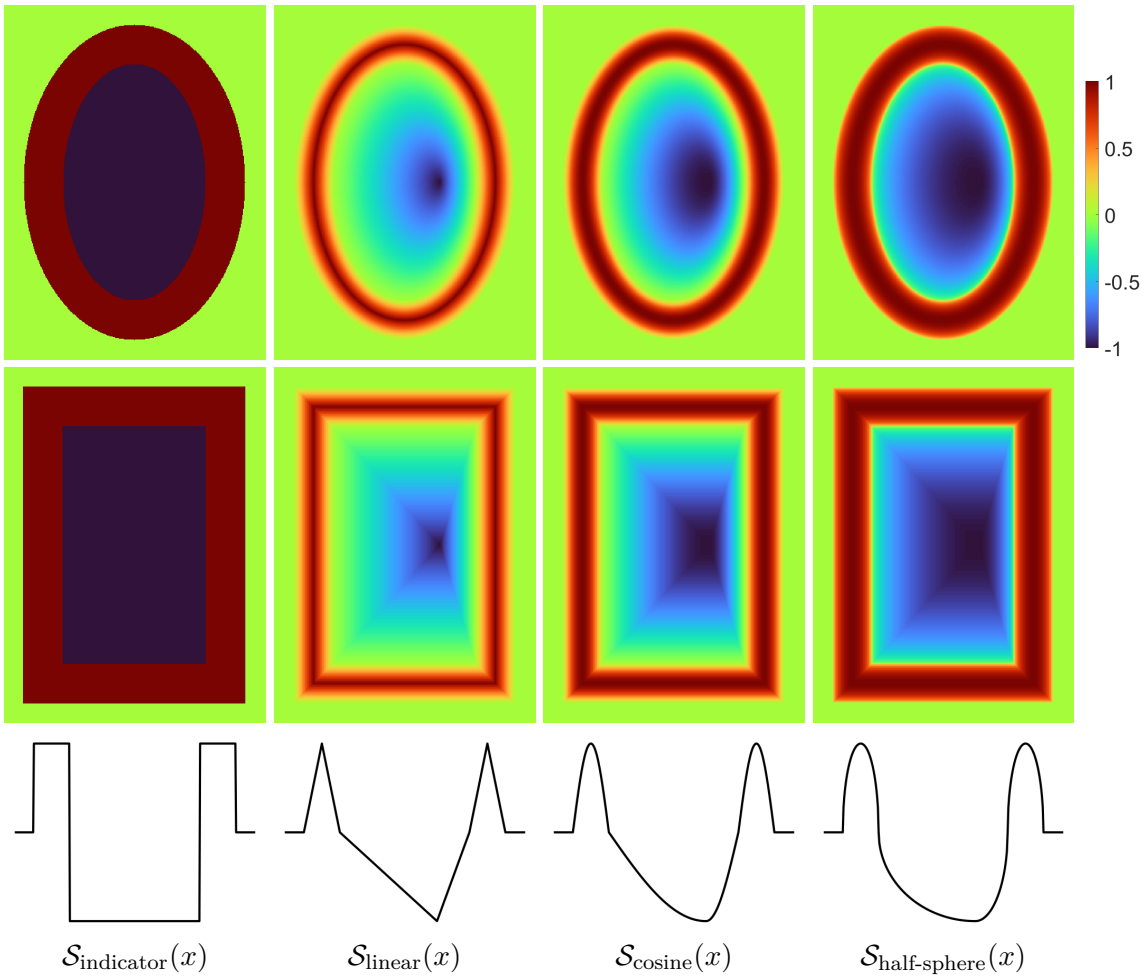


Figure 10.5: Illustration of the shape functions applied on the 2d basic shape of a rotated ellipse and rectangle using $\beta = \pi/2$, top and middle row, respectively. The height profile for $x = 0$ is shown in the bottom row. For the indentation minimum $a_{\min} = 0.5$ and $\varphi_{\min} = 0$ are chosen.

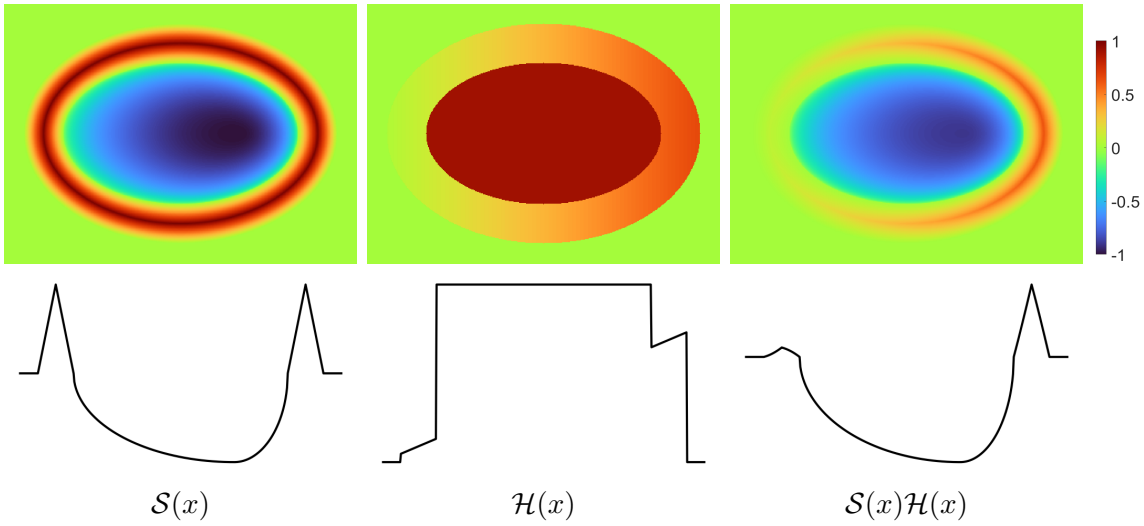


Figure 10.6: Illustration of the scaling function and its impact on the 3d shape using $h_{\min} = 0.9$, $h_{\max}(\varphi) = 0.6$ and $h_{\max}(\varphi + \pi) = 0.1$. The parameters for the minimum are $a_{\min} = 0.5$ and $\varphi = 0$. A combination of 3d shapes is taken, namely half-sphere shape for the indentation and linear shape for the accumulation.

$0 \leqslant h_{\max}(\varphi_{\max} + \pi) \leqslant h_{\max}(\varphi_{\max})$ be the scaling factor reached at $\rho^{-1}(r_{\max}, \varphi_{\max} + \pi)$. Then, the scaling factors for the accumulation area are obtained by a tilted plane similar as defined in Equation (6.1), namely

$$\text{plane}(x) = \frac{h}{2l + w/2} \left(\cos(\varphi_{\max})x_1 + \sin(\varphi_{\max})x_2\right) + h_{\max}(\varphi_{\max} + \pi) + \frac{h}{2}$$

for $h = h_{\max}(\varphi_{\max}) - h_{\max}(\varphi_{\max} + \pi)$. Finally, the scaling function is defined by

$$\mathcal{H}(x) = \begin{cases} h_{\min} & \text{if } x \in \mathcal{G} \\ \text{plane}(x) & \text{if } x \in \mathcal{A} \\ & \text{else.} \end{cases}$$

Figure 10.6 shows an example of a height function and how it is applied on the 3d shape.

10.3.4 Sampling

So far, a dent defect is modeled as 2d surface in 3d by using $\mathcal{S}(x)\mathcal{H}(x)$ which is defined on the whole domain $\mathbb{R}^2$. However, to imprint the defect geometry into an object provided by its surface triangulation, the 3d defect shape has to be discretized and transformed into a surface mesh. Therefore, we need a set of points $\mathscr{P}$ to evaluate the function $\mathcal{S}(x)\mathcal{H}(x)$ which will then serve as approximation of the 3d defect shape. The more points are considered, the more accurate but also the more complex the resulting approximation. We restrict to a rectangular window $W \subset \mathbb{R}^2$ with $\mathcal{G} \cup \mathcal{A} \subset W$ to sample the points for $\mathscr{P}$. Then, the points in $\mathscr{P}$ are triangulated by applying the Delaunay triangulation. To transfer to 3d, we replace the 2d points by $\{(p, \mathcal{S}(p)\mathcal{H}(p)) : p \in \mathscr{P}\}$. The same triangles still create a valid surface triangulation in 3d.

Two different sampling methods will be discussed in the following. First, uniform sampling is defined via $\mathscr{P} = W \cap \mathbb{L}^2$ for a lattice $\mathbb{L}^2$. It results in a regular representation

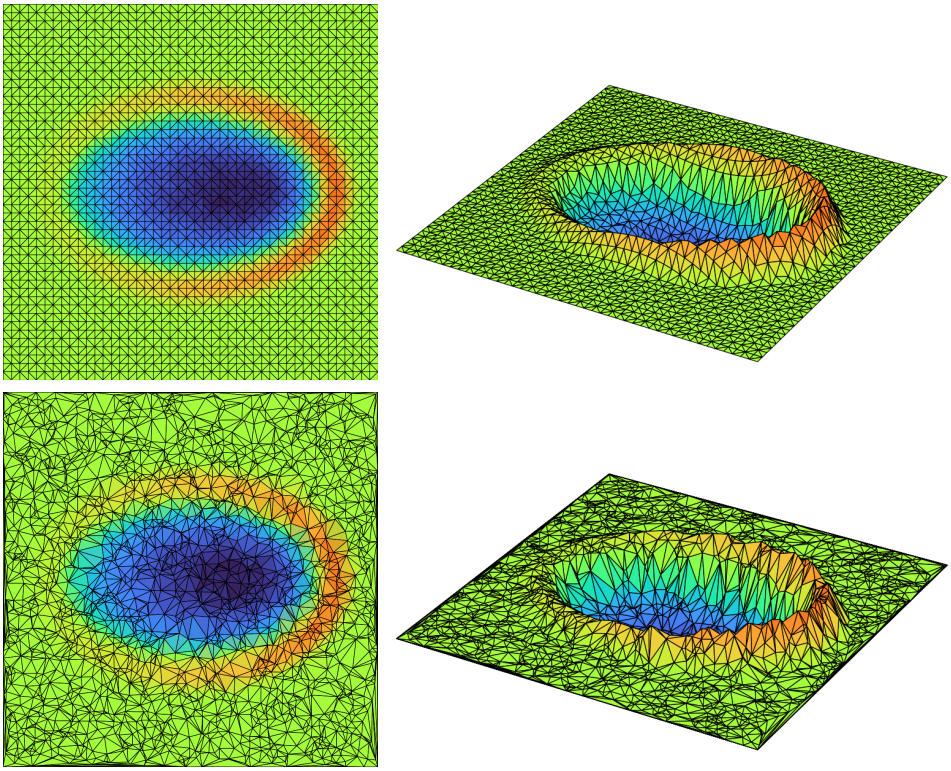


Figure 10.7: Comparison of uniform and random sampling represented in the top and bottom row, respectively. 2d height image (left) and 3d surface (right) are shown. Both methods are applied using 2116 vertices.

of the defect surface. In contrast, random sampling can be used, obtained by sampling from the uniform distribution defined on W, that is $\mathscr{P} = \{p_1, \dots, p_m\}$ for $m \in \mathbb{N}$ with $p_i \sim \mathcal{U}(W)$, $i = 1, \dots, m$. This provides a more random surface triangulation. See Figure 10.7 for a comparison of both methods.

10.3.5 Noise

Although sampling approximates the 3d defect surface, it might still be too smooth to realistically represent a dent defect. Therefore, we add noise to the triangulation vertices. Let n_p be the upwards pointing unit normal vector of the 3d defect shape evaluated at $p \in \mathscr{P}$. Then, noise is added by perturbing the triangulation vertices in direction of the normal vectors, that is $(p, \mathcal{S}(p)\mathcal{H}(p)) + s_p n_p$ for scaling factors $s_p \in \mathbb{R}$. The scaling factors are uniformly sampled from $\mathcal{U}(s_{\min}, s_{\max})$ with $s_{\min} \leqslant s_{\max}$. Here, the minimal and maximal scaling factors are chosen dependent on where p is located in W, that is $s_{\min}, s_{\max} \leqslant 0$ for $p \in \mathcal{G}$, $s_{\min}, s_{\max} \geqslant 0$ for $p \in \mathcal{A}$ and $s_{\min} = s_{\max} = 0$ in the remaining case. We always perturb in direction where the 3d defect surface is convex to prevent the collision of vertices such that the validity of the surface triangulation is maintained. See Figure 10.8 for an example.

10.3.6 Voronoi-Approximation of Defect Surface

Many real-world materials are poly-crystalline and can thus be represented as grain-based structures. 3d Voronoi tessellations, defined in Section 3.1.8, can be used to simulate those structures as they are space-filling partitions into several convex polytopes representing

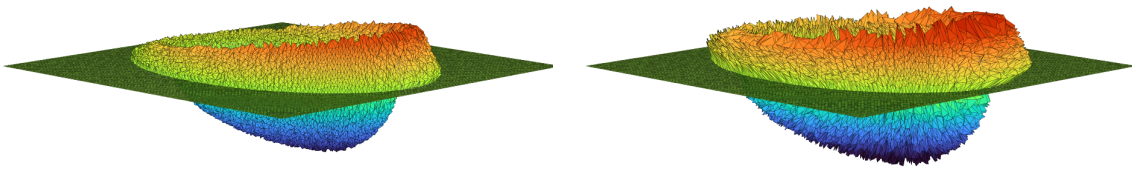


Figure 10.8: Comparison of different noise levels using $s_{\min} = -s$, $s_{\max} = 0$ for $\mathcal{G}$ and $s_{\min} = 0$, $s_{\max} = s$ for $\mathcal{A}$ with $s = 0.1$ and $s = 0.3$ for the left and right simulation, respectively. Uniform sampling is applied.

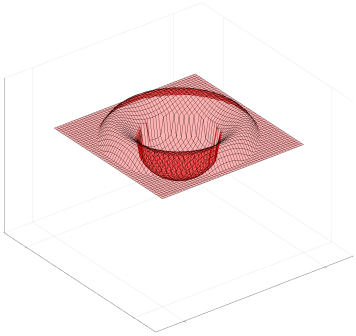
the grains. Therefore, a Voronoi-approximation as discussed for example by Heveling and Reitzner [50] can be used as alternative discretization method of the defect surface. This approach consists of the following steps, see Figure 10.9 for an illustration.

1. Determine the 3d shape of the defect.
2. Generate a Voronoi tessellation restricted to a window in $\mathbb{R}^3$ which covers the 3d shape of the defect.
3. Detect all Voronoi cells whose generator point is located below the 3d shape, called relevant cells.
4. Determine the Voronoi facets creating the upper boundary of the union of all relevant cells, denoted as relevant facets.
5. Triangulate the relevant facets.
6. Extend the triangulation at the border to enable a smooth transition to the object surface the defect will be imprinted. Therefore, new triangles are added using new vertices that have height zero.

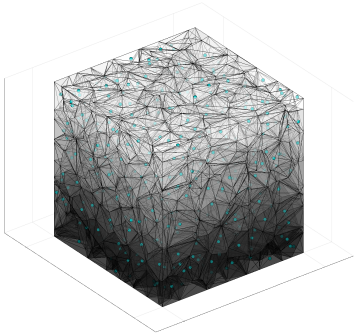
Note that this approach needs plenty of generator points to adequately approximate the 3d shape of a defect. Moreover, the determination of relevant facets is computationally expensive for a sufficient number of generators. Therefore, we restrict to the previously introduced method for approximating the 3d shape.

10.4 Conclusion

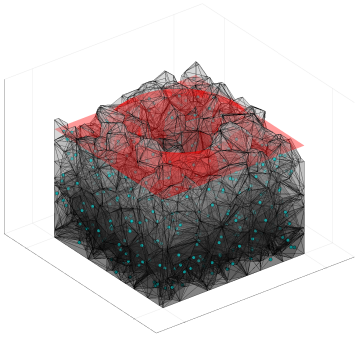
We presented a flexible, parametric multi-step model to simulate dent defects. Different basic 2d shapes are implemented, but can arbitrarily be extended by any other possible shape. We chose ellipse and rectangle as they are simple but can still represent the defect shapes indicated in Section 10.2. Also, different 3d shapes are proposed to enable variable defect simulation and to include different defecting tool geometries. To increase the diversity of the defect generation and also the precision towards the responsible defecting tool, the indentation minimum can be located anywhere in the indented area. Finally, we sample from the 3d defect shape, add noise such that the surface appears more realistic and transform the output into a 3d mesh to enable the defect to be imprinted into an object geometry provided by its surface triangulation. Synthetic images of the resulting defective objects are shown in Figure 10.10 and Figure 10.11. The applied texture of a cast metal surface was implemented by Lovor Bosnar and is based on the cellular texture introduced by Worley [115].



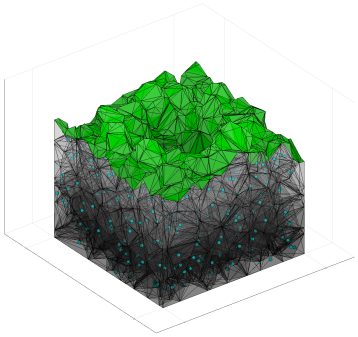
3d shape represented as surface mesh.



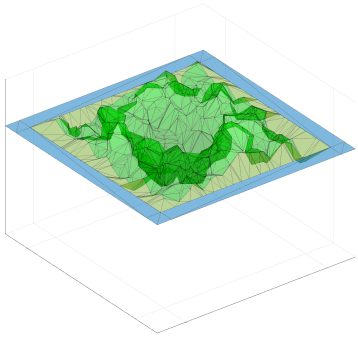
3d Voronoi tessellation using 1000 generator points (blue). The color gradient indicates the height of the generator points.



Intersection of 3d surface and Voronoi tessellation and removal of irrelevant cells.



Labeling of relevant facets.



Triangulation of relevant Voronoi facets to obtain 3d surface approximation.

Figure 10.9: Overview of the pipeline to approximate the 3d surface by using a 3d Voronoi tessellation.

So far, the defects appear too regular. This may be solved by varying the accumulation heights while traversing over the center line. Also, differing the accumulation width while traversing over $\partial\mathcal{G}$ may generate a more realistic defect shape. Moreover, when assuming no compression or destruction of the material during defecting, the total volume of the defect geometry should integrate to zero. This circumstance is not yet included but is considered as future work.

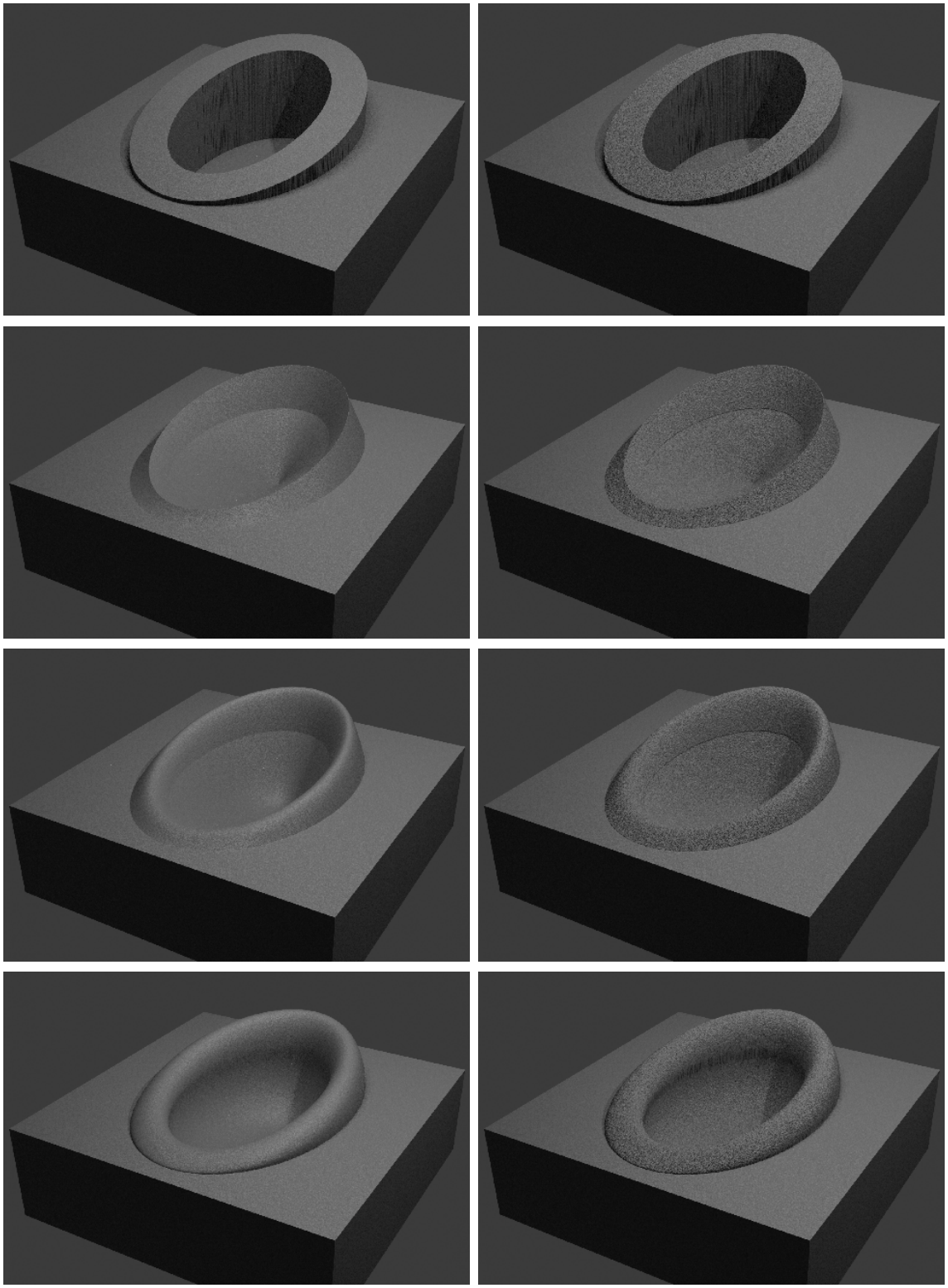


Figure 10.10: Dent defects applied to a cuboid geometry rendered in a virtual environment using Blender [1] (8 spp and 8 light bounces). Casting surface texture is used. Ellipse is chosen as 2d shape with identity, linear, cosine and half-sphere as 3d shape, from top to bottom, simulated using no noise or with noise in the left and right column, respectively. Regular sampling is applied.

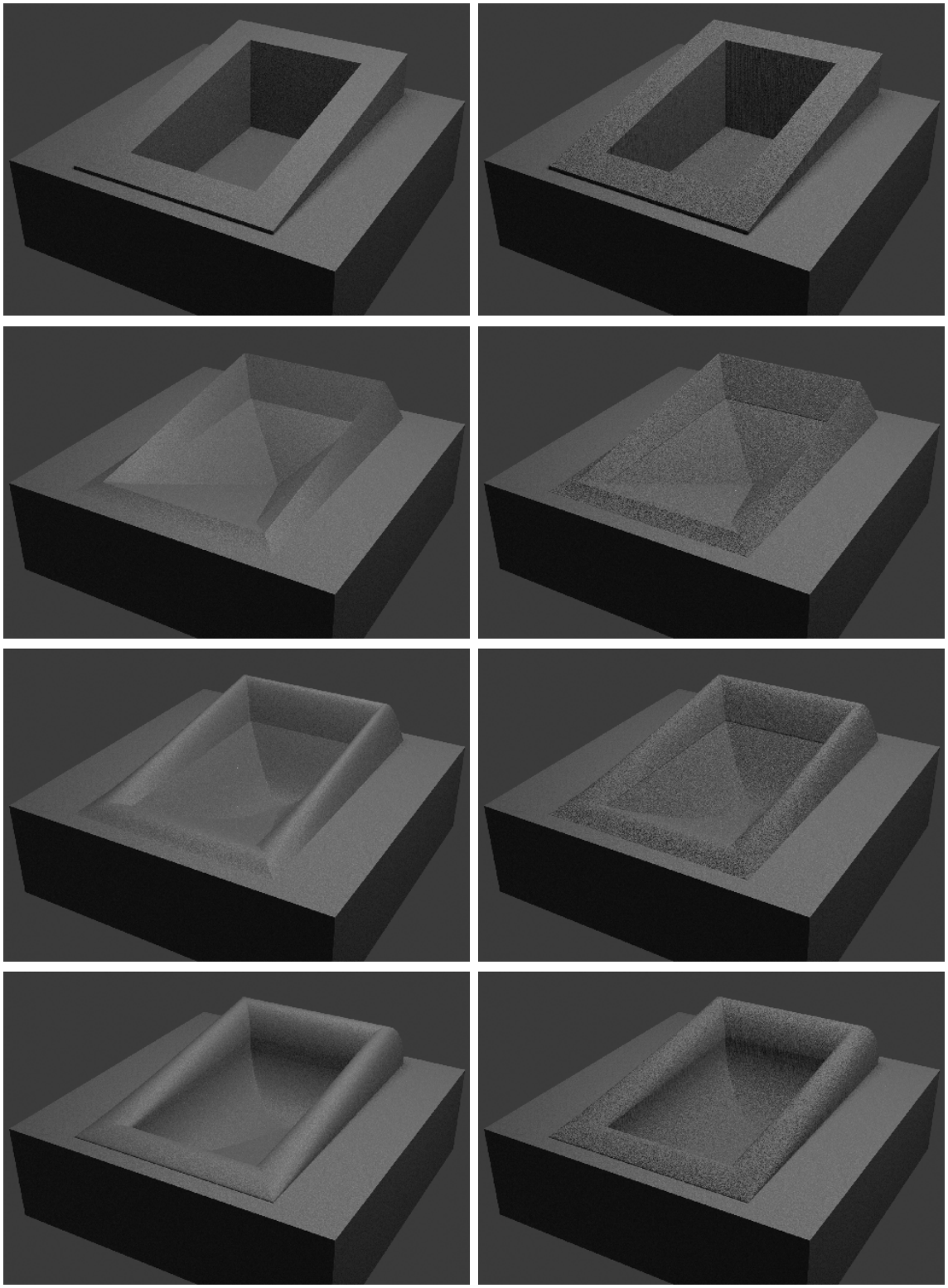


Figure 10.11: Dent defects applied to a cuboid geometry rendered in a virtual environment using Blender [1] (8 spp and 8 light bounces). Casting surface texture is used. Rectangle is chosen as 2d shape with identity, linear, cosine and half-sphere as 3d shape, from top to bottom, simulated using no noise or with noise in the left and right column, respectively. Regular sampling is applied.

11 Casting Defects

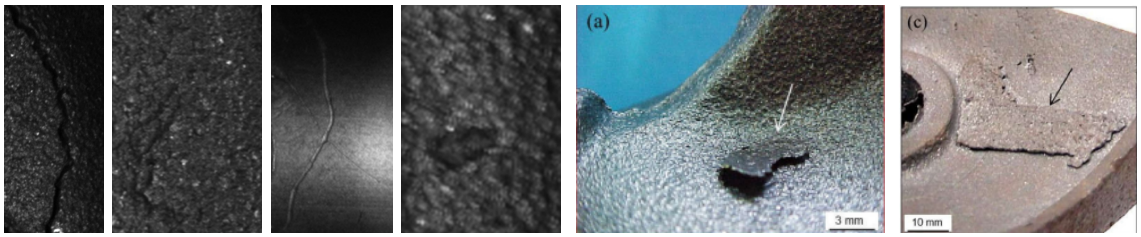
This chapter was published in [55].

11.1 Introduction

Casting is a manufacturing process in which liquid metal is poured into the object mold and then cooled down to solidify. Defects can be caused by a number of reasons: a defective mold, unsuitable object geometries, inappropriate material temperature that influences the fluidity, or too cold ambient air temperature that leads to rapid cooling of the material. We distinguish between elongated defect types (cold cracks, hot tears, buckles, bulges, coat lifts, cold shuts and rat tails) and scab defects, see [72, 93] for an overview. Figure 11.1 shows example images of some defect types.

Models of casting defects are provided in the literature. For instance, Jolly and Katgerman [57] give an extensive overview of modeling approaches for various defect types of cast aluminum products. Most methods are based on modeling the physics of the casting process including the temperature difference during the filling of the mold and the cooling of the material. Physically correct defect geometries are provided by investigating the physical material properties. Including the crystalline structure of the material and its behavior under stress results in complex defect models. Preliminary studies or expert knowledge are needed to obtain exact information on the composition and strength of the used material such that those specialized models cannot easily be transferred to other applications. Sistaninia et al. [96] propose a detailed model for crack formation as metal solidifies. They include the poly-crystalline structure of metal and the heat change during the cooling process. Also, Peng et al. [81] present a grain-based model for micro cracking in granite that includes its mineralogical composition.

We propose parametric mathematical models to generate various types of surface defects that occur in cast metal objects. The development of the models is motivated by the observation of defect structures and the description of defect formation processes [2, 57, 72, 93]. Using stochastic modeling based on random Voronoi tessellations, we introduce a



Cracks Cold shut Bulge
Images provided by Fraunhofer ITWM.

Lustrous carbon inclusion (similar to coat lift). Image taken as crop from Figure 38a in [93].

Scab. Image taken as crop from Figure 23c in [93].

Figure 11.1: Examples of common surface defects in cast metal objects.

flexible pipeline for defect modeling that can be adapted to generate most of the different defect types. Moreover, our models are parameterized to guarantee defect variation using easily interpretable parameter configurations. Furthermore, sampling the parameter values from suitable random distributions ensures variable defect geometries. In this way, defect characteristics such as length, width and depth can be controlled and adapted to the desired parameter ranges. The data set content is controllable in terms of defect properties and appearance, also edge case scenarios can easily be included. Thus, it is possible to systematically vary individual aspects such that the resulting synthetic data can be used to train detection or classification algorithms for specific tasks. This provides a significant advantage over AI-based methods, where systematic data generation and control is difficult to perform.

Although we consider casting defects, similar defect shapes can also occur in other manufacturing processes. Therefore, our models can be transferred to other applications by using the same synthetic defect geometries but adapting the virtual object in terms of material and surface texture.

The modeling procedure differs for elongated and scab defects. But, the same basic framework is used for all elongated defects. Section 11.2 first introduces the general pipeline for elongated defects which is then adapted to the individual defect types. Section 11.3 deals with scab defects.

11.2 Models for Elongated Defects

All defect types, except for the scabs, are characterized by a dominant elongated structure. The behavior of that predominant line can be different, e.g. smooth for cold shuts and rat tails and jagged for the others, and the defects are indented into or added to the product geometry. Nevertheless, one basic model can be used and adapted to generate all the individual defect shapes. The approach is based on Jung and Redenbach's [59] model to create semi-synthetic cracks in 3d µCT images of concrete. In this approach, minimum-weight surfaces computed from a 3d Voronoi tessellation are discretized, dilated and then embedded into real µCT images. For applications in surface inspection, the focus is on modeling the planar (2d) shape of a defect. The third dimension is included by locally assigning height values to the positions of the 2d shape. Our basic defect model thus subdivides into the following steps, see Figure 11.2.

1. Generate a path.
2. Dilate the path using segment-wise widths.
3. Assign height values to the vertices and compute a triangulation such that a valid mesh of a 2d surface is created.
4. Compute the 3d mesh of the defect geometry.

The first step consists of generating the defect-specific 1d shape which can either be a smooth or a jagged path. Then, that path is dilated to obtain the 2d shape of the defect. We use segment-wise widths to better control the spatial extension of the defects. Next, height values are assigned to the vertices. Finally, the triangulation of the 3d vertices is computed to obtain a valid 3d mesh of the defect geometry. Figure 11.2 shows an overview of all adaptations and relations between the defect models. A detailed description of the individual steps and their defect-specific adaptations is given below. Moreover, Figure 11.3 shows synthetic images of the different defect types rendered with parameters of a realistic imaging setup.

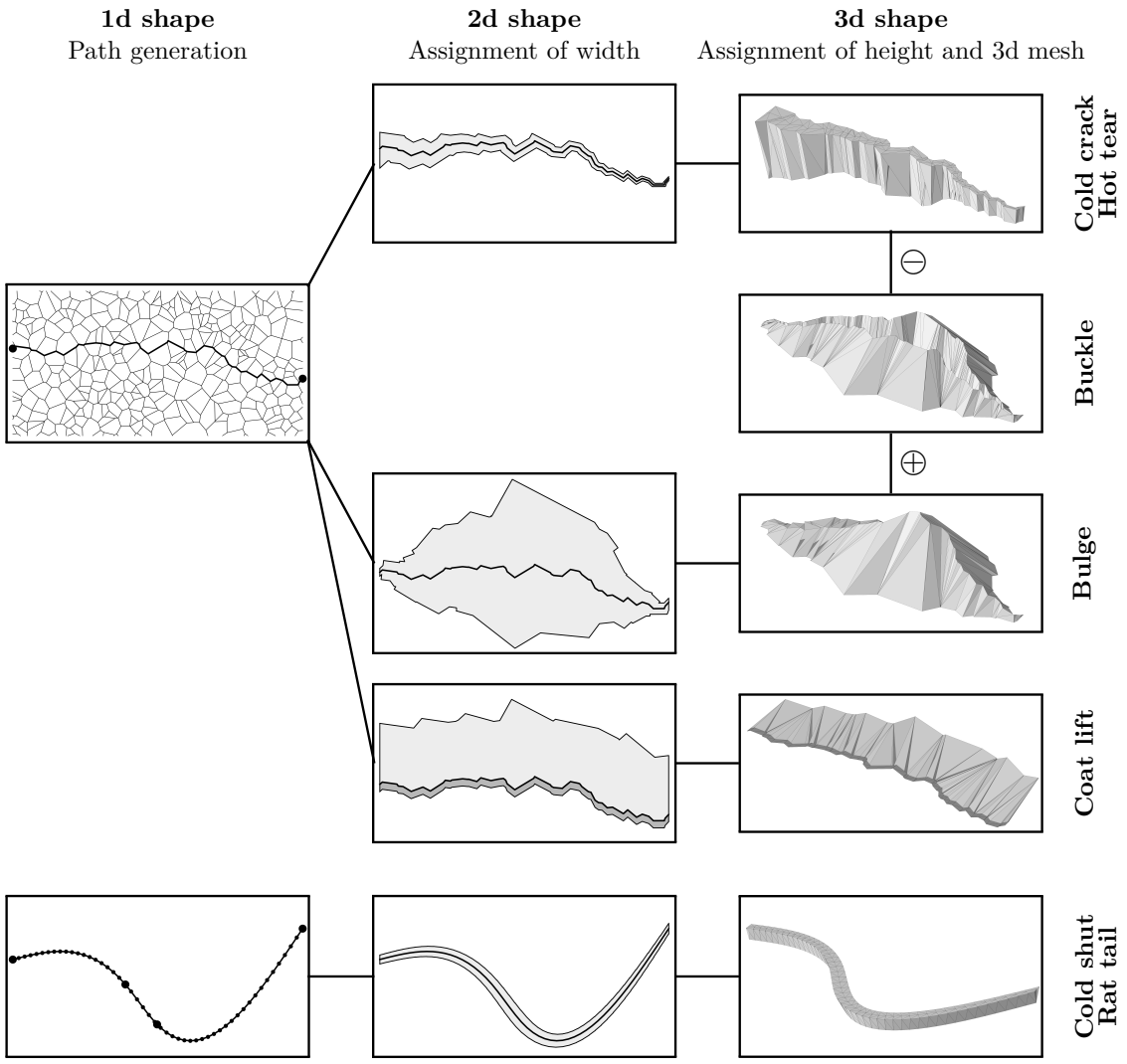


Figure 11.2: Overview of modeling pipeline including the different steps, adaptation to and relations between different defect types.

11.2.1 Details of Individual Modeling Steps

11.2.1.1 Computation of a Path

To obtain the path, two steps are needed. First, a 2d Voronoi tessellation is generated followed by computing a minimal path through its edge system. We use a bounded Voronoi tessellation with i.i.d generator points $p_i \sim \mathcal{U}(W)$ that follow a uniform distribution on a rectangular window $W = \left[w_0^{\min}, w_0^{\max}\right] \times \left[w_1^{\min}, w_1^{\max}\right]$.

For our application, we need shortest paths computed in a 2d Voronoi tessellation. Therefore, the vertices and edges of the tessellation serve as vertices and arcs to define a graph G. Note that for every edge in the Voronoi tessellation we introduce two directed arcs in opposite directions in the planar graph. The weights are given by the lengths of the edges. As start and end vertices of the shortest path we choose Voronoi vertices located on the left and right side of the window, that is, $(\alpha_0)_0 = w_0^{\min}$ and $(\omega_k)_0 = w_0^{\max}$. The shortest path is then computed using Dijkstra's algorithm [22]. See Figure 11.2 for an example.

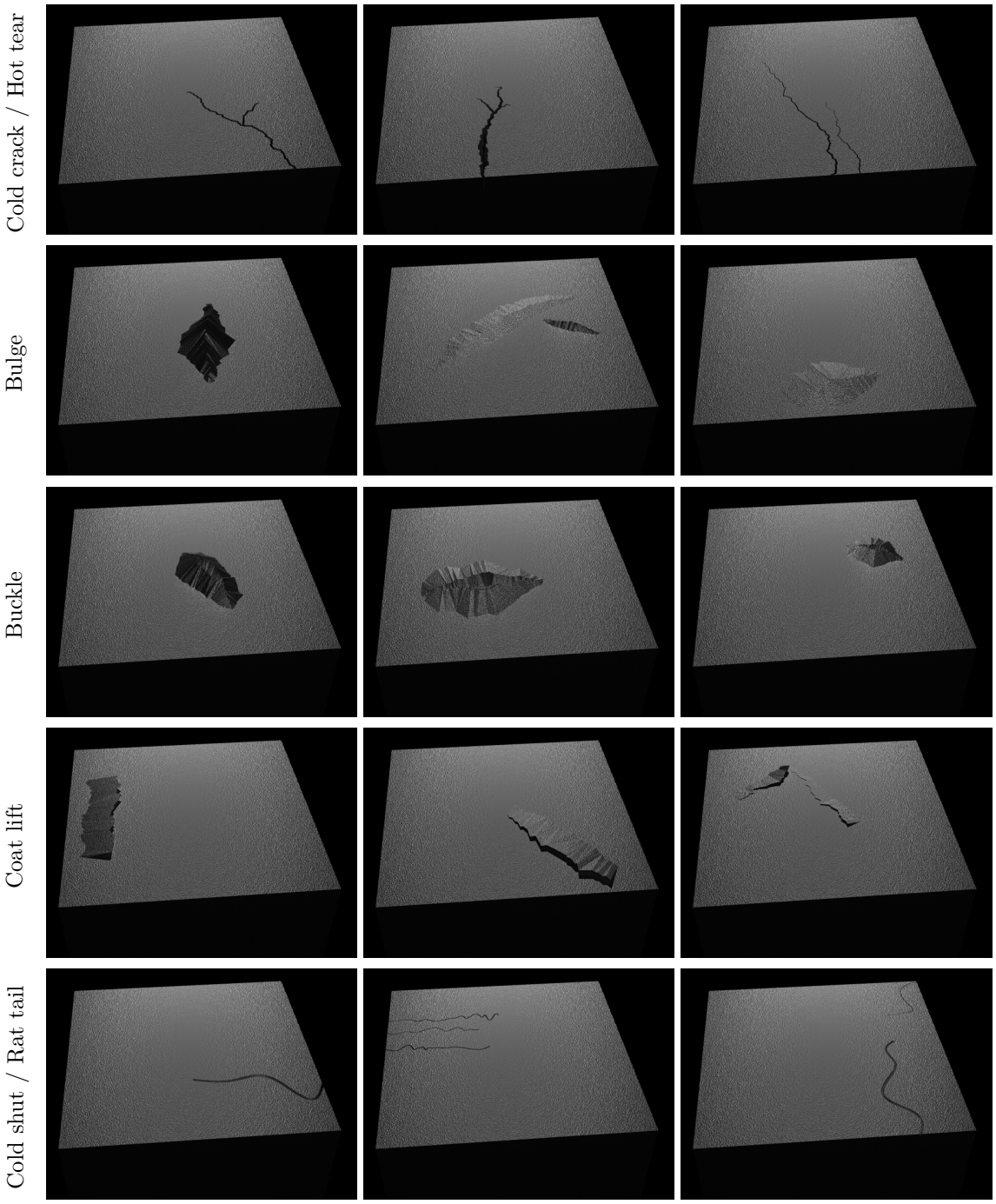
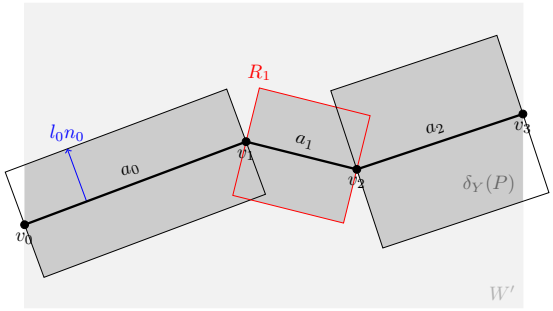
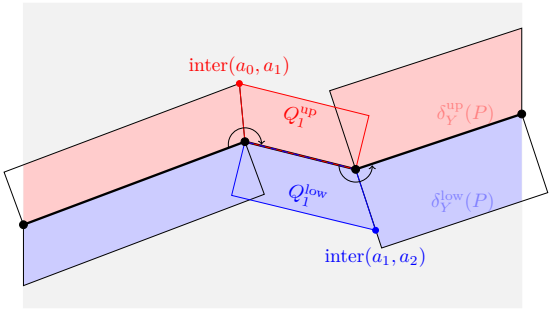


Figure 11.3: Synthetic images of different defect types in a cast metal cube obtained by rendering from a synthetic industrial image acquisition system created in Blender [1]. A pinhole camera and two plane lights are used for illumination. Rendering is performed using 24 spp and 16 light bounces. The surface texture is chosen to resemble the appearance of a cast metal object. The texture model was integrated by Lovro Bosnar from Fraunhofer ITWM and is based on the cellular texture introduced by Worley [115].



(a) Primitive dilation of a path.



(b) Adjustment such that no cone-shaped gaps appear for reflex angles.

Figure 11.4: Illustrations of different definitions for a dilated path.

11.2.1.2 Dilation of a Path

In general, the dilation of $X \subset \mathbb{R}^2$ by a structuring element $Y \subset \mathbb{R}^2$ is defined as

$$\Delta_Y(X) = \bigcup_{y \in Y} (X - y). \tag{11.1}$$

Now, we want to dilate a path $X = P$ as given in Section 11.2.1.1. Assume $P = (a_0, \ldots, a_k)$. Therefore, we adapt Definition (11.1) to an arc-wise dilation with structuring elements $Y = \{Y_0, \ldots, Y_k\}$,

$$\delta_Y(P) = \left(\bigcup_{i=0}^{k} \Delta_{Y_i}(a_i) \right) \cap W'. \tag{11.2}$$

Here, we choose $W' = \left[w_0^{\min}, w_0^{\max}\right] \times \mathbb{R}$ to restrict the dilation on the length of window W. As structuring elements we choose line segments $Y_i = \{t\, l_i\, n_i : t \in [-1, 1]\}$ defined by the upwards-pointing unit normal vectors n_i of the segments and positive half lengths $l_i \in \mathbb{R}_{>0}$. Thus, $\Delta_{Y_i}(a_i)$ is a rectangle R_i with vertices $\alpha_i \pm l_i n_i, \omega_i \pm l_i n_i$, see Figure 11.4a for an illustration. Therefore, Equation (11.2) can be reformulated as

$$\delta_Y(P) = \left(\bigcup_{i=0}^{k} R_i \right) \cap W'.$$

The dilation's contours are then given as an upper path P^{up} and a lower path P^{low}. In analogy, we divide the dilation of P into an upper and a lower dilation $\delta_Y^{\text{up}}(P)$ and $\delta_Y^{\text{low}}(P)$, respectively. They are defined by a separation of the rectangles into an upper rectangle R_i^{up} and a lower rectangle R_i^{low} with the vertices

$$R_i^{\text{up}} : \alpha_i, \alpha_i + l_i n_i, \omega_i, \omega_i + l_i n_i,$$
$$R_i^{\text{low}} : \alpha_i, \alpha_i - l_i n_i, \omega_i, \omega_i - l_i n_i.$$

Now consider the angle $\measuredangle(a_i, a_{i+1})$ between consecutive arcs. For the upper dilation, this angle is computed in clockwise direction and for the lower dilation in counterclockwise direction. In case of a reflex angle $\measuredangle(a_i, a_{i+1}) \in (\pi, 2\pi)$, the union of the corresponding rectangles has a cone-shaped gap. We adapt the corresponding rectangles to fill the gaps, see Figure 11.4b.

It is sufficient to explain the procedure for the upper dilation, since the approach can directly be transferred to the lower dilation. Compute the intersection

$$\text{inter}(a_i, a_{i+1}) = \{\omega_i + l_i n_i + \lambda a_i : \lambda \geqslant 0\} \cap \{\alpha_{i+1} + l_{i+1} n_{i+1} - \tau a_{i+1} : \tau \geqslant 0\}.$$

If $l_i \neq l_{i+1}$, this intersection can be empty. In such a case, elongate the edge having the smaller scaling factor until it intersects the nearest lateral side of the rectangle corresponding to the other edge. For simplicity, the intersection is also denoted by $\text{inter}(a_i, a_{i+1})$. Assuming $l_i < l_{i+1}$ this reads

$$\text{inter}(a_i, a_{i+1}) = \{\omega_i + l_i n_i + \lambda a_i : \lambda \geqslant 0\} \cap \{\alpha_{i+1} + \tau n_{i+1} : \tau \in [0, l_{i+1}]\}.$$

Now, rectangle R_i^{up} is replaced by a quadrangle Q_i^{up} with vertices $\alpha_i, \omega_i, \tilde{v}_i, \tilde{v}_{i+1}$ with

$$\tilde{v}_i = \begin{cases} \alpha_i + l_i n_i & \text{if } \measuredangle(a_{i-1}, a_i) \in (0, \pi] \\ \text{inter}(a_{i-1}, a_i) & \text{else} \end{cases} \quad \text{for } i = 1, \ldots, k,$$

$$\tilde{v}_{i+1} = \begin{cases} \omega_i + l_i n_i & \text{if } \measuredangle(a_i, a_{i+1}) \in (0, \pi] \\ \text{inter}(a_i, a_{i+1}) & \text{else} \end{cases} \quad \text{for } i = 0, \ldots, k-1.$$

Furthermore, note that cone-shaped gaps can occur at the start and end of the dilated path. In that case, the upper edge of the quadrangle is extended until it hits the boundary of the window, see Figure 11.4b. That is,

- if $\measuredangle(y\text{-axis}, a_0) \in (\pi/2, \pi)$, then replace $\alpha_0 + l_0 n_0$ by the intersection of $\alpha_0 + (0, \lambda)$ and $\alpha_0 + l_0 n_0 - \tau a_0$ for $\lambda, \tau \geqslant 0$,
- if $\measuredangle(a_k, y\text{-axis}) \in (\pi/2, \pi)$, then replace $\omega_k + l_k n_k$ by the intersection of $\omega_k + (0, \lambda)$ and $\omega_k + l_k n_k + \tau a_k$ for $\lambda, \tau \geqslant 0$.

Finally, the dilation of a path is given by

$$\tilde{\delta}_Y(P) = \bigcup_{i=0}^{k} \left(Q_i^{\text{up}} \cup Q_i^{\text{low}} \right) \cap W'.$$

Note that the dilation may contain holes, see for example Figure 11.5, which will be filled when computing the triangulation.

11.2.1.3 Transition from 2d to 3d

The shape of the resulting defect is determined by the path P and the dilation's upper and lower contours P^{up} and P^{low}. To obtain three-dimensional defects, height values are assigned to the vertices that form those paths. Denote the set of path vertices by $V = \left\{ v_0, \ldots, v_{k+1}, v_0^{\text{up}}, \ldots, v_{k^{\text{up}}+1}^{\text{up}}, v_0^{\text{low}}, \ldots, v_{k^{\text{low}}+1}^{\text{low}} \right\}$ and by h_i, $h_{i^{\text{up}}}^{\text{up}}$ and $h_{i^{\text{low}}}^{\text{low}}$ the assigned height values. Assume that defects are imprinted or added to an object with a planar surface of constant height $h > 0$. To obtain a smooth transition after imprinting, assign height h to the outermost vertices originating from the contours P^{up} and P^{low}. For the inner vertices belonging to P, we follow a defect-specific pipeline detailed in Section 11.2.2.

11.2.1.4 Triangulation of 2d Surface in 3d

In the next step, we compute a triangulation of the 3d vertices to obtain a mesh of the surface. In general, let $T(t_1, t_2, t_3)$ be a triangle with vertices t_1, t_2 and t_3. We consider the upper and lower dilation separately and introduce our approach for the upper dilation only. Represent the contour of the upper dilation as a path $P^{\text{up}} = \left(a_0^{\text{up}}, a_1^{\text{up}}, \ldots, a_{k^{\text{up}}}^{\text{up}}\right)$ with $\alpha_i^{\text{up}} = v_i^{\text{up}}$, $\omega_i^{\text{up}} = v_{i+1}^{\text{up}}$ for $i = 0, \ldots, k^{\text{up}}$, $(v_0^{\text{up}})_0 = w_0^{\min}$ and $(v_{k^{\text{up}}+1}^{\text{up}})_0 = w_0^{\max}$. For consistency, we use the same notation for the 3d vertices including heights as for the 2d vertices.

We say that a_j^{up} originates from a_i if a_j^{up} is part of the upper contour of Q_i^{up}. Note that an arc a_i in P does not necessarily give rise to an arc in P^{up}. Furthermore, the dilation polygon Q_i^{up} of an arc a_i can generate multiple arcs of P^{up}. We differentiate between different cases depending on the edge of Q_i^{up} that generates the dilation arc in P^{up}. First assume that the left lateral edge of Q_i^{up} contributes at most one arc to P^{up} and that, if it exists, this arc is connected to a dilation arc formed by the upper edge of Q_i^{up}. Furthermore, assume that Q_l^{up} with $l > i$ does not give rise to any arcs in P^{up} that lie further to the left than the arcs formed by Q_i^{up}. Then, the cases are defined as follows.

Case 1: This case refers to the leftmost (and often the only) dilation arc formed by the top edge of a dilation polygon. This is the most common case.

Case 2: This case covers all dilation arcs formed by the top edge of a dilation polygon which are not covered in Case 1. Case 2 arcs only appear if a top edge forms multiple arcs.

Case 3: This case is defined by the arcs generated by all lateral edges of a dilation polygon.

Case 4: This case covers all arcs a_i in P where the top edge of Q_i^{up} does not contribute an arc to P^{up}.

Now, assume that there is at least one arc a_j^{up} formed by the left lateral edge of Q_i^{up} that is not connected to an arc formed by the upper edge of Q_i^{up}. Moreover, no Q_l^{up} with $l > i$ forms an arc that lies further to the left than a_j^{up}. All such arcs formed by the lateral edges belong to Case 3. All arcs generated by the upper edge of Q_i^{up} are defined to be Case 2. We extend the definition of Case 4 by the cases where it is the first time that Q_i^{up} forms an arc in P^{up}, but not from its top edge.

Finally, assume that at least one arc in P^{up} is formed by Q_i^{up} but there exists a Q_l^{up} with $l > i$ that gives rise to arcs in P^{up} further to the left. Then, all arcs formed by the lateral and top edge of Q_i^{up} are added to Case 3 and Case 2, respectively. Moreover, Case 4 is applied before visiting the first arc of Q_l^{up}.

The triangulation is computed by iterating over the arcs a_j^{up} while simultaneously keeping track of the arcs a_i. To do so, let $i_{\text{last}} = 0$ be the last visited polygon among the Q_i^{up} and iterate over the arcs $a_j^{\text{up}}, j = 0, \ldots, k^{\text{up}}$. An example of the algorithm including the classification into the four cases is provided in Figure 11.5.

First, check from which arc a_i the current arc a_j^{up} originates.

If $i > i_{\text{last}}$, then the polygon Q_i^{up} is visited for the first time. In this case, proceed as follows. If $j = 0$ and $i > 0$, then $T\left(\alpha_0, \omega_0, \alpha_0^{\text{up}}\right)$ must be added first. If there are edges in P located between $a_{i_{\text{last}}}$ and a_i that do not generate edges in P^{up} prior to a_j^{up}, then add all triangles $T\left(\alpha_m, \omega_m, \alpha_j^{\text{up}}\right)$ for $m = i_{\text{last}} + 1, \ldots, i - 1$ (**Case 4**). Then, we have to differentiate whether a_j^{up} originates from the top, left or right edge of Q_i^{up}:

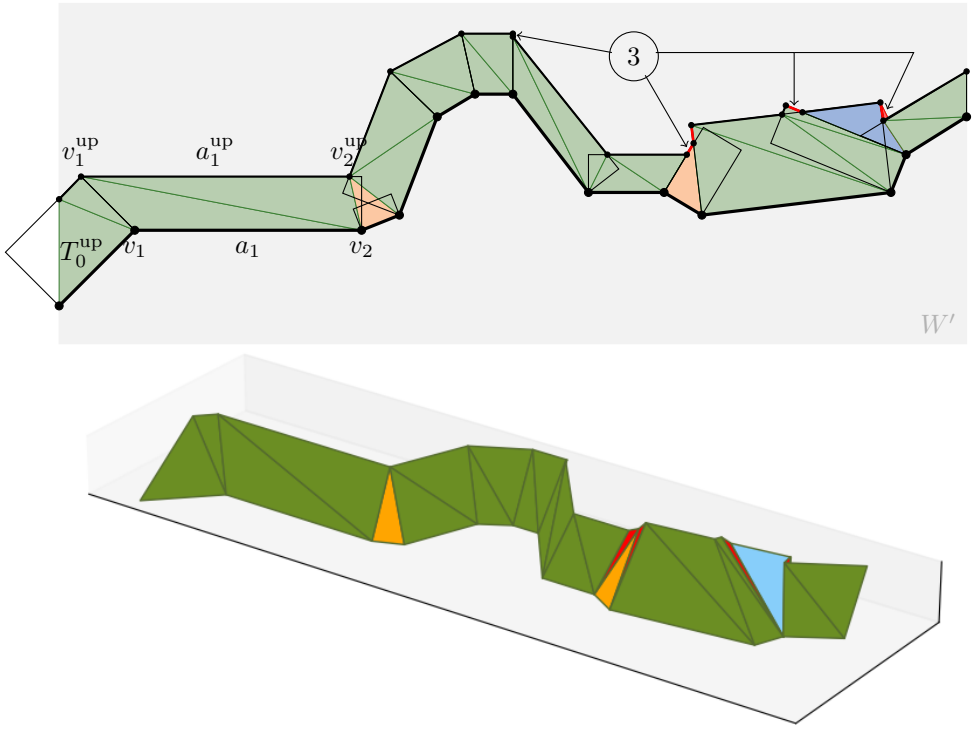


Figure 11.5: Illustration of triangulation with marking the four cases described in Section 11.2.1.4: 1. green, 2. blue, 3. red and 4. orange. Top: 2d drawing as projection on the x-y-plane. Bottom: 3d view with the path vertices deeper than the dilated path. Note that in 2d the blue triangle is partially and almost all red triangles are totally covered by the subsequent triangles. Therefore, the corresponding arcs of all Case 3 triangles are highlighted in the contour of the upper dilation.

Top: Add the triangles $T\left(\alpha_i, \omega_i, \alpha_j^{\text{up}}\right)$ and $T\left(\omega_i, \alpha_j^{\text{up}}, \omega_j^{\text{up}}\right)$ (**Case 1**). Update $i_{\text{last}} = i$.

Left: Add triangle $T\left(\alpha_i, \alpha_j^{\text{up}}, \omega_j^{\text{up}}\right)$ (**Case 3**). If the subsequent arc a_{j+1}^{up} is also a lateral edge of its corresponding polygon (left or right is irrelevant), further add triangle $T\left(\alpha_i, \omega_i, \omega_j^{\text{up}}\right)$ (**Case 4**) and update $i_{\text{last}} = i$. Otherwise, if a_{j+1}^{up} originates from a_i as well, meaning from the top edge of Q_i^{up}, then set $i_{\text{last}} = i - 1$ such that the triangles of Case 1 are added properly.

Right: If a_{j-1}^{up} is also a lateral edge of its corresponding polygon (left or right is irrelevant), first add triangle $T\left(\alpha_i, \omega_i, \alpha_j^{\text{up}}\right)$ (**Case 4**). Then, independently of a_{j-1}^{up}, triangle $T\left(\omega_i, \alpha_j^{\text{up}}, \omega_j^{\text{up}}\right)$ is added (**Case 3**). Update $i_{\text{last}} = i$.

If $i \leqslant i_{\text{last}}$, that is, polygon i was already visited, then just triangle $T\left(\omega_{i_{\text{last}}}, \alpha_j^{\text{up}}, \omega_j^{\text{up}}\right)$ is added, regardless of whether it is part of the top (**Case 2**), left or right edge (**Case 3**) of Q_i^{up}.

Finally, if $i_{\text{last}} < k$ after finalizing the iteration over j, then not all path elements of P were visited. To complete the triangulation, add all triangles $T\left(\alpha_i, \omega_i, \omega_{k^{\text{up}}}^{\text{up}}\right)$ for $i = i_{\text{last}} + 1, \ldots, k$ (**Case 4**).

As the 3d mesh of the defect has to be closed for proper imprinting or adding to the object surface, extra triangles and possibly also extra vertices are needed to obtain a closed surface triangulation of the defect geometry. Details are given in the description for the specific defects below.

11.2.2 Defect-Specific Adaptations

11.2.2.1 Cold Crack and Hot Tear

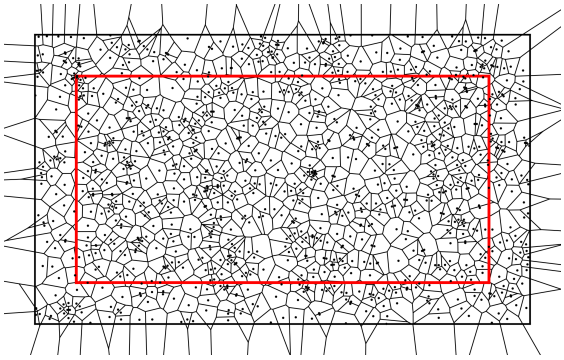
Cracks are caused by internal tensile stress during the cooling and solidification of the molten metal [2, 57, 93]. Two types of cracks are distinguished depending on the predominant temperature in the casting: hot tears occur before the solidification temperature is reached and cold cracks afterwards. The volume reduction during the cooling process starts at different locations and can thus cause solid material separations. Hot tears appear as a result of material shrinkage during solidification of the remaining melt. Cold cracks emerge if the stress of the solidified material exceeds its strength. Thus, cold cracks are trans-crystalline which means that metal grains can be destroyed. In contrast, hot tears are inter-crystalline.

Cracks are linear structures of irregular shape caused by taking the path of least resistance. Usually, cracks are deeper than wide with a symmetric depth profile when looking from the object surface. However, the width and depth within a crack can vary and additional branches can occur. Our model is parameterized so that all those properties are captured. In general, the model is divided into the following steps, which are illustrated in Figure 11.6:

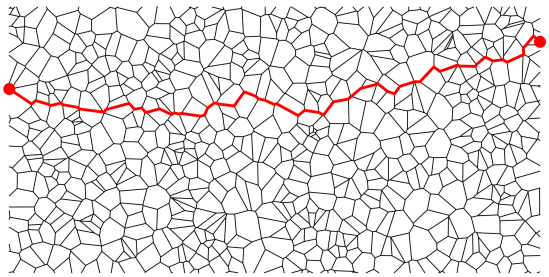
1. Choose a window W such that $w_0^{\max} - w_0^{\min}$ equals the desired expansion of the defect.
2. Generate n random points from a uniform distribution in the dilated window $\Delta_\gamma(W) = \left[w_0^{\min} - \gamma, w_0^{\max} + \gamma\right] \times \left[w_1^{\min} - \gamma, w_1^{\max} + \gamma\right]$ with $\gamma > 0$ and compute the corresponding Voronoi tessellation. The dilation is necessary to obtain regular cell shapes even at the window borders.
3. Choose a start and an end vertex at the left and right border of the window, respectively, and compute the associated minimal path.
4. Dilate the path using segment-wise width values l_i.
5. Assign height values $h_i \in (0, h]$ to the path vertices v_i that define the depth of the crack.
6. Compute the 3d surface triangulation.
7. Compute the mesh $C = (V, T)$ of the 3d crack, where V is the set of vertices and T the set of triangles. In addition to the triangles provided in Step 6, the crack needs a cover at the top and the sides to be a closed surface mesh. Therefore, add a triangulation T_{surface} of $\tilde{\delta}_Y(P)$ at the level of the object surface, i.e., at height h, with vertices $\left\{\left(v_0^{\text{up}}, h\right), \ldots, \left(v_{k^{\text{up}}+1}^{\text{up}}, h\right), \left(v_0^{\text{low}}, h\right), \ldots, \left(v_{k^{\text{low}}+1}^{\text{low}}, h\right)\right\}$. Note that extra vertices may be added for easier computation. To close the mesh, add triangles at the path ends via T_{end}. Thus, the mesh is defined by

$$V = \{(v_i, h_i)\}_{i=0,\ldots,k+1} \cup \left\{\left(v_i^{\text{up}}, h\right)\right\}_{i=0,\ldots,k^{\text{up}}+1} \cup \left\{\left(v_i^{\text{low}}, h\right)\right\}_{i=0,\ldots,k^{\text{low}}+1},$$

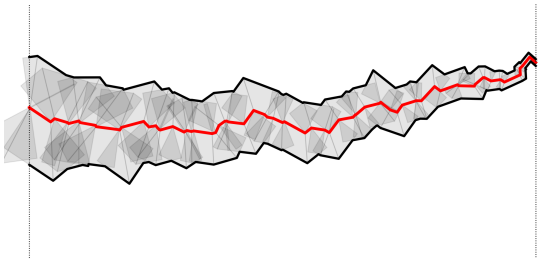
$$T = T_{\text{Step 6}} \cup T_{\text{surface}} \cup T_{\text{end}}.$$



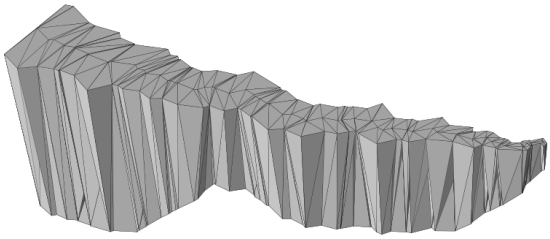
(a) Voronoi tessellation with its generator points in the dilated window (black), the original window is marked in red.



(b) Minimal path (red) between the start and end vertex (circles) in the Voronoi tessellation restricted to the window.



(c) Dilated path (black) bounded to the window edges (dotted). Dilation polygons Q_i in gray.



(d) 3d mesh of the crack defect with surface triangulation.

Figure 11.6: Example of the crack defect generation according to the model steps using $n = 1000$ generator points and $\gamma = 0.1\left(w_0^{\max} - w_0^{\min}\right)$.

In addition to the values for width and depth, the number of generator points for the Voronoi tessellation influences the shape of the crack, see Figure 11.7. With an increasing number of generator points, the crack structure gets finer. However, details may be lost as the width increases.

Assignment of Width and Height Values So far, we assumed a segment-wise specification of width and height values. Alternatively, a global value or a sequence of values can be provided. We introduce the procedure for the width values but it can directly be transferred to the height values.

Let $(\tilde{l}_0, \dots, \tilde{l}_{\tilde{k}})$ be a sequence of arbitrary width values $\tilde{l}_i \in \mathbb{R}_{>0}$, $i = 0, \dots, \tilde{k}$. Note that the sequence must not be longer than the path, that is $0 \leqslant \tilde{k} \leqslant k$. Then, if a global width value is chosen, so $\tilde{k} = 0$, it holds $l_i = \tilde{l}_0$ for all $i = 0, \dots, k$. Contrary, if every segment is assigned an individual value, so $\tilde{k} = k$, then it holds $l_i = \tilde{l}_i$ for all i. Otherwise, if $0 < \tilde{k} < k$, we assign $\tilde{l}_i$ approximately to every $\lfloor k/\tilde{k} \rceil$-th path segment, that is

$$l_{\lfloor (i/\tilde{k}) \cdot k \rceil} = \tilde{l}_i$$

for $i = 0, \dots, \tilde{k}$. The remaining values are obtained by linear interpolation.

Branching Branching is a common crack property and should thus be included in the model. In the following, we describe the procedure to include one branch, but it can easily be extended to an arbitrary number of branches. Let $C = (V_C, T_C)$ be the mesh of a crack

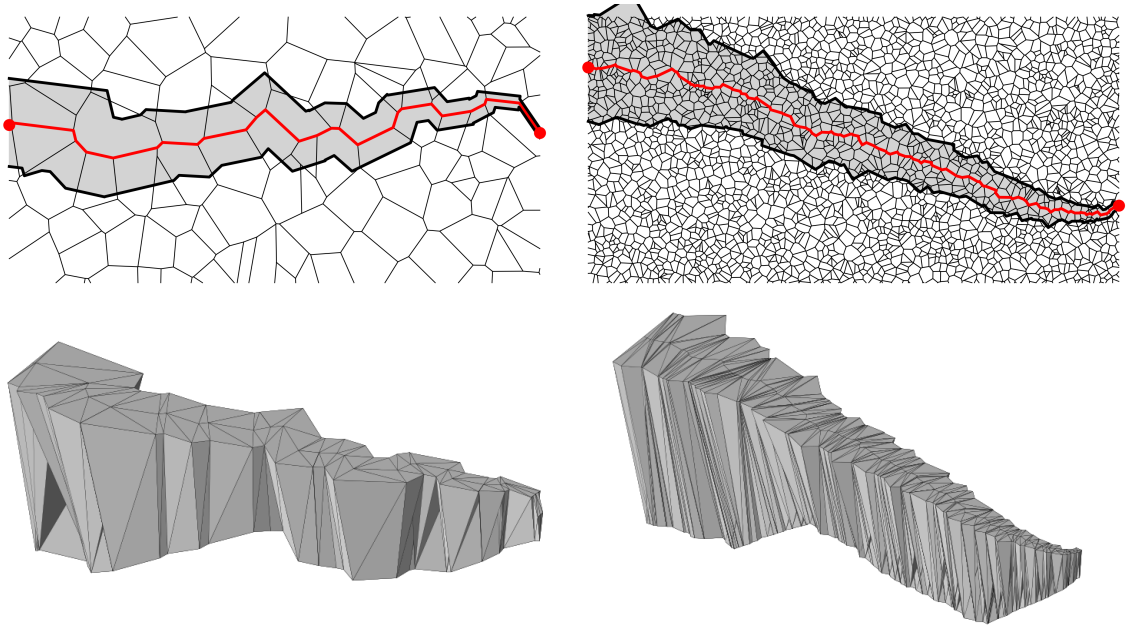


Figure 11.7: Crack defects using different numbers of generator points. Left: $n = 150$, right: $n = 2000$. Top: Voronoi tessellation with minimal path (red) and its dilation (gray). Bottom: 3d mesh of the defect.

defect. First, choose a random Voronoi vertex on the crack path and use it as the starting point of the branch. When considering more than one branch, the start vertices may also be chosen from the previously generated branches. Next, choose an arbitrary Voronoi vertex outside the path as the end point of the branch. Use the same Voronoi tessellation as for generating the crack and compute the minimal path. Then, proceed with Steps 4-7 of the crack simulation procedure to obtain a branch $B = (V_B, T_B)$. Note that the starting width and depth of the new branch should not exceed those of the associated vertex on the crack branch. To compute the dilation, the branch path is rotated such that its start and end vertex are aligned with the x-axis followed by reversing the rotation afterwards. To connect crack and branch, the triangles of $T_C \cup T_B$ have to be re-meshed to form a valid surface triangulation of $C \cup B$, see Figure 11.8 for an example. For this purpose, we use the Boolean union function of the Python package TRIMESH [19].

11.2.2.2 Bulge

Bulges are material accumulations caused by indents in the casting mold and appear as local pyramidal-shaped thickenings of the object surface [72]. They resemble a reverse crack, but with larger width and smaller depth. Therefore, we reuse the crack model from Section 11.2.2.1 with an adaptation in Step 6 to generate positive defects, that is $h_i > h$. Furthermore, the width values should be chosen so that the resulting dilation is roughly eye-shaped. See an example of a synthetic bulge in Figure 11.9a.

11.2.2.3 Buckle

Buckling describes a local lift-off of the top layer of the object surface caused by insufficient bond strength [72, 93]. It can take different forms: linear buckling of small width and long length or pyramidal shaped lift-offs. Furthermore, a buckle can be closed or open. The latter means that the layer is torn off.

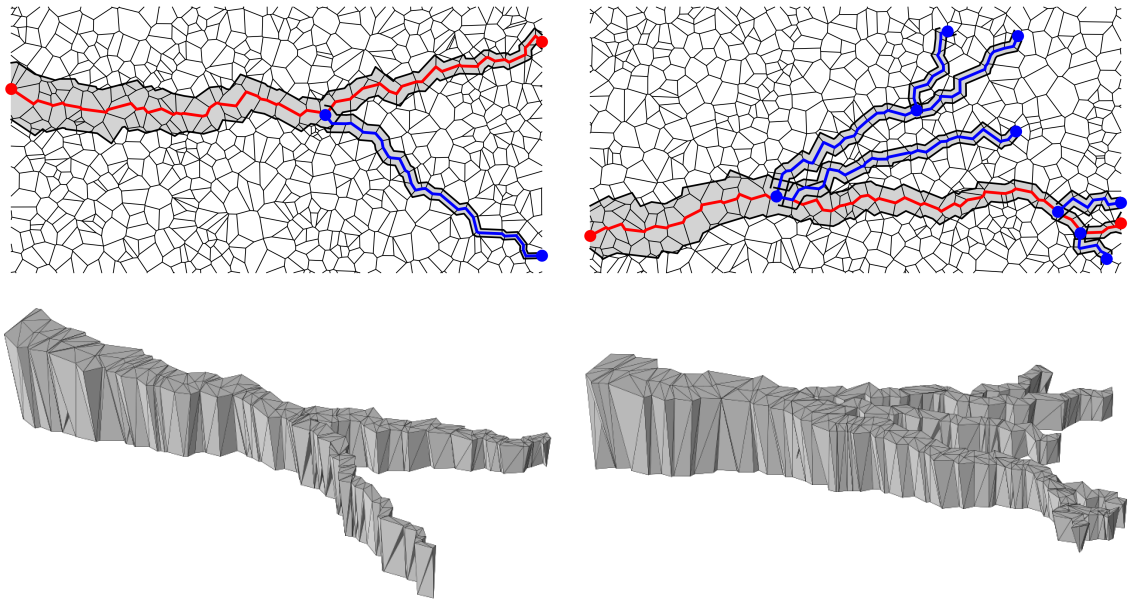
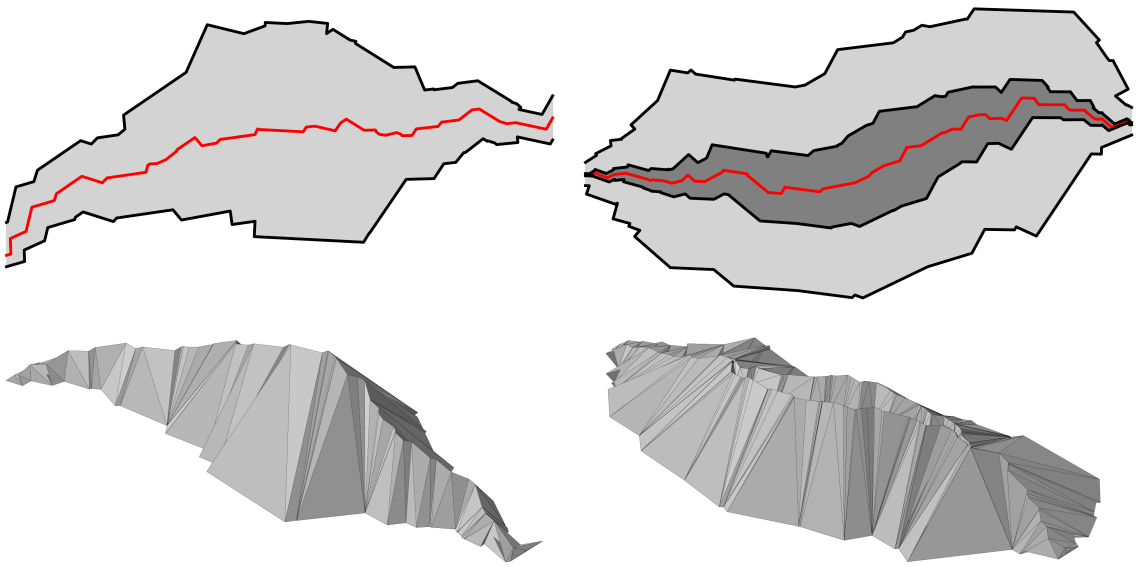


Figure 11.8: Crack with one branch (left) and five branches (right). Top: Voronoi tessellation with path of the main crack in red and the paths of the branches in blue. Associated start and end vertices are marked in the corresponding color. The dilations are colored in gray. Bottom: 3d mesh of the defect.



(a) Bulge. Coloring of the top image: Minimal path in red and its dilation in gray.

(b) Buckle. Coloring of the top image: Minimal path in red, its small dilation in dark gray and its big dilation in light gray.

Figure 11.9: Examples of bulge and buckle defects. Top: Visualization of the 2d model steps. Bottom: 3d mesh of the object.

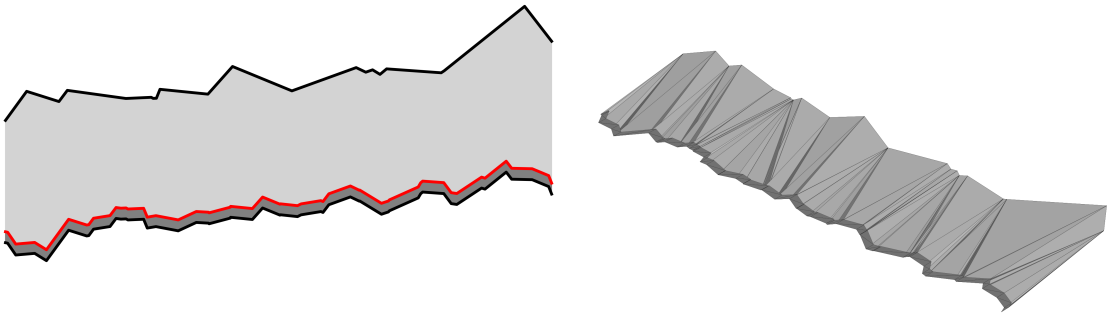


Figure 11.10: Example of coat lift defect. Left: Visualization of the 2d model steps with the minimal path in red, its one-sided dilation in light gray and its shift in dark gray. Right: 3d mesh of the object.

A closed buckle can be modeled using the same approach as for bulges when choosing appropriate parameter configurations. Open buckles are modeled using a Boolean subtraction of a bulge-like and a crack-like defect. Therefore, let B_P be a positive defect shape generated by the bulge model described in Section 11.2.2.2 and B_N be a negative defect shape generated by the crack model in Section 11.2.2.1 generated under the following constraints:

- Both B_P and B_N are modeled using the same underlying minimal path.
- Let l_i^P and l_i^N be the widths of the positive and negative defects, respectively. We require $l_i^P > l_i^N$ for all i, so that the width of B_P is greater than that of B_N for every arc in the path.
- The depth of the crack should be smaller than the height of the bulge such that the buckle is never deeper than the object surface.

Then, an open buckle is computed by $B_P \backslash B_N$. To perform that operation, re-meshing of the surface triangulation is necessary. Again, the Boolean difference function given in the TRIMESH package for Python can be used [19]. An exemplary defect geometry is given in Figure 11.9b.

11.2.2.4 Coat Lift

Coat lifts are caused by buckles in the casting shell. Its primary layer lifts off and cracks, allowing melt to fill the space created [72]. Sertucha and Lacaze [93] describe lustrous carbon inclusions, which are surface defects that look similar, but arise in different ways. Coal is a common component of sand molds. Consequently, carbon films may form on the inner mold surface, which can then be enclosed by the metal melt. This results in protruding metal sheets after solidification.

Coat lifts are therefore one-sided elevations on the surface of the part. Both the contours on the object surface and on the elevated side are jagged. Hence, the model uses the same basis as the previous defect models, a minimal path through a Voronoi tessellation, but the subsequent steps differ, see Figure 11.10 for illustration.

1.-3. Steps 1-3 are equal to crack modeling in Section 11.2.2.1.

4. Compute the upper dilation of the path P with contour P^{up}.

5. Compute a lower shift P^{low} of the path P with vertices $v_i^{\text{low}} = v_i - \left(0, l^{\text{low}}\right)$ using a constant width $l^{\text{low}} > 0$. That width models the thickness of the elevated metal layer.

6. Assign the object surface height to the vertices of P and P^{low}, namely $h_i = h_i^{\text{low}} = h$, and heights $h_i^{\text{up}} \geqslant h$ to the vertices of P^{up}.

7. Compute the triangulation T_{up} of the vertices belonging to P and P^{up}, that are $\left\{(v_0, h), \ldots, (v_{k+1}, h), \left(v_0^{\text{up}}, h_0^{\text{up}}\right), \ldots, \left(v_{k^{\text{up}}+1}^{\text{up}}, h_{k^{\text{up}}+1}^{\text{up}}\right)\right\}$, to connect the elevated contour of the upper dilation and the path as described in Section 11.2.1.4. The triangulation T_{low} of the area between P and P^{low} is given by the triangles

$$\left\{T\left(v_i, v_{i+1}, v_i^{\text{low}}\right)\right\}_{i=0,\ldots,k} \cup \left\{T\left(v_{i+1}^{\text{low}}, v_i^{\text{low}}, v_{i+1}\right)\right\}_{i=0,\ldots,k}.$$

8. Compute the mesh of the coat lift $C = (V, T)$ with

$$V = \{(v_i, h)\}_{i=0,\ldots,k+1} \cup \left\{\left(v_i^{\text{up}}, h_i^{\text{up}}\right)\right\}_{i=0,\ldots,k^{\text{up}}+1} \cup \left\{\left(v_i^{\text{low}}, h\right)\right\}_{i=0,\ldots,k^{\text{low}}+1},$$
$$T = T_{\text{up}} \cup T_{\text{low}} \cup T_{\text{cover}} \cup T_{\text{end}},$$

where T_{cover} connects the vertices of P^{up} and P^{low} to form the top coverage of the defect. Triangles at the ends of the path are again added via T_{end}.

11.2.2.5 Cold Shut and Rat Tail

Aside from cracks, there exist other types of linear defects that have also an irregular path but round edges instead of a jagged behavior. Cold shuts appear if two molten metal streams cannot merge before solidifying. This happens if the melt is not fluid enough, which can be caused by low temperature [2, 57, 72, 93]. Furthermore, rat tails arise from defected shells. For example, a crack in the mold that is too thin to be filled with melt contains air and causes oxidation of the hot metal, which forms an indentation in the part. While cold shuts commonly appear individually, rat tails occur accumulated. Moreover, rat tails are more curved.

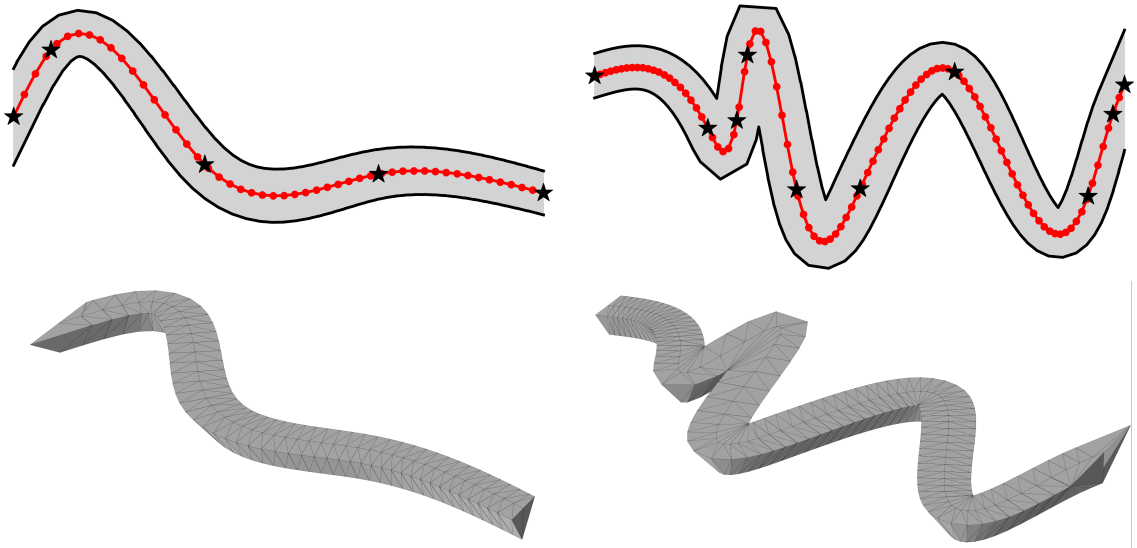
Although the causes of both types of defects are different, their appearance is similar. Thus, a joint model can be used by choosing suitable parameter configurations. The model is related to the crack model, but includes round edges. Therefore, instead of the piece-wise linear path in the Voronoi tessellation, we use a natural cubic spline interpolation of randomly sampled nodes. The model is then defined by the following steps, see Figure 11.11 for an illustration.

1. Choose a window W such that $w_0^{\max} - w_0^{\min}$ equals the desired linear expansion of the defect.

2. Generate $n+1$ random points $p_0, \ldots, p_n$ from a uniform distribution on W and sort them according to their x-values. Generate the points such that the first and the last are located on the left and right border of the window, that is, $(p_0)_0 = w_0^{\min}$ and $(p_n)_0 = w_0^{\max}$.

3. Compute a natural cubic spline interpolation using $\{p_0, \ldots, p_n\}$ as control points. Discretize the curve by computing points $v_0, \ldots, v_{k+1}$ with $k+1 \gg n$ located on the curve with $v_0 = p_0$ and $v_{k+1} = p_n$. Define a path by arcs a_i with $\alpha_i = v_i$ and $\omega_i = v_{i+1}$ for $i = 0, \ldots, k$.

4.-7. Steps 4-7 are equal to crack modeling in Section 11.2.2.1.



$n + 1 = 5$, $k + 2 = 50$. $n + 1 = 10$, $k + 2 = 100$.

Figure 11.11: Realization of the model for cold shuts and rat tails using different parameter configurations. Top: Visualization of the generating procedure of the 2d shape with control points (stars), interpolation with discrete points (red) and the dilated path (gray). Bottom: 3d mesh of the object.

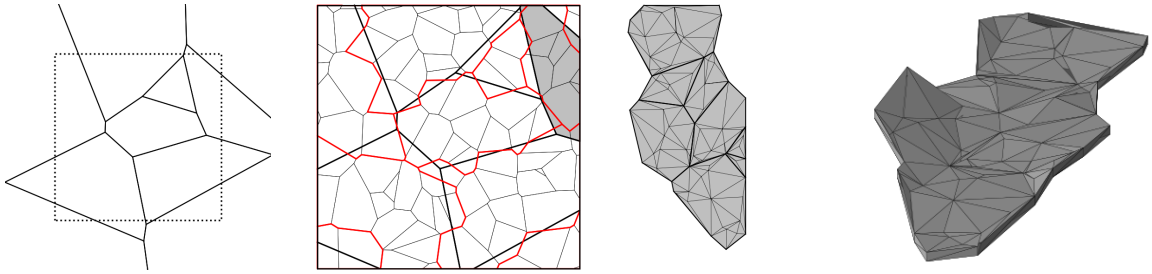
11.3 Model for Scab Defects

Scabs usually occur if the top layer of the mold is defected such that molten metal can fill the space between the layers similarly to coat lifts [93]. In [72], this type of defect is called delamination because the top layer of the mold is delaminated. In general, delaminations appear when the material breaks into layers. For the purpose of this work, we only consider surface delaminations, that is, the breaking of the top surface layer into individual pieces whose outer regions can be elevated. Surface delaminations can occur individually, in clusters or as surface-filling defects that cover the entire object surface. Total surface fracturing rather commonly occurs for coated parts, but is unusual for cast metal products. We propose a general approach that is applicable for modeling scabs (similar to individual delaminations), clustered and surface-filling delaminations.

Again, we use Voronoi cells for the simulation, which cover the whole surface by construction. Our model creates textures of delaminated surfaces by generating cells on the object surface whose outer regions may be elevated. Since these parts result from breaking of the surface layer, the edges of neighboring cells must fit together. Sometimes, just certain parts of the surface are affected by delaminations, for example, in scabs. In this case, the delamination model can be restricted to a subset of the delamination cells. We first outline the main steps of the model generation procedure, see also Figure 11.12. Details are explained in the subsequent sections. Figure 11.13 shows synthetic images of the different delamination types rendered with parameters of a realistic imaging setup.

1. Select an observation window W that represents the part of the object surface containing the delamination.

2. Compute a Voronoi tessellation in the dilated window to obtain the individual delamination cells. The dilation is necessary to obtain regular cell shapes even at the



Voronoi tessellation in a delaminated window. | Border refinement of individual cells. | Triangulation. | Assignment of height values and 3d mesh.

Figure 11.12: Schematic overview of the model for scab and surface delamination defects.

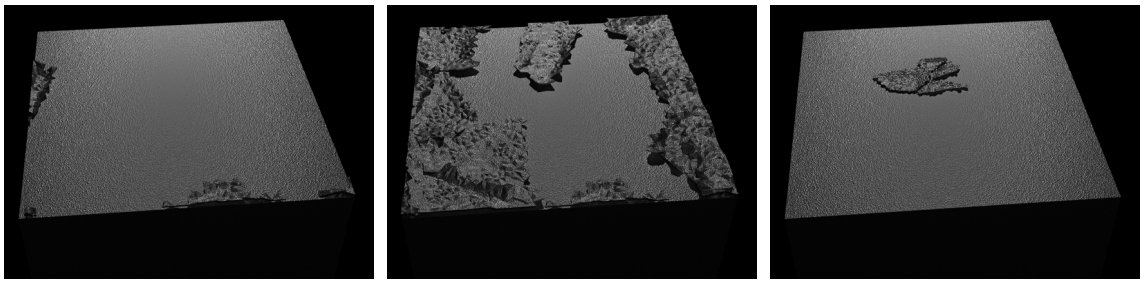


Figure 11.13: Synthetic images of delaminated surfaces of a cast metal cube rendered as described in Figure 11.3.

window borders.

3. Refine the cell borders by using a finer Voronoi tessellation and choose how many and which cells should be considered further.

4. For all chosen delamination cells:

   4.1. Triangulate the refined cell using additional vertices.

   4.2. Assign height values to all vertices.

   4.3. Compute the 3d mesh.

11.3.1 Voronoi Tessellation in the Dilated Window

We call the Voronoi tessellation that defines the individual delamination cells the coarse tessellation. Let $\mathscr{P}^c = \{p_1^c, \dots, p_{n^c}^c\}$ be a set of generator points with p_i^c uniformly sampled in the dilated window $\Delta_\gamma(W)$. Let $D_i^c = C_i^c \cap W$ be the restriction of cell C_i^c to the window. The number of vertices of D_i^c is given by $\#D_i^c$. As some generator points can be located outside W, each nonempty cell D_i^c is assigned its center point

$$c_i^c = \frac{1}{\#D_i^c} \sum_{v \text{ vertex of } D_i^c} v$$

as reference points in W. Let $m^c \leqslant n^c$ be the number of remaining cells. An example is given in Figure 11.15a.

11.3.2 Refinement of Cell Borders

Generate a finer Voronoi tessellation in the same manner as described in Section 11.3.1 but with $n^f \gg n^c$ generator points. Choose reference points c_j^f as described above and define the adjusted reference points

$$\tilde{c}_j^f = \begin{cases} p_j^f & \text{if } p_j^f \in W \\ c_j^f & \text{else} \end{cases}$$

for $D_j^f \neq \varnothing$. To refine the borders of the coarse cells, pick all fine cells whose adjusted reference point is located in the coarse cell, that is, for cell D_i^c, $i \in \{1, \ldots, m^c\}$,

$$\mathscr{D}_i = \left\{ D_j^f : \tilde{c}_j^f \in D_i^c, j = 1, \ldots, m^f \right\}.$$

An example is given in Figure 11.15b.

11.3.3 Triangulation of Fine Cells

Let D be a fine cell with vertices $v_1, \ldots, v_K$ in clockwise order with $v_{K+1} = v_1$ and center point c. To obtain a triangulation of D, additional points are added depending on the relative area of the cell. The number of newly added points is given by

$$R = \left\lfloor \frac{\text{area}(D)}{\max_{j=1,\ldots,m^f} \text{area}\left(D_j^f\right)} \cdot R_{\max} \right\rceil,$$

where $R_{\max} \in \mathbb{N}$ is the number of new points for the largest cell in the fine Voronoi tessellation. The computation of the triangulation $T^{2\text{d}}$ differs depending on R, see also Figure 11.15c for an example.

$\mathbf{R = 0}$: Compute a triangulation of the points $V^{2\text{d}} = \{v_1, \ldots, v_K, c\}$ by $T^{2\text{d}} = \bigcup_{k=1}^{K} T(v_k, v_{k+1}, c)$.

$\mathbf{R > 0}$: Generate new points $u_r, r = 1, \ldots, R$, from a uniform distribution on D. Then, compute the Delaunay triangulation of the points $V^{2\text{d}} = \{v_1, \ldots, v_K, c, u_1, \ldots, u_R\}$, that is, no further points lie in the circumcircle of a triangle.

11.3.4 Assignment of Height Values

Height values $H(v)$ are assigned to all vertices v in the triangulation of $\mathscr{D}_i$, that is the union of triangulations of the associated fine cells, by

$$v \mapsto h + H_{\text{elev}}(v) + H_{\text{tex}}(v)$$

where $H_{\text{elev}}(\cdot)$ determines the amount of elevation dependent on the distance to the boundary of D_i^c and $H_{\text{tex}}(\cdot)$ gives the opportunity to add a surface texture depending on the fine cell that v is located in, see also Figure 11.15d. To define both height functions, let $b(p, D) = \partial D \cap \{\lambda(p - c) : \lambda > 0\}$ be the intersection of the border of cell D and the half-line starting in the center point c that passes through an arbitrary point $p \in \mathbb{R}^2$. See Figure 11.14 for an illustration. The height in v will depend on the distance between $b(v, D)$ and v as described in the following sections.

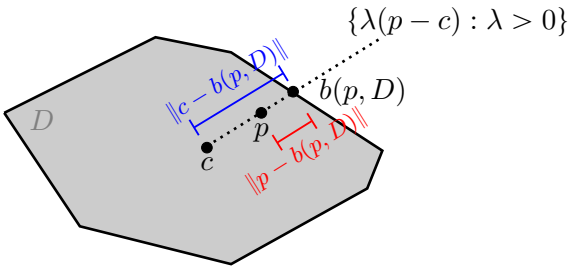


Figure 11.14: Illustration of intersection $b(p, D)$ and computation of $\mathrm{dist}_{\mathrm{rel}}(p, \partial D)$.

11.3.4.1 Elevation of Outer Areas $H_{\mathsf{elev}}(\cdot)$

This height models the defect-specific shape, that is, the outer part of a cell $\mathscr{D}_i$ is elevated while its center is not, so that the defect remains connected to the object surface. To ensure a natural appearance of the defect, the maximum elevation heights at the boundary of $\mathscr{D}_i$ and the diameter of the non-elevated area in the center of the cell can vary for different directions in $\mathscr{D}_i$.

The size of the non-elevated area in direction of a vertex $v \in \mathscr{D}_i$ is characterized by the distance $\|v - b(v, D_i^c)\|$. Since that distance varies when traversing over the cell's contour, we use the relative distance

$$\mathrm{dist}_{\mathrm{rel}}(p, \partial D_i^c) = \frac{\|p - b(p, D_i^c)\|}{\|c_i^c - b(p, D_i^c)\|} \begin{cases} \leqslant 1 & \text{if } p \in D_i^c \\ > 1 & \text{else} \end{cases}$$

for arbitrary $p \in \mathbb{R}^2$. Note that $\mathrm{dist}_{\mathrm{rel}}(v, \partial D_i^c) > 1$ may occur because $\mathscr{D}_i \nsubseteq D_i^c$. Vertex v will be elevated if $\mathrm{dist}_{\mathrm{rel}}(v, \partial D_i^c)$ exceeds a certain threshold $d(v) \in [0, 1]$. Thus, all vertices $v \notin D_i^c$ are elevated for sure.

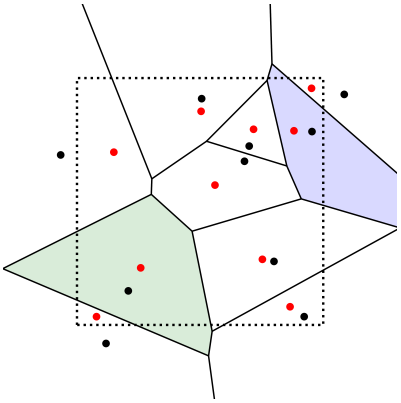
By setting $H_{\mathrm{elev}}(v) = 0$ for all vertices with $\mathrm{dist}_{\mathrm{rel}}(v, \partial D_i^c) \leqslant d(v)$, all such vertices stay at the height of the object surface. Height values for all points on the contour of $\mathscr{D}_i$ must be given to define the maximal elevation. Then, all vertices located on the contour are assigned these maximal height values. The heights of the remaining intermediate vertices between the non-elevated area and the contour are obtained by linear interpolation.

To assign the maximal height values to the points on the contour, let $(h_0, \ldots, h_k)$ be a sequence of intermediate maximal height values $h_j \in \mathbb{R}_{\geqslant 0}$ for $j = 0, \ldots, k$ reached at certain points on the contour. To calculate those points we have to switch to polar coordinates. Assume that $\mathscr{D}_i$ is shifted so that it is centered in $(0, 0)$. Then, h_j is reached at $p = \rho^{-1}(r, \varphi) \in \partial\mathscr{D}_i$ with $\varphi = j\frac{2\pi}{k+1}$ for $j = 0, \ldots, k$. All intermediate values on $\partial\mathscr{D}_i$ are determined using linear interpolation based on their angle.

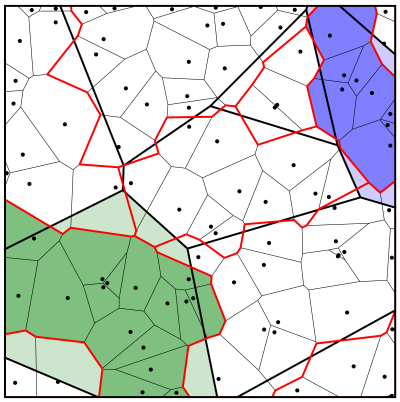
11.3.4.2 Texture Height $H_{\mathsf{tex}}(\cdot)$

This height component gives the opportunity to assign a specific texture to the fine cells by individually elevating them using local bulges with maximal height in the center points of the cells. Assume that a vertex v is contained in the fine cell D_j^f and choose a maximal height $t_j \in \mathbb{R}_{\geqslant 0}$ reached at c_j^f. The height at v then depends on its relative distance to the center point c_j^f and is thus defined via

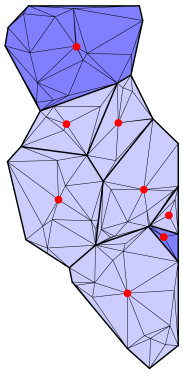
$$H_{\mathrm{tex}}(v) = t_j \cdot \frac{1}{2}\left(\cos\left(\pi \frac{\left\|c_j^f - v\right\|}{\left\|c_j^f - b\left(v, D_j^f\right)\right\|}\right) + 1\right) \in [0, t_j].$$



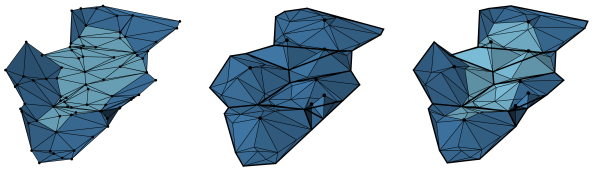
(a) Procedure to restrict the Voronoi tessellation to the window (dotted line). Generator (black) and reference points (red) are marked.



(b) Coarse and fine Voronoi tessellation are represented by bold and thin black lines, respectively. Adjusted reference points of the fine tessellation are marked. Borders of $\mathscr{D}_i$ are illustrated in red. Two cells are highlighted to mark the change from D_i (light) to $\mathscr{D}_i$ (dark).

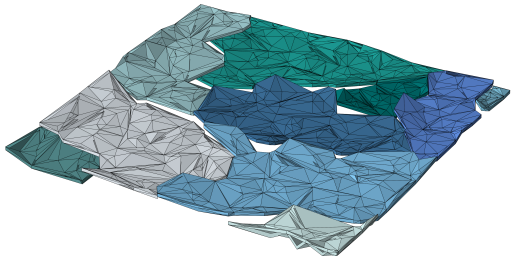


(c) Triangulation of $\mathscr{D}_i$ using $R_{\max}$=12. The upper highlighted cell gets $R = 11$ new points while for the lower highlighted cell we have $R = 0$. Reference points are marked in red.



H_{elev} H_{tex} $H = H_{\text{elev}} + H_{\text{tex}}$

(d) Assignment of height values by the composition into elevation and texture. Dark blue marks the elevated triangles.



(e) All delamination cells of the Voronoi tessellation as separate 3d volumes.

Figure 11.15: Visualization of the pipeline to generate a delaminated surface or scabs with $n^c = 10$, $n^f = 110$ and a squared window with dilation width $\gamma = 0.1\left(w^{\max} - w^{\min}\right)$.

Note that also other shape functions may be used.

11.3.5 Computation of 3d Mesh

To generate the 3d mesh of the delamination cell determined by $\mathscr{D}_i$, its vertices and triangles are duplicated first to define the lower and the upper layer. In the first copy, the vertices get height values $h + H_{elev}(v)$. The texture height is removed such that the non-elevated cell parts are smoothly connected to the object surface. The second copy of the vertices gets the height values $H(v) + l$ where $l \in \mathbb{R}_{>0}$ is the thickness of the delaminated surface layer. Finally, additional triangles are added to connect the upper and lower layer, similar to the coat lifts described in Section 11.2.2.4. Examples of 3d meshes of delaminated cells are given in Figure 11.15e.

11.4 Conclusion

We presented parametric models to generate a variety of different surface defects of cast metal objects. We use Voronoi tessellations to simulate the fundamental defect behavior. Thus, the poly-crystalline material structure is included, but on a coarse scale to reduce the model complexity compared to physically correct defect models. This enables fast computation times for both the individual defect geometries and their addition to the product geometry.

Defect variability is guaranteed by using different parameter configurations. Providing real parameter ranges increases the realism of defect shapes. Moreover, defect edge case scenarios are covered by using appropriate parameter configurations. Thus, a diverse and arbitrarily large synthetic data set can be generated. In addition, particular defect features can be varied individually to fine-tune defect detection algorithms. This provides a great advantage over AI-based image generation methods, where systematic data generation is difficult. Automatic data annotation is also provided.

Although our models are motivated by defect geometries for cast metal objects, they can be transferred to other manufacturing processes. For example, cracks are frequently occurring defects in various applications and delaminations appear commonly on painted surfaces. Synthetic data generation is also not limited to visual surface inspection. Placing the digital twins of the inspected object into another virtual inspection environment can provide synthetic data for any NDT method.

12 Adaptations and Expansions of Certain Casting Defect Models

In the following we present different model adaptations such as using other point process types to generate the Voronoi tessellation that is used to determine the path of a crack defect, see Section 12.1, and an alternative approach to model buckle defects in Section 12.2. Moreover, the proposed models are used and expanded to simulate progressive defecting of cracks and buckles at different time steps, see Section 12.3.

12.1 Model Adaptation to Other Point Process Types for Generating the Voronoi Tessellation

So far, we used Voronoi tessellations generated with an underlying binomial point process for computing the path which determines the basic shape of a crack defect. To increase the variability of the underlying paths, different point process types may be used for generating the Voronoi tessellation, see Section 3.1.7 for the definitions of Matérn cluster and Matérn hardcore point processes. Figure 12.2 and Figure 12.3 show examples of the resulting crack structures using those types of point processes with varying parameters. These realizations can be compared to crack structures based on Poisson point processes of different intensities shown in Figure 12.1.

Independent of the point process type, the number of path vertices obviously increases with a rising number of generator points, see Figure 12.4a and Figure 12.4b for comparison. However, Matérn cluster processes have big variations in the number of points and also in the cell size distribution which both influence the number of path vertices. As all tessellations are computed in the same window, the crack has to migrate over the same distance. Thus, one can expect a path generated from a Matérn cluster process to have a larger variance in the length of its path segments than the paths obtained from the other point process types. This assumption is confirmed by Figure 12.4c when comparing the estimated distribution of the paths' segment lengths. The uneven distribution of segment lengths for paths originating from Matérn cluster processes is especially apparent for a large number of generator points. However, the large variance in segment lengths is not necessarily present in every realization but depends on the individual clusters within a particular simulation. In contrast, the most uniform segment lengths are obtained when applying Matérn hardcore processes due to similar cell sizes in the resulting Voronoi tessellation. The number of long segments decreases rapidly with increasing intensity. Additionally, the box plots are overall thinner indicating a smaller variance. Poisson point processes produce Voronoi tessellations whose variation in cell sizes is bigger compared to Matérn hardcore processes but smaller compared to Matérn cluster processes. Thus, the distribution of segment lengths behaves similarly.

To summarize, certain path features are influenced by the underlying point process type and its intensities which can thus be chosen depending on desired characteristics on the cracks. If, for example, a crack should avoid a certain region on the object surface, a Matérn cluster process having no cluster in that area may be the best option. Otherwise,

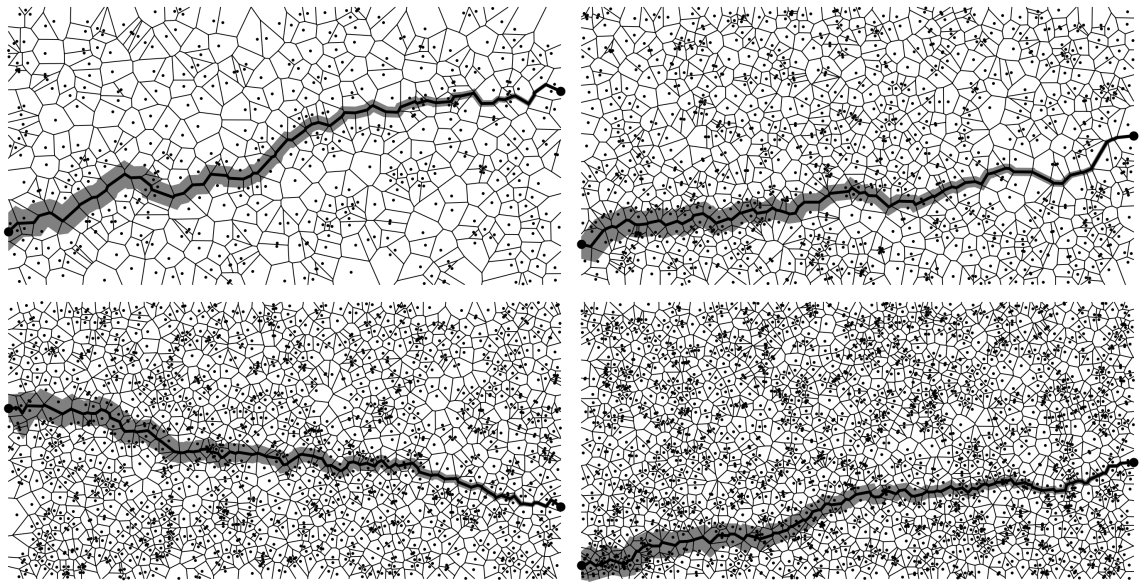


Figure 12.1: Voronoi tessellations with 2d cracks based on Poisson processes with growing intensity $\lambda = 250, 500, 750, 1000$ from top left to bottom right. Window size is 2×1.

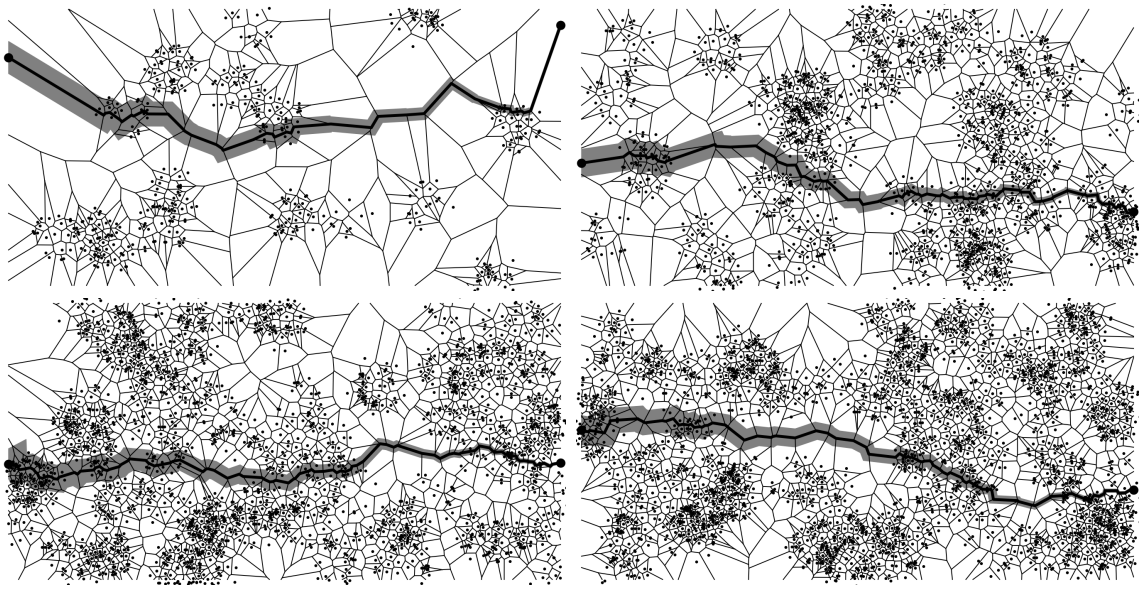


Figure 12.2: Voronoi tessellations with 2d cracks based on Matérn cluster processes with growing intensity $\lambda_{\text{parent}} = 10, 20, 30, 40$ of the underlying parent process from top left to bottom right. The remaining parameters are chosen as $\lambda_{\text{child}} = 25$ and $r = 0.1$. Window size is 2×1.

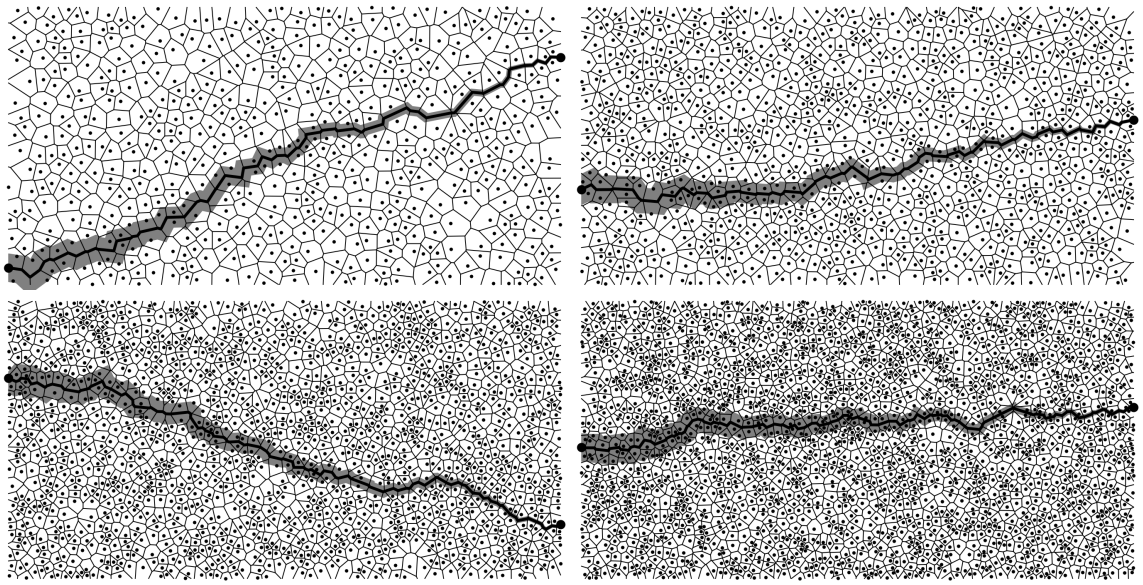


Figure 12.3: Voronoi tessellations with 2d cracks based on Matérn hardcore processes of type II with growing intensity $\lambda = 700, 1200, 1700, 2200$ of the underlying Poisson process and decreasing distance $r = 0.033, 0.024, 0.019, 0.016$ from top left to bottom right. Window size is 2×1.

a Matérn hardcore process can be helpful to generate very regular cracks. Furthermore, if the synthetic crack is to be imprinted into an object with a poly-crystalline micro-structure, knowledge about the material-specific structure can be integrated by selecting an appropriate point process type so that the resulting tessellation reflects the material micro-structure best possible.

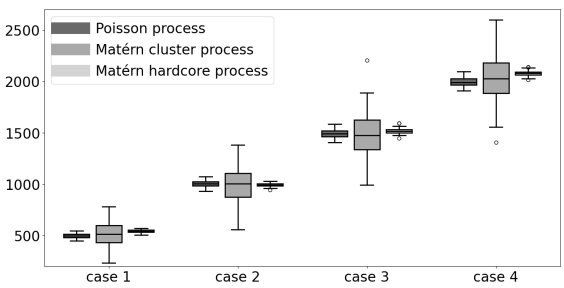
12.2 Alternative to Model Buckle Defects

As described in Section 11.2.2.3, an open buckle defect is modeled as Boolean difference of a crack defect geometry from a bulge defect geometry. Thus, a volumetric surface accumulation with open ridge results. However, buckles are defined as local, pyramidal-shaped lift-offs of the top layer which might be open in the central part. Usually, the elevated material resembles thin metal sheets instead of solid metal accumulations as it is modeled currently. Therefore, a simulation for open buckle defects based on the model for coat lifts provides a promising alternative.

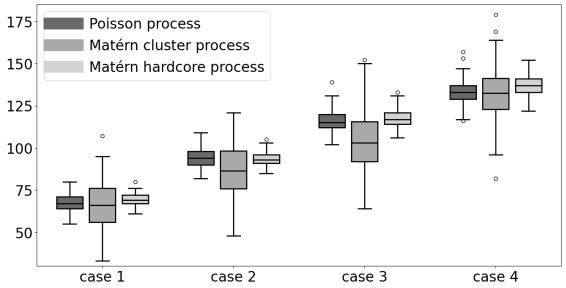
In the following, the model's individual steps are described in detail. See also Figure 12.5 for a visualization of the required structures and the resulting geometry of a buckle defect.

1.-3. Steps 1-3 are equal to crack modeling in Section 11.2.2.1. Let P be the generated path which represents the center line of the open ridge.

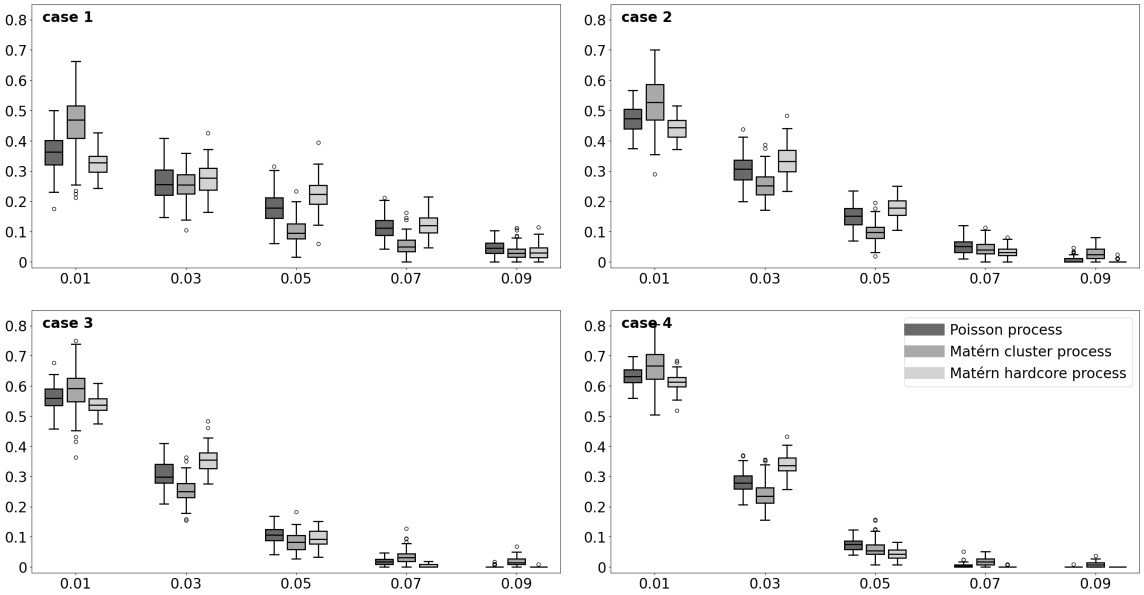
4. **Edges of open part**
   Dilate path P using segment-wise width values l_i^{open} to obtain the paths $P_{\text{open}}^{\text{up}}$ and $P_{\text{open}}^{\text{low}}$ describing the upper and lower contour of the dilation $\tilde{\delta}_Y(P)$, respectively. The width values at the beginning and end of the path should be small as the buckle defect usually is not yet opened widely, if at all, in those areas.



(a) Number of generator points.

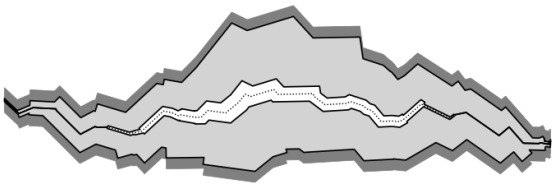


(b) Number of path vertices.

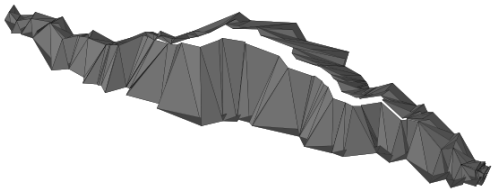


(c) Relative frequencies of the path segment lengths.

Figure 12.4: Comparison of different properties of Poisson, Matérn cluster and Matérn hardcore type II processes and the resulting crack paths. The cases are defined by the same parameter choices as for generating Figure 12.1, Figure 12.2 and Figure 12.3. Each parameter configuration was performed 100 times. Boxes mark the first and third quartiles with median in black. Whiskers extend to the farthest point within the 1.5×inter-quartile range starting at the box. Circles mark the outliers.



2d shape of the alternative simulation of a buckle defect with the basic path P (dotted line), its dilation $\tilde{\delta}(P)$ in white, the upper and lower dilations $\tilde{\delta}^{\text{up}}\left(P^{\text{up}}_{\text{open}}\right), \tilde{\delta}^{\text{low}}\left(P^{\text{low}}_{\text{open}}\right)$ in light gray and the path shifts in dark gray.



3d shape represented as surface mesh.

Figure 12.5: Visualization of the alternative simulation of buckle defects.

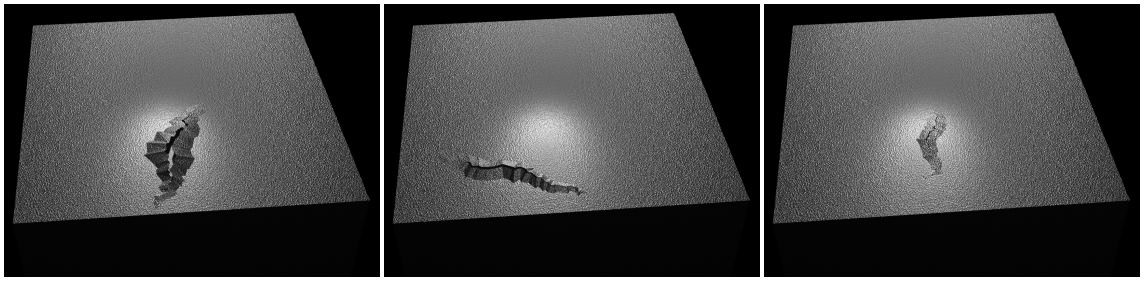


Figure 12.6: Synthetic images of buckle defects in a cast metal object rendered in Blender [1]. The alternative approach is used to simulate buckle defects. A pinhole camera and three plane lights are used for illumination. Rendering is performed using 30 spp and 12 light bounces. The surface texture is chosen to resemble the appearance of a cast metal object. The texture model was implemented by Lovro Bosnar from Fraunhofer ITWM and is based on the cellular texture introduced by Worley [115].

5. **Edges to object surface**
   Compute the upper dilation $\tilde{\delta}^{\text{up}}_{Y}\left(P^{\text{up}}_{\text{open}}\right)$ and the lower dilation $\tilde{\delta}^{\text{low}}_{Y}\left(P^{\text{low}}_{\text{open}}\right)$ using segment-wise width values l^{border}_{i}. The contours $P^{\text{up}}_{\text{border}}$ and $P^{\text{low}}_{\text{border}}$ of the respective dilations define the upper and lower edge that describe the transition of the defect to the object surface.

6. **Area connected to object surface**
   Compute an upper shift $P^{\text{up}}_{\text{shift}}$ of $P^{\text{up}}_{\text{border}}$ and a lower shift $P^{\text{low}}_{\text{shift}}$ of $P^{\text{low}}_{\text{border}}$ using a constant width l^{shift} as described in Step 5 for coat lift modeling in Section 11.2.2.4.

7.-9. To simulate the top part of the buckle defect, apply Steps 6-8 from coat lift modeling in Section 11.2.2.4 using $P^{\text{up}}_{\text{open}}$, $P^{\text{up}}_{\text{border}}$ and $P^{\text{up}}_{\text{shift}}$. Proceed similarly with $P^{\text{low}}_{\text{open}}$, $P^{\text{low}}_{\text{border}}$ and $P^{\text{low}}_{\text{shift}}$ to obtain the lower part of the buckle defect.

10. Unite the meshes for the top and lower part.

See Figure 12.6 for synthetic images of objects having buckle defects generated using the alternative modeling approach. As the defect is modeled using two lift-offs originating from the same path, light is allowed to enter the open buckle and gets under the elevated metal sheets. This is not possible for defect geometries modeled with the previous method as the opening is restricted and closed to the sides. Thus, the new approach generates

more realistic defect geometries especially for buckles having wide openings where the underlying object surface is visible in the renderings.

12.3 Modeling Progressive Defecting by Growing 3d Mesh Objects

Crack defects occur in many domains and should optimally be detected at an early state. However, not every crack may be critical from the beginning. Thus, their progression can be followed to decide whether it becomes a negative, safety-relevant defect. Hence, learning-based methods can be used to forecast the failure progression in order to determine already in an early state whether the defect may become critical.

To generate synthetic data of progressive defect formation, appropriate models are necessary that simulate the growth of defects within an object. In Section 12.3.1 we give real-world examples where the observation of cracks at different states is important and how progressive cracking can be simulated using procedural approaches form computer graphics. Then, in Section 12.3.2, we present a novel approach to generate 3d mesh objects of progressive cracking at intermediate states based on the model for simulating cracks introduced in Section 11.2.2.1. Cracking also appears for buckle defects. Therefore, in Section 12.3.3, we propose a method to compute 3d mesh objects of progressive buckling.

12.3.1 Related Work

When analyzing the progressive cracking process in granite, it is categorized into different cracking levels dependent on characteristic stress thresholds. Zhou et al. [124] investigate the transition from micro-cracking to macro-cracking and to the full fracturing of the rock. The fatigue crack growth rate is relevant to investigate fatigue failure of additive manufacturing of titanium alloys [111, 123]. Therefore, Wang et al. [111] use in-situ scanning electron microscopy images, that is, images are taken sequentially with slightly increasing load.

In computer graphics, different approaches for progressive object fracturing exist [16, 45, 76, 77]. Therefore, the object geometry is considered to be tessellated by tetrahedrons. Then, the modeling of the fracturing process is commonly considered in two steps. First, coarse fracturing is applied on the object tessellation which makes it practicable in real-time needed for generating movies or computer games. Second, the coarse boundaries are post-processed to receive realistic simulations. However, those methods focus on the realistic appearance of the progressive deformation mostly without integrating real material properties and information about its composition. Also, we are not focusing on fracturing of the complete object, but rather on surface defects such as cracks.

12.3.2 Modeling 3d Crack Objects at Different Time Steps

According to Section 11.2.2.1, assume the path of the crack to be denoted via its vertices by $B_0 = \left(v_0^0, \dots, v_{K_0}^0\right)$. Define the paths of the branches by $B_i = \left(v_0^i, \dots, v_{K_i}^i\right)$ for $i = 1, \dots, I$, where $K_i \in \mathbb{N}_{>0}$ defines the length of the corresponding path and $I \in \mathbb{N}_0$ is the number of occurring branches. Note that the start vertex v_0^i of branch i should be contained in the main crack or in one of the previously generated branches, that is $v_0^i \in \bigcup_{j=0}^{i-1} B_j$. Let $\beta_i \in \{0, \dots, i-1\}$ be the branch of the start vertex and $\theta_i \in \left\{v_0^{\beta_i}, \dots, v_{K_{\beta_i}-1}^{\beta_i}\right\}$ be the location of that vertex on the branch. We exclude $\theta_i = v_{K_{\beta_i}}^{\beta_i}$ as possible start vertex since

it would be an extension of an existing branch rather than creating a new one. From now, the main crack and the branches are called cracks if not explicitly differentiated.

The 3d mesh of the crack structure described in Section 11.2.2.1 defines the final state of the cracking process. Temporal evolution is now introduced by subsequently adding the edges representing the cracks. In addition, the width and depth of the cracks gradually increase over time.

To model progressive cracking we need to determine the discrete time when a path starts and ends growing, that are $\tau_i \in \mathbb{N}$ and $\psi_i \in \mathbb{N}$, respectively, for crack $i = 0, \dots, I$. It holds $\tau_i > \psi_i$. The main path starts at time $\tau_0 = 0$, but the initiation of the branches depends on the growth of the paths they originate from. Therefore, assign crack initiation times

$$\tau_i = \begin{cases} 0 & \text{if } i = 0 \\ \tau_{\beta_i} + \theta_i + \epsilon_i & \text{else} \end{cases}$$

with $\epsilon_i \sim \mathcal{U}(\{0, \dots, \min(\epsilon_{\max}, K_{\beta_i} - \theta_i)\})$ a random delay between reaching the start vertex of a branch and the start of its growth. The parameter $\epsilon_{\max} \in \mathbb{N}$ models the maximal delay. Once a branch starts growing, one line segment arrives per time step. Here, the time steps are considered to be discrete. Hence, the growth of branch B_i terminates at time $\psi_i = \tau_i + K_i$. The total duration of the cracking process is $\Delta = \max_{i=0,\dots,I} \psi_i \in \mathbb{N}$.

Now, the cracks are generated at several time steps by continuously increasing the length, thickness and depth of the intermediate cracks. Therefore, let $t_s = \lfloor s/S \cdot \Delta \rceil \in \mathbb{N}$ for $s = 1, \dots, S$ be the time steps to be investigated for a fixed number $S \in \{1, \dots, \Delta\}$. For every s, determine the visible branches and generate the intermediate 3d crack objects by adapting the final crack structure as follows. See Figure 12.7 for a visualization.

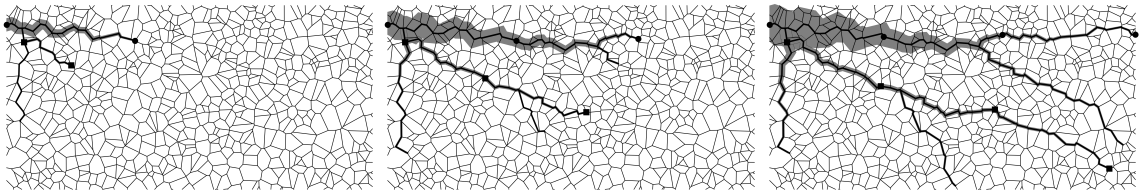
1. Crack B_i is visible at time s if $\tau_i < t_s$. Proceed with these branches.
2. Identify the length $\lambda_{i,s} = \min(t_s - \tau_i, K_i)$ of the intermediate path $B_i^s = \left(v_0^i, \dots, v_{\lambda_{i,s}}^i\right)$.
3. Determine the scaling factors $l_k^{i,s}$ for $k = 0, \dots, \lambda_{i,s}$ of the dilation of the intermediate path. Note that by simply scaling all final widths to obtain $l_k^{i,s}$, the crack can be unrealistically wide for the early time steps. See Figure 12.7a for an example and a description of how scaling factors are selected in our case using a sequence of intermediate scaling factors as described in Section 11.2.2.1. The formal details are as follows.

   Let $J_{i,s}$ denote a set of indices of branch segments of branch B_i that are assigned scaling factors at time t_s.

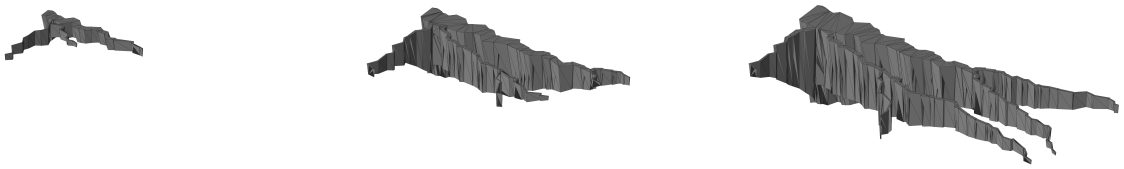
   a) Determine $\rho_i \in \{1, \dots, s\}$ such that B_i was visible at time step t_{ρ_i} for the first time. Set the number of intermediate scaling factors to $|J_{i,s}| = s - \rho_i + 2$. For a branch that is always visible ($\rho_i = 1$), this yields $|J_{i,s}| = s + 1$.

   b) Let
   $$J_{i,s} = \left\{ \left\lfloor \frac{j}{|J_{i,s}| - 1} \cdot \lambda_{i,s} \right\rceil : j = 0, \dots, |J_{j,s}| - 1 \right\}.$$
   Figure 12.7a marks those points for the main crack and an exemplary branch.

   c) To assign values to $l_k^{i,s}$ for $k \in J_{i,s}$ we scale the final width values l_k^i by the number of time steps where the branch is visible. The width at the last visible vertex is always chosen as the smallest thickness of the final branch. Thus, the



(a) Voronoi tessellation in W with the intermediate paths and their dilation. Black circles and squares mark the vertices of the growing main crack and of a growing exemplary branch, respectively, that are directly assigned width values. All other widths are obtained by linear interpolation.



(b) 3d meshes of the progressive cracking.



(c) Renderings of a cuboid with imprinted crack objects from Figure 12.7b. We used a texture for cast metallic objects, implemented by Lovro Bosnar from Fraunhofer ITWM, which is based on the cellular texture introduced by Worley [115]. The renderings are generated in Blender [1] using a plane light and a pinhole camera. 30 spp and 12 light bounces are applied. Rendering was performed by Fraunhofer ITWM.

Figure 12.7: Procedure to generate a progressive cracking visualized for three time steps (from left to right).

intermediate scaling factors are given by

$$l_k^{i,s} = \begin{cases} l_k^i \cdot r_{i,s} & \text{if } k < \lambda_{i,s} \\ l_{K_i}^i & \text{if } k = \lambda_{i,s} \end{cases} \quad \text{with } r_{i,s} = \frac{|J_{i,s}| - 1}{S - \rho_i + 1}.$$

4. Determine the depth values $h_k^{i,s}$ similar to the widths in Step 3.

5. Compute the surface triangulation of the closed 3d shape of the intermediate crack object, see Figure 12.7b.

6. Unite the current 3d object with all the previously obtained meshes at time t_s.

3d mesh objects of progressive cracking obtained by that method can now be used to generate synthetic optical images of breaking products. An example is given in Figure 12.7c showing synthetic images of the cracking of a rectangular object.

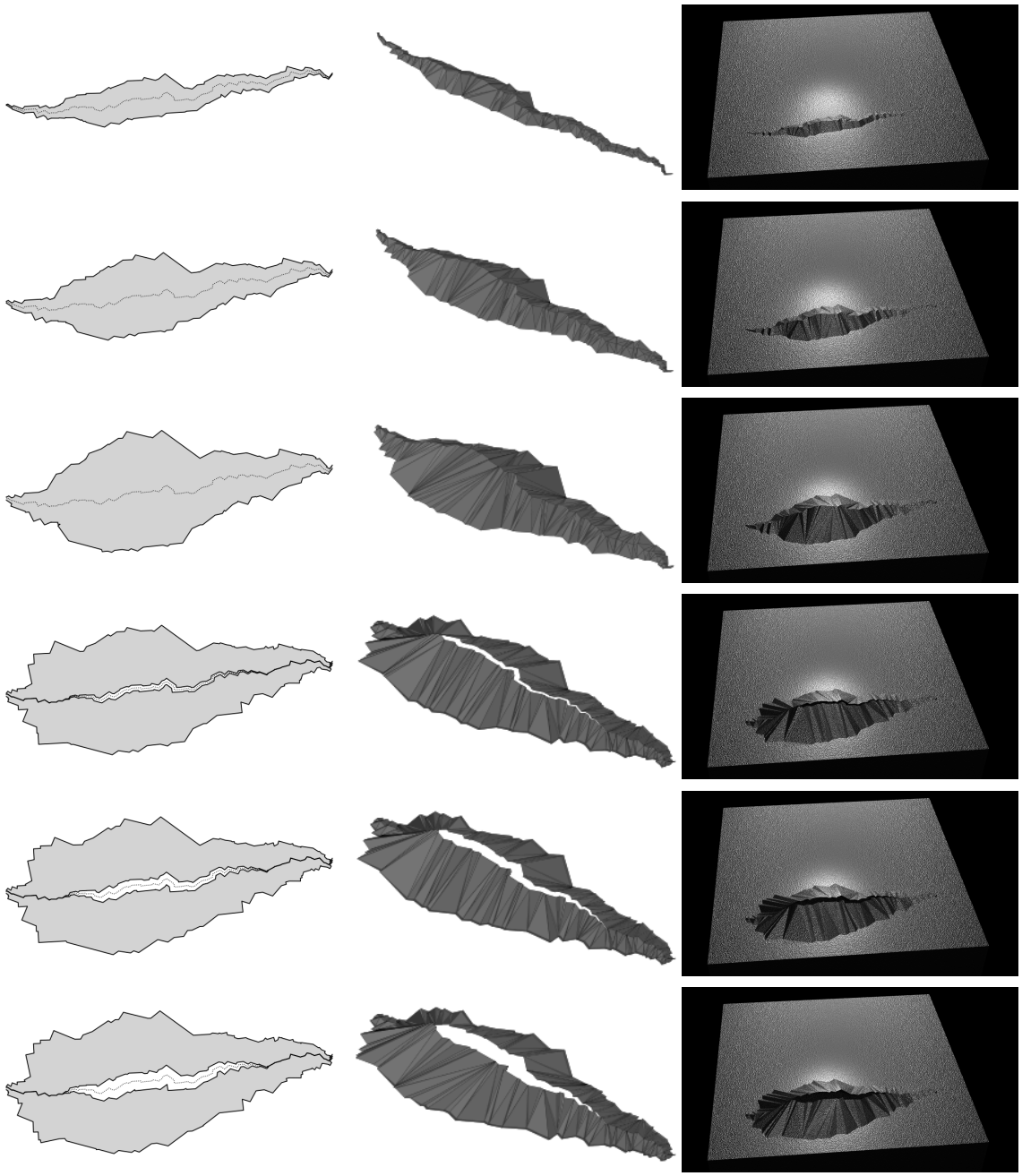
12.3.3 Modeling 3d Buckle Objects at Different Time Steps

As second possibility to model progressive defecting we propose buckle defect formation simulated at different intermediate time steps. Therefore, we use the new approach for buckle modeling introduced in Section 12.2. In contrast to cracks, we assume buckles not to expand in their lengths, only in width and height. Also, the buckle grows primarily as a closed geometry just in width and height, that is a bulge, followed by cracking the ridge such that the characteristic gap opens slowly.

The process of growth is evaluated at S equally-spaced discrete time steps $t_s \in \mathbb{N}$, $s = 1, \ldots, S$. Assume steps $t_1, \ldots, t_b$ to determine the growing of the closed geometry for an appropriate $b \leqslant S$ and $t_{b+1}, \ldots, t_S$ the growing of the open geometry. Further, assume the final dilation widths to be provided by the sequences $\tilde{l}_0^{\text{open}}, \ldots, \tilde{l}_{\tilde{k}^{\text{open}}}^{\text{open}}$ and $\tilde{l}_0^{\text{border}}, \ldots, \tilde{l}_{\tilde{k}^{\text{border}}}^{\text{border}}$, respectively, as described in Section 11.2.2.1. The same holds for heights h_i computed from $\tilde{h}_0, \ldots, \tilde{h}_{\tilde{k}^{\text{height}}}$. Since the number of segments of $P_{\text{border}}^{\text{up}}$ and $P_{\text{border}}^{\text{low}}$ varies for the different time steps as they depend on $P_{\text{open}}^{\text{up}}$ and $P_{\text{open}}^{\text{low}}$, respectively, the dilation widths and heights are obtained by scaling the sequences that determine the final values. Note that in contrast to growing crack defects, uniform scaling of widths and heights can be applied because we assume buckles not to expand in their length and no branches exist which may start growing at a later moment. Then, the modeling of progressive buckling divides into the following steps, see also Figure 12.8 for illustration.

1. **Growth of closed geometry:** $s = 1, \ldots, b$
   Scale $\tilde{l}_0^{\text{border}}, \ldots, \tilde{l}_{\tilde{k}^{\text{border}}}^{\text{border}}$ and $\tilde{h}_0, \ldots, \tilde{h}_{\tilde{k}^{\text{height}}}$ by $\frac{s}{b}$ and use those values to compute a bulge defect, see Section 11.2.2.2 for more details.

2. **Growth of open geometry:** $s = b + 1, \ldots, S$
   Let $\tilde{l}_0^{\text{border}}, \ldots, \tilde{l}_{\tilde{k}^{\text{border}}}^{\text{border}}$ and $\tilde{h}_0, \ldots, \tilde{h}_{\tilde{k}^{\text{height}}}$ be unchanged but scale $\tilde{l}_0^{\text{open}}, \ldots, \tilde{l}_{\tilde{k}^{\text{open}}}^{\text{open}}$ by $\frac{s}{S-b}$ and use those values to compute a buckle defect as described in Section 12.2.



2d shape with path (dotted line), dilation (gray area) and opening (white).

3d surface mesh.

Synthetic images rendered in Blender [1] using a cast metallic object texture, three plane lights and a pinhole camera. Rendering is applied with 24 spp and 16 light bounces. The texture model was implemented by Lovro Bosnar from Fraunhofer ITWM based on the cellular texture from Worley [115].

Figure 12.8: Procedure to generate progressive buckling visualized for six time steps (increasing from top to bottom) and $b = 3$ marks the opening of the buckle defect.

13 Conclusion

In this thesis we presented models to simulate surface textures and surface defects of rigid metal parts with planar surfaces. Using these methods, digital twins of defective and defect-free objects can be generated. This approach is essential to generate synthetic data for visual surface inspection via rule-based methods. Therefore, the real inspection environment is virtually recreated including simulations of the camera, the light source and the object to be inspected followed by photo-realistic rendering, that is, virtually taking 2d images of the 3d scene. Those images can then be used to train machine learning approaches for automatic defect detection. Note that the quality and robustness of those algorithms rely on the quality of the provided input data set in terms of completeness and correct annotation of defects to be detected. The main advantages of a synthetic training data set in contrast to real image data are the fast data generation in sufficient quantity and automatic pixel-perfect annotation as the defect locations and shapes are known.

As we focus on visual surface inspection, a sufficiently precise simulation of the interaction of the object's surface with light is crucial for realistic image synthesis. This is affected by the surface topography of the object in meso scale which is mainly determined by the applied surface finishing method. Thus, precise simulations of the resulting surface patterns are required. Those are applied to the object surface using textures, that are functions specifying the small-scale changes of a surface topography. Here, texture images are used. We restricted on texture synthesis models for two manufacturing methods resulting in completely different surface patterns, namely sandblasting and milling. As a common model was not feasible due to the predominantly differing properties of the surface patterns, two different approaches based on topography measurements of real surfaces are proposed.

Sandblasted surface textures are modeled by a combination of existing data-based texture models that rely entirely on the measurements. First, image patches are generated using the RPN with crop-outs of the associated measurement as input. Those are stitched afterwards using the method of Efros and Freeman. The second step can be neglected if the output texture is to be smaller than the measurement. Texture images of arbitrary size and with a pixel spacing coarser or equal to that of the measurement can be generated. A procedural model was difficult to develop as only one machining parameter was provided and varied for the real surface treatment. However, for future research a parametric texture synthesis model would be beneficial that is independent of input images.

In contrast, for milled surface textures we proposed a procedural, multi-step model that includes all process-related parameters known from the machine settings. It is defined as a function on a continuous domain and has thus to be evaluated on given grid points to obtain texture images. The pattern of a milled surface is determined by ring-shaped structures resulting from the rotating movement of the milling head. The model subdivides into determining the locations of the milling rings, creating the shapes of the individual rings and simulating the interaction of several overlapping rings. For efficient use of the model in rendering as truly procedural texture, the intermediate image generating step must be avoided. Therefore, fast approaches for computing the height value at a given surface point are required where the crucial point is to develop a fast searching algorithm to find all rings affecting one surface point. Moreover, an adaptation of the model for

milled surfaces was presented, which models textures of polished objects.

An annotated dual data set consisting of real and synthetic images of non-defective and defective sandblasted and milled objects was published using the presented texture synthesis models. For synthetic images of sandblasted parts, the topography measurements of the associated real object surfaces were used as input for the corresponding texture simulation model. Different values of various milling parameters have been used to process the real objects such that the variability of milled surface patterns is represented in the data set. The same parameter configurations have been chosen to create the synthetic data set of milled objects. Surface defects considered in the data set are dents and scratches, simulated via the approach from Bosnar et al. [12]. Therefore, 3d geometries of the defects were generated and then imprinted into the 3d geometry of the object to be inspected to obtain digital twins of defective parts. The synthetic data set can be used to fully train or to fine-tune a pre-trained learning-based defect detection algorithm. However, the domain gap between real and synthetic data is visible when comparing a dual image pair.

In this thesis we further presented a parametric model to simulate dent defects different to the approach introduced by Bosnar et al. [12]. Furthermore, models for casting defects such as cracks, rat tails and delaminations were proposed to increase the diversity of surface defects that can be simulated.

Dents are usually caused by a defecting tool that hits the object surface, resulting in an indentation that resembles the negative of the tool geometry. Further, we assume material relocation modeled by accumulations around the indentation. We proposed a multi-step model that integrates interchangeable components which can arbitrarily be combined to generate various defect shapes. So far, the resulting dent defects appear too regular which may be solved for example by varying width values for the accumulation within a realization.

Casting defects can be caused by a number of reasons, for example a defected mold or a too cold ambient air temperature that leads to rapid cooling of the material. We distinguished between elongated defect types (cold cracks, hot tears, buckles, bulges, coat lifts, cold shuts and rat tails) and scab defects. The development of the proposed models is motivated by the observation of defect structures and the description of defect formation processes. As most of the considered defect types are influenced by the material's microstructure, the models are based on Voronoi tessellations to include the poly-crystalline structure of metal on a coarse scale. However, the underlying tessellations have not yet been simulated in a way that incorporates existing knowledge about the material's structure. This is the subject of future research. Moreover, the models are parameterized using easily interpretable parameters. Note that the introduced defect types can also occur in other manufacturing processes in a similar form. Cracks for example are frequently occurring defects in various applications.

Often, defect formation is gradually progressing and not every deformation may be critical from the beginning. However, it can be beneficial to detect defects in an early state and follow their progression to decide whether they will have a negative, safety-relevant impact. Therefore, we proposed methods to compute 3d mesh objects of progressive cracking and buckling. This enables synthetic data generation to train machine learning methods that forecast the failure progression in order to determine already in an early state whether the defect may become critical.

The proposed texture model for milled surfaces and the models for surface defects are all parameterized. Thus, a wide variety of texture images and defect geometries can be generated simply by using different parameter settings. Furthermore, randomization of the models ensures variability, that is, sampling the individual parameter values from

suitable probability distributions, for example random sampling from fixed parameter ranges. Realistic synthetic images can be created by choosing real situation-dependent parameter restrictions for textures and defects. Moreover, edge case scenarios such as exaggeratedly long cracks or too narrow milling rings can easily be generated by using appropriate parameter configurations. This data can be useful to increase the model training performance of a defect detection algorithm to make it as robust as possible also in rarely occurring scenarios. Thus, it is possible to systematically vary individual aspects of the milling texture or the surface defects, so that defect detection algorithms can be specifically trained with the resulting synthetic data. For example, particular defect features can be varied individually to fine-tune defect detection algorithms. This provides a significant advantage over AI-based methods, where systematic data generation and control is difficult to perform.

Bibliography

[1] Blender. v4.3.2. BlenderFoundation. https://www.blender.org/.

[2] Foundry Lexicon. https://www.giessereilexikon.com/. Accessed: 2024-12-19.

[3] Severstal: Steel defect detection, Kaggle. 2019. https://kaggle.com/competitions/severstal-steel-defect-detection.

[4] A. G. Abubakr, I. Jovančević, N. I. Mokhtari, H. B. Abdallah, and J.-J. Orteu. On learning deep domain-invariant features from 2D synthetic images for industrial visual inspection. In *Fifteenth International Conference on Quality Control by Artificial Vision*. 11794:1179418, SPIE, 2021. https://doi.org/10.1117/12.2589040.

[5] T. Barisin, C. Jung, A. Nowacka, K. Schladitz, and C. Redenbach. Crack segmentation in 3d concrete images: perspectives and challenges. In *International Symposium on Non-Destructive Testing in Civil Engineering (NDT-CE 2022)*. e-Journal of Nondestructive Testing, 27(9), 2022. https://doi.org/10.58286/27207.

[6] F. Beaune, E. Tovagliari, L. Barrancos, and et. al. appleseed. appleseed: v2.1.0-beta. Zenodo. 2019. http://dx.doi.org/10.5281/zenodo.3456967.

[7] A. Bechikh, O. Klinkova, Y. Maalej, I. Tawfiq, and R. Nasri. Sandblasting parameter variation effect on galvanized steel surface chemical composition, roughness and free energy. *International Journal of Adhesion and Adhesives*, 102:102653, 2020. https://doi.org/10.1016/j.ijadhadh.2020.102653.

[8] R. Bellman. On a Routing Problem. *Quarterly of Applied Mathematics*, 16(1):87–90, 1958. https://api.semanticscholar.org/CorpusID:123639971.

[9] A. Boikov, V. Payor, R. Savelev, and A. Kolesnikov. Synthetic Data Generation for Steel Defect Detection and Classification Using Deep Learning. *Symmetry*, 13(7):1176, 2021. https://doi.org/10.3390/sym13071176.

[10] C. Bosch. *Realistic Image Synthesis of Surface Scratches and Grooves*. PhD Thesis, Université de Limoges, 2007.

[11] L. Bosnar. *Procedural Modeling and Image Synthesis for Virtual Surface Inspection Planning*. PhD Thesis, RPTU University Kaiserslautern-Landau, 2024.

[12] L. Bosnar, H. Hagen, and P. Gospodnetić. Procedural Defect Modeling for Virtual Surface Inspection Environments. *IEEE Computer Graphics and Applications*, 43(2):13–22, 2023. https://doi.org/10.1109/MCG.2023.3243276.

[13] L. Bosnar, M. Rauhut, H. Hagen, and P. Gospodnetić. Texture Synthesis for Surface Inspection. *LEVIA 22: Leipzig Symposium on Visualization in 2022*, 2022. https://doi.org/10.36730/2022.1.levia.4.

[14] L. Bosnar, D. Šarić, S. Dutta, T. Weibel, M. Rauhut, and P. Gospodnetić. Image Synthesis Pipeline for Surface Inspection. In *LEVIA'20: Leipzig Symposium on Visualization in Applications 2020*, 2020. https://doi.org/10.31219/osf.io/kqt8w.

[15] J. Božič, D. Tabernik, and D. Skočaj. Mixed supervision for surface-defect detection: From weakly to fully supervised learning. *Computers in Industry*, 129:103459, 2021. https://doi.org/10.1016/j.compind.2021.103459.

[16] Z. Chen, M. Yao, R. Feng, and H. Wang. Physics-Inspired Adaptive Fracture Refinement. *ACM Transactions on Graphics*, 33(4):113, 2014. https://doi.org/10.1145/2601097.2601115.

[17] S. N. Chiu, D. Stoyan, W. S. Kendall, and J. Mecke. *Stochastic Geometry and Its Applications*. Wiley Series in Probability and Statistics, John Wiley & Sons, 2013. https://doi.org/10.1002/9781118658222.

[18] T. Dahmen, P. Trampert, F. Boughorbel, J. Sprenger, M. Klusch, K. Fischer, C. Kübel, and P. Slusallek. Digital reality: a model-based approach to supervised learning from synthetic data. *AI Perspectives*, 1:2, 2019. https://doi.org/10.1186/s42467-019-0002-0.

[19] Dawson-Haggerty et al. trimesh. v3.2.0. https://trimesh.org/.

[20] L. N. L. de Lacalle, F. J. Campa, and A. Lamikiz. 3 - Milling. In *Modern Machining Technology*, pages 213–303. Woodhead Publishing, 2011. https://doi.org/10.1533/9780857094940.213.

[21] P. De Roovere, S. Moonen, N. Michiels, and F. wyffels. Sim-to-Real Dataset of Industrial Metal Objects. *Machines*, 12(2):99, 2024. https://doi.org/10.3390/machines12020099.

[22] E. W. Dijkstra. A Note on Two Problems in Connexion with Graphs. *Numerische Mathematik*, 1:269–271, 1959. https://doi.org/10.1007/BF01386390.

[23] K. J. Dogahe, I. Kurz, P. Binkele, S. Schmauder, M. Mlikota, and Z. Božić. Physically-based modelling of the fatigue crack initiation life of stent components under cyclic loading employing the Finite-Element-Method (FEM). *International Journal of Fatigue*, 171:107594, 2023. https://doi.org/10.1016/j.ijfatigue.2023.107594.

[24] J. Dorsey, H. Rushmeier, and F. Sillion. *Digital Modeling of Material Appearance*. Elsevier, 2008. https://doi.org/10.1016/B978-0-12-221181-2.X5001-0.

[25] A. A. Efros and W. T. Freeman. Image Quilting for Texture Synthesis and Transfer. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '01. Association for Computing Machinery, 2001. https://doi.org/10.1145/383259.383296.

[26] A. A. Efros and T. K. Leung. Texture Synthesis by Non-parametric Sampling. In *Proceedings of the Seventh IEEE International Conference on Computer Vision*, vol. 2, pages 1033-1038, 1999. https://doi.org/10.1109/ICCV.1999.790383.

[27] L. Fan, F. M. Chitalu, and T. Komura. Simulating Brittle Fracture with Material Points. *ACM Transactions on Graphics*, 41(5):177, 2022. https://doi.org/10.1145/3522573.

[28] C. Felhő, B. Karpuschewski, and J. Kundrák. Surface roughness modelling in face milling. *Procedia CIRP*, 31:136–141, 2015. https://doi.org/10.1016/j.procir.2015.03.075.

[29] C. Felhő and J. Kundrák. Effects of Setting Errors (Insert Run-Outs) on Surface Roughness in Face Milling When Using Circular Inserts. *Machines*, 6(2):14, 2018. https://doi.org/10.3390/machines6020014.

[30] W. T. Freeman and E. H. Adelson. The design and use of steerable filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(9):891–906, 1991. https://doi.org/10.1109/34.93808.

[31] J. Friemann, L. P. Mikkelsen, C. Oddy, and M. Fagerström. Synthetic, automatically labelled training data for machine learning based X-ray CT image segmentation: Application to 3D-textile carbon fibre reinforced composites. *Composites Part B: Engineering*, 305:112656, 2025. https://doi.org/10.1016/j.compositesb.2025.112656.

[32] P. Fuchs, T. Kröger, and C. S. Garbe. Defect detection in CT scans of cast aluminum parts: A machine vision perspective. *Neurocomputing*, 453:85–96, 2021. https://doi.org/10.1016/j.neucom.2021.04.094.

[33] J. Fulir, L. Bosnar, H. Hagen, and P. Gospodnetić. Synthetic Data for Defect Segmentation on Complex Metal Surfaces. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2023. https://doi.org/10.1109/CVPRW59228.2023.00465.

[34] J. Fulir, N. Jeziorski, L. Bosnar, H. Hagen, C. Redenbach, T. Herrfurth, M. Trost, T. Gischkat, and P. Gospodnetić. SynosIs data set. 2024. http://dx.doi.org/10.24406/fordatis/370.

[35] J. Fulir, N. Jeziorski, L. Bosnar, H. Hagen, C. Redenbach, T. Herrfurth, M. Trost, T. Gischkat, and P. Gospodnetić. SYNOSIS: Image Synthesis Pipeline for Machine Vision in Metal Surface Inspection. *Sensors*, 25(19):6016, 2025. https://doi.org/10.3390/s25196016.

[36] B. Galerne, Y. Gousseau, and J.-M. Morel. Micro-Texture Synthesis by Phase Randomization. *Image Processing On Line*, 1:213–237, 2011. https://doi.org/10.5201/ipol.2011.ggm_rpn.

[37] B. Galerne, Y. Gousseau, and J.-M. Morel. Random Phase Textures: Theory and Synthesis. *IEEE Transactions on Image Processing*, 20(1):257–267, 2011. https://doi.org/10.1109/TIP.2010.2052822.

[38] L. A. Gatys, A. S. Ecker, and M. Bethge. Texture Synthesis Using Convolutional Neural Networks. In *Advances in Neural Information Processing Systems*. Curran Associates, Inc., vol. 28, pages 262-270, 2015. https://proceedings.neurips.cc/paper/2015/file/a5e00132373a7031000fd987a3c9f87b-Paper.pdf.

[39] E. Ghazvinian, M. S. Diederichs, and R. Quey. 3D random Voronoi grain-based models for simulation of brittle rock damage and fabric-guided micro-fracturing. *Journal of Rock Mechanics and Geotechnical Engineering*, 6(6):506–521, 2014. https://doi.org/10.1016/j.jrmge.2014.09.001.

[40] R. C. Gonzalez and R. E. Woods. *Digital Image Processing (3rd Edition)*. Prentice-Hall, Inc., 2006.

[41] P. Gospodnetić, D. Mosbach, M. Rauhut, and H. Hagen. Viewpoint placement for inspection planning. *Machine Vision and Applications*, 33:2, 2022. https://doi.org/10.1007/s00138-021-01252-z.

[42] P. Gospodnetić, D. Mosbach, M. Rauhut, and H. Hagen. Flexible Surface Inspection Planning Pipeline. In *2020 6th International Conference on Control, Automation and Robotics (ICCAR)*, pages 644-652, 2020. https://doi.org/10.1109/ICCAR49639.2020.9107983.

[43] P. Gospodnetić, M. Rauhut, and H. Hagen. Surface Inspection Planning Using 3D Visualization. *LEVIA'19: Leipzig Symposium on Visualization in Applications 2019*, 2020. https://doi.org/10.31219/osf.io/6ad2b.

[44] M. Hadad and M. Ramezani. Modeling and analysis of a novel approach in machining and structuring of flat surfaces using face milling process. *International Journal of Machine Tools and Manufacture*, 105:32–44, 2016. https://doi.org/10.1016/j.ijmachtools.2016.03.005.

[45] D. Hahn and C. Wojtan. High-Resolution Brittle Fracture Simulation with Boundary Elements. *ACM Transactions on Graphics*, 34(4):151, 2015. https://doi.org/10.1145/2766896.

[46] R. M. Haralick, K. Shanmugam, and I. Dinstein. Textural Features for Image Classification. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-3(6):610–621, 1973. https://doi.org/10.1109/TSMC.1973.4309314.

[47] P. E. Hart, N. J. Nilsson, and B. Raphael. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968. https://doi.org/10.1109/TSSC.1968.300136.

[48] M. Haselmann and D. P. Gruber. Pixel-Wise Defect Detection by CNNs without Manually Labeled Training Data. *Applied Artificial Intelligence*, 33(6):548–566, 2019. https://doi.org/10.1080/08839514.2019.1583862.

[49] D. J. Heeger and J. R. Bergen. Pyramid-based Texture Analysis/Synthesis. In *Proceedings., International Conference on Image Processing*, vol. 3, pages 648-651, 1995. https://doi.org/10.1109/ICIP.1995.537718.

[50] M. Heveling and M. Reitzner. Poisson–Voronoi approximation. *The Annals of Applied Probability*, 19(2):719–736, 2009. https://doi.org/10.1214/08-AAP561.

[51] N. Hoffman. Background: Physics and Math of Shading. In *SIGGRAPH 2013 Course: Physically Based Shading in Theory and Practice*, 2013. https://blog.selfshadow.com/publications/s2013-shading-course/hoffman/s2013_pbs_physics_math_notes.pdf.

[52] D. Holten, J. J. Van Wijk, and J.-B. Martens. A perceptually based spectral model for isotropic textures. *ACM Transactions on Applied Perception*, 3(4):376–398, 2006. https://doi.org/10.1145/1190036.1190039.

[53] D. Honzátko, E. Türetken, S. A. Bigdeli, L. A. Dunbar, and P. Fua. Defect segmentation for multi-illumination quality control systems. *Machine Vision and Applications*, 32:118, 2021. https://doi.org/10.1007/s00138-021-01244-z.

[54] S. Jain, G. Seth, A. Paruthi, U. Soni, and G. Kumar. Synthetic data augmentation for surface defect detection and classification using deep learning. *Journal of Intelligent Manufacturing*, 33:1007–1020, 2020. https://doi.org/10.1007/s10845-020-01710-x.

[55] N. Jeziorski, P. Gospodnetić, and C. Redenbach. Synthetic Defect Geometries of Cast Metal Objects Modeled via 2d Voronoi Tessellations. *arXiv*, preprint arXiv:[2602.05440], 2026. https://doi.org/10.48550/arXiv.2602.05440.

[56] N. Jeziorski and C. Redenbach. Stochastic geometry models for texture synthesis of machined metallic surfaces: sandblasting and milling. *Journal of Mathematics in Industry*, 14:17, 2024. https://doi.org/10.1186/s13362-024-00155-8.

[57] M. Jolly and L. Katgerman. Modelling of defects in aluminium cast products. *Progress in Materials Science*, 123:100824, 2022. https://doi.org/10.1016/j.pmatsci.2021.100824.

[58] C. Jung, F. Müsebeck, T. Barisin, K. Schladitz, C. Redenbach, M. Kiesche, and M. Pahn. Towards automatic crack segmentation in 3d concrete images. In *11th Conference on Industrial Computed Tomography (iCT)*. e-Journal of Nondestructive Testing (eJNDT), 27(3), 2022. https://doi.org/10.58286/26620.

[59] C. Jung and C. Redenbach. Crack modeling via minimum-weight surfaces in 3d Voronoi diagrams. *Journal of Mathematics in Industry*, 13:10, 2023. https://doi.org/10.1186/s13362-023-00138-1.

[60] C. Jung, C. Redenbach, and K. Schladitz. VoroCrack3d: An annotated semi-synthetic 3d image data set of cracked concrete. *Data in Brief*, 54:110474, 2024. https://doi.org/10.1016/j.dib.2024.110474.

[61] J. T. Kajiya. The Rendering Equation. In *Proceedings of the 13th Annual Conference on Computer Graphics and Interactive Techniques*. ACM SIGGRAPH Computer Graphics, 29(4):143-150, 1986. https://doi.org/10.1145/15886.15902.

[62] A. Kim, K. Lee, S. Lee, J. Song, S. Kwon, and S. Chung. Synthetic Data and Computer-Vision-Based Automated Quality Inspection System for Reused Scaffolding. *Applied Sciences*, 12(19):10097, 2022. https://doi.org/10.3390/app121910097.

[63] F. Klocke. *Fertigungsverfahren 1: Zerspanung mit geometrisch bestimmter Schneide*. Springer Vieweg Berlin, Heidelberg, 2018. https://doi.org/10.1007/978-3-662-54207-1.

[64] J. Kundrák, C. Felhő, and A. Nagy. Analysis and Prediction of Roughness of Face Milled Surfaces using CAD Model. *Manufacturing Technology*, 22(5):558–572, 2022. https://doi.org/10.21062/mft.2022.061.

[65] A. Leclaire. *Random Phase Fields and Gaussian Fields for Image Sharpness Assessment and Fast Texture Synthesis*. PhD Thesis, Université Paris Descartes, 2015.

[66] T. Löfstedt, P. Brynolfsson, T. Asklund, T. Nyholm, and A. Garpebring. Gray-level invariant Haralick texture features. *PLOS ONE*, 14(2):e0212110, 2019. https://doi.org/10.1371/journal.pone.0212110.

[67] M. Lu and C.-L. Chen. Detection and Classification of Bearing Surface Defects Based on Machine Vision. *Applied Sciences*, 11(4):1825, 2021. https://doi.org/10.3390/app11041825.

[68] Q. Luo, X. Fang, L. Liu, C. Yang, and Y. Sun. Automated Visual Defect Detection for Flat Steel Surface: A Survey. *IEEE Transactions on Instrumentation and Measurement*, 69(3):626–644, 2020. https://doi.org/10.1109/TIM.2019.2963555.

[69] Y. Ma, X. Gu, and Y. Wang. Histogram similarity measure using variable bin size distance. *Computer Vision and Image Understanding*, 114(8):981–989, 2010. https://doi.org/10.1016/j.cviu.2010.03.006.

[70] K. Mao, P. Wei, Y. Wang, M. Liu, S. Wang, and N. Zheng. CSDD: A Benchmark Dataset for Casting Surface Defect Detection and Segmentation. *IEEE/CAA Journal of Automatica Sinica*, 12(5):947–960, 2025. https://doi.org/10.1109/JAS.2025.125228.

[71] R. Mao, J. Fulir, C. Garth, and P. Gospodnetić. Sequential PatchCore: Anomaly Detection for Surface Inspection Using Synthetic Impurities. In *Computer Vision – ECCV 2024 Workshops, Proceedings, Part IV*. Springer-Verlag, 360-377, 2025. https://doi.org/10.1007/978-3-031-92805-5_23.

[72] J. Markee. *Atlas of Casting Defects*. An Investment Casting Institute Publication, 2017. https://61746c6173.investmentcasting.org/casting/defect/delamination/index.html.

[73] S. Merillou, J. M. Dischler, and D. Ghazanfarpour. Surface scratches: measuring, modeling and rendering. *The Visual Computer*, 17:30–45, 2001. https://doi.org/10.1007/s003710000093.

[74] D. Mery, V. Riffo, U. Zscherpel, G. Mondragón, I. Lillo, I. Zuccar, H. Lobel, and M. Carrasco. GDXray: The Database of X-ray Images for Nondestructive Testing. *Journal of Nondestructive Evaluation*, 34:42, 2015. https://doi.org/10.1007/s10921-015-0315-7.

[75] L. Moisan. Periodic Plus Smooth Image Decomposition. *Journal of Mathematical Imaging and Vision*, 39:161–179, 2011. https://doi.org/10.1007/s10851-010-0227-1.

[76] M. Müller, N. Chentanez, and T.-Y. Kim. Real Time Dynamic Fracture with Volumetric Approximate Convex Decompositions. *ACM Transactions on Graphics*, 32(4):115, 2013. https://doi.org/10.1145/2461912.2461934.

[77] M. Müller, L. McMillan, J. Dorsey, and R. Jagnow. Real-Time Simulation of Deformation and Fracture of Stiff Materials. In *Computer Animation and Simulation 2001, 113-124*. Eurographics. Springer Vienna, 2001. https://doi.org/10.1007/978-3-7091-6240-8_11.

[78] S. I. Nikolenko. *Synthetic Data for Deep Learning*. Springer Optimization and Its Applications, Springer, 2021. https://doi.org/10.1007/978-3-030-75178-4.

[79] A. Okabe, B. Boots, K. Sugihara, S. N. Chiu, and D. Kendall. *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*. Wiley Series in Probability and Statistics, John Wiley & Sons, 2000. https://doi.org/10.1002/9780470317013.

[80] E. Oranli, N. Gungoren, A. H. Astaraee, E. Maleki, S. Bagherifard, and M. Guagliano. Numerical and experimental analysis of sand blasting on polymeric substrates. *Forces in Mechanics*, 12:100208, 2023. https://doi.org/10.1016/j.finmec.2023.100208.

[81] J. Peng, L. N. Y. Wong, C. T. Teh, and Z. Li. Modeling Micro-cracking Behavior of Bukit Timah Granite Using Grain-Based Model. *Rock Mechanics and Rock Engineering*, 51:135–154, 2018. https://doi.org/10.1007/s00603-017-1316-x.

[82] M. Pharr, W. Jakob, and G. Humphreys. *Physically Based Rendering: From Theory to Implementation*. Morgan Kaufmann Publishers Inc., 2016. https://doi.org/10.1016/C2013-0-15557-2.

[83] B. T. Phong. Illumination for Computer Generated Pictures. *Communications of the ACM*, 18(6):311–317, 1975. https://doi.org/10.1145/360825.360839.

[84] R. Plato. *Basiswissen Numerik: Ein kompakter Einstieg*. Springer Spektrum Berlin, Heidelberg, 2023. https://doi.org/10.1007/978-3-662-66570-1.

[85] J. Portilla and E. P. Simoncelli. A Parametric Texture Model Based on Joint Statistics of Complex Wavelet Coefficients. *International Journal of Computer Vision*, 40(1):49–70, 2000. https://doi.org/10.1023/A:1026553619983.

[86] L. Raad, A. Davy, A. Desolneux, and J.-M. Morel. A survey of exemplar-based texture synthesis. *Annals of Mathematical Sciences and Applications*, 3(1):89–148, 2018. https://doi.org/10.4310/AMSA.2018.v3.n1.a4.

[87] L. Raad and B. Galerne. Efros and Freeman Image Quilting Algorithm for Texture Synthesis. *Image Processing On Line*, 7:1–22, 2017. https://doi.org/10.5201/ipol.2017.171.

[88] B. Ramos Barbero and E. Santos Ureta. Comparative study of different digitization techniques and their accuracy. *Computer-Aided Design*, 43(2):188–206, 2011. https://doi.org/10.1016/j.cad.2010.11.005.

[89] K. Schladitz, C. Redenbach, T. Barisin, C. Jung, N. Jeziorski, L. Bosnar, J. Fulir, and P. Gospodnetić. Simulation of Microstructures and Machine Learning. In *Continuum Models and Discrete Systems (CMDS 2023)*. 457:243-256, Springer Proceedings in Mathematics & Statistics, 2024. https://doi.org/10.1007/978-3-031-58665-1_19.

[90] O. Schmedemann, S. Schlodinski, D. Holst, and T. Schüppstuhl. Adapting synthetic training data in deep learning-based visual surface inspection to improve transferability of simulations to real-world environments. In *Automated Visual Inspection and Machine Vision V*. 12623:1262307, SPIE, 2023. https://doi.org/10.1117/12.2673857.

[91] R. Schneider and W. Weil. *Stochastic and Integral Geometry*. Springer Berlin, Heidelberg, 2008. https://doi.org/10.1007/978-3-540-78859-1.

[92] L. Schumaker. *Spline Functions: Basic Theory*. Cambridge University Press, 2007. https://doi.org/10.1017/CBO9780511618994.

[93] J. Sertucha and J. Lacaze. Casting Defects in Sand-Mold Cast Irons — An Illustrated Review with Emphasis on Spheroidal Graphite Cast Irons. *Metals*, 12(3):504, 2022. https://doi.org/10.3390/met12030504.

[94] E. P. Simoncelli and W. T. Freeman. The Steerable Pyramid: A Flexible Architecture for Multi-scale Derivative Computation. In *Proceedings., International Conference on Image Processing*, vol. 3, pages 444-447, 1995. https://doi.org/10.1109/ICIP.1995.537667.

[95] E. P. Simoncelli, W. T. Freeman, E. H. Adelson, and D. J. Heeger. Shiftable Multiscale Transforms. *IEEE Transactions on Information Theory*, 38(2):587–607, 1992. https://doi.org/10.1109/18.119725.

[96] M. Sistaninia, J.-M. Drezet, A. B. Phillion, and M. Rappaz. Prediction of Hot Tear Formation in Vertical DC Casting of Aluminum Billets Using a Granular Approach. *JOM*, 65(9):1131–1137, 2013. https://doi.org/10.1007/s11837-013-0662-8.

[97] R. Sizyakin, V. Voronin, N. Gapon, A. Zelensky, and A. Pižurica. Automatic detection of welding defects using the convolutional neural network. In *Automated Visual Inspection and Machine Vision III*. 11061:110610E, SPIE, 2019. https://doi.org/10.1117/12.2525643.

[98] P. Soille. *Morphological Image Analysis: Principles and Applications*. Springer Berlin, Heidelberg, 2004. https://doi.org/10.1007/978-3-662-05088-0.

[99] F. Sukowski, D. Suth, A. Waldyra, T. Grulich, and A. Jung. Automated 3D Defect Detection based on Simulated Reference. In *12th Conference on Industrial Computed Tomography (iCT)*. e-Journal of Nondestructive Testing (eJNDT), 28(3), 2023. https://doi.org/10.58286/27727.

[100] D. Tabernik, S. Šela, J. Skvarč, and D. Skočaj. Segmentation-based deep-learning approach for surface-defect detection. *Journal of Intelligent Manufacturing*, 31:759–776, 2020. https://doi.org/10.1007/s10845-019-01476-x.

[101] B. Tang, L. Chen, W. Sun, and Z.-k. Lin. Review of surface defect detection of steel products based on machine vision. *IET Image Processing*, 17(2):303–322, 2023. https://doi.org/10.1049/ipr2.12647.

[102] K. E. Torrance and E. M. Sparrow. Theory for Off-Specular Reflection From Roughened Surfaces. *Journal of the Optical Society of America*, 57(9):1105–1114, 1967. https://doi.org/10.1364/JOSA.57.001105.

[103] G. Tosello, G. Bissacco, J. Cao, and D. Axinte. Modeling and simulation of surface generation in manufacturing. *CIRP Annals*, 72(2):753–779, 2023. https://doi.org/10.1016/j.cirp.2023.05.002.

[104] B. Totino, F. Spagnolo, and S. Perri. RIAWELC: A Novel Dataset of Radiographic Images for Automatic Weld Defects Classification. *International Journal of Electrical and Computer Engineering Research*, 3(1), 2023. https://doi.org/10.53375/ijecer.2023.320.

[105] A. Tsirikoglou, G. Eilertsen, and J. Unger. A Survey of Image Synthesis Methods for Visual Machine Learning. *Computer Graphics Forum*, 39(6):426–451, 2020. https://doi.org/10.1111/cgf.14047.

[106] T. Tyystjärvi, P. Fridolf, A. Rosell, and I. Virkkunen. Deploying Machine Learning for Radiography of Aerospace Welds. *Journal of Nondestructive Evaluation*, 43:24, 2024. https://doi.org/10.1007/s10921-023-01041-w.

[107] D. Ulyanov, A. Vedaldi, and V. Lempitsky. Improved Texture Networks: Maximizing Quality and Diversity in Feed-Forward Stylization and Texture Synthesis. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4105–4113, 2017. https://doi.org/10.1109/CVPR.2017.437.

[108] J. Vacher and T. Briand. The Portilla-Simoncelli Texture Model: towards Understanding the Early Visual Cortex. *Image Processing On Line*, 11:170–211, 2021. https://doi.org/10.5201/ipol.2021.324.

[109] J. Vacher, T. Briand, B. Galerne, and J. Rabin. The Heeger & Bergen Pyramid Based Texture Synthesis Algorithm. *Image Processing On Line*, 4:276–299, 2014. https://doi.org/10.5201/ipol.2014.79.

[110] K. Wada, mpitid, M. Buijs, and et. al. Labelme: Image Polygonal Annotation with Python. wkentaro/labelme: v4.6.0. Zenodo. 2021. http://dx.doi.org/10.5281/zenodo.5711226.

[111] X. Wang, Y. Zhao, L. Wang, L. Wei, J. He, and X. Guan. In-situ sem investigation and modeling of small crack growth behavior of additively manufactured titanium alloy. *International Journal of Fatigue*, 149:106303, 2021. https://doi.org/10.1016/j.ijfatigue.2021.106303.

[112] Z. Wang, A. Bovik, H. Sheikh, and E. Simoncelli. Image Quality Assessment: From Error Visibility to Structural Similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004. https://doi.org/10.1109/TIP.2003.819861.

[113] Z.-M. Wang, M.-H. Li, and G.-S. Xia. Conditional Generative ConvNets for Exemplar-Based Texture Synthesis. *IEEE Transactions on Image Processing*, 30:2461–2475, 2021. https://doi.org/10.1109/TIP.2021.3052075.

[114] J. Wei, F. Shen, C. Lv, Z. Zhang, F. Zhang, and H. Yang. Diversified and Multi-Class Controllable Industrial Defect Synthesis for Data Augmentation and Transfer. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 4445-4453, 2023. https://doi.org/10.1109/CVPRW59228.2023.00467.

[115] S. Worley. A cellular texture basis function. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '96*, page 291–294. Association for Computing Machinery, 1996. https://doi.org/10.1145/237170.237267.

[116] J. Wu, R. Westermann, and C. Dick. A Survey of Physically Based Simulation of Cuts in Deformable Bodies. *Computer Graphics Forum*, 34(6):161–187, 2015. https://doi.org/10.1111/cgf.12528.

[117] J. Xie, Y. Lu, R. Gao, S.-C. Zhu, and Y. N. Wu. Cooperative Training of Descriptor and Generator Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(1):27–45, 2020. https://doi.org/10.1109/TPAMI.2018.2879081.

[118] J. Xie, Y. Lu, S.-C. Zhu, and Y. Wu. A Theory of Generative ConvNet. In *Proceedings of The 33rd International Conference on Machine Learning*, Proceedings of Machine Learning Research. PMLR, 48:2635-2644, 2016. http://proceedings.mlr.press/v48/xiec16.pdf.

[119] X. Xie. A Review of Recent Advances in Surface Defect Detection using Texture analysis Techniques. *Electronic Letters on Computer Vision and Image Analysis*, 7(3):1–22, 2008. https://doi.org/10.5565/rev/elcvia.268.

[120] X. Yu, X. Wei, X. Huang, and L. Zhang. Study on the Modeling Method and Influencing Parameters of Sandblasting Process for Blade Grinding. *Advances in Materials Science and Engineering*, 2022(1):7905927, 2022. https://doi.org/10.1155/2022/7905927.

[121] L. Zeng, S. Zhang, P. Wang, Z. Li, Y. Hu, and T. Xie. Defect detection algorithm for magnetic particle inspection of aviation ferromagnetic parts based on improved deeplabv3+. *Measurement Science and Technology*, 34:065401, 2023. https://doi.org/10.1088/1361-6501/acb9ae.

[122] G. Zhang, K. Cui, T.-Y. Hung, and S. Lu. Defect-GAN: High-Fidelity Defect Synthesis for Automated Defect Inspection. In *2021 IEEE Winter Conference on Applications of Computer Vision (WACV)*. 2524-2534, IEEE Computer Society, 2021. https://doi.org/10.1109/WACV48630.2021.00257.

[123] X. Zhang, F. Martina, J. Ding, X. Wang, and S. W. Williams. Fracture toughness and fatigue crack growth rate properties in wire + arc additive manufactured Ti-6Al-4V. *Fatigue & Fracture of Engineering Materials & Structures*, 40(5):790–803, 2016. https://doi.org/10.1111/ffe.12547.

[124] X.-P. Zhou, J.-Z. Zhang, Q.-H. Qian, and Y. Niu. Experimental investigation of progressive cracking processes in granite under uniaxial loading using digital imaging and ae techniques. *Journal of Structural Geology*, 126:129–145, 2019. https://doi.org/10.1016/j.jsg.2019.06.003.

Scientific Career

List of Publications

Published

[34] J. Fulir, **N. Jeziorski**, L. Bosnar, H. Hagen, C. Redenbach, T. Herffurth, M. Trost, T. Gischkat, and P. Gospodnetić. SyonsIs data set. 2024. `http://dx.doi.org/10.24406/fordatis/370`.

[35] J. Fulir, **N. Jeziorski**, L. Bosnar, H. Hagen, C. Redenbach, T. Herffurth, M. Trost, T. Gischkat, and P. Gospodnetić. SYNOSIS: Image Synthesis Pipeline for Machine Vision in Metal Surface Inspection. *Sensors*, 25(19):6016, 2025. `https://doi.org/10.3390/s25196016`.

[56] **N. Jeziorski** and C. Redenbach. Stochastic geometry models for texture synthesis of machined metallic surfaces: sandblasting and milling. *Journal of Mathematics in Industry*, 14:17, 2024. `https://doi.org/10.1186/s13362-024-00155-8`.

[89] K. Schladitz, C. Redenbach, T. Barisin, C. Jung, **N. Jeziorski**, L. Bosnar, J. Fulir, and P. Gospodnetić. Simulation of Microstructures and Machine Learning. In *Continuum Models and Discrete Systems (CMDS 2022)*, 457:243–256, Springer Proceedings in Mathematics & Statistics, 2024. `https://doi.org/10.1007/978-3-031-58665-1_19`.

Preprint

[55] **N. Jeziorski**, P. Gospodnetić, and C. Redenbach. Synthetic Defect Geometries of Cast Metal Objects Modeled via 2d Voronoi Tessellations. *arXiv*, preprint arXiv:[2602.05440], 2026. `https://doi.org/10.48550/arXiv.2602.05440`

Curriculum Vitae

01/2024 - 12/2025	Scientific Assistant Fraunhofer Institute for Industrial Mathematics ITWM Kaiserslautern, Germany
02/2022 - 12/2023	Scientific Assistant RPTU University Kaiserslautern-Landau, Germany
since 02/2022	PhD in Mathematics RPTU University Kaiserslautern-Landau, Germany
10/2018 - 12/2021	M. Sc. in Mathematics University of Kaiserslautern, Germany
10/2014 - 01/2019	B. Sc. in Mathematics University of Kaiserslautern, Germany

Akademischer Werdegang

01/2024 - 12/2025	Wissenschaftliche Mitarbeiterin Fraunhofer-Institut für Techno- und Wirtschaftsmathematik ITWM Kaiserslautern, Deutschland
02/2022 - 12/2023	Wissenschaftliche Mitarbeiterin Rheinland-Pfälzische Technische Universität Kaiserslautern-Landau, Deutschland
seit 02/2022	Promotionsstudium in Mathematik Rheinland-Pfälzische Technische Universität Kaiserslautern-Landau, Deutschland
10/2018 - 12/2021	M. Sc. in Mathematik Technische Universität Kaiserslautern, Deutschland
10/2014 - 01/2019	B. Sc. in Mathematik Technische Universität Kaiserslautern, Deutschland