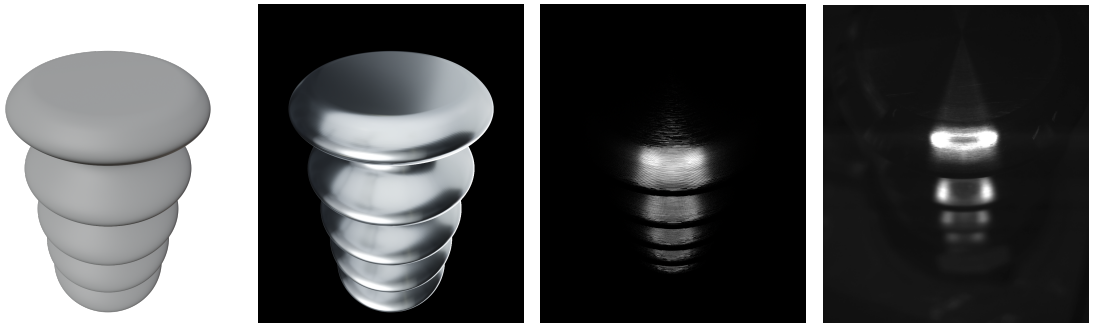
Texture Synthesis for Automated Visual Surface Inspection Planning

Lovro Bosnar
lovro.bosnar@itwm.fraunhofer.de

Hans Hagen
hagen@informatik.uni-kl.de

Petra Gospodnetic
petra.gospodnetic@itwm.fraunhofer.de

03.12.2021.



(a) CAD model (b) Reflectance model with uniform surface parameters (c) Reflectance model with varying parameters defined with our texturing model (d) Original image

Figure 1: Comparison of synthesized images and original image.

Abstract

Automated surface inspection planning is a research area, aimed to enhance inspection in automated and custom product manufacturing. Visual surface inspection focuses on two main tasks. First, camera and illumination placement for achieving the required coverage of the inspected product. Second, the development of defect detection algorithms. Both tasks greatly benefit from realistic and automated image synthesis of the inspected object using computer graphics modeling and rendering methods. The realism of synthesized images greatly depends on object material, whose properties are greatly influenced by texture. In our work, we focus on texture synthesis and its application for visual surface inspection planning. Our research was motivated by the complexity and vastness of textures that have to be recreated as well as their consistent application on complex, free-form geometry for the purposes of automated texture synthesis. In this paper, a framework that stemmed from our research is discussed. Furthermore, we provide an overview and shortly discuss several texture synthesis models that we have developed. Presented methods enable non-computer graphics experts, such as inspection planning experts, to recreate a range of patterns often present in the machined surfaces, using only several parameters, during the synthesis of representative product image data.

1 Introduction

The goal of the surface inspection is to find defects on a product. In order to do so, the positioning of acquisition hardware (i.e. illumination and camera), relative to the inspected object, is needed in order to obtain required coverage (see Figure 2). Also, image processing algorithms for defect detection must be developed and utilized on covered surface areas. Although the inspection process is automated once the configuration is present, the planning of such configuration is still done by experts which is time-consuming and error-prone as discussed by Gospodnetic et al. [1]. Furthermore, in the context of Industry 4.0, where automation of customized products manufacturing is introduced, manual inspection planning becomes infeasible.

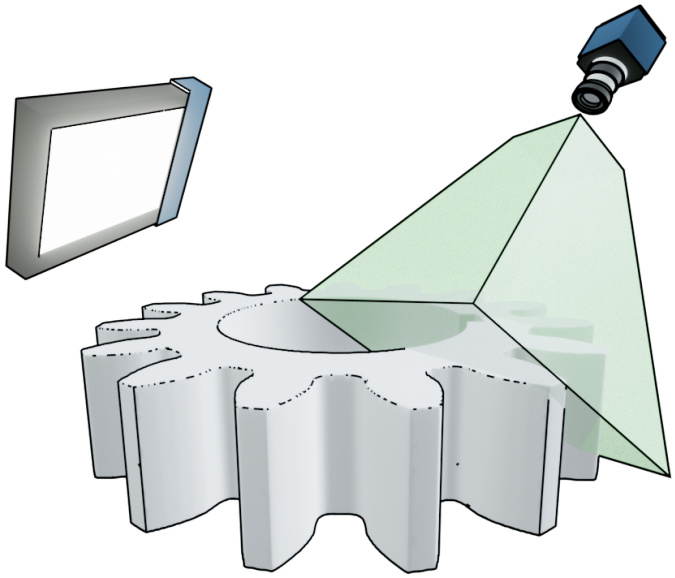


Figure 2: Illustration of surface inspection environment: illumination, camera and inspected object

Research aiming to automate the inspection planning process is an active area discussed by Gospodnetic et al. [2, 3], Mohammadikaji et al. [4] and Mosbach et al. [5]. Planning of camera and illumination placement, as well as the development of image processing algorithms for defect detection, requires a large amount of representative product image data. Representative images are images that are taken under controlled conditions and for which is known exactly which part of the object they show. Due to challenges such as the lack of product samples, complex environment, lack of particular defects (which also have to be acquired), obtaining the right amount of images from the right position becomes difficult very fast. Therefore, the process of obtaining representative images can greatly benefit from image synthesis using computer graphics modeling and rendering techniques discussed by Mohammadikaji [6], Bosnar et al. [7] and Reiner [8]. Synthesized product image data show inspected objects as they would have been seen from a real camera in a real inspection environment. Therefore, synthesized images can be used for the automated development and evaluation of visual inspection plan.

To synthesize representative product image data, it is required to model and render a 3D scene containing inspected object, illumination and camera used in inspection. In order to achieve realism,

modeling and rendering of the 3D scene must be performed in a physically based manner. The material definition of the inspected object greatly determines its realism in synthesized images. For the purpose of our work, we decompose the material in the microfacet-based reflection model discussed in Walter et al. [9] and texture which defines material properties over the surface. To further explain material decomposition and highlight the importance of texture, let's consider Figure 1. A synthesized image of the inspected object assigned only with reflectance model, and therefore uniform parameters over a surface, is presented in Figure 1(b). Comparing this synthesized image with the real image in Figure 1(d), we can see that both spatial variation and complex reflection is missing. On the other hand, the synthesized image of the inspected object assigned with reflectance model and our texture synthesis models is presented in Figure 1(c). Comparing this synthesized image with the real image in Figure 1(d), we can see that both spatial variations, as well as complex, anisotropic reflection, is present.

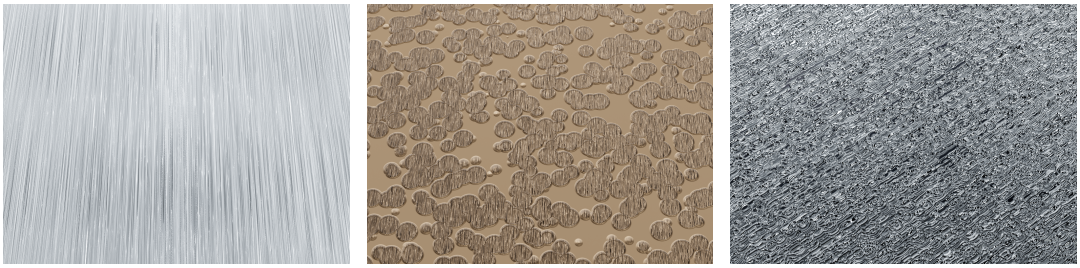
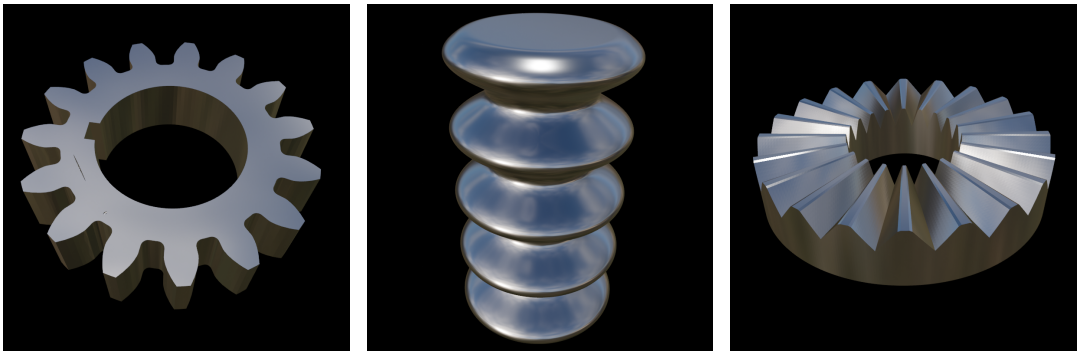


Figure 3: Examples of machined surface textures manually recreated using Blender [10] based on Kikukawa image samples [11]



(a) Gear object (b) Spring object (c) Hirth object

Figure 4: Synthesized images depicting complex, free-form product geometries simulated with uniform reflectance parameters

Texture synthesis for representative product image data in surface inspection is constrained by two main requirements. First, the texture synthesis model must generate physically based, spatially varying surface information, in an automated manner. Second, texture synthesis model must be accessible to inspection planning expert who is not necessarily an artist or computer-graphics expert. This implies that the texture synthesis model must solve two main tasks. First, it must recreate

texture patterns as similar as possible to the real texture pattern that appears in the line of products being inspected. This problem is hard because texture patterns are complex and vast (see Figure 3). Second, it is not only enough to recreate texture patterns on a planar surface. Texture must be consistently recreated over complex, free-form object surface (see Figure 4).

The two main contributions of this paper are the following. First, we will discuss the framework that stemmed from our research on automated texture synthesis models development. This framework consists of four steps, namely classification, analysis, synthesis and application which will be discussed. Second, we will provide an overview of texture synthesis models that we have developed. Each model was used to synthesize several images that will be presented. Presented models are aimed to be used by inspection planning expert, for automated synthesis of texture patterns often present on machined surfaces without artistic or computer graphics knowledge.

2 Method

2.1 Image Synthesis Environment

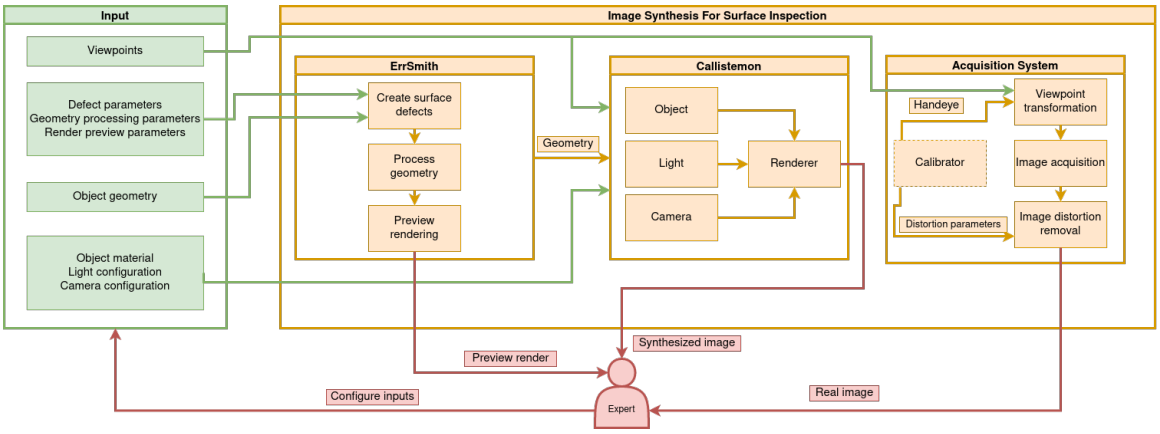


Figure 5: Image synthesis pipeline presented by Bosnar et al. [7]

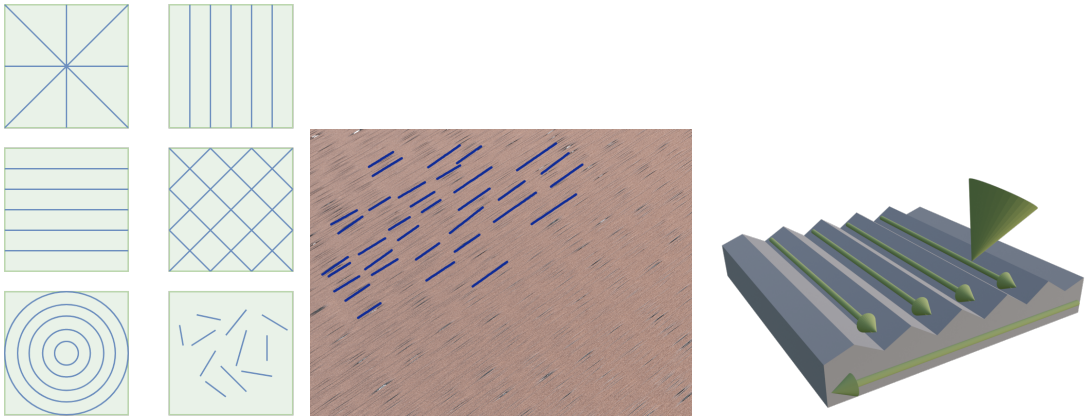
The image synthesis environment used for the texture synthesis framework is introduced by Bosnar et al. [7] in form of a pipeline. The general idea of the pipeline is given in Figure 5. Inspection planning expert provides relevant information as well as expected defects. The relevant information consists of viewpoints (i.e. camera positions), inspected object, illumination and camera used in the inspection process. The pipeline is then used for obtaining both synthesized and real representative images of the inspected product. The first module, ErrSmith, is used to imprint defects on inspected object mesh based on expected defect information. The second module, Callistemon, is the bridge between surface inspection data and appleseed rendering engine [12]. In this module, a 3D scene containing inspected object, illumination and camera used in the inspection environment is simulated and rendered from given viewpoints. Both original and defected inspected object mesh can be used. Finally, the acquisition system is used to obtain the real images, from the same viewpoints as used in the Callistemon module. All obtained images are available to the expert for evaluation and development of the inspection plan. In this paper, we focus on the Callistemon module because it is, among others, responsible for material and texture modeling for realistic image synthesis.

2.2 Development Framework for Texture Synthesis Models

Framework for texture synthesis models development consists consists of four tightly intertwined steps:

- Classification
- Analysis
- Synthesis
- Application

The first step, classification, is concerned with determining which classes of textures are required to be modeled. The choice depends on the surface texture present in the line of the products being inspected. Generally, the surface texture is determined with a certain manufacturing process as discussed in Black et al. [13]. In our work, we focus on metal machining which is a subset of manufacturing processes. Machining processes are deterministic and standardized which implies that resulting surfaces are also deterministic and standardized as discussed by Groover [14]. Standardization of surfaces is discussed in machining theory and surface metrology, e.g., ISO [15]. However, it has not been utilized in Computer graphics until now. For example, the surface can be decomposed in roughness, waviness and lay. Surface roughness refers to finely spaced surface irregularities. Waviness describes surface irregularities with spacing greater than that of surface roughness. Lay is determined by production method and it described predominant surface pattern direction. Standardization of lay is given in Figure 6(a).



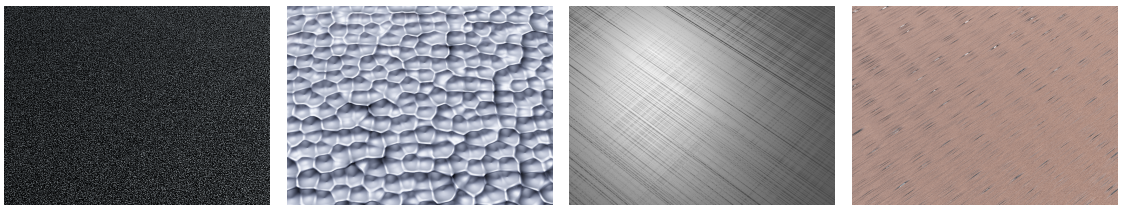
(a) Standardized surface lay types Black et al. [13]

(b) Image processing: texture elements and their relationships

(c) Tool movement

Figure 6: Texture analysis

Examples of machined surfaces are given in Figure 3. Here is also important to note that texture class features determine texture modeling which will be covered later. For example, textures can contain anisotropic or isotropic features that will determine the texture modeling approach (see Figure 7). For the purpose of our work, we consider scratched or grooved surfaces anisotropic because they form certain structures determined by the shape and orientation of texture elements. Isotropic textures are, on the other hand, more stochastic and without structure.



(a) Isotropic pattern 1 (b) Isotropic pattern 2 (c) Anisotropic pattern 1 (d) Anisotropic pattern 2

Figure 7: Isotropic and anisotropic examples of machined surface textures manually recreated using Blender [10] based on Kikukawa image samples [11]

The analysis step is concerned with understanding the texture pattern of the selected texture class. Understanding texture pattern can be gained in several intertwined ways. First, surface metrology and machining theory contain valuable information regarding standardized surfaces, e.g. machining theory discussed by Groover [14]. Next, image samples of real textures can be both observed directly or analyzed using image processing techniques to understand texture elements and their relationships (see Figure 6(b)). Finally, since the surface texture is completely defined with the machining process, insight into texture can be gained by understanding the machining tool movement (see Figure 6(c)).

Once the insight into texture pattern is gained, the texture modeling can begin. For the purpose of our work, we use procedural texture synthesis methods discussed by Ebert et al. [16]. These methods allow algorithmical encoding of the texture pattern. The resulting model is a black box with several parameters which fully determine the resulting texture. The procedural texture synthesis model is evaluated during rendering and it produces spatially varying surface information. For example, Figures 1(c) and 1(b) show spring object with and without algorithmically defined texture.

Finally, the application is concerned with consistent texture mapping or encoding over complex, free form geometry. It is worth noting that the application step is particularly intertwined with the modeling step and here is presented separately for the sake of clearness. As mentioned, texture features determine modeling and therefore application step. For example, modeling anisotropic texture features require encoding information about texture element orientation and relationship, which is easier to perform on 2D than 3D surface. Therefore, if modeling is done in 2D texture space, the application step must take care of mapping texture from 2D to 3D space (e.g., using triplanar projection discussed by Weiss et al. [17]). On the other hand, we found that isotropic textures can be more easily encoded directly on the 3D surface which doesn't require additional mapping as it is a case with, e.g., Worley cellular noise [18]. Besides texture mapping and encoding, the texture application step must also take into account correct texture rendering. For example, texture introduces additional geometrical details for which is required to take into account shadowing, masking and interreflection effects as discussed by Schussler et al. [19]. Also, texture introduces high-frequency details which must be efficiently sampled during shading as discussed by Jakob et al. [20].

2.3 Texture Synthesis Models

Based on the framework described in Section 2.2 we have developed several texture synthesis models. The first set of models is used to recreate circular, parallel and radial texture patterns, according to the standardized surface lays given in Figure 6(a). Models are introduced by Bosnar et al.

[21]. The circular pattern (see Figure 9) is modeled using a large number of torus elements placed concentrically. Additional pattern complexity is introduced by minor torus radius perturbation using noise. The parallel pattern (see Figure 10) is modeled using a large number of parallelly placed cylinder elements. Additional pattern complexity is introduced by perturbation, i.e., adding noise to cylinder radius. Radial pattern (see Figure 11) is modeled using a large number of cylinder elements that are placed radially. Additional complexity is introduced by perturbing cylinder radius using noise.

The second set of models is used to recreate straight and curved scratches. Models are introduced by Bosnar et al. [22]. Straight scratches (see Figure 12) are modeled as follows. First, the surface is divided into a square lattice where each lattice cell is of equal size (i.e. scale). Each lattice cell can contain an arbitrary number of cylinders lying on the surface. The cylinder scale is defined by the cell scale. The final texture is a result of stacking multiple, scale-decreasing lattices with varying cylinder parameters. That is, if we take a look at highest scale lattice cells, then they can contain cylinder elements or/and be further divided into smaller lattices whose cells follow the same rules. Curved scratches (see Figure 14) are following the same conceptual idea but instead of straight cylinders, curved cylinders are used. Cylinder curving is achieved by displacing cylinder position using noise.

One particular advantage of procedural texture synthesis models is that the positions of generated texture elements are known. Using this knowledge it is possible to perform color-coding of texture elements and effectively achieve texture labeling. Color-coding of texture elements is also discussed in [22].

The third set of textures are misc textures that were implemented based on either existing methods or a combination of existing methods. The knurling pattern (see Figure 17(c)) is based on two triangle waves that are placed at different angles. Checker pattern (see Figure 17(b)) is very common procedural texture often available in 3D modeling programs. This method is dividing a object into regular cubes effectively resulting in checkerboard texture. Bumpy texture (see Figure 17(a)) is based on cellular basis function introduced by Worley [16]. Knurling, Checker and Bumpy patterns are also shortly discussed by Bosnar et al. [21]. Finally, Tsuchime pattern (see Figure 17(d)) is based on texture bombing procedural texturing method discussed by Glanville [23].

3 Results

Models described in Section 2.3 are used for image synthesis via Callistemon module described in Section 2.1. The simulated scene contains inspected object, illumination and camera. The 3D scene is rendered using appleseed rendering engine which is an offline, physically based, production rendering engine. For rendering, we have used path tracing light transport with 150 samples per pixel.

Inspected objects were represented by gear, spring and hirth geometry (see Figure 4). Material is defined by appleseed's metal reflection BRDF model and texture synthesis model which generates surface properties (i.e. perturbed normals) for reflection model evaluation. Texture synthesis models, discussed in Section 2.3, are implemented in OSL [24] which is a standard shading language in the computer graphics industry.

The illumination source is defined by torus-shaped mesh geometry and diffuse-emission material which is placed in a black, non-emissive environment. Emissive material is also specified using OSL. Light position is relative to the camera, meaning that it is mounted on the camera and its position transformation follows the camera transformation.

The camera is defined by appleseed's pinhole camera model with a pixel size of 0.00824 mm,

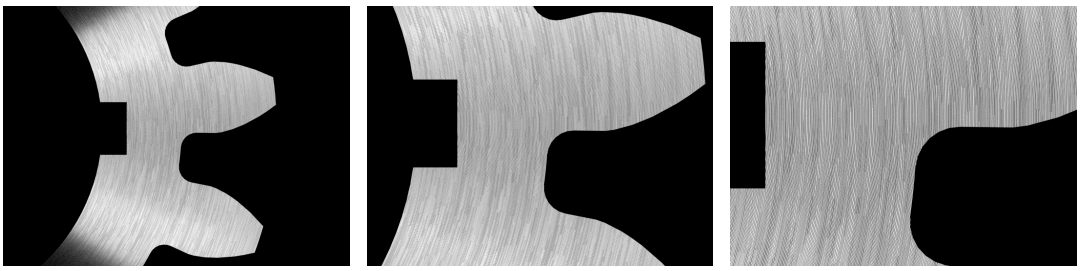


Figure 8: Zoom in on one circular texture pattern on gear object

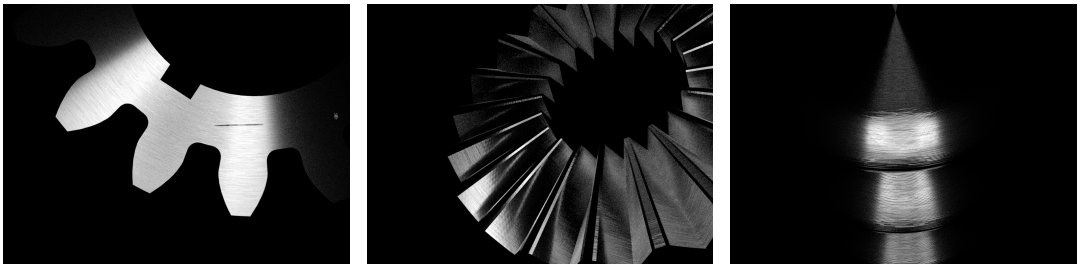


Figure 9: Circular pattern applied gear, hirth and spring objects

focal length 12.93 (if not otherwise stated) and resolution 1024x768.

Figure 8 shows the circular pattern applied on the gear mesh object with camera focal lengths of 18, 30 and 50, effectively zooming in on the pattern. The circular pattern consists of a large number of concentrically placed torus elements simulating circular brushing. Further, in Figure 9, the circular pattern is used for texturing gear, hirth and spring mesh objects. Figure 10 show a parallel texture pattern, consisting of a large number of parallelly placed cylinders simulating parallel brushing, on the gear, hirth and spring mesh objects respectively. Figure 11 shows the radial pattern, consisting of a large number of radially placed cylinders simulating radial brushing, on gear, hirth and spring objects respectively.

Figure 12 shows straight scratches on gear object rendered with camera focal lengths of 18, 30 and 50, effectively zooming in on the pattern. Straight scratches pattern consists of multiple layers of scale-decreasing cylinder segments with different orientations. Figure 13 shows straight scratches applied on gear, hirth and spring objects. Figure 14 show curved scratches on gear object rendered with camera focal lengths 18, 30 and 50, effectively zooming in on on the pattern. Curved scratches pattern consists of multiple layers of scale-decreasing, curved cylinder segments with different orientations. Figure 15 show curved scratches texture applied on gear, hirth and spring objects.

Figure 17 show several misc textures. Bumpy texture, simulating casting surface (i.e. bumps and dents), is presented on hirth object in Figure 17(a). Checker texture, depicting surface consisting of square elements, is presented on spring object in Figure 17(b). Knurling texture, depicting pyramid-like bumps and dents, is presented on hirth object in Figure 17(c). Tsuchime texture, depicting surface consisting of disk-like elements, is presented on spring object in Figure 17(d).

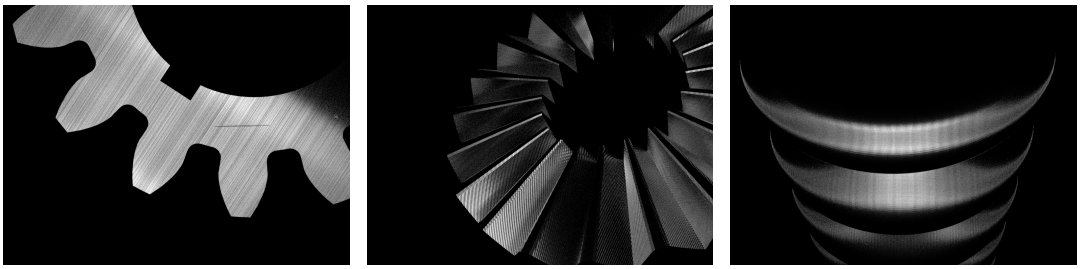


Figure 10: Parallel pattern applied gear, hirth and spring objects

Figure 11: Radial pattern applied on gear, hirth and spring objects.

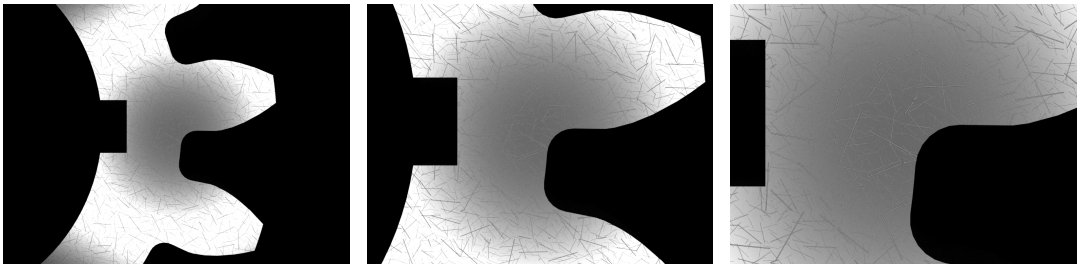


Figure 12: Zoom in on one straight scratches texture pattern on gear object.

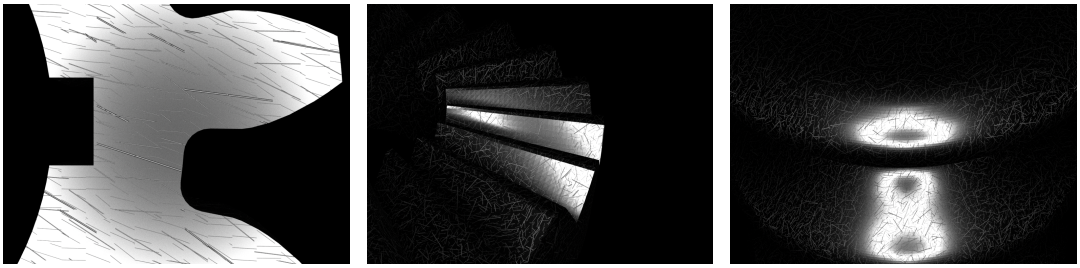


Figure 13: Straight scratches texture pattern on gear, hirth and spring objects

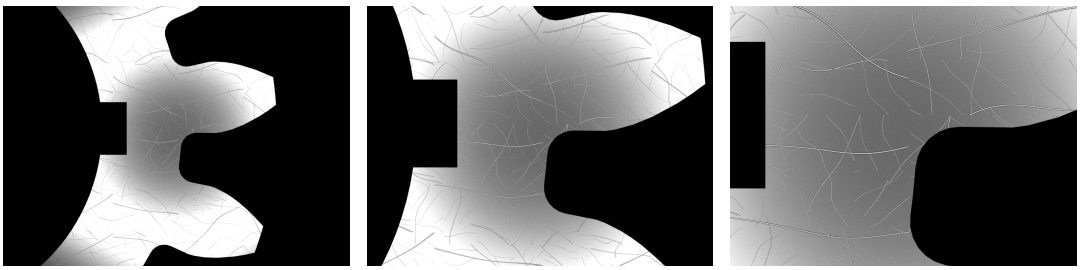


Figure 14: Zoom in on one curved scratches texture pattern on gear object.

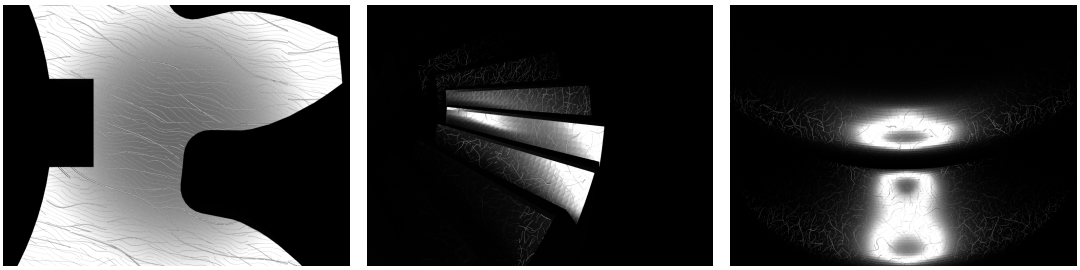
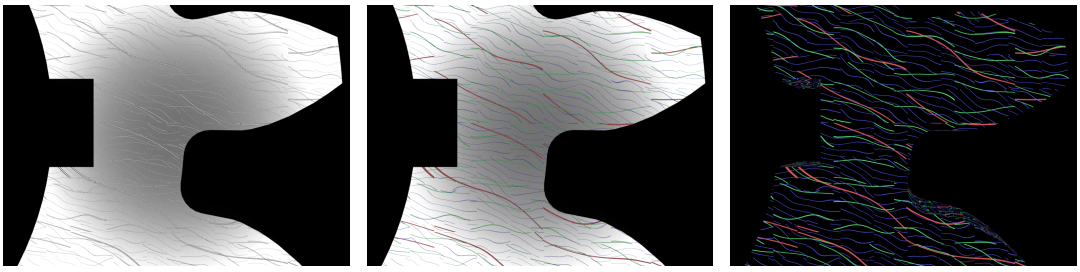


Figure 15: Curved scratches texture pattern on gear, hirth and spring objects

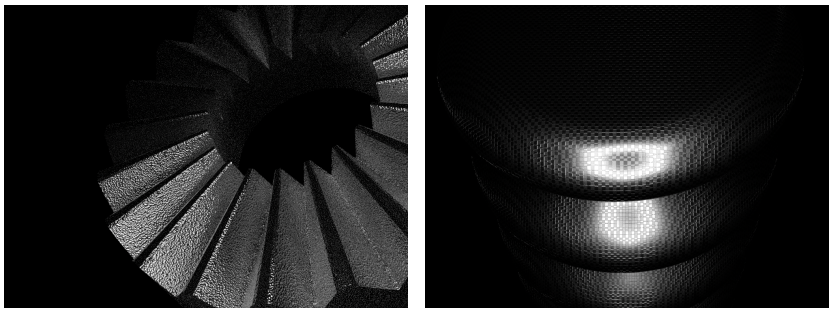


(a) Original curved scratches texture

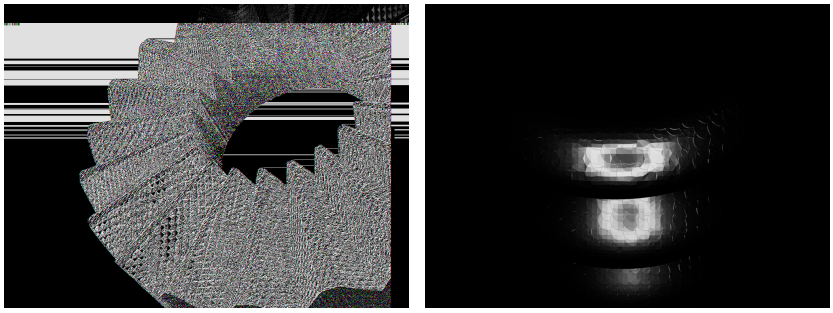
(b) Color-coded scratches on original texture

(c) Color-coded scratches only

Figure 16: Example of labeling texture elements per layers.

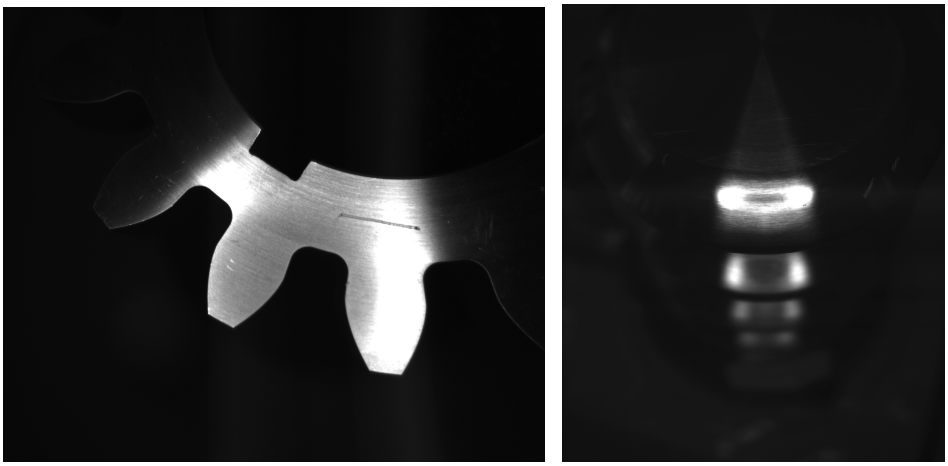


(a) Bumpy pattern, hirth object (b) Checker pattern, spring object



(c) Knurling pattern, hirth object (d) Tsuchime pattern, torus object

Figure 17: Misc texture patterns on complex, free-form objects.



(a) Gear object (b) Spring object

Figure 18: Real images of inspected product objects

4 Discussion

Texture synthesis models for surface inspection must automatically recreate realistic texture pattern over complex, free-form geometry and be usable by inspection planning experts. Development of such models is faced with the challenge of both complexity of the texture pattern that has to be recreated, as well as consistent application on complex, free-form geometry. We have decomposed this problem into four steps: classification, analysis, modeling and application. Following these steps, we have developed several texture synthesis models that recreate common machining surface textures present in surface inspection.

Presented models define geometrical surface properties, that is, surface normal vectors. The defined geometry is then used during physically based shading and light transport. Therefore, the resulting pixel color is physically plausible.

From Figures 8, 12 and 14 it is observable that despite different imaging distances, models produce high quality surface features resolution.

Presented models can cover an arbitrary surface area without tiling artifacts and without problems caused by free-form surface, e.g., see Figures 13 and 15. Next, models are configurable by a set of predetermined parameters. This means that a wide range of texture instances belonging to a certain texture class can be recreated. For example, compare Figures 12 and 13 or 14 and 15. Finally, models only require the specification of the predetermined parameters. Using those parameters, the models synthesize texture relying only on intersection/shading point position and normal which are known during rendering without mesh preprocessing or any other user input. For more details, refer to work done by Bosnar et al. [21] and [22].

Surface textures like grooves cause different types of anisotropic reflectances depending on grooves' orientation and shape (see Figure 18). For example, circular, parallel or radial grooves will cause the different shapes of reflected light and highlights, e.g., see Figures 9, 10, 11. As we can see, the presented models inherently cause anisotropic reflectance due to the visible geometry which they generate.

The presence of texture greatly influences the object's appearance and thus visual surface inspection. In Figures 9, 10, 11 we can see that surface texture causes wide range of different bright and dark areas which can hide portions of the surface. Therefore, surface coverage is influenced by the texture which must be taken into account during illumination and camera positioning. Also, visible texture patterns (e.g., see Figure 12) must be present in representative images so that the development of defect detection algorithms can take all the surface details into account. The development of defect detection algorithms often requires images with labeled texture elements. In figure 16 we can see that presented models are capable of color-coding the texture elements which can be used as labels.

5 Conclusion

Development of automated texture synthesis models for applications in surface inspection planning must tackle both recreation of complex texture pattern as well as its consistent application on complex, free-form geometry. The resulting model must produce physically based, spatially varying surface information for photo-realistic rendering and be usable by inspection planning expert without computer graphics or artistic knowledge. To make this problem more tractable, we have presented a framework for approaching the automated texture synthesis models development. This framework stemmed from our research and serves as the foundation for further research directions in automated texture synthesis. Furthermore, we have presented an overview of texture synthesis

models that we have developed using this framework. Presented models enable inspection planning expert to automatically recreate texture patterns frequently found in inspected machined surfaces in a consistent and repeatable manner.

Acknowledgment

Authors would like to thank Fraunhofer ITWM, Image processing department for the support as well as appleseed developers for continuous support.

References

[1] P. Gospodnetic, D. Mosbach, M. Rauhut, and H. Hagen, "Flexible surface inspection planning pipeline," in *2020 6th International Conference on Control, Automation and Robotics (ICCAR)*. IEEE, 2020, pp. 644–652.

[2] P. Gospodnetic, M. Rauhut, and H. Hagen, "Surface inspection planning using 3d visualization," 2020.

[3] P. Gospodnetic, D. Mosbach, M. Rauhut, and H. Hagen, "Viewpoint placement for inspection planning," *Machine Vision and Applications*, 2021.

[4] M. Mohammadikaji, S. Bergmann, J. Beyerer, J. Burke, and C. Dachsbacher, "Sensor-realistic simulations for evaluation and planning of optical measurement systems with an application to laser triangulation," *IEEE Sensors Journal*, vol. 20, no. 10, pp. 5336–5349, 2020.

[5] D. Mosbach, P. Gospodnetic, M. Rauhut, B. Hamann, and H. Hagen, "Feature-driven viewpoint placement for model-based surface inspection," *Machine Vision and Applications*, vol. 32, no. 1, pp. 1–21, 2020.

[6] M. Mohammadikaji, "Simulation-based planning of machine vision inspection systems with an application to laser triangulation," 2020.

[7] L. Bosnar, D. Saric, S. Dutta, T. Weibel, M. Rauhut, H. Hagen, and P. Gospodnetic, "Image synthesis pipeline for surface inspection," 2020.

[8] J. Reiner, "Rendering for machine vision prototyping," in *Optical Design and Engineering III*, vol. 7100. International Society for Optics and Photonics, 2008, p. 710009.

[9] B. Walter, S. R. Marschner, H. Li, and K. E. Torrance, "Microfacet models for refraction through rough surfaces." *Rendering techniques*, vol. 2007, p. 18th, 2007.

[10] Blender Online Community, *Blender - a 3D modelling and rendering package*, Blender Foundation, Blender Institute, Amsterdam, 2021. [Online]. Available: http://www.blender.org

[11] [Online]. Available: https://www.kikukawa.com/en/

[12] F. Beaune, E. Tovagliari, L. Barrancos, S. Agyemang, S. Basu, M. Bhutani, L. Bosnar, R. Brune, M. Chan, J. M. M. Costa, H. Crepaz, J. Deng, J. Dent, M. Dhiman, D. Fevrier, K. R. Iyer, D. Lahiri, K. Masson, G. Olson, A. Pandey, J. Park, S. Pogosyan, B. Samir, O. Smolin, T. Vergne, L. Wilimitis, and L. Zawallich, "appleseed," Sep. 2019. [Online]. Available: https://doi.org/10.5281/zenodo.3456967

[13] J. T. Black and R. A. Kohser, *DeGarmo's materials and processes in manufacturing.* John Wiley & Sons, 2020.

[14] M. P. Groover, *Fundamentals of modern manufacturing: materials, processes, and systems.* John Wiley & Sons, 2020.

[15] ISO, "Iso 1302 (2002) geometrical product specifications (gps)—indication of surface texture in technical product documentation," 2002.

[16] D. S. Ebert, F. K. Musgrave, D. Peachey, K. Perlin, J. C. Hart, and S. Worley, *Texturing & modeling: a procedural approach.* Morgan Kaufmann, 2003.

[17] S. Weiss, F. Bayer, and R. Westermann, "Triplanar displacement mapping for terrain rendering," 2020.

[18] S. Worley, "A cellular texture basis function," in *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, 1996, pp. 291–294.

[19] V. Schüssler, E. Heitz, J. Hanika, and C. Dachsbacher, "Microfacet-based normal mapping for robust monte carlo path tracing," *ACM Transactions on Graphics (TOG)*, vol. 36, no. 6, pp. 1–12, 2017.

[20] W. Jakob, M. Hašan, L.-Q. Yan, J. Lawrence, R. Ramamoorthi, and S. Marschner, "Discrete stochastic microfacet models," *ACM Transactions on Graphics (TOG)*, vol. 33, no. 4, pp. 1–10, 2014.

[21] L. Bosnar, M. Rauhut, H. Hagen, and P. Gospodnetic, "Texture synthesis for surface inspection," to appear.

[22] ——, "Parameter-centered texture synthesis for surface inspection," to appear.

[23] R. Fernando *et al.*, *GPU gems: programming techniques, tips, and tricks for real-time graphics.* Addison-Wesley Reading, 2004, vol. 590.

[24] L. Gritz, C. Stein, C. Kulla, and A. Conty, "Open shading language," in *ACM SIGGRAPH 2010 Talks*, 2010, pp. 1–1.