Visual Prototyping of Knitwear

Dissertation

zur
Erlangung des Doktorgrades (Dr. rer. nat.)
der
Mathematisch-Naturwissenschaftlichen Fakultät
der
Rheinischen Friedrich-Wilhelms-Universität Bonn

vorgelegt von

Elena Trunz

aus
Moskau, Russland

Bonn 2023

Angefertigt mit Genehmigung der Mathematisch-Naturwissenschaftlichen Fakultät
der Rheinischen Friedrich-Wilhelms-Universität Bonn

1. Gutachter: Prof. Dr. Reinhard Klein
2. Gutachter: Prof. Dr. Michael Weinmann

Tag der Promotion: 24.04.2023
Erscheinungsjahr: 2023

Acknowledgements

First of all, I would like to thank Reinhard Klein for giving me the opportunity to work in his department, for supervising and supporting my Ph.D. studies and for giving me a lot of valuable advice over the past years. In this context, I would also like to thank Norbert Blum for establishing a connection to Reinhard Klein in the first place.

Furthermore, thanks to Michael Weinmann, Jonathan Klein, Sebastian Merzbach and Lukas Bode for our many discussions and laughs, and for your co-authorship and support, especially before the paper deadlines. I also thank my other co-authors Jan Müller, Reinhard Klein, Ralf Sarlette, Thomas Schulze, Stefan Hartmann, Björn Krüger, Ilia Mazlov and Matthias Hullin.

Additionally, I would like to thank all my colleagues in the Visual Computing Group. It has been a pleasure to work with you.

Moreover, I want to thank Michael Weinmann and Reinhard Klein for proofreading this thesis and Patrick Stotko for his awesome thesis template.

Finally, I am very thankful to all my family and friends for their support and endless patience during my studies and especially during the last year.

Abstract

In this thesis, we address the problem of prediction and editing of the virtual appearance of knitted cloth. Among the most accurate approaches to model fabrics and other complex materials is to represent them in terms of dense Bidirectional Texture Functions (BTFs) that describe the spatially varying material appearance under different viewing and lighting conditions parametrized on a proxy geometry. This requires taking an exhaustive set of photographs under different light and view directions to accurately reproduce the appearance of an existing material, and thus comes at a high cost regarding storage requirements, motivating the research of convenient compression methods. The current state-of-the-art neural approach did not fully exploit the capabilities of the proposed architecture, as the compression ratio did not depend on the complexity of the material. Therefore, in the first project presented in this thesis, we developed an approach to even further compress compact BTFs, that were compressed based on state-of-the-art neural compression, and let the compression ratio depend on the complexity of the material.

Besides high memory requirements, the usage of BTFs does not allow for easy editing operations. Moreover, BTFs are not suitable for macro-scale geometry representation. Both these limitations pose a problem in the context of visual prototyping of knitwear. There are three different levels of knitted cloth design that are important for our task. First, there is the choice of the yarn, which in turn consists of fibers of the same type or a mix of different fiber types. Then, the selected yarn is knitted into a chosen pattern producing the actual knitted cloth. This thesis aims to enable editing operations on all three scales: patterns, yarns and fibers. To this end, in two projects that constitute the second part of this thesis, we develop algorithms and models for macroscale representation of knitting patterns, mesoscale modeling of yarns, and visualization and easy editing of knitting yarns, fibers and patterns that we automatically induce from images of real yarns or knitted pieces, respectively.

Contents

Part I

Introduction

Chapter 1

Introduction

Fabrics are omnipresent in our daily life. They are used not only in the domain of clothing and fashion but also in the entertainment industry or for upholstery. Therefore, virtual design and modeling of fabrics and garments are very important for many applications. For example, there are many new designs of clothing created by the fashion industry every season. Such a creation process usually requires several iterative steps, many of them depending on the visualization of the garments or their fragments and without computer-aided design, those visualizations can only be achieved during the actual fabrication process. Similar examples can be observed for upholstery products, such as furniture or car seats. *Visual prototyping*, which denotes the prediction and editing of the appearance of virtual objects, following [Schröder, 2015], aims to replace some of these manufacturing steps by virtual ones. In this work, we focus on the visual prototyping of knitted cloth as a representative of one of the most complex and versatile types of fabrics, which is not only popular among designers but also among a large group of society interested in producing handcrafted clothing according to their own preferences.

One common way to approach the problem of appearance modeling of fabrics is to use an image-driven material representation technique, such as dense Bidirectional Texture Functions (BTFs)[Dana et al., 1999]. For each material, many photographs with cameras positioned at different angles and directional light coming from various directions need to be taken and stored. Using these measurements, one can reproduce the appearance of the corresponding original material. The accurateness of the reproduction depends on the number of measurements. The more images under different light and view directions are captured, the more truthful the reproduction becomes. BTFs have proven to accurately reproduce a wide range of real-world materials, including complex materials such as fabrics. However, one of the main drawbacks of this technique is its high memory complexity, which drastically reduces its practicability and arouses the requirement for a compression strategy. In order to address this problem, Rainer et al. [2019] developed a new compression strategy for BTFs, where the compact representation for BTFs is learned by an encoder-decoder network. During the training, the high-resolution textures are transformed into low-dimensional latent variables. After the learning procedure, only the latent variables, together with the decoder network need to be stored. This compressed representation can be directly used for rendering by combining the latent variables with the angles of the light and view directions

as an input query for the decoder and receiving the corresponding RGB values as an output. Since compression rates and storage requirements of a material sample in this approach are directly influenced by the dimension of the latent variables, a critical question of the BTF compression problem is: Which number of latent dimensions is the most suitable for each BTF? Rainer et al. [2019] heuristically chose eight as the number of dimensions, independent of the material, leaving the detailed material-dependent analysis as an open question. We address this question in the first part of the thesis and make neural BTF representation for fabrics even more compact by analyzing the latent space depending on the complexity of each fabric individually, ordering the latent variables according to their contribution to the reconstruction and determining the most suitable latent space dimensionality.

Another drawback of image-based appearance representation approaches, such as BTFs, is that performing editing operations is not easily possible. This is a major problem for the visual prototyping of clothes. In addition, BTFs cannot be used to represent macro-scale geometry, which is important when representing garments such as knitwear due to their inherent 3D structure. Besides setting and following color trends in fashion, the design of knitted cloth takes place at different levels of detail. On the one hand, the composition of the fabric needs to be chosen, and for knitwear this means deciding on the yarn, which can, in turn, contain either only one type of fiber or a mixture of different fibers. On the other hand, there is a vast amount of possible knitting patterns into which the yarn can be knitted. Ideally, we would like to be able to visualize and edit all these levels of detail (fibers, yarns and patterns). In order to do this, we need to model these three scales explicitly. Therefore, in the second part of the thesis, we focus on the macro-scale representation of knitwear in terms of knitting patterns, mesoscale representations of yarns and the plausible editing and visualization of knitting yarns and patterns. We also present algorithms for automatically deriving these representations of yarns and knitting patterns from images of real yarns and knitting patches.

1.1 Challenges

In the following, the major challenges that are tackled in this thesis are summarized.

Challenges in Determining Task-Dependent Latent Dimension Compact data representation techniques often transform the input data into a lower-dimensional latent space that represents the most important features of the input in a compact way. The main challenge of such compression methods is to find the best trade-off between the dimensionality of the latent space, which is directly linked to the compression rate, and the reconstruction error that occurs when approximating the input from the latent code. Due to runtime issues, this trade-off should be found without multiple training processes with different numbers of latent dimensions and the subsequent calculation of the reconstruction errors for each model. The nonlinearity of encoder-decoder functions and the fact that the latent variables are usually not independent present additional challenges because they make linear approaches like principal component analysis not suitable for the analysis of latent space.

Challenges in Inverse Procedural Modeling of Knitwear from a Single Image In the case of knitwear, the corresponding procedural model is represented through knitting instructions for patterns. In order to derive a correct knitting instruction from an image of a garment, it is necessary to both identify and localize the stitches contained in the pattern. The localization is required in order for the knitting instruction to be valid. In the case of knitting patterns consisting of the two most common stitch types, purl and knit, a valid stitch pattern is a regular grid of stitches. The main challenge of stitch identification from images is the vast amount of variations in appearance. Even though there exists only a small number of stitch types, the actual knitting pieces can be made of yarns of various thicknesses, fuzziness, material and other different properties. Depending on the pattern, the neighboring stitches often partially or sometimes even completely hide adjacent stitches or cast shadows on them. The problem becomes more challenging for hand-made knitting clothing. Depending on the style, skill and other characteristics of the person, who knitted the cloth, the stitches exhibit various deformations like stretching and can range from too tight to very loose throughout the piece. The challenge in localizing all the stitches in an image consists of recognizing the correct number of rows and columns in the pattern, which is especially difficult because, due to the aforementioned variations of stitches, the size of the stitches varies by a considerable amount. Currently, to the best of our knowledge, there exists no labeled database that would help to address these challenges with deep learning methods.

Challenges in Image-based Inverse Procedural Modeling of Yarns While for knitwear, the corresponding knitting instructions could be described based on a well-established procedural model, a broadly accepted model for the generation of knitting yarns is still under development. Although some procedural yarn models have been proposed in the literature, it remains a challenge to generate realistic yarns. After creating or selecting an appropriate procedural model for yarn generation, the next challenge is to gain control over the output. The most convenient way for an inexperienced user is to input an image of a desired yarn, and the system automatically infers the corresponding parameters. This inverse procedural modeling task is particularly challenging due to the previously mentioned huge appearance variations of yarns. Hairiness and other irregularities of yarns constitute, at this level of detail, an even more significant disturbance than in the case of textile pieces. Different illuminations and other light and camera parameter variations must also be taken into account. Similar to the case of knitwear, to the best of our knowledge, no labeled database of knitting yarns exist at the time of completion of this thesis.

1.2 Contributions

In the scope of this thesis, we developed methods to address the previously mentioned challenges. In particular, the main contributions of this thesis are:

- **Task-dependent Specification of the Latent Space Dimensionality in Encoder-Decoder Architectures** We developed a novel approach for analyzing, structuring and determining a suitable dimension of the latent space in compression schemes that utilize encoder-decoder architectures. We compute the contributions of latent

variables towards the reconstruction result using Shapley values and order the variables according to their contribution. The subsequent reconstruction of the data with ordered subsets of latent variables and the computation of the cumulative contribution provides a good measure of how many dimensions are suitable for the particular task, thus making the latent representation of the data even more compact. We evaluate our method on several compression applications to demonstrate its effectiveness.

- **Connection Between Shapley Values and Principal Component Analysis (PCA)** We establish a link between Shapley values and singular values and prove that in the case of a linear relation $Ax = y$ between the input data x and the output y of a model A, where a singular value decomposition of A and a PCA can be applied and the Eckard-Young-Mirsky theorem states how optimal low-rank approximation A_k can be approached, the ordering of the eigenvectors according to their Shapley values is equal to the ordering of the corresponding singular values. As a direct corollary, we follow that the optimal rank-k approximation of a linear model A can be computed from the first k elements according to the ordering based on Shapley values.
- **Inverse Procedural Modeling of Knitwear** We propose a novel approach for inferring knitting instructions from a single image that does not rely on a labeled database. Our method includes a template matching step to identify of the stitch types in the image as well as an integer linear programming step to find the correct stitch positions. It ensures that the resulting stitch grid is regular and, thus, constitutes a valid knitting instruction. Since knitted garments are produced by repetitions of the pattern and the actual pattern usually appears several times in the image, we developed a pattern size detection step for our pipeline that simultaneously corrects the possible errors that may result from the template matching step.
- **Extended Best Buddies Similarity Measure for Template Matching** We improved the existing similarity measure, which was successfully used in template matching tasks of various computer vision applications, by enhancing it with a gradient penalty. The extended version of the similarity measure has proven to outperform many state-of-the-art techniques when tested in the application of stitch type identification and localization in an image of knitted cloth.
- **Inverse procedural modeling of Yarns** We provide a database of very detailed, annotated synthetic yarn images of high resolution. In order to generate this database, we enhanced an existing yarn generator with several meaningful parameters to enable the synthesis of even more realistic yarns. Using our database of synthetic but realistic yarns, we developed a neural method for the induction of the required procedural parameters for the yarn generator from images of real-world yarns.

1.3 List of Publications

I have contributed towards the goal of visual prototyping of knitwear over the course of several research projects. These projects are part of this thesis, and the corresponding

publications are listed in the following.

- **Elena Trunz**, Michael Weinmann, Sebastian Merzbach, and Reinhard Klein.
  "Efficient structuring of the latent space for controllable data reconstruction and compression."
  Graphics and Visual Computing (GVC), 7, page 200059, 2022.
  DOI: 10.1016/j.gvc.2022.200059
- **Elena Trunz**, Sebastian Merzbach, Jonathan Klein, Thomas Schulze, Michael Weinmann, and Reinhard Klein.
  "Inverse Procedural Modeling of Knitwear."
  2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 8622–8631, 2019.
  DOI: 10.1109/CVPR.2019.00883
- **Elena Trunz**, Jonathan Klein, Jan Müller, Lukas Bode, Ralf Sarlette, Michael Weinmann, and Reinhard Klein.
  "Neural Inverse Procedural Modeling of Knitting Yarns from Images."
  arXiv:2303.00154 (under review), submitted to Computers & Graphics (CG), 2023.
  DOI: 10.48550/arXiv.2303.00154

Additionally, I also contributed to the following publications, which, however, are not part of this thesis:

- Stefan Hartmann, **Elena Trunz**, Björn Krüger, Reinhard Klein, and Matthias B. Hullin.
  "Efficient Multi-Constrained Optimization for Example-Based Synthesis."
  The Visual Computer / Proc. Computer Graphics International (CGI 2015), pages 893–904, 2015.
  DOI: 10.1007/s00371-015-1114-y
- Ilia Mazlov, Sebastian Merzbach, **Elena Trunz**, and Reinhard Klein.
  "Neural Appearance Synthesis and Transfer."
  Workshop on Material Appearance Modeling, pages 35–39, 2019.
  DOI: 10.2312/mam.20191311

1.4 Thesis Outline

The remainder of this thesis consists of the following chapters:

Chapter 2 contains the summary of the background information and discussion of the previous works that are related to our research. Next, Chapter 3 summarizes our peer-reviewed publication "Efficient structuring of the latent space for controllable data reconstruction and compression" [Trunz et al., 2022], where we introduced the idea of analyzing, ordering and finding a suitable dimension for the task-dependent latent space, based on the importance of the corresponding latent variables towards the reconstruction, with the help of the Shapley values. In the following Chapter 4, we summarize our peer-reviewed publication "Inverse Procedural Modeling of Knitwear" [Trunz et al., 2019], which presents a pipeline for a robust

inference of knitting instructions from a single image of a knitting patch. Chapter 5 contains our publication "Inverse Procedural modeling of knitted yarns" [Trunz et al., 2023], which is currently under review and available online as a preprint. It presents an approach for the inference of geometric yarn and flyaway parameters from a photo of knitting yarn and a corresponding procedural yarn generator, together with an annotated yarn database. Finally, the thesis concludes with a discussion of the contribution of the introduced techniques and remaining limitations as well as an outlook regarding potential future developments.

Chapter 2

Background and Related Work

In this chapter, we give an overview of previous works on the appearance and geometry modeling of fabrics and briefly review some basic information about the elements of knitted cloth that is relevant in the context of this thesis. We assume the reader to be familiar with basic deep-learning concepts, which are explained in detail by Goodfellow et al. [2016].

2.1 Appearance Modeling of Fabrics

The appearance of fabrics and other materials is determined by the complex interactions of light, optical material properties of the surface and surface geometry. These light-scattering interactions can be categorized according to the size of the geometric features that affect them. In the Computer Graphics community, it is common to differentiate between features on three scales, although one can come across different terminologies for those. For example, Westin et al. [1992] use the expressions *microscale*, the *milliscale* and the *object scale*, whereas Fournier [1992] denotes the corresponding scales as *microscopic*, *mesocsopic* and *macroscopic* scale. Throughout this thesis, we will follow the latter terminology:

- On the level of the microscopic scale, there are object features that correspond to microscopic surface structures (e.g. roughness) and physical properties of the material, such as the index of refraction and absorption coefficient. These features are not visible to the human eye, but are responsible for the reflectance of the object (Fig. 2.1, left).
- Structures at the mesoscopic scale correspond to fine details in the surface geometry that are barely visible and are still considered part of the material rather than the geometric shape of the object. Examples include scratches, engravings or yarns and fibers as mesoscopic features of knitted cloth (Fig. 2.1, center).
- The coarsest level is the macroscopic scale. Its structures define the geometric shape of the object. However, the boundaries between the scales are not clearly distinguished and can be chosen based on some prior information about the task, such as the expected distance of the user. For our visual prototyping task, knitting patterns are viewed as structures of the macroscopic scale (Fig. 2.1, right).

Figure 2.1: Illustration of geometric features of knitwear at different scales. Left: The microimage taken with an electron microscope depicts the features of a single wool fiber on the microscopic scale (image taken from [Robbins, 2013]); Middle: On the photograph of the yarn, its virgin wool fibers are clearly visible. Right: A patch of knitwear shows the geometric pattern.

One can further group these scales into two classes and separate the material of an object (the microscopic and the mesoscopic scales) and its shape (the macroscopic scale). We expect the designer to view the knitting cloth at a relatively close distance, therefore, in our case, knitting yarns and fibers serve as material and knitting patterns define the shape of the object. Whereas the usual representation of the shape is a polygon mesh, there are several commonly used models to represent optical material properties. In the following, we briefly describe representations that are most relevant to the content of the thesis. For extensive surveys on reflectance modeling, we refer to [Haindl and Filip, 2013; Weinmann et al., 2016].

Surface Reflectance There are two classical ways to model material appearance. One of them is to derive analytical reflectance models, and the other is to use real-world measurements directly.

One of the most popular ways to represent appearance at the microscale is through *Bidirectional Reflectance Distribution Functions* (BRDFs) [Nicodemus et al., 1977]. BRDFs belong to the class of analytical appearance models and describe the transformation of incoming irradiance to outgoing radiance at the considered surface point depending on only four parameters: two spherical angles for the direction of incoming light and another two spherical angles for the outgoing direction. These models achieve good results for the materials where light scattering is a local phenomenon. However, they cannot model the effects of the mesoscopic scale and are therefore not sufficient to describe materials with subsurface scattering, such as fabrics.

A very powerful approach for realistic appearance modeling of complex materials, such as fabrics, at both the microscale and the mesoscale, is the image-based technique of *Bidirectional Texture Functions (BTFs)* [Dana et al., 1999]. An exhaustive set of photographs is taken under a directional light source by densely sampling different light and viewing directions. BTFs are six-dimensional functions that take as input two spherical angles for the incoming light direction, two angles for the camera direction and 2D coordinates of the position on the surface and output the corresponding reflectance value. These measurements capture effects such as subsurface scattering, occlusion or self-shadowing, which allow accurate reproduction of fabrics and other complex materials.

Four-dimensional functions defined by the evaluation of the BTF at a given 2D point on the surface are called *apparent BRDFs (ABRDFs)*. One main difference between BRDFs and ABRDFs is the support of several non-local phenomena, such as subsurface-scattering, as in the case of BTFs.

The faithful material reconstruction through the BTF approach comes at the cost of high memory requirements both for storing the data measurements as well as for loading operations of the rendering process since the number of measurements directly influences the accuracy of the reconstruction and suffers if the database of measured samples is not exhaustive enough. One of the primary openly accessible BTF datasets at the moment is the UBO2014 database [Weinmann et al., 2014] of the University of Bonn. It contains bidirectional texture functions for 84 various materials, where each BTF consists of 22801 (= 151 light directions × 151 view directions) HDR photographs of a $5cm^2$ material patch with the resolution of 512×512 texture pixels (texels). Such a large amount of data explains the need for compression strategies in order to maintain the practicability of the BTF approach. We describe previous work on BTF compression techniques in the following paragraph.

An interesting approach to the appearance of woven fabrics was presented by Montazeri et al. [2020], which results in a more efficient rendering. The approach combines a new geometric model for ply-based woven cloth representation with a new light scattering model, which was custom tailored for this application.

During the last few years, with the increasing advances in the field of deep neural networks, another class of material appearance methods has emerged. The neural rendering approach, which is orthogonal to the two previously described ways of appearance modeling, took over the focus of researchers. For extensive surveys on neural rendering, we refer to [Tewari et al., 2020] and the most recent state-of-the-art report [Tewari et al., 2022].

BTF Compression As mentioned above, BTFs require a high amount of memory for storing and loading the data, which has motivated research towards developing different BTF compression strategies. A relatively simple but very effective way to compress BTFs is to apply standard Principle Component Analysis (PCA) to the measured data. As mentioned before, the BTF stores a four-dimensional ABRDF for every two-dimensional surface point and is, therefore, originally a tensor. In order to perform a PCA, the data needs to be transformed into two-dimensional matrices, therefore all n ABRDFs are arranged as columns of a matrix $A \in \mathbb{R}^{m \times n}$, where m denotes the number of BTF texels. Then, a linear matrix factorization in the form of a Singular Value Decomposition (SVD) is performed on A:

$$A = U \Sigma V^T$$

Subsequently, only the most important components are kept. In other words, the data is compressed by computing the best k-rank approximation A_k to the data matrix A:

$$A_k = \sum_{i=1}^{k} \sigma_i u_i v_i^T,$$

where $k << n$. Because of its simplicity, this PCA-based approach is a very common method for BTF compression, which was, for example, used in [Koudelka et al., 2003; Liu et al., 2004] and also implemented to store the data in the aforementioned UBO2014 database with $k = 101$. Another possibility is first to divide the complete database into convenient subsets and perform the PCA on them independently [Sattler et al., 2003; Westermann et al., 2003; Kim et al., 2018]. Assuming that the entire data matrix fits into memory, the full matrix factorization usually achieves better compression ratios than the factorization of data subsets.

As an alternative to the matrix representation, several works have performed the decomposition directly on the tensor representation [Furukawa et al., 2002; Wang et al., 2005; Wu et al., 2008; Ruiters and Klein, 2009]. However, all factorization approaches assume only linear dependencies in the data and do not exploit the fact that reflectance data is significantly more structured.

In order to detect more complex dependencies Rainer et al. [2019] utilized a neural approach for BTF compression. An asymmetric encoder-decoder is utilized to transform a preprocessed d-dimensional data vector into latent vectors of dimension k, where $k << d$. For each BTF, a separate model needs to be trained. After the transformation, only the decoder and the latent vectors are stored. The decoder is then used to recover the approximated dataset by obtaining the corresponding latent vector and the light and view direction angles ω_i and ω_o. While this method significantly outperformed PCA-based BTF compression, it did not exploit the possibility of determining the number of latent vectors in dependence on the BTF complexity and chose the same number for all BTFs instead. Since the number of latent vectors is directly connected to the compression ratio, determining it based on the data leads to better compression. In the subsequent work [Rainer et al., 2020], the authors unified multiple BTFs into one model, but the question of a suitable number of latent dimensions still remained. In the first part of the thesis, we explore this question by analyzing the latent space. Therefore, in the following, we discuss previous approaches for determining a suitable dimension of the latent space and other compression techniques based on network architecture adaptation.

2.2 Data Compression Through Network Architecture Adjustment

To date, most methods, which rely on an autoencoder or other encoder-decoder architectures, specified the number of latent variables in the corresponding network by heuristically setting it to some power of two without any explanation on why this particular number was chosen. In contrast, some previous works addressed the question of determining the most convenient latent dimensionality for their encoder-decoder in the context of their specific task [Rainer et al., 2019; Rainer et al., 2020]. The network was trained for each number of latent variables anew, and after each training, the results of the output reconstructions were evaluated. Based on the trade-off between the resulting compression ratio and the reconstruction error, the suitable number of latent dimensions was chosen. However, this exhaustive approach leads

to a very high runtime consumption and, in the case of high-dimensional data, might be intractable.

A more complex approach to structure the latent space of an autoencoder [Ladjal et al., 2019; Pham et al., 2020] is based on the basic idea of the PCA. The architecture of this autoencoder is built iteratively, such that all latent variables are statistically independent and arranged in decreasing order of importance regarding the input data. However, this approach also has a very high runtime complexity, requiring almost a complete learning cycle per latent variable. In contrast, our method directly computes the contribution of each latent variable based on their Shapley values [Shapley, 1953], a concept introduced in game theory, in a single step, thus avoiding many time-consuming training iterations.

In the context of neural image compression, an approach based on slimmable neural networks [Yu et al., 2018] has been applied [Yang et al., 2021]. It adjusts the width of the hidden layers of a compressive autoencoder, leading to even better compression rates. For the purpose of automatic architecture adjustment, several alternatives of slimmable generalized divisive normalization (GDN) [Ballé et al., 2016a] layers together with slimmable probability models were evaluated. While this method has shown impressive results in the field of image compression, our approach (Trunz et al., 2022, see Chapter 3) based on the contributions of latent variables towards the reconstruction result provides an effective alternative that can be used in combination with such slimmable neural networks. Furthermore, our method can improve any compression method which utilizes an encoder–decoder.

2.3 Geometric Modeling of Fabrics

In this section, we discuss previous work on procedural and inverse procedural modeling and give basic information on the elements of knitted cloth.

Composition of Knitted Cloth Information about the compositional elements of knitted cloth is essential for any kind of procedural modeling of knitwear. In the following, we introduce some basic concepts of textile yarns and knitting in general.

In essence, there are three scales of knitted cloth. The basic microscopic scale refers to the level of fibers and their arrangement. The most exquisite are natural fibers such as cashmere wool, mohair, virgin wool, cotton or silk. But also synthetic fibers like acrylic or nylon are very popular choices in the knitwear industry. Many fibers are twisted to form a long strand called a ply. A ply may consist of only one type of fiber or may contain several different types. There are various ways of twisting fibers to plys, we refer to [Shaikh, 2002] for a detailed explanation. For the purpose of realistic modeling, we differentiate between two classes of fibers: raw yarn fibers and *flyaways*. Flyaways are fibers that have been displaced from their original position in the yarn e.g. due to friction. Such fibers contribute significantly to the hairiness of the yarn. According to [Zhao et al., 2016], we distinguish between *loop* flyaways and *hair* flyaways. Fibers where one side completely left the ply are denoted as

hairs, whereas fibers that significantly deviate from the main ply structure but where both ends are still within the ply are denoted as loop flyaways.

When two or more plys are twisted together, they form a yarn, which constitutes the second level of cloth. Whereas yarns for woven cloth are often made out of two or three plys, the most common number of plys in a knitted cloth without further twisting is four, followed by three and five. If the yarn is too thin, it is usually made thicker by twisting several thinner yarns together (Fig. 2.2).

Figure 2.2: Left: two plys are twisted together to form a thin yarn. Middle: a four-ply yarn that is thick enough for knitting. Right: two thin yarns are twisted together to form a thicker yarn.

Knitwear is made out of yarns by repeatedly following some knitting instructions, which results in a pattern that represents the third level of cloth. This process is done either by hand or with the help of knitting machines. Knitting instructions consist of a regular grid of symbols, where each symbol corresponds to a three-dimensional loop. Such loops are called *stitches*. Each pattern starts with an initial top row of stitches, and all subsequent rows are generated below by pulling the yarn through the stitches of the previous row. There is only a small amount of stitch types with which all of the knitting instructions are produced, but the two fundamental and by far the most common types are a *knit* and a *purl* stitch, as shown in Figure 2.3. A knit stitch occurs when the yarn is pulled through a loop of a stitch of the upper row from below, while creating a purl stitch requires the yarn to be pulled from above. If the knitted cloth is viewed from the other side, then purl stitches look like knit stitches and vice versa. Three simple common knit-purl patterns are called *Stockinette*, *Garter* and *2-2 Rib*. However, there exists a huge amount of patterns that can be made only with these two types of stitches.

Procedural Modeling of Fabrics As already mentioned, for visual prototyping of knitted cloth, we ideally want to be able to perform editing operations on each of the three scales: fibers, yarns and patterns. For this purpose, we need to model these scales explicitly and not only by means of an image-based representation, such as the BTF described in Section 2.1. Procedural modeling is an established technique in computer graphics that helps to efficiently create a lot of 3D content employing production rules and parameters instead of the time-consuming manual generation of each shape individually with some modeling

Figure 2.3: Examples of knit (top) and purl (bottom) stiches.

software. Such production rules have been developed for example for buildings [Müller et al., 2006], cities [Vanegas et al., 2012] and trees [Longay et al., 2012; Stava et al., 2014]. Recently, Guerrero et al. [Guerrero et al., 2022] presented the first generative model for procedural materials.

At the pattern level, for woven cloth, such rules can be represented in terms of a weave pattern matrix, and in the case of knitwear, production rules correspond to knitting instructions. In order to create any desired knitting pattern, one merely requires a 3D model of each type of stitch. These stitch shapes are then assembled according to the corresponding knitting instruction. Due to physical forces, the same stitches in a pattern behave differently when combined with different types of neighboring stitches (see Figure 2.3). Thus, to increase realism, several approaches for physical simulation of knitwear and woven cloth have been proposed [Kaldor et al., 2008; Yuksel et al., 2012; Leaf et al., 2018; Sperl et al., 2020]. After the whole pattern has been modeled, subsequent iterative physical simulation gives the stitch loops a more realistic shape and can be combined with the approaches presented in this thesis.

Jin et al. [2022] focused their work on woven cloth and introduced a procedural model that approximates yarn geometry with smooth bent cylinders combined with a shading model for reflectance. In order to edit at the level of fibers and yarns, i.e. the components the stitches were made of, several variations of procedural models for yarns have been developed. Sreprateep and Bohez [2006] presented a detailed model of fibers twisted into a single ply. This model already took into account the fact that fibers do not follow a uniform distribution within a ply. Furthermore, the idea of fiber migration (i.e. parts of fibers migrating from their positions within the yarn) was also incorporated into the model. Schröder et al. [2015] took this model as an inspiration and introduced a procedural yarn model that included hair flyaways and was able to generate more than one ply. Further extensions and modifications to this model were made by Zhao et al. [2016]. First, their new model included support for loop flyaways. Furthermore, instead of simulating hair flyaways with Perlin Noise, the authors introduced several parameters to control all flyaways. Both modifications result in a more realistic appearance of yarns and fabrics.

Luan et al. [2017] introduced an approach for efficient and fast rendering of procedural textiles that does not require the full realization of the procedural model and, thus, can be applied to large-scale procedural textiles.

In our work, we even further extend the model of Zhao et al. [2016] by including an elliptical

cross-section of fibers instead of a circular one, changing the orientation of the plys from the global to the local one and, most importantly, modifying the parameters for the hair flyaways to better reflect the actual manufacturing process. These enhancements lead to an even more realistic yarn appearance, which enables the training of neural network models based solely on synthetic data and still be successfully applied to real-world yarns.

Inverse Procedural Modeling of Fabrics Although forward procedural modeling has been successfully and widely used in the computer graphics community to efficiently create a large amount of 3D content, controlling the output remains a well-known open problem. Even for an experienced user, it is not trivial to correctly set all the parameters of a model in order to get the desired output. Inverse procedural modeling aims to allow the user to control the modeling process in terms of the specification of the output. Instead of guessing how the parameters should be set in order to produce the needed output, the user specifies how the output should look, usually in terms of an image, a 3D volume or some other specification. Subsequently, the system itself discovers how to set the corresponding parameters of the procedural model so as to generate the desired output. Such inverse systems have been successfully introduced for facades [Weissenberg et al., 2013; Wu et al., 2014; Lienhard et al., 2017], buildings [Demir et al., 2016; Nishida et al., 2018], trees [Stava et al., 2014] and in the domain of urban design [Vanegas et al., 2012], among others. We refer to the course of Aliaga et al. [Aliaga et al., 2016] for an extensive survey on the topic of inverse procedural modeling.

In the area of cloth modeling, several inverse procedural modeling approaches have been proposed. A complete pipeline for reverse engineering of woven cloth has been developed by Schröder et al. [2015]. Guarnera et al. [2017] introduced an alternative approach that required less runtime for its execution. Both methods demonstrated promising results in the reverse engineering of woven cloth but were not designed for the analysis of knitwear. Hussain et al. [2020] cast the problem of weave pattern identification as a classification problem for a neural network and restricted the number of possible classes to three, based on the three most commonly used weave patterns: plain, twill, and satin. Recently, Ali et al. [2022] proposed a neural segmentation approach to detect a weave pattern in a set of micro CT images.

Knitted stitches have three-dimensional structures, and the overall shape of the loops, especially in the case of hand-made garments, do not exhibit the similarity and overly regular structures, without non-rigid deformations, as the warp and weft of woven cloth do. Parallel to our work on inverse procedural modeling of knitwear, Kaspar et al. [2019] proposed neural inverse knitwear modeling. However, this approach was designed to handle machine-knitted textiles and was not able to analyze hand-made patches, which are usually more deformed than their machine-knitted counterparts. Furthermore, to obtain good results, the number of rows and columns of the patterns has to be specified in advance, which is a difficult task for an inexperienced user. In contrast, our approach (Trunz et al., 2019, see Chapter 4) achieves good results on both machine- and hand-knitted fabrics, and the number of columns and rows of the pattern is determined automatically.

Inverse Procedural Modeling of Yarns Already in 2004, Voborova et al. [2004] used an imaging system containing a microscope, optical fiber lighting and a CCD camera as input and estimated the hairiness along with the effective diameter and a twist of the yarn. The work of Zhao et al. [2016] focused on inferring the parameters for their procedural model for yarns from a given CT scan of a small piece of the yarn. However, this approach requires expensive hardware and can only be applied for yarns, where all characteristics (such as helix pitch, yarn radius, etc.) fit into one scan. In order to make an approach to be accessible to more users, other works focused on determining yarn characteristics from images. Saalfeld et al. [2018] inferred some of the parameters of the procedural yarn model of Zhao et al. [2016] by means of gradient descent with momentum approach on artificially generated yarn images. However, the approach did not achieve good results on the images of real yarns. Given a single image of a small textile patch and a yarn database as a prior, Wu et al. [2019] estimated the yarn-level geometry of the fabric. However, the authors conclude that such input information is not sufficient to extract yarn parameters other than fiber twist and the number of fibers. To the best of our knowledge, our neural approach (Trunz et al., 2023, see Chapter 5) for procedural modeling of yarns is the first to infer all required geometry yarn parameters from a single image.

Part II

Publications

Chapter 3

Efficient structuring of the latent space for controllable data reconstruction and compression

In this chapter, the contributions and results developed in the following peer-reviewed publication are discussed:

Elena Trunz, Michael Weinmann, Sebastian Merzbach, and Reinhard Klein.
"Efficient structuring of the latent space for controllable data reconstruction and compression."
Graphics and Visual Computing (GVC), 7, page 200059, 2022.
DOI: 10.1016/j.gvc.2022.200059

3.1 Summary of the Publication

In the last years, deep neural networks received vast popularity, and much attention has been devoted to gaining an understanding of their design and behavior. In the case of autoencoders, it is difficult to explain the contribution of individual latent variables to the model performance, which complicates the choice of an appropriate dimensionality of the latent space. In turn, since the most common application of autoencoders and other encoder-decoder architectures lies in compression, the choice of the number of latent variables directly influences the compression rates and the compactness of the data representation.

Most methods, which rely on the utilization of an encoder-decoder, specify the number of the latent variables heuristically, usually setting it to some power of two and providing no explanation regarding the choice or analysis on the question of whether the choice was a suitable one. Only a few attempts have been made to determine the most suitable size of the latent dimension. Rainer et al. [2019] trained their encoder-decoder models for compression of bidirectional texture functions (BTFs) several times anew, choosing a different latent size

each time and calculating the resulting reconstruction error. The final size was chosen based on the compression-reconstruction trade-off. This approach results in high computational costs, which is why Rainer et al. did not perform the computations for each BTF but chose the same number of latent variables for all BTFs. However, since each BTF exhibits different complexity, choosing the same latent dimension for all BTFs is not the optimal decision, as we demonstrated in our work.

In contrast to this time-consuming approach, the method we presented in this work is focused on the exploration of the structure of the latent space in encoder-decoder schemes. We analyze the contribution of the individual latent dimensions to the reconstruction results and subsequently order them based on their Shapley values [Shapley, 1953] during only a single training process, thereby avoiding a time-consuming stepwise training process. The idea of the Shapley values, which has origins in the cooperative game theory domain, has been successfully applied in the field of interpretable machine learning in applications such as feature attribution. Given a linear network, we have a direct relation to Principal Component Analysis (PCA), which provides a natural ordering of the components regarding their singular values. Starting with the linear models, in our work, we derived the theorem and gave the theoretical proof that in the linear case, the ordering which results based on the Shapley values is the same as the order obtained from the singular value decomposition, thus proving it to be the optimal one.

However, autoencoders and other encoder-decoder models are not linear. While the involved non-linearity allows encoder-decoders to find more flexible and more powerful latent spaces that exhibit compactness with respect to the original data domain, it does not allow to gain structural information of the latent space by means of a PCA. On the other hand, the non-linearity of a function (encoder-decoder in our case) does not pose a problem for the computation of the Shapley values for this function. Thus, based on the properties of the Shapley values, we derived the following algorithm for finding the most suitable number of latent variables for each particular application: First, the encoder-decoder is trained with a sufficiently large number of latent variables for a certain number of epochs (usually half of the intended training). Then, approximate Shapley values for the latent variables are computed, subsequently ranking the variables in descending order. After the computation of the cumulative contribution of the ordered sets of latent variables and after the visualization of to the sets corresponding reconstructions the user can decide how many of the higher-ranked latent variables will be kept. Subsequently, the last encoder layer and the first decoder layer are modified, discarding the remaining latent variables and the training is resumed with the selected variables.

We exemplary demonstrated the beneficial combination of Shapley values and autoencoders regarding the choice of the dimensionality of the latent space, the ordering of the involved latent variables according to their importance and the respective capability for reconstruction and compression of reflectance data (BTFs), as well as images. In the case of the BTF compression, we demonstrated that different BTFs, due to their different complexity, require different numbers of latent variables for a similar representation in contrast to the use of a fixed number of 8 as chosen by Rainer et al. Rainer et al. [2019].

In our work, we also report, for both applications, the relationship between the running time for computations of the Shapley values and the number of the latent variables from which the values were computed. Furthermore, we performed the evaluation of the reconstruction error depending on the number of latent variables that were ordered based on Shapley analysis in the middle of the training, at the end of the training and a random ordering.

3.2 Author Contributions of the Publication

In this work, I derived the theorem and the theoretical proof that in the case of a linear model, the ordering of the elements involved in the reconstruction according to their Shapley values is optimal. Furthermore, I developed the algorithm for finding the required number of latent dimensions in an encoder-decoder based on the computation of the Shapley values for the latent variables and ranking them according to their contribution, with subsequent analysis of cumulative contribution and visualizations. Additionally, I performed the evaluation of the proposed approach on different applications: a compact representation of images and high-dimensional reflectance data.

Chapter 4

Inverse Procedural Modeling of Knitwear

In this chapter, the contributions and the results developed in the following peer-reviewed publication are discussed:

Elena Trunz, Sebastian Merzbach, Jonathan Klein, Thomas Schulze, Michael Weinmann, and Reinhard Klein.
"Inverse Procedural Modeling of Knitwear."
2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 8622–8631, 2019.
DOI: 10.1109/CVPR.2019.00883

4.1 Summary of the Publication

In order to produce nice and interesting-looking knitted clothing, one requires, among other things, nice knitting patterns. There are books and websites that describe various patterns. But what if we see a picture of a clothing piece for which we do not have instruction on how to knit it but want to use its pattern in order to knit this or possibly some other clothing piece? How can we recognize the desired pattern? This problem is very challenging, even for an expert in knitting. Even though there is quite a small amount of stitch types, each stitch can take several possible shapes, depending on the shapes of the neighboring stitches and the yarn it is made of. Especially stitches might be partially or completely covered by the adjacent stitches, making them hard to recognize. To be sure of the final instruction of the pattern, an expert would often have to stretch the clothing piece and turn it around to be able to see all the covered stitches. With only one image available, it is not possible to perform such operations. Additional difficulties arise when analyzing an image of hand-knitted clothing. While the shapes of the stitches of machine-knitted clothing depend mostly on the yarn and the adjacent stitches, the shapes of the stitches of hand-knitted clothing include

various deformations (like stretching, holes, tight stitches, etc.), depending on the behavior of the person, who knitted the piece.

Recently some approaches on reverse-engineering of weave patterns [Schröder et al., 2015; Guarnera et al., 2017] were proposed, but, as the authors state themselves, these methods are not suitable for analyzing knitted textiles. In contrast to the very regularly shaped woven cloth, knitted textiles are inherently three-dimensional and exhibit occlusions and non-rigid transformations, which need to be accounted for in order to accurately infer the correct stitch patterns from a two-dimensional image. Although there are several approaches for inverse procedural modeling of objects from images, these approaches are designed for a particular application and cannot be easily applied to the inference of knitting patterns.

In this paper, we tackled this inverse procedural modeling problem of finding a correct procedural description (knitting instruction) for a given design on an image by proposing a four-step approach for a stitch pattern analysis of the given image. In this work, we limited our algorithm to handle the two main and the most common types of stitches, knit and purl, since there is a large number of knitting designs using only these. Given an input image, the user initiates the process by manually choosing one sample for each type of stitch occurring in the image. All other steps of the algorithm are performed automatically. First, for each pixel of the image, we used template matching in order to determine, which stitch type it is more likely to belong to. For the computation of the similarity between the stitch template and the image patch, we used Best Buddies Similarity (BBS [Dekel et al., 2015]), which we, for our purposes, extended with an additional gradient constraint (BBSg). The result of the overall template matching procedure was then used to detect the correct positions of the stitches, which form a regular grid but with irregular row and column placement. Since there is a huge amount of possible grids, we formulated the search as an integer linear programming (ILP) problem and solved for the optimum with the Gurobi Optimizer software [Gurobi Optimization, 2016]. In the resulting optimal regular stitch grid, possible stitch type detection errors can occur. In order to correct such errors, we took advantage of the fact that a knitted piece always contains repetitions of the underlying pattern. During the third step of our pipeline, possible type errors are corrected using the maximum voting approach, resulting in the size of the desired pattern and the stitches that it contains. Since each row and column of the pattern constitutes a possible beginning of the actual knitting pattern, as the last step of our pipeline, we used the Law of Prägnanz and the Law of Symmetry from Gestalt theory [Bradley, 2014] to find such beginning that results in a nice clothing piece, when knitted by repeating the pattern. The result is then transformed into the desired knitting instruction.

We tested our approach on 25 photos and scans of mostly hand-made knitting samples with different patterns of various complexity. Despite the relatively high complexity of the patterns, our approach was able to correctly infer the corresponding knitting instructions. As there were no previous approaches for inverse modeling of knitting patterns, we compared our approach with a greedy strategy, where instead of optimizing over all possible stitch grids, we iteratively selected the best local match according to the similarity measure. However, due to the iterative local instead of global optimization, this strategy did not yield acceptable results. Furthermore, we compared our extended Best Buddy Similarity (BBSg) measure

against some other popular template matching measures. Our experiments have shown how BBSg reduced the matching error. We also reported on the computational time of our pipeline and especially the ILP solver, and observed that it was less than the running time of the similarity computations, making the approach feasible for the intended application.

4.2 Author Contributions of the Publication

In this work, I designed a pipeline for the induction of knitting instruction from a single image. For the first step of the pipeline, I evaluated different metrics for pattern matching, which were required to find coarse localizations of the stitches in the image. For the second step, I formulated the integer linear programming (optimization) problem to find an optimal regular grid structure out of the irregular row and column placement and implemented the solution with the help of the Gurobi software. For the third and fourth steps, I developed the error correction and pattern induction algorithm to find the final knitting instruction from the grid.

Chapter 5

Neural Inverse Procedural Modeling of Knitting Yarns from Images

In this chapter, we discuss the contributions and results developed in the following publication which already appeared as a preprint and is currently under review:

Elena Trunz, Jonathan Klein, Jan Müller, Lukas Bode, Ralf Sarlette, Michael Weinmann, and Reinhard Klein.
"Neural Inverse Procedural Modeling of Knitting Yarns from Images."
arXiv:2303.00154 (under review), submitted to Computers & Graphics (CG), 2023.
DOI: 10.48550/arXiv.2303.00154

In the following, we include a verbatim copy of the content of this work subject to some minor editorial changes.

Author Contributions of the Publication In this work, I developed the idea of utilizing neural networks together with a database of synthetic yarns to estimate geometry parameters of real-world yarns. Together with my co-authors, I designed a procedural yarn model that extends the state-of-the-art model in terms of generation of more realistic synthetic yarns. I also developed a yarn sampler that automatically generates plausible-looking yarns and used it to generate a database of annotated images of synthetic yarns. Furthermore, I developed and implemented the network to infer flyaway parameters and a set of parameter-specific networks for inference of other yarn parameters from images of real yarns. I used the generated database to train and validate these neural models.

5.1 Abstract

We investigate the capabilities of neural inverse procedural modeling to infer high-quality procedural yarn models with fiber-level details from single images of depicted yarn samples. While directly inferring all parameters of the underlying yarn model based on a single neural

network may seem an intuitive choice, we show that the complexity of yarn structures in terms of twisting and migration characteristics of the involved fibers can be better encountered in terms of ensembles of networks that focus on individual characteristics. We analyze the effect of different loss functions including a parameter loss to penalize the deviation of inferred parameters to ground truth annotations, a reconstruction loss to enforce similar statistics of the image generated for the estimated parameters in comparison to training images as well as an additional regularization term to explicitly penalize deviations between latent codes of synthetic images and the average latent code of real images in the encoder's latent space. We demonstrate that the combination of a carefully designed parametric, procedural yarn model with respective network ensembles as well as loss functions even allows robust parameter inference when solely trained on synthetic data. Since our approach relies on the availability of a yarn database with parameter annotations and we are not aware of such a respectively available dataset, we additionally provide, to the best of our knowledge, the first dataset of yarn images with annotations regarding the respective yarn parameters. For this purpose, we use a novel yarn generator that improves the realism of the produced results over previous approaches.

5.2 Introduction

Due to their ubiquitous presence, fabrics have a great importance in domains like entertainment, advertisement, fashion and design. In the era of digitization, numerous applications rely on virtual design and modeling of fabrics and cloths. Besides the use of fabrics in games and movies, further examples include online retail with its focus on more accurately depicting the appearance of the respective clothes in images, videos or even virtual try-on solutions, as well as virtual prototyping and advertisement applications to provide pre-views on respective product designs.
The accurate digital reproduction of the appearance of fabrics and cloth relies on a fiber-level based modeling to allow accurately representing light exchange in the fiber and yarn levels. However, due to their structural and optical complexity imposed by the arrangement of fibers with diverse characteristics within yarns and the interaction between yarns in the scope of weave and knitting patterns – where small changes in the fiber and yarn arrangement may result in significant appearance variations – as well as due to the numerous partial occlusions of the involved fibers and yarns, capturing and modeling the appearance of yarns, fabrics and cloth remains a challenge. In the context of reconstructing yarns, Zhao et al. [2016] addressed the difficulty of scanning the self-occluding fiber arrangements based on computer-tomography (CT) scans to get accurate 3D reconstructions of the individual yarns. However, this imposes the need for special hardware. Instead, in this paper, we aim at the capture and modeling of the appearance of yarns by inferring individual yarn parameters from a single photograph depicting a small part of a yarn.
To address this goal, we investigate the capabilities of neural inverse procedural modeling. Whereas directly optimizing all the parameters that determine a yarn's geometry (including flyaways i.e. fibers that migrate from the yarn, contributing to the fuzziness of the yarn) with a single neural network may seem an intuitive choice, the complexity of the depictions

of yarns, where twisting characteristics dominate the appearance in the yarn center and flyaway statistics dominate the appearance in the yarns' border regions, imposes that the network has the capacity to understand where which parameters can be predominantly inferred from. This observation might indicate that other strategies such as training separate networks for inferring the structural parameters for the main yarn and the characteristics of flyaways or even using an ensemble of networks, where each of these networks is only responsible for estimating a single parameter of the underlying yarn model, could be reasonable alternatives. Therefore, we investigate the potential of these approaches for the task of inverse yarn modeling from a single image. Furthermore, we investigate the effect of different loss functions including a parameter loss to penalize the deviation of inferred parameters to ground truth annotations, a reconstruction loss to enforce similar statistics of the image generated for the estimated parameters in comparison to training images as well as an additional regularization term to explicitly penalize deviations between latent codes of synthetic images and the average latent code of real images in the encoder's latent space. Thereby, we also analyze to what extent such models can be trained from solely using synthetic training data.
All of these models are trained based on synthetic training data generated from a high-quality yarn simulator that improves upon the generator by Zhao et al. [2016] in terms of a more realistic modeling of hair flyaways, fiber cross-section characteristics and the orientation of the fibers' twisting axis. As our approach relies on the availability of a dataset of yarn images with respective annotations regarding characteristic yarn parameters, such as the number of plys, the twisting length etc., we introduce – to the best of our knowledge – the first dataset of synthetic yarns with respective yarn parameter annotations. Both the dataset and the yarn generator used for the automatic generation of this dataset will be released upon acceptance of the paper. Our approach for neural inverse procedural modeling of yarns exhibits robustness to variations in appearance induced by varying capture conditions such as different exposure times as long as strong over-exposure and under-exposure are avoided during capture.
In summary, the key contributions of our work are:

- We present a novel neural inverse modeling approach that allows the inference of accurate yarn parameters including flyaways from a single image of a small part of a yarn.
- We investigate the effect of different loss formulations on the performance based on different configurations of a (yarn) parameter loss to penalize deviations in the inferred parameters with respect to the ground truth, a reconstruction loss to enforce the statistics of a rendering with the estimated parameters to match the statistics of given images, and regularization term to explicitly penalize deviations between latent codes of synthetic images and the average latent code of real images in the encoder's latent space.
- We provide, to the best of our knowledge, the first dataset of realistic synthetic yarn images with annotations regarding the respective yarn parameters.
- We present a yarn generator that supports a large range of input parameters as well as a yarn sampler that guides the selection of parameter configurations for the automatic

generation of realistic yarns.

5.3 Related Work

Respective surveys [Schröder et al., 2012; Castillo et al., 2017; Castillo et al., 2019; Pagán et al., 2020; Amor et al., 2021; Mohammadi and Kalhor, 2021; Noor et al., 2021] indicate the opportunities of computational approaches for the cloth and apparel industries as well as challenges regarding the capture, modeling, representation and analysis of cloth. Some approaches approximate fabrics as 2D sheets. Wang et al. [2008] and Dong et al. [2010] leverage spatially varying BRDF (SVBRDFs) based on tabulated normal distributions to represent the appearance of captured materials including embroidered silk satin, whereas others focused on appearance modeling in terms of bidirectional texture functions (BTFs) [Dana et al., 1997; Weinmann et al., 2014; Filip et al., 2018].

For scenarios with a focus on efficient simulation and editing or respective manipulation, yarn-based models [Kaldor et al., 2008; Cirio et al., 2014; Martin-Garrido et al., 2018] have been shown to be more amenable [Yuksel et al., 2012], however, at the cost of not offering the capabilities to accurately capture details of fiber-level structures and the resulting lack of realism. Drago and Chiba [2004] focused on simulating the macro- and microgeometry of woven painting canvases based on procedural displacement for modeling the arrangement of the woven yarns (i.e. a spline-based representation) and surface shading. The model by Irawan and Marschner [2012] also predicts yarn geometry (in terms of curved cylinders made of spiralling fibers) and yarnwise BRDF modeling to represent the appearance of different yarn segments within a weaving pattern. However, this approach does not model shadowing and masking between different threads. The latter has been addressed with the appearance model for woven cloth by Sadeghi et al. [2013] that relies on extensive measurements of light scattering from individual threads, thereby taking into account for shadowing and masking between neighboring threads. However, these approaches are suitable for scenarios where cloth is viewed from a larger distance, since reproducing the appearance characteristics oberservable under close-up inspection would additionally require the capability to handle thick yarns or fuzzy silhouettes as well as the generalization capability to handle fabrics with strongly varying appearances. To increase the degree of realism, Guarnera et al. [2017] augment the yarns extracted for woven cloth in terms of micro-cylinders with adjustments regarding yarn width and misalignments according to the statistics of real cloth in combination with the simulation of the effect of yarn fibers by adding 3D Perlin Noise [Perlin, 1985] to the micro-cylinder derived normal map. Several approaches focus on fitting an appearance model like Bidirectional Reflectance Distribution Functions (BRDFs) [Dobashi et al., 2019; Jin et al., 2022] to inferred micro-cylinder yarn models or Bidirectional Curve Scattering Distribution Function (BCSDF) [Schroder et al., 2011] to simulate the appearance from the fibers within each ply curve extracted for a pattern without explicitly modeling each individual fiber or applying a pre-computed fiber simulation [Montazeri et al., 2021]. Extracting yarn paths from image data can be approached by leveraging the prior of perpendicularly running yarns for woven cloth (e.g., [Schröder et al., 2015]) as well as based on knitting primitive detection inspired by template matching

with a refinement according to an underlying knitting pattern structure [Trunz et al., 2022] or deep learning based program synthesis [Kaspar et al., 2019]. While such approaches allow the modeling of the underlying yarn arrangements, the detailed yarn modeling with yarn widths, yarn composition, yarn twisting, hairiness, etc. is not explicitly modeled.
Following investigations on the geometric structure of fabrics in the domain of the textile research community [Keefe, 1994; Tao, 1996; Morris et al., 1999; Shinohara et al., 2010], several works focused on a more detailed modeling of the underlying cloth micro-appearance characteristics to more accurately model the underlying cloth characteristics such as thickness and fuzziness. This includes volumetric cloth models [Xu et al., 2001; Jakob et al., 2010; Zhao et al., 2011; Zhao et al., 2012; Zhao et al., 2013], that describe cloth in terms of 3D volumes with spatially varying density, as well as fiber-based cloth models [Khungurn et al., 2015; Schröder et al., 2015] that infer the detailed 3D structure of woven cloth at the yarn level with its fiber arrangement. Zhao et al. [Zhao et al., 2011; Zhao et al., 2012; Zhao et al., 2016] leveraged a micro-computed tomography (CT) scanner to capture 3D volumetric data. The detailed volumetric scan allows them to trace the individual fibers and, hence, provides a an accurate volumetric yarn model that captures high-resolution volumetric yarn structure. For instance, Zhao et al. [2016] present an automatic yarn fitting approach that allows creating high-quality procedural yarn models of fabrics with fiber-level details by fitting procedural models to CT data that are additionally augmented by a measurement-based model of flyaway fibers. Instead of involving expensive hardware setups such as based on CT scanning, others focused on inferring yarn parameters from images, thereby representing more practical approaches for a wide range of users. Voborova et al. [2004] focused on estimating yarn properties like the effective diameter, hairiness and twist based on initially fitting the yarn's main axis based on an imaging system consisting of a CCD Camera, a microscope, and optical fiber lighting. Furthermore, with a focus on providing accurate models at less computational costs and memory requirements than required for volumetric models, Schröder et al. [2015] introduced a procedural yarn model based on several intuitive parameters as well as an image-based analysis for for the structural patterns of woven cloth. The generalization of this approach to other types of cloth, such as knitwear, however, has not been provided but still needs further investigation. Saalfeld et al. [2018] used gradient descent with momentum to predict some of the procedural yarn parameters used by Zhao et al. [2016] from images of synthetically generated yarns. Although the results were promising for some of the parameters, the approach still could not be applied to the real yarn images. Wu et al. [2019] estimate yarn-level geometry of cloth given a single micro-image taken by a consumer digital camera with a macro lens, leveraging prior information in terms of a given yarn database for yarn layout estimation. Large-scale yarn geometry is estimated based on image shading, whereas fine-scale fiber details are obtained based on fiber tracing and generation algorithms. However, the authors mention that the use of a single micro-image does not suffice for the estimation of all relevant yarn parameters of complex procedural yarn models like the ones by Zhao et al. [2016] or Schröder et al. [2015], and, hence, the authors only consider the two parameters of fiber twisting and fiber count. Whereas our yarn generator is conceptually similar to the one by Zhao et al. [2016], there is an important difference in how we model the orientation of the twisting axis of the fibers. Instead of using the global z-axis, we align the twisting axis with the relative z-axis of the next hierarchy level, resulting in a more realistic yarn structure. This relative implementation allows adding additional hierarchy levels, i.e.

especially for the hand knitting it is common to twist different yarns if they are too thin, thus creating the next level. Furthermore, our model's realism is further increased by also considering elliptic fiber cross-sections as occurring for natural hair fibers like wool and by considering a more natural modeling of flyaways.
In the context of inferring physical yarn properties from visual information, Bouman et al. [2013] estimated cloth density and stiffness from the video-based dynamics information of wind-blown cloth. Others focused on a neural network based classification of cloths according how stretching and bending stiffness influence their dynamics. Furthermore, Rasheed et al. [2020] focused on the estimation of the friction coefficient between cloth and other objects. Based on the combination of neural networks with physically-based cloth simulation, Runia et al. [2020] trained a neural network to fit the parameters used for simulation to make the simulated cloth match to the one observed in video data. Liang et al. [2019] and Li et al. [2022] presented approaches for cloth parameter estimation based on sheet-level differentiable cloth models. Gong et al. [2022] introduced a differentiable physics model at a more fine-grained level, where yarns are modelled individually, thereby allowing to model cloth with mixed yarns and different woven patterns. Their model leverages differentiable forces on or between yarns, including contact, friction and shear.

5.4 Generation of synthetic training data

Our learning-based approach to infer yarn parameters from images relies on the availability of a database of images of yarns with respective annotations. However, to the best of our knowledge, no such database exists to this day. Since the exact measurements of the parameters of a real yarn is a complex task that requires experts as well as additional hardware such as a CT scanner, we overcome this problem by leveraging modeling and rendering tools from the field of computer graphics to create images of synthetic yarns with known parameters that can be directly used for learning applications.

To enable robust parameter inference from photographs of real yarns, the synthetic yarn images used for training the underlying neural model must be highly realistic, i.e. they must accurately model the yarn structure with its underlying arrangement of individual fibers. Similar to Zhao et al. [2016], we chose a fiber-based model rather than a volumetric one to gain more control over the generation and achieve higher quality. However, to increase the realism of the synthetic yarns, we extended the yarn model of Zhao et al. by including elliptical fiber cross-sections, local coordinate frame transformation for helix mapping and considering more complex modeling of hair flyaways.

We mimic the actual manufacturing process by introducing a hierarchical approach. Multiple fibers are twisted together to form a ply, and, in turn, multiple plies are twisted together to form a yarn. If necessary, multiple thinner yarns can be twisted into a thicker yarn, which is sometimes the case in knitwear manufacturing. We denote the yarn resulting from this hierarchical procedure as *raw yarn*.

In addition to capturing the characteristics of the fiber arrangement of the yarn structure, we must also consider that some of the fibers, referred to as *flyaways*, may deviate from their

intended arrangement within yarns and run outside the thread. These deviations are caused by friction, aging or errors in the manufacturing process and play a central role in the overall appearance of yarns and the fabrics made from them.

Therefore, our yarn model is controlled by a number of parameters, which belong to two types: raw yarn parameters and flyaway parameters. During generation, these parameters are stored along with the respective resulting images, and later serve as training labels for the network training.

The raw yarn is recursively built from multiple hierarchical levels (see Fig. 5.1 and Alg. 1). In the next step, flyaways are added (Alg. 2) and detailed fiber parameters such as material and cross section are defined. The yarn is then ready to be rendered. We generate and render the synthetic yarn images using *Blender*, which offers advanced modeling capabilities that can be fully controlled by Python scripts, making it suitable for procedural modeling. Especially, the high level mesh modifiers allow for relatively compact scripts. Additionally, it also contains a path tracer capable of rendering photo-realistic images, which allows us to build an all-in-one pipeline.

5.4.1 Hierarchical yarn model

During the first step, yarns, plies and fibers are represented as polygonal lines, i.e. a tuple (V, E) that stores the vertex positions $V = \left\{v_i \in \mathbb{R}^3 | i \in \mathbb{N}\right\}$ and their edges $E = \{(i, j) | i, j \in \mathbb{N}\}$. Note that our generation process allows for an arbitrary number of levels. However, in the rest of the paper, we will demonstrate the concept using three levels, fibers, plies, and yarns. Algorithm 1 presents an overview over our recursive hierarchical generation of the raw yarn. The input of the first level, the fiber level, is a simple straight polygonal line that must be chosen large enough to allow for the required resolution. The vertices v_i of the line are given by

$$v_i = \begin{pmatrix} 0 & 0 & i\alpha_f \end{pmatrix}^T \tag{5.1}$$

Here, α_f denotes the distance between two consecutive vertices of a fiber. In each of the higher levels, we start by creating a set of N 2D instance start positions p_i. We define two variations of this procedure, one for small amounts of instances (~ 7), and one for larger (in practice up to 200). Both are illustrated in Fig. 5.2. In both cases, we add some jitter j_{xy} to the sample positions. For small numbers N of instances, we generate a regular pattern on a circle with radius r:

$$p_i = r\begin{pmatrix} \sin\theta_i \\ \cos\theta_i \end{pmatrix} + j_{xy}R\begin{pmatrix} 1 \\ 1 \end{pmatrix}, \quad \theta_i = 2\pi\frac{i}{N} \tag{5.2}$$

Here and in the following, R is a zero-mean, normal distributed random variable with a standard deviation of 1 that is redrawn for each occurrence.

Algorithm 1 Recursive hierarchical generation of raw yarn

Require: level of raw yarn $level$

procedure BUILDLEVEL($level$)
  if $level = 0$ **then**
    create straight polygonal line $line$
    return $line$
  else
    $template \leftarrow$ BUILDLEVEL($level - 1$)
  end if
  $P \leftarrow$ create N instance positions using Eq. 5.2 or Eq. 5.3
  $output \leftarrow \emptyset$
  for all $p \in P$ **do**
    $I \leftarrow copy(template)$
    $I \leftarrow$ scale x coordinate of I with e for elliptical cross-section
    $I \leftarrow$ rotate I using Eq. 5.4
    center I at position p
    generate helix at position p using Eq. 5.5-5.7 and let I follow the helix
    $output \leftarrow output \cup I$
  end for
  return $output$
end procedure

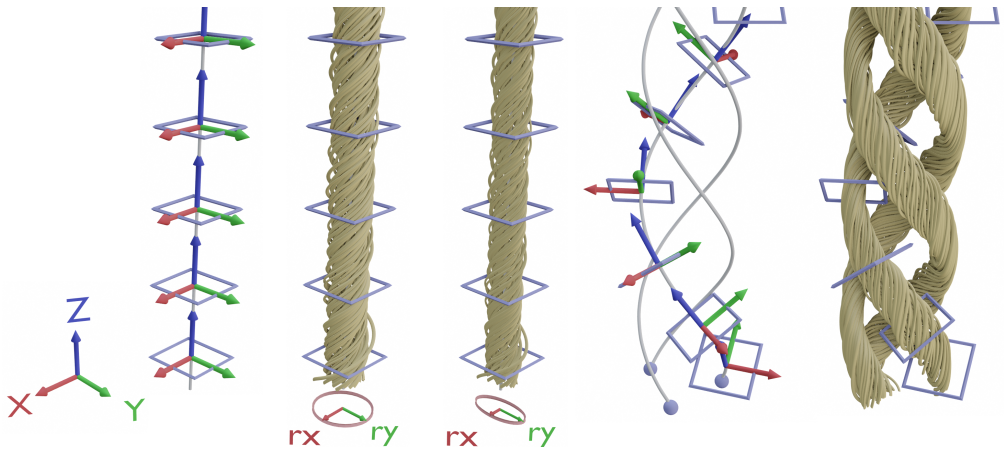

Figure 5.1: Hierarchical twisting process. a) Level 1 corresponds to a straight polygonal line. b) Twisted fibers from level 1 form a ply on the second level. c) Before twisting plies into a yarn, the x-axis of each ply is downscaled to create an elliptical cross-section. d) Multiple initial positions (blue) are sampled, and a helix curve with the specified properties is created at each. These curves, called *center lines*, represent the paths of the different plies. e) Deformed copies of the initial input follow each helix curve, resulting in the yarn on the third level and forming the input for the next step.

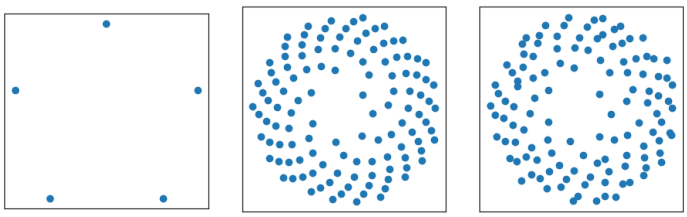

Figure 5.2: Ply and fiber distribution for the process explained in Fig. 5.1. For each level, multiple instances of the previous level are created and placed at initial positions according to a specified distribution. We use more randomness (middle) and jitter (right) on the fiber level and more structure on the ply level (left).

For larger numbers of instances, we sample the whole area of a disc. We distribute fewer samples towards the center, as instances in the middle are mostly occluded by the outer ones:

$$p_i = r_i \begin{pmatrix} \sin\theta_i \\ \cos\theta_i \end{pmatrix} + j_{xy} R \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \quad r_i = r\frac{i^{0.3}}{N^{0.3}}, \quad \theta_i = 2\pi \cdot 0.137 \cdot i \tag{5.3}$$

The heuristically chosen constants create a slightly pseudo-random distribution that is enhanced by the added jitter.

Next, for each sample point, we copy the instance template from the previous level, and then each instance is transformed as follows: Since the sampling patterns are roughly circular, we downscale the template along the x-axis, transforming its cross-section into an ellipse (see Fig. 5.1, c). The rotation ensures that the smaller radius of the ellipse is oriented toward the center, which simulates the squeezing of the individual fibers for dense packing. As a last step, the template is translated to the position of the sampling point.

To simulate the twisting that occurs during the production of real yarns, we create a helix in the z-direction at each sample point $p = (p_x, p_y)^T$ and transform the template to follow it accordingly (Fig. 5.1). The helix is given by:

$$\theta_i = \frac{i}{H} 2\pi + \arctan2(p_y, p_x) \tag{5.4}$$

$$r_h = \sqrt{p_x^2 + p_y^2} \tag{5.5}$$

$$s_i = 1 + \max(0, R_h) \cdot \cos\left(2R_h + \frac{i}{H} 2\pi R_h\right) \tag{5.6}$$

$$v_i = \begin{pmatrix} r_h s_i \sin\theta_i & r_h s_i \cos\theta_i & \frac{i}{H}\alpha_h + j_z R_i \end{pmatrix}^T \tag{5.7}$$

Here, α_h is the height of each complete turn, called the *pitch* of a helix. H is the helix resolution, i.e. the number of vertices per turn. The number of turns for the helix depends

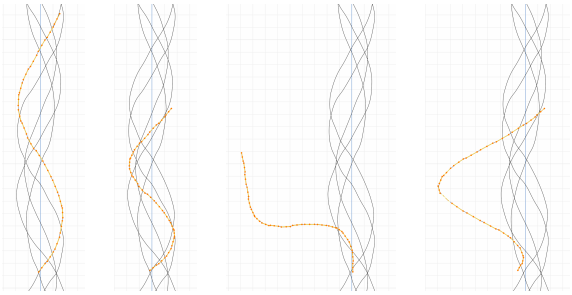

Figure 5.3: Generation of flyaways. a) A random vertex strip is selected and duplicated to become the new flyaway. b) The flyaway is scaled along its up-axis to exaggerate details. c) Hair flyaway: The flyaway from b) is rotated along its lowest point. d) Loop flyaway: The flyaway from b), where the vertices are moved radially according to a sine function, except for the first and last vertices, which remain at their previous locations, while the middle vertex is offset the most to simulate a loop.

on the desired total length of the generated yarn. Since the helix is always curved around the center line, its radius r_h is determined by the position of the sample point p. The angle θ_i has an offset that ensures that the 0th vertex coincides with p. The random variables R_i and R_h are drawn once per vertex and once per helix, respectively. However, different occurrences of R_i and R_h are drawn independently. s_i is the fiber migration value, modulation of the helix radius that varies along the vertical axis. It is realized by scaling the radius with a height-dependent cosine function with random amplitude, offset and phase speed.

Note that each template point is transformed to a local coordinate frame given by the helix at the corresponding height. We do not perform an actual physical simulation for the twisting process, as this would require a complex numerical simulation and thus increase computation time drastically.

For our recursive hierarchical raw yarn generation, we used generic variable names, such as N, R and α_h. The parameters of each level for the three-level fiber-ply-yarn model, used in the following, are summarised in Table 5.B.1.

5.4.2 Flyaway generation

After creating the raw yarn structure according to the previous section, we now model the flyaways. Flyaways are fibers that got displaced from their original position within the yarn. Following the previous work [Schröder et al., 2015; Zhao et al., 2016], we distinguish between two different categories of flyaways. *Hair flyaways* are fibers where one side is completely outside the yarn, while *loop flyaways* are fibers where both ends of the fiber are still inside the yarn, but the middle part is outside the main yarn. Both types of flyaways and the key steps of their creation are shown in Figure 5.3. The generation of flyaways is summarized in Algorithm 2. Flyaways are created by copying and transforming parts of the yarn. First, we determine whether the new flyaway will be a loop or a hair flyaway by drawing a uniformly distributed random number in $[0, 1]$ and determining whether it is greater or less than the

loop probability p_l. In both cases, the flyaway length is determined from a given mean and a

Algorithm 2 Flyaway generation

Require: flyaway parameter g, p_l, β, l_{hair}, s, l_{loop}, d_{mean}, d_{std}
1: **procedure** ADDFLYAWAYS(g, p_l, β, l_{hair}, s, l_{loop}, d_{mean}, d_{std})
2: **for** $k \in [1, g]$ **do**
3: $fly \leftarrow loop$ with probability p_l, or $fly \leftarrow hair$ else
4: **end for**
5: **if** $fly = loop$ **then**
6: $length \leftarrow l_{loop} + 0.01R$ (R as explained in 5.4.1)
7: **else**
8: $length \leftarrow l_{hair} + 0.05R$
9: **end if**
10: find fiber segment S of length $length$ via rejection sampling
11: **if** $fly = loop$ **then**
12: create loop flyaway using Eq. 5.8 on S
13: **else**
14: scale z coordinates of S and rotate by β to create hair flyaways (Fig. 5.3)
15: **end if**
16: **end procedure**

fixed standard deviation. Note that typical means are of the same order of magnitude as the standard deviations used. To find a fiber segment for the new flyaway, a random vertex is selected and the chain of connected vertices is followed in a random direction. If this chain ends before the desired length is reached, the process is repeated with a different starting vertex (rejection sampling). Once a suitable segment is found, it is copied and transformed according to its type in the next step. Copying a segment from the original yarn, rather than creating a new vertex line, preserves the deformation from the overlapping helixes from different levels, adding realistic detail.

Loop flyaways are created by overlaying the segment with a sine wave by adding an offset to each vertex:

$$o_i = d \sin\left(\frac{i\pi}{j}\right) \begin{pmatrix} v_x & v_y & 0 \end{pmatrix}^T, \quad d = d_{mean} + d_{std}R \tag{5.8}$$

The sine wave moves the vertex in a radial direction, keeping its vertical coordinate untouched. j is the total number of vertices in the segment, so exactly half a period of the sine wave is used, ensuring that the first and last vertices remain at their original positions, thus creating the loop shape. The amplitude d is chosen per flyaway, not per vertex.

Hair flyaways are created by rotating the segment by the angle β (see Fig. 5.3). Prior to rotation, they are scaled along the vertical axis by a value of s to amplify their shape.

Table 5.1: Parameters of our procedural Blender yarn model. Top: Fiber parameters, Middle: Ply parameters, Bottom: Flyaway parameters. Although fiber distribution and migration are not technically flyaway parameters, we consider them as such for our parameter prediction due to their probabilistic nature.

Parameter type	Parameter name	Explained
Fiber amount	m	Number of fibers in each ply
Fiber ellipse	t_x, t_y	Radii of fiber ellipse
Fiber twist	α	Pitch of the ply helix
Number of plies	n	Number of plies in the yarn
Ply ellipse	r_x, r_y	Radii of ply ellipse
Ply twist	α_{ply}, R_{ply}	Pitch and radius of the yarn helix
Fiber migration	j_z,j	Jitter of the fibers in z and in radial direction
Fiber distribution	j_{xy}	Jitter of fibers in xy plane direction
Flyaway amount	g	Number of flyaways
Loop probability	p	Probability for loop type flyaway
Hair flyaways	β, l_{hair}, s	Angle, hair length, fuzziness
Loop flyaways	l_{loop}, d_{mean}, d_{std}	Loop length, Mean and std of distance from ply center

Once all levels and flyaways are created, the bevel parameter is set to control the thickness and ellipticity of each fiber, giving the object a proper volume. All learnable parameters for the yarn and the flyaways are summarized in Table 5.B.1.

5.4.3 Further Implementation Details

To increase the realism of the resulting yarn appearance, we apply a reflectance model to the individual fibers, which describes their view- and illumination-dependent appearance. This allow us to obtain synthetic images of yarns by placing the yarn in a pre-built scene that resembles our measurement environment in the lab where we took the photos of real yarns.

We implement the yarn generation as a Python script inside the 3D modeling suite Blender, since it not only provides many of the operations needed during the generation, but also has powerful rendering capabilities. In particular, we leveraged Blender's *principled hair BSDF shader*, which is particularly relevant for our scenario of fiber-based yarn representation, as well as the built-in path tracer capable of rendering photo-realistic images with full global illumination to generate images depicting the synthesized yarns according to the conditions we expect to occur in photographs of real yarns.

5.4.4 Extensions to State-of-the-art Yarn Generator

Whereas Zhao et al. [2016] focused on woven cloth made of cotton, silk, rayon and polyester yarns, we observed that in addition to these fiber types, knitwear is often made of various

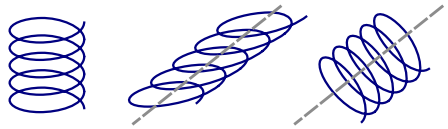

Figure 5.4: A ply (blue) is mapped to a helix segment (grey). The figure shows a very similar scene to Fig. 5.1, but drastically simplified and with exaggerated dimensions. a) The ply before mapping. b) Mapping by shifting orthogonal to the global vertical axis, as implemented in [Zhao et al., 2016]. c) Mapping by applying a local coordinate frame transformation, as implemented in our generator.

types of natural wool (cashmere, virgin wool, etc.) and acrylic a as wool substitute, as they offer exceptional warming properties and knitwear is mainly worn or used in the colder months. These and most other fiber types have longer flyaways, and their fibers exhibit elliptical cross-sections rather than circular ones, as assumed by Zhao et al. [2016]. These observations inspired us to make the following extensions to the current state-of-the-art models [Schröder et al., 2015; Zhao et al., 2016]:

- Hair flyaways: Instead of implementing hair flyaways in terms of adding hair arcs, we simulate them similarly to loop flyaways in terms of being pulled out of the plies. Hence, the twist characteristics are preserved (see Fig. 5.3, c)). Furthermore, we leverage hair squeezing to simulate the effect that when flyaways are released from the twist, they are less stretched and contract slightly (see Fig. 5.3, b). These two steps make even the longer flyaways look realistic (see Fig. 5.5, d-f).
- Elliptical fiber cross-sections: We implement the ellipticity of the cross-section of many types of fibers, which is particularly prominent in natural hair fibers such as wool. Although the geometric changes are too small to be seen directly, the shape of the cross-section affects the shading during the rendering, especially the prominence of specular highlights (see Fig. 5.5, a-c).
- Local coordinate frame transformation for helix mapping: Previously, in [Zhao et al., 2016], plies were twisted by sliding individual vertices orthogonal to the global vertical axis. Instead, we introduce a proper coordinate system transformation, which leads to more plausible results. In some cases, the differences are small, in others they are much more obvious. Fig. 5.4 shows an exaggerated case to illustrate the difference.
- Hierarchical generation: Sometimes, when multiple thinner threads are twisted into a thicker thread, yarns with more than three levels occur. Our hierarchical generator allows for any number of levels.

Evaluation of performance Although in its current implementation, the yarn generation process is more optimized for clarity and ease of use rather than efficiency, the time for generating all fiber and flyaway curves (about 6-12 seconds per image) is significantly less than the rendering time (about 1 to 4 minutes). This makes it suitable for our purpose of generating a database of yarns, but further optimization of the generation process may be an aspect for future developement.

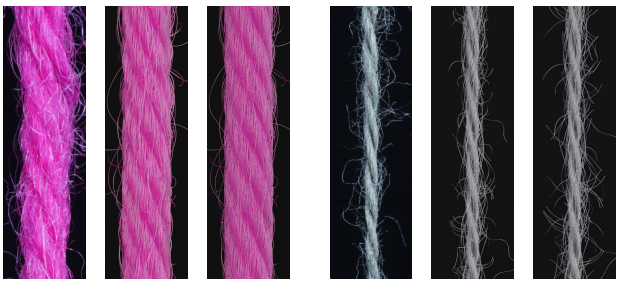

Figure 5.5: Left: Comparisson of fiber cross section. a) Photograph of a wool yarn with elliptical cross section. b) Virtual yarn generated with elliptical fiber cross-section. c) Virtual yarn generated with circular fiber cross-section. The changes in geometry are hard to spot when zoomed out, however the shading and in particular the strength of the specular highlights is clearly affected by the cross-section shape. Right: Effect of the squeeze parameter s. d) Reference, e) With squeeze, f) Without squeeze.

5.4.5 Yarn dataset

To represent the variations in color and reflective characteristics encountered in real yarns in our synthesized yarn dataset, we sample different of these parameter configurations by uniformly sampling the parameters within the corresponding, heuristically determined intervals shown in Table 5.B.1 and then rendering the resulting yarns in different conditions that we expect to occur when considering photos of real yarns. We provide details of our guided parameter sampling procedure in the Section 5.B. All yarns in our database consist of two to six plies. Note that our yarn generator allows the generation of yarns with more plies, but our observations indicate that three, four and five plies are the most common scenarios in the case of knitting yarns. Fig. 5.6 depicts some of the yarns from the database. In total, we sampled 4000 parameter configurations for the synthetic training set and 345 parameter configurations for the synthetic validation set, resulting in 4000 images with a resolution of 2000x600 pixels for training and 345 images with a resolution of 2000x600 pixels for validation.

Although our yarn generator can generate many levels of hierarchy, for proof-of-concept purposes, in this paper, we focused on yarns made up of plies and did not investigate learning the next level, where multiple thinner yarns are twisted into a thicker yarn. Therefore, our database does not include such yarns. Furthermore, by rendering the yarn in different scenes, including various indoor and outdoor settings, training data for *in-the-wild* yarn parameter estimation could be generated.

5.5 Inference of yarn characteristics from input images

We model the prediction of the parameters for our procedural yarn model from images as a regression problem. Our training and validation dataset consists of annotated synthetic yarn images that we use to train a model that allows inferring yarn parameters from novel images

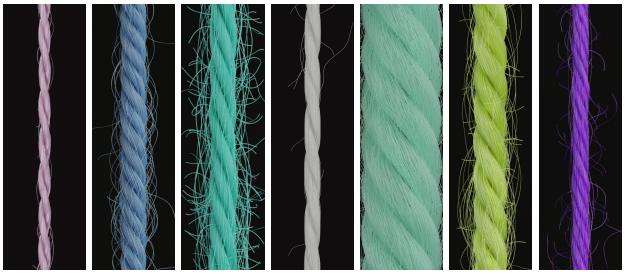

Figure 5.6: Examples of synthetic yarns in our database.

of yarns not seen during training or validation. Before providing details of our respective approach (see Section 5.5.1), we motivate our choice of a suitable network architecture that is capable of handling the challenging nature of the underlying problem.We considered the saliency maps [Simonyan et al., 2014] of networks trained to predict the set of raw yarn's parameters and the set of the flyaway parameters with two independent models. An entry $m_{i,j}$ in a saliency map for a model f that has been trained on a subset of P parameters is the maximum derivative of the average value of the predicted parameters with respect to a pixel $x_{i,j,c}$ in the input image over the color channels c, i.e.

$$m_{i,j} = \max_c \left| \frac{\partial}{\partial x_{i,j,c}} \left(\frac{1}{P} \sum_p f_p(x) \right) \right|. \tag{5.9}$$

In contrast to saliency maps that have been proposed within the context of classification networks, we consider the derivative of the mean of the predicted parameters because we need to investigate the effect of a pixel on the entire subset on which the network was trained.

The saliency maps (Figure 5.7) for the network trained to predict the raw yarn parameters illustrate a higher susceptibility to changes in the yarn center region of the image. On the other hand, the saliency maps for the network that predicts the flyaway parameters indicate that such a network exhibits a higher sensitivity towards the border regions within the input images. Motivated by these saliency maps, we concluded that it is better to train separate models for the raw yarn parameters and the flyaway parameters.

5.5.1 Inference of yarn parameters

As already mentioned before, we formulate the problem in terms of a regression problem. Here an encoder f maps an input image to a latent code which then becomes the input to a regression head h (Fig. 5.8) which performs the parameter regression. This regression path within our model is trained to minimize an L_1 loss between the prediction obtained on synthetic yarn images $x^{(i)}$ and their ground truth parameter $y^{(i)}$ used in the generation model.

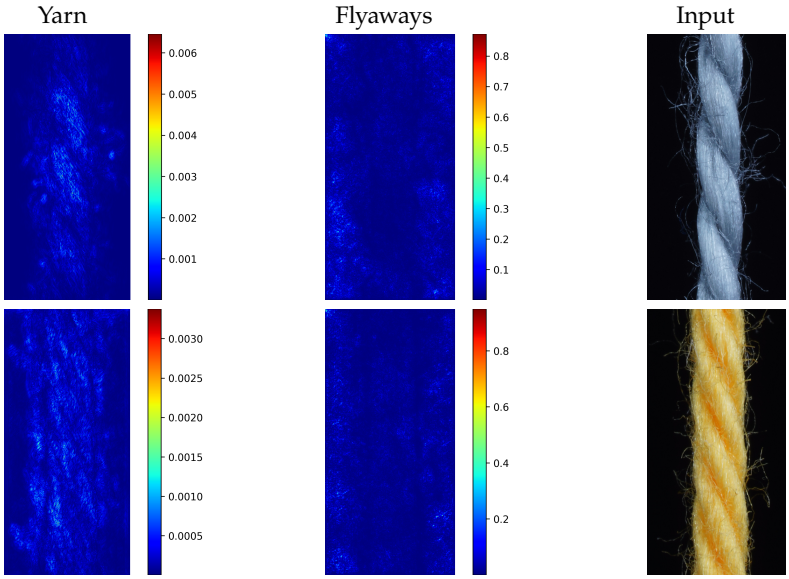

Figure 5.7: Saliency maps computed for network configurations that are trained either to predict geometry (columns 1) or flyaway parameters (column 2) of the yarn model and either respective inputs (column 3). The color temperature in a saliency map indicates an input pixel's influence on the predicated parameter. Lighter/warmer colors correspond to a stronger influence.

$$\mathcal{L}_{\text{regress}} = \mathbb{E}\left[\left\| h(f(x^{(i)})) - \hat{y}^{(i)} \right\|_1\right] \tag{5.10}$$

We will refer to this network simply as Reg.

Although the synthetic training data was carefully generated to match the appearance of real yarn photographs as accurately as possible, a domain gap between the synthetic and real images cannot be ruled out. To address the domain gap, we investigated the impact of adding some not annotated real images to the training and utilized a semi-supervised training process which interleaves synthetic and real images in the training process to improve the extrapolation from synthetic images with known yarn parameters to real photographs. For this purpose, we extended our aforementioned regression model into an autoencoder with an additional regression by adding a decoder model d. The autoencoder of the path is trained to minimize a simple image reconstruction loss both on synthetic and real images:

$$\mathcal{L}_{\text{recon}} = \mathbb{E}\left[\| d(f(x^{(i)})) - x^{(i)} \|_F^2\right]. \tag{5.11}$$

This unsupervised training process enables our encoder to be trained to map synthetic and real images into the same latent space from which the regression head predicts the yarn parameters for synthetic data points. During inference, only the encoder and regression head are required to predict inputs for the parametric yarn model. In an ideal case, the encoder maps the synthetic and real images of similar yarn to vectors that are within a close proximity

in its latent space. However, such a behavior is not guaranteed by the reconstruction loss. On the contrary, the encoder might learn to distinguish between the synthetic and real images so that their latent vectors from two distinctive clusters. We propose an additional regularization term which explicitly penalizes the distance between latent codes of synthetic images and the average latent code of real images in the encoder's latent space:

$$\mathcal{L}_{\text{latent}} = \mathbb{E}\left[\|f(x^{(i)}) - \text{sg}(\mu_{\text{SMA}})\|_F^2\right] \tag{5.12}$$

where sg denotes the stop gradient function (i.e. μ_{SMA} is considered to be a constant in the backward step) and μ_{SMA} is a simple moving average of latent codes of real images for the last 5 batches. With this additional regularization term the average latent code of synthetic and real images are close to each other, and distinctive clusters become energetically less optimal. Note that the evidence lower bound (ELBO) in the objective function of variational autoencoder could be considered as an alternative regularization. However, it is more restrictive by forcing the latent code to be roughly normal distributed which is not necessary in this application.

In three different networks Reg_{latent}, Reg^{ae} and Reg^{ae}_{latent} we investigated the following three combinations of those losses:

- Network Reg_{latent}: $\mathcal{L}_{\text{reglat}} = \mathcal{L}_{\text{regress}} + \lambda_{\text{latent1}}\mathcal{L}_{\text{latent}}$.
- Network Reg^{ae}: $\mathcal{L}_{\text{regrec}} = \mathcal{L}_{\text{regress}} + \lambda_{\text{recon1}}\mathcal{L}_{\text{recon}}$.
- Network Reg^{ae}_{latent}: The combination of both previous variants, i.e.: $\mathcal{L}_{\text{combined}} = \mathcal{L}_{\text{regress}} + \lambda_{\text{recon}}\mathcal{L}_{\text{recon}} + \lambda_{\text{latent}}\mathcal{L}_{\text{latent}}$.

where $\lambda_{\text{recon}}, \lambda_{\text{recon1}}, \lambda_{\text{latent}}, \lambda_{\text{latent1}}$ are hyper-parameters of the models. The combined architecture is provided in Figure 5.8.

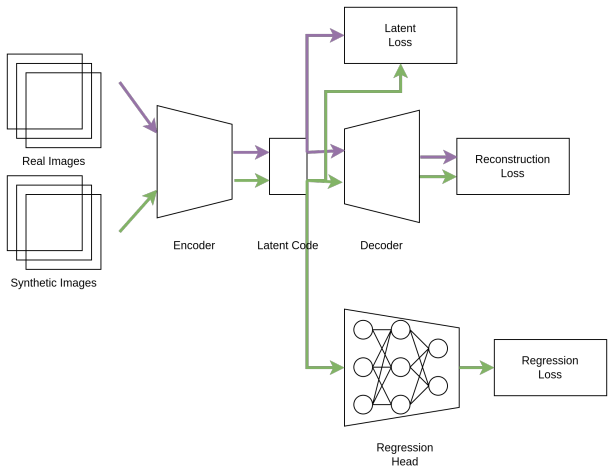

Figure 5.8: Overview of the interleaved training process. The depicted architecture combines all the different networks we investigated: The networks Reg and Reg_{latent} consist of the encoder and the regression head, while the networks Reg^{ae} and Reg^{ae}_{latent} additionally include the decoder.

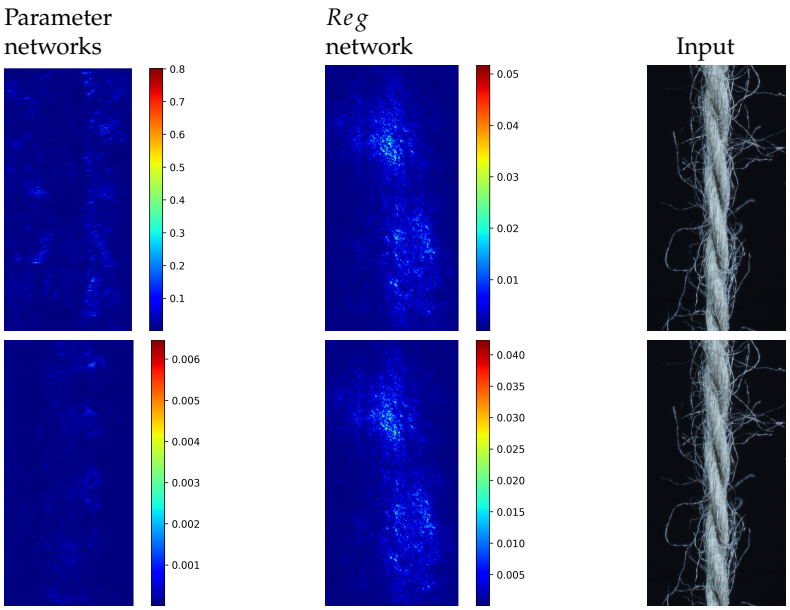

Figure 5.9: Sailiency maps for the yarn twist pitch α_{ply} (top row) and the yarn radius R_{ply} (bottom row).

Network Architecture The encoder architecture is a pure CNN model based on ResNet [He et al., 2015] were the average pooling has been moved to the regression head i.e. the latent codes are the tensors which result from the convolution stack. We explore both the ResNet18 and ResNet34 configurations with the standard ResNet block as proposed in [He et al., 2015] as well as the more recently proposed convnext blocks which also replaces the batch normalization with layer normalization [Liu et al., 2022]. If the encoder uses a ResNet18 or ResNet34 architecture, the regression head h is a two layer MLP with a hidden dimension of 512 and Exponential Linear Unit activation function after the first layer. Otherwise, the regression head is a linear projection of the 512-dimensional input onto the required output dimension.

The decoder g consists of four transposed convolutions with kernel sizes $k_1 = 2$, $k_2 = 2$, $k_3 = 2$, $k_4 = 2$, the stride $s_i = k_i/2$ and output kernel sizes 256, 128, 64, 3. The first three layers use ReLU activations, while the last layer uses the Tanh function to ensure that the output values are within the range of the pixel values.

Whereas small modifications of the parameters of the basic yarn structure (i.e. without flyaways) already have a significant visual effect on the resulting yarn (see Fig. 5.16 for different values of the parameter α_{ply}), small variations of the parameters for the flyaway characteristics do not have such a significant impact on the generated yarn since only the distribution of the flyaway characteristics has to be matched to obtain plausible flyaways. For this reason, we compared the saliency maps from models trained for different raw yarn parameters individually (e.g., Fig. 5.9 shows the maps for the yarn radius and parameter α_{ply}) and found that they differ. Motivated by the results of the individual saliency maps,

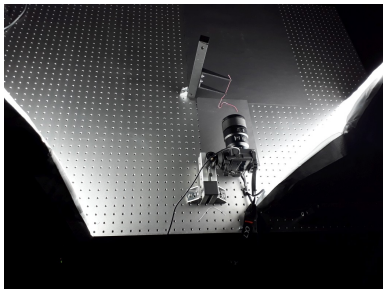

Figure 5.10: Our setup for capturing the test yarns.

we investigated the use of separate networks for the separate prediction of each of the raw yarn parameters. A similar separation of models has already been observed by Nishida et al. [2018]. We compared the previously described approach of using separate networks to predict the raw yarn parameters and the flyaway characteristics with the approach of using separate networks to predict each raw yarn parameter separately along with a network to predict the flyaway characteristics.

In this context, we leveraged further priors for some of the parameters to exploit their underlying nature. For the parameter *number of plys*, we changed the last layer from the identity function as used for regression to a softmax function, thereby framing the prediction of this discrete parameter as a classification problem. The underlying motivation is that most knitting yarns have 2 to 6 plys and the estimation of the number of plys based a classification might be easier that based on a regression. Furthermore, we distinguish fibers according to their elliptic cross-section characteristics into *thin fibers* ($t_x = 0.01$, $t_y = 0.007$) and *thick fibers* ($t_x = 0.018$, $t_y = 0.01$), which we also frame as a classification problem since considering all intermediate states seems tricky and there seems to be no such significant perceptual difference for these intermediate states.

5.6 Experiments

Training, validation and test data We use 4000 synthetic yarns for training and 345 synthetic yarns for validation as mentioned in Section 5.4.5. To get insights on the performance of our method for parameter inference for real yarns depicted in photographs, we tested our approach on different knitting yarns, which were made either of one type of fiber such as wool, acrylic, cotton, polyamide, etc. or of a mix of different types of fibers (Fig. 5.12, top row, second yarn). We took the corresponding photos of the yarns under a simple lab setup (see Fig. 5.10) with a Sony α7RIII camera using the makrolens Makro G OSS with FE 90 mm F2.8. Then we cropped the photos to the size of 600×2000 pixels, ensuring that the yarn roughly runs through the center of the image. These cropped photos served as an input for the parameter inference. As our networks were trained for inputs of 1200 times 584 pixels, we again crop the previously cropped photos randomly to the required size before performing the forward pass and hence determining the parameters.

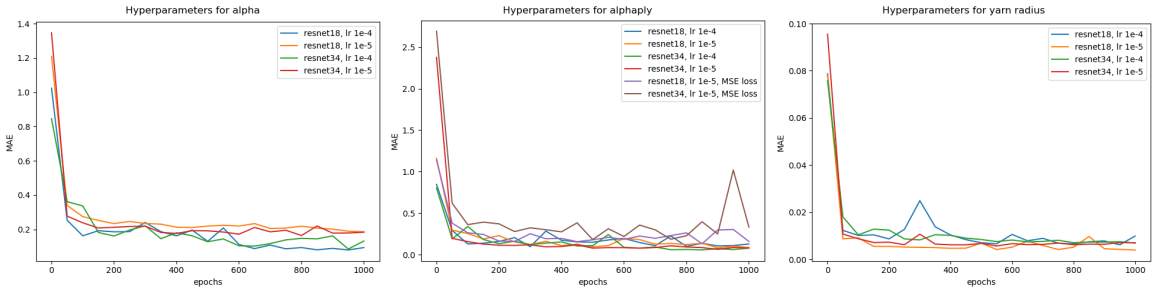

Figure 5.11: Examples of validation loss comparisons for hyperparameter determination for parameters α (left), α_{ply} (middle) and R_{ply} (right). Based on the loss values we chose the model of ResNet18, learning rate = $1e^{-4}$ and epoch 850 for α, ResNet34, learning rate = $1e^{-4}$ and epoch 850 for α_{ply} and ResNet18, learning rate = $1e^{-5}$ and epoch 1000 for R_{ply}.

Details of the training process To improve the robustness of the trained models, we increase the variety of the training data by randomly cropping the 4000 images of a size of 2000x600 pixels to the network input size of 1200x584 pixels during each epoch. Then we run the training for 1000 epochs with a batch size of 32 and a learning rate of 0.0001 based on the Adam optimizer [Kingma and Ba, 2015]. For this purpose, we used three Nvidia Titan XP GPUs, each having 12 GB of RAM. Based on this hardware, the training for the flyaway network and the full regression network took each approx. 11 hours. When training only for one parameter, the training for the ResNet34 took ca. 4 hours, while for the ResNet18 it was 2.5 hours.

5.6.1 Parameter inference on real data

Validation of training process First, we need to validate whether the trained model shows the potential to perform well on synthetic validation data. For this purpose, we compared the inference of yarn parameters for validation data through a set of different models against one model for all parameters. In this scope, we compared the the approaches of using two separate networks for predicting the raw yarn parameters and flyaway characteristics and using separate parameter specific networks for predicting each individual raw yarn parameter separately together with a network for predicting the flyaway characteristics, and have chosen the best hyperparameters and the best epoch based on the validation loss computed on synthetic validation set. Figure 5.11 illustrates the validation losses for the twisting parameters α, α_{ply} and yarn radius R_{ply}. Table 5.2 shows the comparison of the best models of every case. We can see that the loss over each parameter is bigger when training one model for all parameters of the raw yarn instead of training specific models with different hyperparameters for each parameter separately. While this indicates a better capability to infer yarn parameters on synthetic data, we did not yet analyze the generalization to images depicting real yarns, which will follow with the experiments regarding performance analysis on real data.

Table 5.2: Validation loss of different networks. Note that especially the important yarn twist parameters α, α_{ply} and R_{ply} are better learned with parameter specific networks.

	parameter specific networks	Reg	Reg^{ae}	Reg_{latent}	Reg^{ae}_{latent}
r_x	0.0080	0.0082	0.0080	0.0097	0.0087
r_y	0.0066	0.0066	0.0074	0.0083	0.0079
m	12	12	13	14	14
α	0.0807	0.2493	0.2587	0.3230	0.3026
α_{ply}	0.0614	0.1953	0.2101	0.2400	0.2433
R_{ply}	0.00444	0.0082	0.0092	0.0092	0.0095

Performance evaluation We now provide an evaluation of the performance of the different approaches of using a single network for predicting all parameters of the raw yarn and a network for the prediction of the flyaway parameters when using different loss formulations as discussed before in comparison to using also separate networks for the prediction of the raw yarn parameters. This means the model for prediction of the flyaway parameters is the same for all these approaches. In our experiments, the flyaway model with the lowest validation loss was the ResNet18 model trained with a learning rate of 0.001 and MAE loss, which we therefore use for the subsequent experiments. Exemplary results of our experiments including a comparison between the investigated approaches can be observed in Figure 5.12. The corresponding inferred parameters are presented in Table 5.A. The renderings of the parameters inferred from parameter specific networks for each parameter of the raw yarn look more similar to the input image, than the renderings from the other approaches.Since we do not have the ground truth parameters for our real world yarns, we can only compare the geometry appearance of the yarns. Based on the appearance comparison to the input image, we conclude that the approach of the parameter specific networks is most suitable for the given task.

Additionally, we utilize the parameter inference for renderings of knitting samples. In Figure 5.13, we show the renderings of knitting samples, made with the three yarns of the top row from Figure 5.12 with the parameters from the ensemble of per-parameter networks for the raw yarn structure. We observe that yarns with different geometry lead to entirely different appearances of the same pattern. Furthermore, we can see that if the inferred yarn looks similar to the yarn in the image, the pattern rendered with the inferred yarn will also look similar to the pattern knitted with the real yarn.

Once the parameters are inferred, we can use them also for editing and for the creation of new yarns. Some examples for modification of the twist parameters α and α_{ply} as well as some flyaways parameters are depicted in Figure 5.15.

Ablation study regarding effect of resolution To get insights on the effect of the resolution of the input images, we trained the networks for the different yarn parameters on images of significantly lower resolution. We experimented with the reduction to 50 and 25 percent of the original resolution of 1200 times 584 pixels. Figure 5.14 shows the validation loss

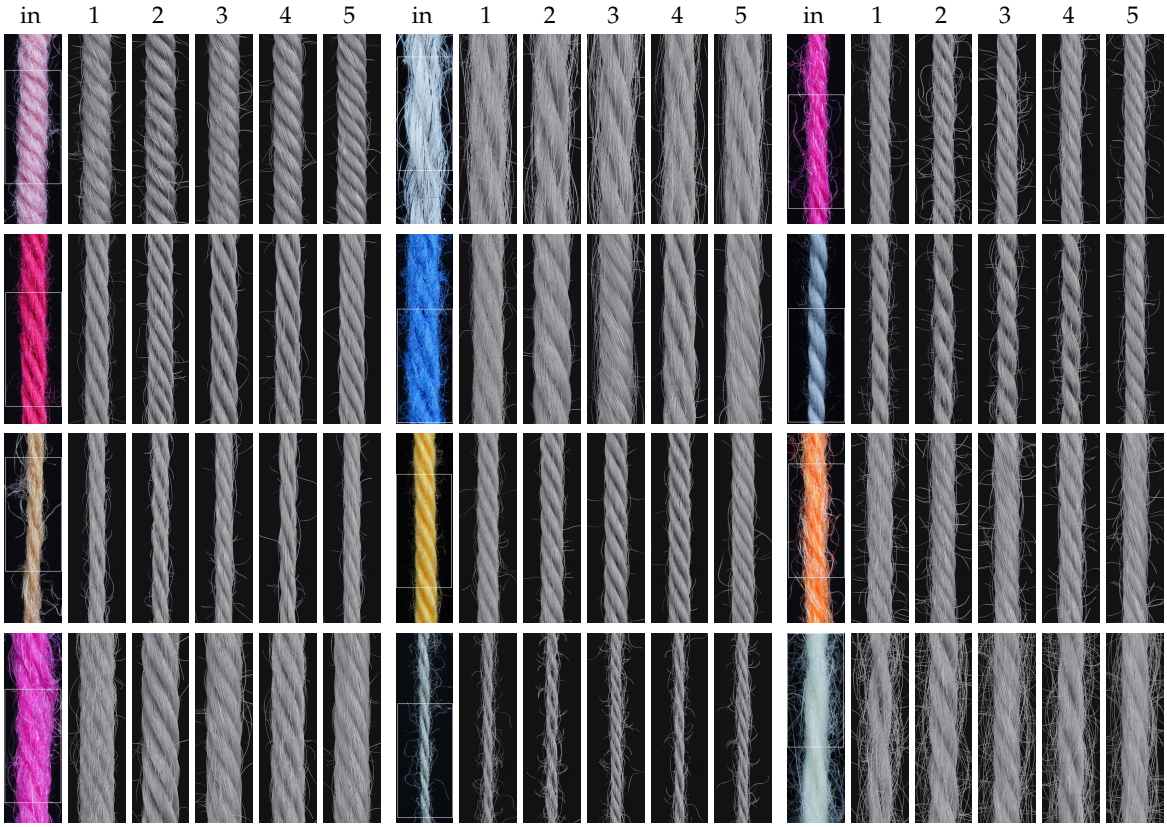

Figure 5.12: in = input image, 1 = reconstruction image from parameter specific models, trained for each yarn parameter separately, 2 = Reg, 3 = Reg_{latent}, 4 = Reg^{ae}, 5 = Reg^{ae}_{latent}. The rectangle region shows the input image, which was randomly cropped from the whole image.

plots for the alphaply and yarn radius parameters for different resolutions. Figure 5.16 shows some visual comparisons of reconstructed yarns with the corresponding parameters for alphaply. As can be observed, the achieved accuracy decreases with decreasing image resolution. We expect this to be a result of the lower quality of the depiction of the individual fiber arrangements that can be seen in terms of a blurring of the yarn structure. In order to demonstrate the robustness of our approach to different exposure times we made exposure series of the input yarns and tested images with different exposure times. The results show that as long as the images are not too dark or over-exposed, the inferred parameters vary only insignificantly and the reconstructions are very similar.

5.6.2 Limitations

In addition to the dependence on the quality of the depicted fiber arrangements (as shown in the previous section), our approach depends on having the variations to be expected in the test data included in the training data. Note that our dataset includes only yarns with a normal (helix-like) fiber twisting. However, other fiber twisting-types could also occur as

Figure 5.13: 1st and 3rd rows: images of a real knitted cloth (made with yarns from the top row of Fig. 5.12) for the pattern consisting of knit (1st row) and purl (3rd row) stitches. 2nd and 4th rows: rendering of the same stitch pattern with the inferred yarn with default material settings.

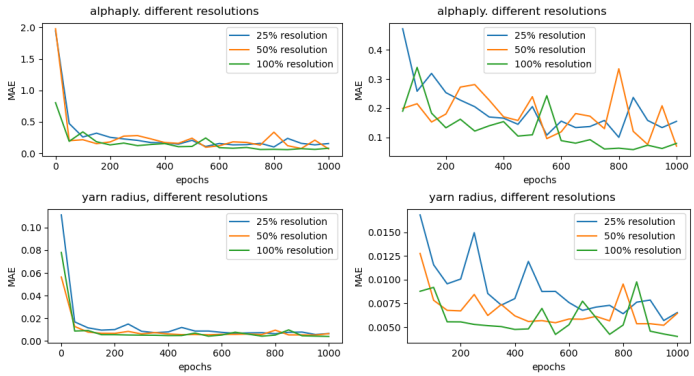

Figure 5.14: Validation loss comparisons for trainings with different resolution of input images. Left: full loss curves, right: loss curves without the first element.

shown with the example in Figure 5.17. The depiction shows a reconstruction that exhibits a high similarity to the input yarn. The thickness on both ply- and yarn-level as well as the number of twists closely follow the original structure.Since we did not consider this type of ply-twist in our yarn generator, there is also some deviation. We expect that such deviations might be handled by further extending the dataset regarding further types of yarn variations.

Furthermore, despite the fact that our yarn generator also supports the fourth hierarchical level (i.e., where thinner yarns are twisted into thicker yarns, see Figure 5.17 d)-e)), we only

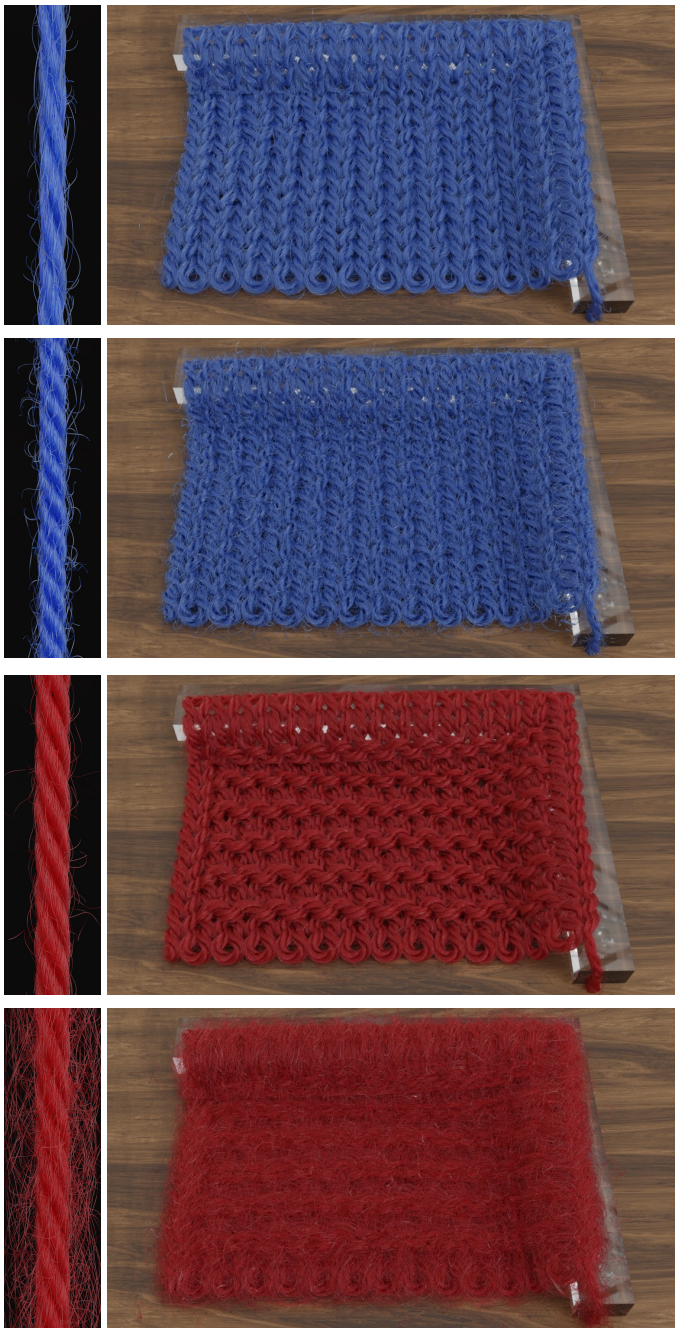

Figure 5.15: Two examples of editing operations for yarns with original inferred parameters and the edited ones together with corresponding renderings of knitted patches. Reflectance parameters were not part of the inference but chosen arbitrarily for demonstration. 1st row: golden yarn from Figure 5.12 in the 3rd row, left. 2nd row: the same yarn but with both pitch parameters α and α_{ply} divided by 2. 3rd row: yellow yarn from Figure 5.12 in the 3rd row. 4th row: the same yarn but with parameters for flyaway amount and length multiplied by 2.

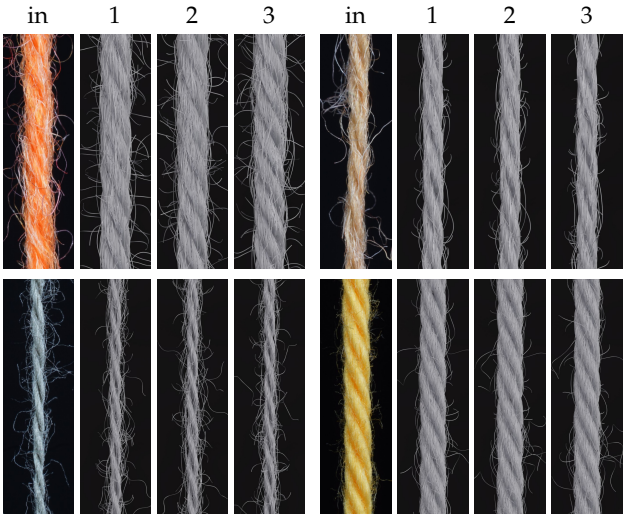

Figure 5.16: in = input image, 1 = reconstruction image from different models trained for each yarn parameter separately, where the α_{ply} parameter was trained on images with full resolution, 2 = α_{ply} parameter was trained on images with 50% of the full resolution, 3 = α_{ply} parameter was trained on images with 25% of the full resolution.

included yarns represented based on the first three levels, which limits our approach to the prediction of the characteristics up to the third level. However, the extension to the level of also twisting yarns is straightforward and we leave it for future work.

5.7 Conclusions

We presented an investigation of different neural inverse procedural modeling methods with different architectures and loss formulations to infer procedural yarn parameters from a single yarn image. The key to our approach was the accurate hierarchical parametric modeling of yarns, enhanced by handling elliptic fiber cross-sections, as occurring in many types of natural hair fibers, as well as more accurately handling flyaway characteristics and the twisting axis and the respective generation of synthetic yarns that are realistic enough so that the trained model can extrapolate to the real yarn inputs. Our experiments indicate that that the complexity of yarn structures in terms of twisting and migration characteristics of the involved fibers can be better encountered in terms of ensembles of networks that focus on individual characteristics than in terms of a single neural network that estimates all parameters. In addition, we demonstrated that carefully designed parametric, procedural yarn models in combination with respective neural architectures and respective loss functions even allow robust parameter inference based on models trained on purely synthetic data. In the scope of this paper we focused solely on the geometric fiber arrangement including migration characteristics (i.e. flyaways) and left the prediction of the reflectance characteristics of knitting yarns for future work. Further developments may also consider a further hierarchical level of yarns, i.e. thinner yarns twisted to thicker yarns. Whereas we

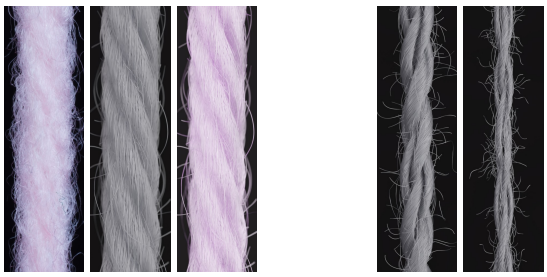

Figure 5.17: a) Input image of a yarn made by unusual (non-helical) fiber twisting procedure. b) Rendering of a yarn with infered parameters with default material. c) Rendering with color. d) and e) Examples of yarns of fourth level, where two thinner yarns are twisted into one to make it thicker and better suitable for knitting: d) Two yarns of the thin grey yarn from Figure 5.12, fourth row, e) Two yarns of the thick grey yarn from second row of Figure 5.12

did not focus on inferring parameters for this kind of yarns, our yarn generator would be able to produce the respective characteristics and might allow enriching the dataset accordingly in future work.

Appendix

5.A Inferred yarn parameters

In Section 5.6, we presented exemplary results of our experiments on yarn parameter inference, including a comparison between all investigated approaches (Figure 5.12). The corresponding inferred parameters are presented in the Tables 5.A.1 (raw yarn parameters) and 5.A.2 (flyaway parameters).

5.B Yarn sampler

In the course of our experiments, we heuristically determined parameter sampling intervals that produced natural-looking yarns. These intervals are presented below. For some of the parameters, we defined sampling intervals directly, while for others, the sampling was implemented through auxiliary variables.

The intervals for fiber thickness were chosen as follows:

$$t_y = [0.006, 0.01] \tag{5.13}$$
$$t_x = [t_y, 2.5 \cdot t_y] \tag{5.14}$$

For the number n of plies, we sampled integers between 2 and 6, while for the number m of fibers, we sampled integers between 40 and 200. All other raw yarn parameters were sampled indirectly using auxiliary variables.

Let $r_{frac} = \frac{r_y}{r_x}$ be the parameter that reflects how much a ply has been squeezed during the twisting and how much it deviates from its original circular cross-section (Fig. 1 in the paper). The fewer plies there are in the yarn, the less they resemble a circle after twisting:

$$n = 2 \Rightarrow r_{frac} = [0.67, 0.9] \tag{5.15}$$
$$n = 3 \Rightarrow r_{frac} = [0.72, 0.91] \tag{5.16}$$
$$n > 3 \Rightarrow r_{frac} = [0.85, 0.95] \tag{5.17}$$

Table 5.A.1: Inferred raw yarn parameters for the yarns of the Figure 5.12 from top to bottom and from left to right. For each yarn there are 5 rows, each corresponding to parameter detection from different experiments: from top to bottom: parameter specific models, Reg, Reg^{ae}, Reg_{latent}, Reg^{ae}_{latent}

yarn	n	α_{ply}	R_{ply}	α	r_x	r_y	m	thickness
	4	4.515	0.555	-3.611	0.329	0.283	72	2
	4	4.591	0.614	-3.217	0.305	0.285	102	0.021,0.008
rose	5	5.186	0.603	-3.989	0.327	0.286	107	0.020,0.008
	4	5.053	0.546	-3.545	0.332	0.295	100	0.018,0.008
	5	5.110	0.581	-3.474	0.304	0.274	118	0.018,0.007
	4	7.815	0.449	-2.442	0.297	0.240	131	1
	5	6.636	0.518	-2.647	0.250	0.225	106	0.018,0.008
red	4	7.760	0.520	-3.273	0.282	0.249	132	0.017,0.008
	4	7.911	0.477	-3.033	0.282	0.246	139	0.018,0.008
	5	7.663	0.504	-3.786	0.280	0.238	135	0.016,0.008
	3	6.474	0.289	-2.392	0.241	0.195	43	2
	3	5.823	0.291	-2.326	0.236	0.200	62	0.022,0.008
golden	4	5.708	0.292	-3.175	0.228	0.192	87	0.020,0.008
	3	6.714	0.261	-3.223	0.238	0.206	81	0.019,0.008
	4	6.021	0.280	-3.150	0.227	0.194	72	0.017,0.007
	6	10.679	0.672	-3.288	0.390	0.350	64	2
	5	9.884	0.758	-4.457	0.358	0.373	86	0.021,0.008
pink 6ply	5	9.079	0.669	-5.512	0.399	0.363	92	0.021,0.008
	5	11.173	0.636	-3.868	0.371	0.328	106	0.018,0.008
	6	11.000	0.688	-6.077	0.377	0.332	113	0.017,0.007
	3	14.961	0.639	-6.152	0.521	0.423	99	2
	3	13.159	0.598	-5.445	0.472	0.415	112	0.022,0.008
mixed	3	11.965	0.572	-6.160	0.478	0.409	129	0.020,0.008
	4	14.522	0.540	-4.994	0.441	0.386	135	0.019,0.008
	4	15.316	0.635	-6.440	0.433	0.391	124	0.018,0.007
	4	11.009	0.602	-2.866	0.406	0.298	74	1
	4	10.108	0.663	-6.308	0.458	0.407	151	0.017,0.008
blue	5	7.545	0.631	-7.674	0.395	0.359	120	0.017,0.008
	4	11.868	0.606	-4.054	0.383	0.335	114	0.019,0.008
	6	9.505	0.638	-6.250	0.366	0.325	123	0.017,0.007
	4	6.398	0.393	-2.023	0.278	0.236	65	1
	4	6.176	0.425	-2.838	0.272	0.237	92	0.015,0.007
yellow	4	6.087	0.435	-3.384	0.254	0.222	110	0.014,0.007
	4	5.897	0.424	-1.953	0.261	0.233	103	0.014,0.007
	5	5.996	0.415	-3.054	0.247	0.217	105	0.014,0.007
	2	3.348	0.139	-1.175	0.146	0.115	82	1
	2	3.230	0.179	-1.329	0.155	0.122	105	0.017,0.008
grey thin	3	3.005	0.171	-1.430	0.147	0.118	97	0.017,0.007
	2	3.648	0.154	-1.320	0.162	0.130	84	0.016,0.008
	3	3.456	0.168	-1.306	0.134	0.108	102	0.015,0.007
	4	5.605	0.364	-2.419	0.249	0.230	49	2
	4	4.867	0.400	-2.529	0.248	0.222	67	0.021,0.008
pink 4ply	4	5.206	0.367	-3.360	0.264	0.222	91	0.020,0.008
	4	5.381	0.359	-3.160	0.251	0.218	72	0.019,0.008
	5	5.704	0.381	-3.189	0.251	0.219	83	0.018,0.007
	2	3.825	0.254	-2.525	0.334	0.232	148	1
	2	4.142	0.308	-3.109	0.322	0.250	188	0.014,0.007
grey thick	2	3.630	0.284	-3.015	0.342	0.274	163	0.016,0.008
	2	3.770	0.291	-3.057	0.326	0.246	173	0.016,0.008
	3	4.126	0.296	-3.328	0.317	0.260	176	0.014,0.007
	4	6.995	0.440	-3.181	0.309	0.275	52	2
	5	6.275	0.447	-3.745	0.270	0.246	66	0.021,0.008
orange	5	6.762	0.454	-5.513	0.299	0.234	70	0.020,0.008
	4	6.932	0.405	-3.506	0.312	0.276	71	0.018,0.008
	5	6.921	0.440	-5.403	0.290	0.256	69	0.018,0.007
	2	10.705	0.369	-3.654	0.380	0.318	93	2
	3	8.471	0.477	-5.574	0.433	0.380	101	0.017,0.008
light	4	7.926	0.465	-6.81	0.369	0.324	101	0.016,0.008
	3	10.433	0.409	-4.501	0.410	0.357	113	0.017,0.008
	4	9.913	0.472	-5.625	0.363	0.319	90	0.014,0.007

Table 5.A.2: Inferred flyaway parameters for the Figure 5.12.

yarn	m	p	l_{hair}	β	s	l_{loop}	d_{mean}	d_{std}	j_{xy}	j
red	151	0.48	2.41	0.86	0.65	4.75	7.77	2.35	0.014	0.20
golden	123	0.50	2.26	0.77	0.44	4.48	4.44	2.08	0.014	0.19
light 3ply	160	0.53	4.71	0.77	0.55	12.38	11.10	2.68	0.019	0.19
orange	163	0.54	3.50	1.07	0.46	5.14	7.83	1.78	0.015	0.20
rose	147	0.51	2.61	0.83	0.49	4.40	8.30	1.53	0.014	0.21
grey-blue	177	0.44	1.93	0.97	0.43	4.03	5.42	1.70	0.010	0.16
pink 4ply	154	0.56	3.1	0.82	0.45	4.66	4.69	2.35	0.018	0.23
blue	182	0.45	2.91	0.85	0.73	7.20	10.61	1.68	0.014	0.19
grey	200	0.42	1.72	0.85	0.36	3.22	3.38	1.52	0.015	0.17
yellow	183	0.35	1.78	0.88	0.52	4.14	7.79	1.45	0.011	0.16
pink 6ply	175	0.54	3.70	0.97	0.62	7.04	10.74	2.14	0.016	0.22
light 2ply	223	0.47	4.75	1.12	0.48	8.14	13.12	2.08	0.011	0.23

Let the parameter $area_{frac}^{ply}$ represent different fiber densities in a ply:

$$area_{frac}^{ply} = \frac{m \cdot t_x \cdot t_y \cdot \pi}{r_x \cdot r_y \cdot \pi} = \frac{m \cdot t_x \cdot t_y}{r_x^2 \cdot r_{frac}} \tag{5.18}$$

We sample it from the following interval:

$$area_{frac}^{ply} = [0.035, 0.215] \tag{5.19}$$

Then the parameters r_x and r_y can be computed as

$$r_x = \sqrt{\frac{m \cdot t_x \cdot t_y}{area_{frac}^{ply} \cdot r_{frac}}} \tag{5.20}$$

$$r_y = r_{frac} \cdot r_x \tag{5.21}$$

The auxiliary variable $area_{frac}^{yarn}$ depicts the ply density in the yarn:

$$area_{frac}^{yarn} = \frac{m \cdot t_x \cdot t_y \cdot \pi}{r_x \cdot r_y \cdot \pi} = \frac{m \cdot t_x \cdot t_y}{r_x^2 \cdot r_{frac}} \tag{5.22}$$

We sample from the following interval:

$$area_{frac}^{yarn} = [0.55, 0.82] \tag{5.23}$$

Furthermore, we sample both the auxiliary variable γ of the helix angle of the fiber twist in a

Table 5.B.1: Value intervals of the learnable parameters in our synthetic yarn database.

Parameter	Value interval
m	[20,200]
t_x	[0.006,0.01]
t_y	[0.006,0.02]
α	[-25.778, -0.476]
n	[2,6]
r_x	[0.029,0.789]
r_y	[0.042,0.830]
α_{ply}	[0.639,31.655]
R_{ply}	[0.053,1.486]
j	[0,0.3]
j_{xy}	[0,0.03]
g	[30,300]
p	[0.35,0.65]
β	[0.050,1.571]
l_{hair}	[0.222,14.5]
s	[0,1]
l_{loop}	[0.407,34.627]
d_{mean}	[0.394,30.469]
d_{std}	[0.007,5]

ply and the auxiliary variable γ_{ply} of the helix angle of the ply twist in the yarn as follows (both angles are represented in radians):

$$\gamma_{ply} = [50, 80] \cdot \frac{\pi}{180} \tag{5.24}$$

$$\gamma = [50, 80] \cdot \frac{\pi}{180} \tag{5.25}$$

Then, following the helix formula, we compute the parameters for the pitch α of the ply helix and the pitch α_{ply} and radius R_{ply} of the yarn helix:

$$R_{ply} = \sqrt{\frac{n \cdot r_{frac}(\frac{r_x}{sin(\gamma_{ply})})^2}{area_{frac}^{yarn}}} - r_{frac}\frac{r_x}{sin(\gamma_{ply})} \tag{5.26}$$

$$\alpha_{ply} = 2\pi R_{ply} \cdot tan(\gamma_{ply}) \tag{5.27}$$

$$\alpha = -1 \cdot 2\pi r_x \cdot tan(\gamma) \tag{5.28}$$

Note the opposite signs for the clockwise and counterclockwise directions of the twist of the ply and yarn. This is not true for all existing yarns, but it is true for all knitting yarns that we have observed. If necessary, yarns with other combinations of clockwise and counterclockwise twist can be added to the database.

The overall intervals of all learnable yarn parameters are shown in Table 5.B.1.

Part III

Conclusion

Chapter 6

Conclusion

In this chapter, we summarize the contributions of the publications included in this thesis, discuss the limitations and give an overview of potential future developments.

6.1 Contributions and Impact

In the scope of this thesis, we presented three research projects that were directed towards the goal of visual prototyping of knitted cloth. The first project (Chapter 3) aimed to improve compression rates of a state-of-the-art encoder-decoder-based compression method for bidirectional texture functions (BTFs). BTFs are known to accurately model the appearance of fabrics, but the accuracy comes at the cost of high storage consumption. Our key contributions to this project are the determination of task-dependent latent space dimensionality in encoder-decoder architectures and the establishment of a connection between Shapley values [Shapley, 1953] and principal component analysis. The second and the third project (Chapters 4 and 5 respectively) focused on the editability and controlled image-based reconstruction of all three scales of knitwear: fibers, yarns and patterns. We contributed to these projects with the pipelines for inverse procedural modeling of yarns and knitwear and the extension of the Best Buddies similarity measure [Dekel et al., 2015] for template matching.

Task-dependent Latent Space Dimensionality in Encoder-Decoder Even though BTFs present a well-established technique for appearance modeling of complex materials, such as fabrics, their high memory requirements pose a problem for the practicability of this approach, thus motivating the research on compression methods. While the current state-of-the-art encoder-decoder-based approach significantly outperformed the previously widely used PCA-based compression technique, it did not exploit the full potential of encoder-decoder schemes. Since the dimensionality of the latent variables is directly connected to the compression rate of the method, it is essential to determine which number of latent variables is most convenient for each particular application. For this purpose, we developed an algorithm [Trunz et al., 2022] for efficient task-dependent analysis of the latent space

using Shapley values. It enables the user to decide on the suitable latent dimensionality, thus contributing to a more compact representation of data.

Connection Between Shapley Values and Principal Component Analysis (PCA) In order to motivate the usage of Shapley values for the latent space analysis, we found a direct connection between Shapley values and singular values involved in the principal component analysis [Trunz et al., 2022]. During the PCA, the principal components are ordered according to their singular values. We proved that if for model A, the input data x and the output y of A the following linear relation $Ax = y$ holds and a singular value decomposition, and therefore a PCA can be applied on A, the ordering of the singular values is the same as the ordering of the corresponding eigenvectors according to their Shapley values. As a direct conclusion of our theorem in combination with the Eckard-Young-Mirsky theorem [Eckart and Young, 1936] that states how to calculate the optimal low-rank approximation of A, we follow that using the first k elements in consistency with the ordering based on Shapley values the optimal rank-k approximation of a linear model A can be determined.

Inverse procedural modeling of Yarns BTFs belong to image-based appearance representation techniques and do not allow for easy editing operations on different scales of geometric features. A very convenient way to support such editing is to use procedural models for data generation. The challenge of data generation based on procedural models is determining which parameters yield the desired output. A user-friendly solution to control the output is to allow the user to specify the appearance of the output by means of an image while the algorithm automatically finds all parameter values required to generate a similar output. This technique is known as inverse procedural modeling. In the context of inverse procedural modeling of yarns and fibers [Trunz et al., 2023], we first extended an existing procedural yarn generator [Zhao et al., 2016] resulting in the synthesis of very realistic yarns. Using this enhanced generator, we created, to the best of our knowledge, the first annotated database of synthetic but natural-looking yarns. This database was the foundation of our newly developed neural approach for the automatic inference of parameters from images of real yarns. We demonstrated how the yarns generated from images with this approach could be easily edited and used to synthesize larger knitwear patches.

Inverse Procedural Modeling of Knitwear Given the yarn parameters, we can generate and visualize knitted cloth. However, to match the appearance of a particular knitted garment, apart from the yarn parameters, one requires the same knitting instruction used to produce the garment. In our project for inverse procedural modeling of knitwear [Trunz et al., 2019], we developed a novel approach for the induction of knitting instructions from an image of a knitwear piece, focusing on two fundamental stitch types: knit and purl. Our method is independent of any labeled database and can be easily utilized even by an inexperienced user. The algorithm includes a four-step pipeline, each step solving a sub-problem of the overall challenge. The first step utilizes a template-matching technique targeting per-pixel identification of the stitch types present in the image. After the coarse stitch identification,

the next step is directed toward the refinement of the correct positions of all included stitches and establishing their adjacent relations. Since a valid knitting pattern made of knits and purls is always a regular grid of stitches, we need to ensure that established adjacencies also result in a regular grid. To approach this sub-problem of finding the correct regular grid of stitches, we first determined the correct numbers of rows and columns in the grid and then formulated the task as an integer linear programming (ILP) problem that could be solved with an existing ILP software. Utilizing the key insight that knitted garments are produced by repeating the instructions many times, and therefore the actual pattern commonly appears in the image several times, we subsequently implemented a pattern size detection step, which at the same time corrects the possible template matching errors.

Extended Best Buddies Similarity Measure for Template Matching In the course of the project on inverse procedural modeling of knitwear [Trunz et al., 2019], we tackled the problem of coarse identification of stitch types in an image of knitted cloth. Neither of the existing template-matching approaches we tried produced satisfactory results for our particular task. We introduced a gradient penalty to a state-of-the-art similarity measure called Best Buddies Similarity [Dekel et al., 2015]. Applying template matching with our improved version of this similarity measure led to very good results on our application of identifying and localizing stitches in images of knitted textiles and has outperformed many other state-of-the-art approaches.

6.2 Limitations and Future Work

While many challenges of visual prototyping of knitted cloth have been tackled by the approaches developed in the course of this thesis, there are still some problems that require future research. In the following, we outline some potential directions for future work and discuss further open questions.

BTF Editing Through Latent Space Modification During our first project (Chapter 3) in this thesis, we addressed one of the main problems of the BTFs, their high storage requirement. We enhanced a neural model approach to make BTF representation even more compact. However, the editability of the BTFs still remains an open question. In this neural model, materials are represented through the latent space of an encoder-decoder model, and at this point, there is no direct mapping between latent variables and quantities that have an intuitive visual explanation, like albedo, roughness or glossiness. Recent advances in the field of deep neural networks suggest further development in this area, while the open problem of explainable AI is being extensively researched. Therefore, a potential direction of research would be to link the latent space, possibly in an iterative manner, to some intuitive parameters for material representation, thus enabling easy editing support and taking a further step towards explainable encoder-decoder models.

Inverse Procedural Modeling of Knitwear There are several different aspects that can be improved in our pipeline for inverse procedural modeling of knitwear (Chapter 4). During our research, we focused on two fundamental stitch types: the knit stitch and the purl stitch. Even though there is a vast amount of possible knitting patterns that can be produced with only these two stitch types, it would be interesting to be able to infer knitting instructions that contain other stitch types, such as holes. These knitting instructions still have a regular grid-like structure, but the corresponding knitted garment is usually deformed in a way that such a grid is not visually recognizable. Therefore, additional constraints or rules could be added to our optimization approach. Another research direction could be making the approach completely automatic, without any user interaction. Both of these open problems could be addressed with a deep neural approach. For example, one could create a labeled database of synthetically generated and real-world stitches or stitch pairs made of different yarns under different environmental conditions and pose the problem as classification, segmentation or localization. Subsequently, an optimization could be performed to infer the actual grid of instructions. A further possibility could be to directly target the grid localization utilizing an image database of knitted patches depicting various patterns. Parallel to our work, a neural inverse knitting approach [Kaspar et al., 2019] was presented. This approach was unsuitable for hand-made knitted textiles and the input patches had to have the same pattern size as the patches in the images on which the network was trained. However, it presented a possibility for fast rendering of a database of synthetic knitted patches. Utilizing this idea and extending it to produce more detailed and physically correctly simulated knitted patches could lead to a database that would be sufficient for fully automatic pattern extraction from images.

Inverse Procedural Modeling of Yarns In the course of our project on inverse procedural modeling of yarns (Chapter 5), we inferred geometrical parameters of real-world yarns for our procedural yarn model. A potential future work direction would be to extend the parameter prediction to estimate the reflectance parameters of yarns and fibers from the yarn input images. In our experiments, this task has proven to be particularly challenging and could not be solved in the same manner as inferring the geometry parameters, namely by parameter regression with an L1 loss formulation. One could introduce some additional rendering loss that would express visual similarity since an L1 loss over parameters is not a suitable choice for visual image comparison. Another possibility would be to create a large BTF database of knitted materials. Then, one can either apply a differentiable rendering approach for material appearance estimation or utilize this database to train a network model in the unified neural BTF approach of Rainer et al. [2020] and try reconstructing unseen materials.

Similar to previous works, the generation of our database for natural-looking synthetic yarns [Trunz et al., 2023] did not include physically-based simulation. While at the scale of knitwear patterns, physically-based simulation has been successfully applied [Kaldor et al., 2008; Yuksel et al., 2012; Leaf et al., 2018; Sperl et al., 2020], there are currently no yarn generators that support this feature. The reason could be that such simulations often require numerical integration procedures that can be costly in terms of runtime requirements. However, since physically-based simulation considerably increased the level of realism for

knitting patterns, it might be worth including the simulation in the scale of yarns and plys or even fibers as well.

Moreover, since our current yarn database does not contain yarns of the fourth level, i.e., when thinner yarns are regarded as plys and twisted into thicker yarns, we did not include this yarn type in our yarn parameter inference approach. However, our yarn generator supports this level of the hierarchy, so the straightforward extension would be generating additional yarns for the database and training new models to support the detection of the parameters of this new level of hierarchy.

Bibliography

Aas, Kjersti, Martin Jullum, and Anders Løland (2019). "Explaining individual predictions when features are dependent: More accurate approximations to Shapley values." *arXiv preprint arXiv:1903.10464*.

Adebayo, Julius, Justin Gilmer, Michael Muelly, Ian Goodfellow, Moritz Hardt, and Been Kim (2018). "Sanity Checks for Saliency Maps." *Advances in Neural Information Processing Systems*.

Agustsson, Eirikur, Fabian Mentzer, Michael Tschannen, Lukas Cavigelli, Radu Timofte, Luca Benini, and Luc Van Gool (2017). "Soft-to-hard vector quantization for end-to-end learning compressible representations." *arXiv preprint arXiv:1704.00648*.

Agustsson, Eirikur, Michael Tschannen, Fabian Mentzer, Radu Timofte, and Luc Van Gool (2019). "Generative adversarial networks for extreme learned image compression." *Proceedings of the IEEE/CVF International Conference on Computer Vision*.

Ali, Muhammad A., Qiangshun Guan, Rehan Umer, Wesley J. Cantwell, and Tiejun Zhang (2022). "Efficient processing of CT images using deep learning tools for generating digital material twins of woven fabrics." *Composites Science and Technology*. ISSN: 0266-3538. URL: `https://www.sciencedirect.com/science/article/pii/S0266353821004474`.

Aliaga, Carlos, Carlos Castillo, Diego Gutierrez, Miguel A Otaduy, Jorge Lopez-Moreno, and Adrian Jarabo (2017). "An appearance model for textile fibers." *Computer Graphics Forum*.

Aliaga, D. G., İ. Demir, B. Benes, and M. Wand (2016). "Inverse Procedural Modeling of 3D Models for Virtual Worlds." *ACM SIGGRAPH 2016 Courses*.

Alkhayrat, Maha, Mohamad Aljnidi, and Kadan Aljoumaa (2020). "A comparative dimensionality reduction study in telecom customer segmentation using deep learning and PCA." *Journal of Big Data*.

Amor, Nesrine, Muhammad Tayyab Noman, and Michal Petru (2021). "Classification of Textile Polymer Composites: Recent Trends and Challenges." *Polymers*.

Ancona, Marco, Cengiz Oztireli, and Markus Gross (2019). "Explaining deep neural networks with a polynomial time algorithm for shapley value approximation." *International Conference on Machine Learning*.

Bach, Sebastian, Alexander Binder, Grégoire Montavon, Frederick Klauschen, Klaus-Robert Müller, and Wojciech Samek (2015). "On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation." *PloS one*.

Ballé, Johannes, Valero Laparra, and Eero P Simoncelli (2016a). "Density modeling of images using a generalized normalization transformation." *4th International Conference on Learning Representations, ICLR 2016*.

Ballé, Johannes, Valero Laparra, and Eero P Simoncelli (2016b). "End-to-end optimized image compression." *arXiv preprint arXiv:1611.01704*.

Ballé, Johannes, Valero Laparra, and Eero P. Simoncelli (2017). "End-to-end Optimized Image Compression." *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*.

Ballé, Johannes, David Minnen, Saurabh Singh, Sung Jin Hwang, and Nick Johnston (2018). "Variational image compression with a scale hyperprior." *arXiv preprint arXiv:1802.01436*.

Barnes, C., E. Shechtman, A. Finkelstein, and D. B. Goldman (2009). "PatchMatch: A Randomized Correspondence Algorithm for Structural Image Editing." *ACM Transactions on Graphics (Proc. SIGGRAPH)*.

Barnes, C., E. Shechtman, D. B. Goldman, and A. Finkelstein (2010). "The Generalized PatchMatch Correspondence Algorithm." *Proceedings of the 11th European Conference on Computer Vision Conference on Computer Vision: Part III*. URL: http://dl.acm.org/citation.cfm?id=1927006.1927010.

Barnes, C., F.-L. Zhang, L. Lou, X. Wu, and S.-M. Hu (2015). "PatchTable: Efficient Patch Queries for Large Datasets and Applications." *ACM Trans. Graph.* ISSN: 0730-0301. URL: http://doi.acm.org/10.1145/2766934.

Bartholomew, David J, Martin Knott, and Irini Moustaki (2011). *Latent variable models and factor analysis: A unified approach*.

Biecek, Przemysław (2018). "DALEX: explainers for complex predictive models in R." *The Journal of Machine Learning Research*.

Bokeloh, M., M. Wand, and H.-P. Seidel (2010). "A Connection Between Partial Symmetry and Inverse Procedural Modeling." *ACM Trans. Graph.* ISSN: 0730-0301. URL: http://doi.acm.org/10.1145/1778765.1778841.

Bouman, Katherine L, Bei Xiao, Peter Battaglia, and William T Freeman (2013). "Estimating the material properties of fabric from video." *Proceedings of the IEEE international conference on computer vision*.

Bowen, Dillon and Lyle Ungar (2020). "Generalized SHAP: Generating multiple types of explanations in machine learning." *arXiv preprint arXiv:2006.07155*.

Bradley, S. (2014). *Design Principles: Visual Perception And The Principles Of Gestalt*. URL: https://www.smashingmagazine.com/2014/03/design-principles-visual-perception-and-the-principles-of-gestalt/.

Cai, Chunlei, Li Chen, Xiaoyun Zhang, and Zhiyong Gao (2018). "Efficient variable rate image compression with multi-scale decomposition network." *IEEE Transactions on Circuits and Systems for Video Technology*.

Cai, Chunlei, Li Chen, Xiaoyun Zhang, Guo Lu, and Zhiyong Gao (2019). "A novel deep progressive image compression framework." *2019 Picture Coding Symposium (PCS)*.

Caputo, Barbara, Eric Hayman, and P Mallikarjuna (2005). "Class-specific material categorisation." *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*.

Carroll, J Douglas and Jih-Jie Chang (1970). "Analysis of individual differences in multidimensional scaling via an N-way generalization of "Eckart-Young" decomposition." *Psychometrika*.

Castillo, Carlos, Carlos Aliaga, and Jorge López-Moreno (2017). "Challenges in appearance capture and predictive modeling of textile materials." *Proceedings of the Workshop on Material Appearance Modeling*.

Castillo, Carlos, Jorge López-Moreno, and Carlos Aliaga (2019). "Recent advances in fabric appearance reproduction." *Computers & Graphics*.

Castro, Javier, Daniel Gómez, and Juan Tejada (2009). "Polynomial calculation of the Shapley value based on sampling." *Computers & Operations Research*.

Chen, Hugh, Scott Lundberg, and Su-In Lee (2021). "Explaining models by propagating Shapley values of local components." *Explainable AI in Healthcare and Medicine*.

Chen, J.-H., C.-S. Chen, and Y.-S. Chen (2003). "Fast algorithm for robust template matching with M-estimators." *IEEE Transactions on Signal Processing*. ISSN: 1053-587X.

Chen, Jianbo, Le Song, Martin J. Wainwright, and Michael I. Jordan (2019). "L-Shapley and C-Shapley: Efficient Model Interpretation for Structured Data." *International Conference on Learning Representations*.

Choi, Yoojin, Mostafa El-Khamy, and Jungwon Lee (2019). "Variable rate deep image compression with a conditional autoencoder." *Proceedings of the IEEE/CVF International Conference on Computer Vision*.

Cirio, Gabriel, Jorge Lopez-Moreno, David Miraut, and Miguel A Otaduy (2014). "Yarn-level simulation of woven cloth." *ACM Transactions on Graphics (TOG)*.

Covert, Ian, Scott Lundberg, and Su-In Lee (2020a). "Explaining by Removing: A Unified Framework for Model Explanation." *arXiv preprint arXiv:2011.14878*.

Covert, Ian, Scott Lundberg, and Su-In Lee (2020b). "Understanding global feature contributions with additive importance measures." *Advances in Neural Information Processing Systems*.

Dai, Bin, Yu Wang, John Aston, Gang Hua, and David Wipf (2018). "Connections with robust PCA and the role of emergent sparsity in variational autoencoder models." *The Journal of Machine Learning Research*.

Dai, Bin, Yu Wang, John Aston, Gang Hua, and David Wipf (2019). "Hidden talents of the variational autoencoder." *arXiv preprint arXiv:1706.05148v5*.

Dana, Kristin J, Shree K Nayar, Bram Van Ginneken, and Jan J Koenderink (1997). "Reflectance and texture of real-world surfaces authors." *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*.

Dana, Kristin J., Bram van Ginneken, Shree K. Nayar, and Jan J. Koenderink (1999). "Reflectance and Texture of Real-World Surfaces." *ACM Trans. Graph.* ISSN: 0730-0301. URL: https://doi.org/10.1145/300776.300778.

Datta, Anupam, Shayak Sen, and Yair Zick (2016). "Algorithmic transparency via quantitative input influence: Theory and experiments with learning systems." *2016 IEEE symposium on security and privacy (SP)*.

De Lathauwer, Lieven, Bart De Moor, and Joos Vandewalle (2000a). "A multilinear singular value decomposition." *SIAM journal on Matrix Analysis and Applications*.

De Lathauwer, Lieven, Bart De Moor, and Joos Vandewalle (2000b). "On the best rank-1 and rank-(r 1, r 2,..., rn) approximation of higher-order tensors." *SIAM journal on Matrix Analysis and Applications*.

Dekel, T., S. Oron, M. Rubinstein, S. Avidan, and W. T. Freeman (2015). "Best-Buddies Similarity for robust template matching." *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Demir, I., D. G. Aliaga, and B. Benes (2015). "Procedural editing of 3D building point clouds." *Proceedings of the IEEE International Conference on Computer Vision*.

Demir, I., D. G. Aliaga, and B. Benes (2016). "Proceduralization for Editing 3D Architectural Models." *2016 Fourth International Conference on 3D Vision (3DV)*.

Ding, Zheng, Yifan Xu, Weijian Xu, Gaurav Parmar, Yang Yang, Max Welling, and Zhuowen Tu (2020). "Guided variational autoencoder for disentanglement learning." *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Dobashi, Yoshinori, Kei Iwasaki, Makoto Okabe, Takashi Ijiri, and Hideki Todo (2019). "Inverse appearance modeling of interwoven cloth." *The Visual Computer*.

Dong, Yue, Jiaping Wang, Xin Tong, John Snyder, Yanxiang Lan, Moshe Ben-Ezra, and Baining Guo (2010). "Manifold bootstrapping for SVBRDF capture." *ACM Transactions on Graphics (TOG)*.

Doshi-Velez, Finale and Been Kim (2017). "Towards a rigorous science of interpretable machine learning." *arXiv preprint arXiv:1702.08608*.

Drago, Frédéric and Norishige Chiba (2004). "Painting canvas synthesis." *The Visual Computer*.

Dubuisson, M. P. and A. K. Jain (1994). "A modified Hausdorff distance for object matching." *Proceedings of 12th International Conference on Pattern Recognition*.

Eckart, Carl and Gale Young (1936). "The approximation of one matrix by another of lower rank." *Psychometrika*.

Elboher, E. and M. Werman (2013). "Asymmetric Correlation: A Noise Robust Similarity Measure for Template Matching." *IEEE Transactions on Image Processing*. ISSN: 1057-7149.

F.R.S., Karl Pearson (1901). "LIII. On lines and planes of closest fit to systems of points in space." *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*. eprint: https://doi.org/10.1080/14786440109462720. URL: https://doi.org/10.1080/14786440109462720.

Fatima, Shaheen S, Michael Wooldridge, and Nicholas R Jennings (2008). "A linear approximation method for the Shapley value." *Artificial Intelligence*.

Fedkiw, R., J. Stam, and H. W. Jensen (2001). "Visual Simulation of Smoke." *Proceedings of SIGGRAPH 2001*. Edited by Eugene Fiume.

Filip, J., M. Kolafová, M. Havlıček, R. Vávra, M. Haindl, and Rushmeier H. (2018). "Evaluating Physical and Rendered Material Appearance." *The Visual Computer (Computer Graphics International 2018)*.

Fong, Ruth C. and Andrea Vedaldi (2017). "Interpretable Explanations of Black Boxes by Meaningful Perturbation." *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*. URL: https://doi.org/10.1109/ICCV.2017.371.

Fournier, Alain (1992). "Normal distribution functions and multiple surfaces." *Graphics Interface '92 Workshop on Local Illumination*.

Fryer, Daniel, Inga Strümke, and Hien Nguyen (2021). "Shapley values for feature selection: The good, the bad, and the axioms." *arXiv preprint arXiv:2102.10936*.

Furukawa, R., H. Kawasaki, K. Ikeuchi, and M. Sakauchi (2002). "Appearance based object modeling using texture database: Acquisition, compression and rendering." *Eurographics Workshop on Rendering*. Edited by P. Debevec and S. Gibson.

Gast, Jochen and Stefan Roth (2018). "Lightweight probabilistic deep networks." *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Ghorbani, Amirata, Abubakar Abid, and James Zou (2019). "Interpretation of neural networks is fragile." *Proceedings of the AAAI Conference on Artificial Intelligence*.

Ghorbani, Amirata and James Zou (2019). "Data shapley: Equitable valuation of data for machine learning." *International Conference on Machine Learning*.

Ghorbani, Amirata and James Zou (2020). "Neuron shapley: Discovering the responsible neurons." *arXiv preprint arXiv:2002.09815*.

Giraud, R., V.-T. Ta, N. Papadakis, J. V. Manjn, D. L. Collins, and P. Coup (2016). "An Optimized PatchMatch for multi-scale and multi-feature label fusion." *NeuroImage*. ISSN: 1053-8119. URL: http://www.sciencedirect.com/science/article/pii/S1053811915006965.

Giudici, Paolo and Emanuela Raffinetti (2020). "Shapley-Lorenz eXplainable artificial intelligence." *Expert Systems with Applications*.

Gong, Deshan, Zhanxing Zhu, Andrew J Bulpitt, and He Wang (2022). "Fine-grained differentiable physics: a yarn-level model for fabrics." *arXiv preprint arXiv:2202.00504*.

Goodfellow, Ian, Yoshua Bengio, and Aaron Courville (2016). *Deep Learning*. http://www.deeplearningbook.org.

Guan, Yue and Jennifer Dy (2009). "Sparse probabilistic principal component analysis." *Artificial Intelligence and Statistics*.

Guarnera, Giuseppe Claudio, Peter Hall, Alain Chesnais, and Mashhuda Glencross (2017). "Woven fabric model creation from a single image." *ACM Transactions on Graphics (TOG)*.

Guerrero, Paul, Miloš Hašan, Kalyan Sunkavalli, Radomır Měch, Tamy Boubekeur, and Niloy J. Mitra (2022). "MatFormer: A Generative Model for Procedural Materials." ISSN: 0730-0301. URL: https://doi.org/10.1145/3528223.3530173.

Gurobi Optimization, Inc. (2016). *Gurobi Optimizer Reference Manual*. URL: http://www.gurobi.com.

Haindl, Michal and Jiří Filip (2013). *Visual Texture: Accurate Material Appearance Measurement, Representation and Modeling*.

Harshman, Richard A and Margaret E Lundy (1994). "PARAFAC: Parallel factor analysis." *Computational Statistics & Data Analysis*.

Hartmann, Stefan, Elena Trunz, Björn Krüger, Reinhard Klein, and Matthias B. Hullin (2015). "Efficient Multi-Constrained Optimization for Example-Based Synthesis." *The Visual Computer / Proc. Computer Graphics International (CGI 2015)*. URL: %7Bhttp://dx.doi.org/10.1007/s00371-015-1114-y%7D.

Hayman, Eric, Barbara Caputo, Mario Fritz, and Jan-Olof Eklundh (2004). "On the significance of real-world conditions for material classification." *European conference on computer vision*.

He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun (2015). "Deep Residual Learning for Image Recognition." *CoRR*. arXiv: 1512.03385. URL: http://arxiv.org/abs/1512.03385.

Hel-Or, Y., H. Hel-Or, and E. David (2014). "Matching by Tone Mapping: Photometric Invariant Template Matching." *IEEE Transactions on Pattern Analysis and Machine Intelligence*. ISSN: 0162-8828.

Hotelling, Harold (1936). "Relations Between Two Sets of Variates." *Biometrika*. ISSN: 00063444. URL: http://www.jstor.org/stable/2333955.

Howard, Andrew G, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam (2017). "Mobilenets: Efficient convolutional neural networks for mobile vision applications." *arXiv preprint arXiv:1704.04861*.

Huang, Q., L. J. Guibas, and N. J. Mitra (2014). "Near-Regular Structure Discovery Using Linear Programming." *ACM Trans. Graph.* ISSN: 0730-0301. URL: http://doi.acm.org/10.1145/2535596.

Hussain Iqbal, Muhammad Ather, Babar Khan, Zhijie Wang, and Shenyi Ding (2020). "Woven Fabric Pattern Recognition and Classification Based on Deep Convolutional Neural Networks." *Electronics*. ISSN: 2079-9292. URL: https://www.mdpi.com/2079-9292/9/6/1048.

Huttenlocher, D. P., G. A. Klanderman, and W. J. Rucklidge (1993). "Comparing images using the Hausdorff distance." *IEEE Transactions on Pattern Analysis and Machine Intelligence*. ISSN: 0162-8828.

Irawan, Piti and Steve Marschner (2012). "Specular reflection from woven cloth." *ACM Transactions on Graphics (TOG)*.

Izzo, Cosimo, Aldo Lipani, Ramin Okhrati, and Francesca Medda (2020). "A Baseline for Shapely Values in MLPs: from Missingness to Neutrality." *arXiv preprint arXiv:2006.04896*.

Jacob, Benoit, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko (2018). "Quantization and training of neural networks for efficient integer-arithmetic-only inference." *Proceedings of the IEEE conference on computer vision and pattern recognition*.

Jakob, W., A. Arbree, J. T. Moon, K. Bala, and S. Marschner (2010). "A radiative transfer framework for rendering materials with anisotropic structure." *ACM Transactions on Graphics (TOG)*.

Jia, Ruoxi, David Dao, Boxin Wang, Frances Ann Hubis, Nick Hynes, Nezihe Merve Gürel, Bo Li, Ce Zhang, Dawn Song, and Costas J Spanos (2019). "Towards efficient data valuation based on the shapley value." *The 22nd International Conference on Artificial Intelligence and Statistics*.

Jin, Wenhua, Beibei Wang, Milos Hasan, Yu Guo, Steve Marschner, and Ling-Qi Yan (2022). "Woven Fabric Capture from a Single Photo." *SIGGRAPH Asia 2022 Conference Papers*.

Jing, Li, Jure Zbontar, and Yann LeCun (2020). "Implicit Rank-Minimizing Autoencoder." *arXiv preprint arXiv:2010.00679*.

Jobson, D. J., Z. Rahman, and G. A. Woodell (1995). "Retinex image processing: Improved fidelity to direct visual observation." *Proceedings of the IS&T Fourth Color Imaging Conference: Color Science, Systems, and Applications*.

Johnston, Nick, Elad Eban, Ariel Gordon, and Johannes Ballé (2019). "Computationally efficient neural image compression." *arXiv preprint arXiv:1912.08771*.

Jolliffe, Ian T and Jorge Cadima (2016). "Principal component analysis: a review and recent developments." *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*.

Kaldor, Jonathan M., Doug L. James, and Steve Marschner (2008). "Simulating Knitted Cloth at the Yarn Level." *ACM SIGGRAPH 2008 Papers*. URL: https://doi.org/10.1145/1399504.1360664.

Kartch, D. (2000). "Efficient Rendering and Compression for Full-Parallax Computer-Generated Holographic Stereograms." *Ph.D. thesis*. *Cornell University*.

Kaspar, Alexandre, Tae-Hyun Oh, Liane Makatura, Petr Kellnhofer, and Wojciech Matusik (2019). "Neural Inverse Knitting: From Images to Manufacturing Instructions." *Proceedings of the 36th International Conference on Machine Learning*. Edited by Kamalika Chaudhuri and Ruslan Salakhutdinov. URL: http://proceedings.mlr.press/v97/kaspar19a.html.

Keefe, M (1994). "Solid modeling applied to fibrous assemblies. Part I: Twisted yarns." *The Journal of the Textile Institute*.

Khan, AH and EL Hines (1994). "Integer-weight neural nets." *Electronics Letters*.

Khungurn, Pramook, Daniel Schroeder, Shuang Zhao, Kavita Bala, and Steve Marschner (2015). "Matching Real Fabrics with Micro-Appearance Models." *ACM Trans. Graph.*

Kim, H. Y. and S. A. de Araújo (2007). "Grayscale Template-Matching Invariant to Rotation, Scale, Translation, Brightness and Contrast." *Advances in Image and Video Technology: Second Pacific Rim Symposium, PSIVT 2007 Santiago, Chile, December 17-19, 2007 Proceedings*. Edited by Domingo Mery and Luis Rueda. URL: https://doi.org/10.1007/978-3-540-77129-6_13.

Kim, Yong Hwi, Junho Choi, and Kwan H. Lee (2018). "An Efficient Method for Specular-Enhanced BTF Compression." *Comput. Graph.* ISSN: 0097-8493. URL: https://doi.org/10.1016/j.cag.2018.06.001.

Kindermans, Pieter-Jan, Sara Hooker, Julius Adebayo, Maximilian Alber, Kristof T. Schütt, Sven Dähne, Dumitru Erhan, and Been Kim (2019). "The (Un)reliability of Saliency Methods." *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*.

Kingma, Diederik P and Max Welling (2013). "Auto-encoding variational bayes." *arXiv preprint arXiv:1312.6114*.

Kingma, Diederik P. and Jimmy Ba (2015). "Adam: A Method for Stochastic Optimization." *Proceedings of 3rd International Conference on Learning Representations (ICLR)*. Edited by Yoshua Bengio and Yann LeCun.

Kolda, Tamara G and Brett W Bader (2009). "Tensor decompositions and applications." *SIAM review*.

Korman, S., D. Reichman, G. Tsur, and S. Avidan (2013). "FasT-Match: Fast Affine Template Matching." *2013 IEEE Conference on Computer Vision and Pattern Recognition*.

Koudelka, Melissa, Peter Belhumeur, and David Kriegman (2003). "Acquisition, Compression, and Synthesis of Bidirectional Texture Functions." *Texture 2003*.

Kumar, I Elizabeth, Carlos Scheidegger, Suresh Venkatasubramanian, and S Friedler (2020a). "Shapley Residuals: Quantifying the limits of the Shapley value for explanations." *ICML Workshop on Workshop on Human Interpretability in Machine Learning (WHI)*.

Kumar, I Elizabeth, Suresh Venkatasubramanian, Carlos Scheidegger, and Sorelle Friedler (2020b). "Problems with Shapley-value-based explanations as feature importance measures." *International Conference on Machine Learning*.

Ladjal, Saıd, Alasdair Newson, and Chi-Hieu Pham (2019). "A PCA-like autoencoder." *arXiv preprint arXiv:1904.01277*.

Landis, H. (2002). *Production-Ready Global Illumination*. ACM SIGGRAPH 2002 Course #16 Notes.

Leaf, Jonathan, Rundong Wu, Eston Schweickart, Doug L. James, and Steve Marschner (2018). "Interactive Design of Yarn-Level Cloth Patterns." *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia 2018)*.

Lee, Jooyoung, Seunghyun Cho, and Seung-Kwon Beack (2018). "Context-adaptive entropy model for end-to-end optimized image compression." *arXiv preprint arXiv:1809.10452*.

Levoy, M., K. Pulli, B. Curless, S. Rusinkiewicz, D. Koller, L. Pereira, M. Ginzton, S. Anderson, J. Davis, J. Ginsberg, J. Shade, and D. Fulk (2000). "The Digital Michelangelo Project." *Proceedings of SIGGRAPH 2000*. Edited by Kurt Akeley.

Li, C. and M. Wand (2015). "Approximate Translational Building Blocks for Image Decomposition and Synthesis." *ACM Trans. Graph.* ISSN: 0730-0301. URL: `http://doi.acm.org/10.1145/2757287`.

Li, Mu, Kede Ma, Jane You, David Zhang, and Wangmeng Zuo (2020). "Efficient and effective context-based convolutional entropy modeling for image compression." *IEEE Transactions on Image Processing*.

Li, Y., Y. Hu, R. Song, P. Rao, and Y. Wang (2017). "Coarse-to-fine PatchMatch for Dense Correspondence." *IEEE Transactions on Circuits and Systems for Video Technology*. ISSN: 1051-8215.

Li, Yifei, Tao Du, Kui Wu, Jie Xu, and Wojciech Matusik (2022). "DiffCloth: Differentiable cloth simulation with dry frictional contact." *ACM Transactions on Graphics (TOG)*.

Liang, Junbang, Ming Lin, and Vladlen Koltun (2019). "Differentiable cloth simulation for inverse problems." *Advances in Neural Information Processing Systems*.

Lienhard, S., C. Lau, P. Müller, P. Wonka, and M. Pauly (2017). "Design Transformations for Rule-based Procedural Modeling." *Comput. Graph. Forum*. ISSN: 0167-7055. URL: https://doi.org/10.1111/cgf.13105.

Lipton, Zachary C (2016). "The Mythos of Model Interpretability." *ICML Workshop on Human Interpretability in MachineLearning (WHI)*.

Liu, C., L. Sharan, E. H. Adelson, and R. Rosenholtz (2010). "Exploring features in a Bayesian framework for material recognition." *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*.

Liu, J. and Y. Liu (2014). "Local Regularity-Driven City-Scale Facade Detection from Aerial Images." *2014 IEEE Conference on Computer Vision and Pattern Recognition*.

Liu, S., T. T. Ng, K. Sunkavalli, M. N. Do, E. Shechtman, and N. Carr (2015). "PatchMatch-Based Automatic Lattice Detection for Near-Regular Textures." *2015 IEEE International Conference on Computer Vision (ICCV)*.

Liu, Xinguo, Yaohua Hu, Jingdan Zhang, Xin Tong, Baining Guo, and Heung-Yeung Shum (2004). "Synthesis and Rendering of Bidirectional Texture Functions on Arbitrary Surfaces." *IEEE Transactions on Visualization and Computer Graphics*. ISSN: 1077-2626. URL: https://doi.org/10.1109/TVCG.2004.1272727.

Liu, Zhuang, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie (2022). "A ConvNet for the 2020s." *CoRR*. arXiv: 2201.03545. URL: https://arxiv.org/abs/2201.03545.

Longay, Steven, Adam Runions, Frédéric Boudon, and and Prusinkiewicz (2012). "TreeSketch: Interactive Procedural Modeling of Trees on a Tablet."

Lou, Yin, Rich Caruana, and Johannes Gehrke (2012). "Intelligible models for classification and regression." *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*.

Lou, Yin, Rich Caruana, Johannes Gehrke, and Giles Hooker (2013). "Accurate intelligible models with pairwise interactions." *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*.

Lu, J., Y. Li, H. Yang, D. Min, W. Eng, and M. N. Do (2017). "PatchMatch Filter: Edge-Aware Filtering Meets Randomized Search for Visual Correspondence." *IEEE Transactions on Pattern Analysis and Machine Intelligence*. ISSN: 0162-8828.

Luan, Fujun, Shuang Zhao, and Kavita Bala (2017). "Fiber-Level On-the-Fly Procedural Textiles." *Computer Graphics Forum*. ISSN: 1467-8659.

Lukáč, M., D. Sýkora, K. Sunkavalli, E. Shechtman, O. Jamriška, N. Carr, and T. Pajdla (2017). "Nautilus: Recovering Regional Symmetry Transformations for Image Editing." *ACM Trans. Graph.* ISSN: 0730-0301. URL: http://doi.acm.org/10.1145/3072959.3073661.

Lundberg, Scott M, Gabriel Erion, Hugh Chen, Alex DeGrave, Jordan M Prutkin, Bala Nair, Ronit Katz, Jonathan Himmelfarb, Nisha Bansal, and Su-In Lee (2019). "Explainable AI for trees: From local explanations to global understanding." *arXiv preprint arXiv:1905.04610*.

Lundberg, Scott M and Su-In Lee (2017). "A Unified Approach to Interpreting Model Predictions." *Advances in Neural Information Processing Systems*.

Ma, Sisi and Roshan Tourani (2020). "Predictive and causal implications of using shapley value for model interpretation." *Proceedings of the 2020 KDD Workshop on Causal Discovery*.

Maleki, Sasan, Long Tran-Thanh, Greg Hines, Talal Rahwan, and Alex Rogers (2013). "Bounding the estimation error of sampling-based Shapley value approximation." *arXiv preprint arXiv:1306.4265*.

Mangalathu, Sujith, Seong-Hoon Hwang, and Jong-Su Jeon (2020). "Failure mode and effects analysis of RC members based on machine-learning-based SHapley Additive exPlanations (SHAP) approach." *Engineering Structures*.

Martin-Garrido, Alberto, Eder Miguel, and Miguel Angel Otaduy (2018). "Toward Estimation of Yarn-Level Cloth Simulation Models." *CEIG*.

Martinovic, A. and L. Van Gool (2013). "Bayesian Grammar Learning for Inverse Procedural Modeling." *Proceedings of the 2013 IEEE Conference on Computer Vision and Pattern Recognition*. URL: http://dx.doi.org/10.1109/CVPR.2013.33.

Matsui, Yasuko and Tomomi Matsui (2001). "NP-completeness for calculating power indices of weighted majority games." *Theoretical Computer Science*. Combinatorics and Computer Science. ISSN: 0304-3975. URL: https://www.sciencedirect.com/science/article/pii/S0304397500002516.

Mattei, Pierre-Alexandre and Jes Frellsen (2018). "Leveraging the exact likelihood of deep latent variable models." *arXiv preprint arXiv:1802.04826*.

Mazlov, Ilia, Sebastian Merzbach, Elena Trunz, and Reinhard Klein (2019). "Neural Appearance Synthesis and Transfer." *Workshop on Material Appearance Modeling*.

Mentzer, Fabian, Eirikur Agustsson, Michael Tschannen, Radu Timofte, and Luc Van Gool (2018). "Conditional probability models for deep image compression." *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Michalak, Tomasz P, Karthik V Aadithya, Piotr L Szczepanski, Balaraman Ravindran, and Nicholas R Jennings (2013). "Efficient computation of the Shapley value for game-theoretic network centrality." *Journal of Artificial Intelligence Research*.

Minnen, David, Johannes Ballé, and George Toderici (2018). "Joint autoregressive and hierarchical priors for learned image compression." *arXiv preprint arXiv:1809.02736*.

Minnen, David and Saurabh Singh (2020). "Channel-wise autoregressive entropy models for learned image compression." *2020 IEEE International Conference on Image Processing (ICIP)*.

Mohammadi, Seyed Omid and Ahmad Kalhor (2021). "Smart Fashion: A Review of AI Applications in the Fashion & Apparel Industry." *arXiv preprint arXiv:2111.00905*.

Montavon, Grégoire, Sebastian Lapuschkin, Alexander Binder, Wojciech Samek, and Klaus-Robert Müller (2017). "Explaining nonlinear classification decisions with deep taylor decomposition." *Pattern Recognition*.

Montazeri, Zahra, Søren Gammelmark, Shuang Zhao, and Henrik Wann Jensen (2021). *Systems and methods to compute the appearance of woven and knitted textiles at the ply-level*. US Patent 11,049,291.

Montazeri, Zahra, Søren B. Gammelmark, Shuang Zhao, and Henrik Wann Jensen (2020). "A Practical Ply-Based Appearance Model of Woven Fabrics." *ACM Trans. Graph.* ISSN: 0730-0301. URL: https://doi.org/10.1145/3414685.3417777.

Montazeri, Zahra, Chang Xiao, Yun Fei, Changxi Zheng, and Shuang Zhao (2019). "Mechanics-aware modeling of cloth appearance." *IEEE transactions on visualization and computer graphics*.

Morris, PJ, JH Merkin, and RW Rennell (1999). "Modelling of yarn properties from fibre properties." *Journal of the Textile Institute. Part 1, Fibre science and textile technology*.

Müller, Pascal, Peter Wonka, Simon Haegler, Andreas Ulmer, and Luc Van Gool (2006). "Procedural Modeling of Buildings." *ACM Trans. Graph.* ISSN: 0730-0301. URL: https://doi.org/10.1145/1141911.1141931.

Nakanishi, Ken M, Shin-ichi Maeda, Takeru Miyato, and Daisuke Okanohara (2018). "Neural multi-scale image compression." *Asian Conference on Computer Vision*.

Nicodemus, F E, J C Richmond, J J Hsia, I W Ginsberg, and T Limperis (1977). *Geometrical considerations and nomenclature for reflectance*. Technical report.

Nie, Weili, Yang Zhang, and Ankit Patel (2018). "A theoretical explanation for perplexing behaviors of backpropagation-based visualizations." *International Conference on Machine Learning*.

Nishida, Gen, Adrien Bousseau, and Daniel Aliaga (2018). "Procedural Modeling of a Building from a Single Image." *Computer Graphics Forum*.

Nohara, Yasunobu, Koutarou Matsumoto, Hidehisa Soejima, and Naoki Nakashima (2019). "Explanation of machine learning models using improved Shapley Additive Explanation." *Proceedings of the 10th ACM International Conference on Bioinformatics, Computational Biology and Health Informatics*.

Noor, Abid, Muhammad Asad Saeed, Tehseen Ullah, Zia Uddin, and Raja Muhammad Waseem Ullah Khan (2021). "A review of artificial intelligence applications in apparel industry." *The Journal of The Textile Institute*.

Olson, C. F. (2002). "Maximum-likelihood image matching." *IEEE Transactions on Pattern Analysis and Machine Intelligence*. ISSN: 0162-8828.

Oron, S., A. Bar-Hillel, D. Levi, and S. Avidan (2012). "Locally Orderless Tracking." *2012 IEEE Conference on Computer Vision and Pattern Recognition*.

Pagán, Ester Alba, Marıa del Mar Gaitán Salvatella, Marıa Dolores Pitarch, Arabella León Muñoz, Marıa del Mar Moya Toledo, José Marin Ruiz, Maurizio Vitella, Georgia Lo Cicero, Franz Rottensteiner, Dominic Clermont, et al. (2020). "From silk to digital technologies: a gateway to new opportunities for creative industries, traditional crafts and designers. The SILKNOW case." *Sustainability*.

Pajarola, Renato, Susanne K. Suter, and Roland Ruiters (2013). "Tensor Approximation in Visualization and Computer Graphics." *Eurographics 2013 - Tutorials*. URL: `http://diglib.eg.org/EG/DL/conf/EG2013/tutorials/t6.pdf`.

Park, Jeong Joon, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove (2019). "Deepsdf: Learning continuous signed distance functions for shape representation." *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Park, M., K. Brocklehurst, R. T. Collins, and Y. Liu (2011). "Translation-Symmetry-Based Perceptual Grouping with Applications to Urban Scenes." *Computer Vision – ACCV 2010*. Edited by Ron Kimmel, Reinhard Klette, and Akihiro Sugimoto.

Parke, F. I. and K. Waters (1996). *Computer Facial Animation*.

Patumchat, Puttipong and Keartisak Sriprateep (2019). "New computer geometric modeling approach with filament assembly model for woven fabric structures." *The Journal of The Textile Institute*.

Pele, O. and M. Werman (2008). "Robust Real-Time Pattern Matching Using Bayesian Sequential Hypothesis Testing." *IEEE Transactions on Pattern Analysis and Machine Intelligence*. ISSN: 0162-8828.

Pellacini, F., K. Vidimče, A. Lefohn, A. Mohr, M. Leone, and J. Warren (2005). "Lpics: a Hybrid Hardware-Accelerated Relighting Engine for Computer Cinematography." *ACM Transactions on Graphics*.

Perlin, Ken (1985). "An image synthesizer." *ACM Siggraph Computer Graphics*.

Pham, Chi-Hieu, Saıd Ladjal, and Alasdair Newson (2020). "PCAAE: Principal Component Analysis Autoencoder for organising the latent space of generative networks." *arXiv preprint arXiv:2006.07827*.

Pritts, J., O. Chum, and J. Matas (2014). "Rectification, and Segmentation of Coplanar Repeated Patterns." *2014 IEEE Conference on Computer Vision and Pattern Recognition*.

Rainer, Gilles, Abhijeet Ghosh, Wenzel Jakob, and Tim Weyrich (2020). "Unified neural encoding of btfs." *Computer Graphics Forum*.

Rainer, Gilles, Wenzel Jakob, Abhijeet Ghosh, and Tim Weyrich (2019). "Neural btf compression and interpolation." *Computer Graphics Forum*.

Rasheed, Abdullah Haroon, Victor Romero, Florence Bertails-Descoubes, Stefanie Wuhrer, Jean-Sébastien Franco, and Arnaud Lazarus (2020). "Learning to measure the static friction coefficient in cloth contact." *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Rastegari, Mohammad, Vicente Ordonez, Joseph Redmon, and Ali Farhadi (2016). "Xnor-net: Imagenet classification using binary convolutional neural networks." *European conference on computer vision*.

Rezende, Danilo Jimenez, Shakir Mohamed, and Daan Wierstra (2014). "Stochastic backpropagation and approximate inference in deep generative models." *International conference on machine learning*.

Ribeiro, Marco Tulio, Sameer Singh, and Carlos Guestrin (2016). ""Why should i trust you?" Explaining the predictions of any classifier." *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*.

Rippel, Oren and Lubomir Bourdev (2017). "Real-time adaptive image compression." *International Conference on Machine Learning*.

Robbins, C.R. (2013). *Chemical and Physical Behavior of Human Hair*. URL: `https://books.google.de/books?id=eabTBwAAQBAJ`.

Ruiters, Roland and Reinhard Klein (2009). "BTF Compression via Sparse Tensor Decomposition." *Proceedings of the Twentieth Eurographics Conference on Rendering*. URL: `https://doi.org/10.1111/j.1467-8659.2009.01495.x`.

Runia, Tom FH, Kirill Gavrilyuk, Cees GM Snoek, and Arnold WM Smeulders (2020). "Cloth in the wind: A case study of physical measurement through simulation." *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Saalfeld, Alina, Florian Reibold, and Carsten Dachsbacher (2018). "Image-based Fitting of Procedural Yarn Models." *Workshop on Material Appearance Modeling*. Edited by Reinhard Klein and Holly Rushmeier.

Sadeghi, Iman, Oleg Bisker, Joachim De Deken, and Henrik Wann Jensen (2013). "A practical microcylinder appearance model for cloth rendering." *ACM Transactions on Graphics (TOG)*.

Sako, Y. and K. Fujimura (2000). "Shape Similarity by Homotropic Deformation." *The Visual Computer*.

Sandler, Mark, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen (2018). "Mobilenetv2: Inverted residuals and linear bottlenecks." *Proceedings of the IEEE conference on computer vision and pattern recognition*.

Sattler, Mirko, Ralf Sarlette, and Reinhard Klein (2003). "Efficient and Realistic Visualization of Cloth." *Proceedings of the 14th Eurographics Workshop on Rendering*.

Schroder, K, Reinhard Klein, and Arno Zinke (2011). "A volumetric approach to predictive rendering of fabrics." *Computer Graphics Forum*.

Schröder, K., A. Zinke, and R. Klein (2015). "Image-Based Reverse Engineering and Visual Prototyping of Woven Cloth." *IEEE Transactions on Visualization and Computer Graphics*.

Schröder, Kai, Shuang Zhao, and Arno Zinke (2012). "Recent Advances in Physically-Based Appearance Modeling of Cloth." *SIGGRAPH Asia 2012 Courses*. URL: https://doi.org/10.1145/2407783.2407795.

Schröder, Kai Michael Nikolai (2015). "Visual Prototyping of Cloth." *Ph.D. thesis*. *Rheinische Friedrich-Wilhelms-Universität Bonn*. URL: https://hdl.handle.net/20.500.11811/6459.

Sellereite, Nikolai and Martin Jullum (2019). "shapr: An R-package for explaining machine learning models with dependence-aware Shapley values." *Journal of Open Source Software*. URL: https://doi.org/10.21105/joss.02027.

Shaikh, I.A. (2002). *Pocket Textile Expert: Mini textile encyclopedia*. URL: https://books.google.de/books?id=HFIRmgEACAAJ.

Shapley, Lloyd S (1953). "A value for n-person games." *Contributions to the Theory of Games*.

Sharan, L., C. Liu, R. Rosenholtz, and E. H. Adelson (2013). "Recognizing Materials Using Perceptually Inspired Features." *International Journal of Computer Vision*.

Shin, B. G., S.-Y. Park, and J. J. Lee (2007). "Fast and robust template matching algorithm in noisy image." *2007 International Conference on Control, Automation and Systems*.

Shinohara, Toshihiro, Jun-ya Takayama, Shinji Ohyama, and Akira Kobayashi (2010). "Extraction of yarn positional information from a three-dimensional CT image of textile fabric using yarn tracing with a filament model for structure analysis." *Textile Research Journal*.

Shrikumar, Avanti, Peyton Greenside, and Anshul Kundaje (2017). "Learning Important Features Through Propagating Activation Differences." *Proceedings of the 34th International Conference on Machine Learning*. Edited by Doina Precup and Yee Whye Teh.

Sibiryakov, A. (2011). "Fast and high-performance template matching method." *CVPR 2011*.

Simakov, D., Y. Caspi, E. Shechtman, and M. Irani (2008). "Summarizing visual data using bidirectional similarity." *2008 IEEE Conference on Computer Vision and Pattern Recognition*.

Simonyan, Karen, Andrea Vedaldi, and Andrew Zisserman (2014). "Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps." *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Workshop Track Proceedings*. Edited by Yoshua Bengio and Yann LeCun. URL: http://arxiv.org/abs/1312.6034.

Sperl, Georg, Rahul Narain, and Chris Wojtan (2020). "Homogenized Yarn-Level Cloth." *ACM Transactions on Graphics (TOG)*.

Springenberg, J, Alexey Dosovitskiy, Thomas Brox, and M Riedmiller (2015). "Striving for Simplicity: The All Convolutional Net." *ICLR (workshop track)*.

Sreprateep, Keartisak and Erik L. J. Bohez (2006). "Computer Aided Modeling of Fiber Assemblies." *Computer-Aided Design and Applications*.

Stava, O., B. Benes, R. Mech, D. G. Aliaga, and P. Kristof (2010). "Inverse Procedural Modeling by Automatic Generation of L-systems." *Comput. Graph. Forum*. URL: http://dblp.uni-trier.de/db/journals/cgf/cgf29.html#StavaBMAK10.

Stava, O., S. Pirk, J. Kratt, B. Chen, R. Mech, O. Deussen, and B. Benes (2014). "Inverse Procedural Modelling of Trees." *Computer Graphics Forum*. ISSN: 1467-8659. URL: http://dx.doi.org/10.1111/cgf.12282.

Strumbelj, Erik and Igor Kononenko (2010). "An efficient explanation of individual classifications using game theory." *The Journal of Machine Learning Research*.

Sun, Yi and Mukund Sundararajan (2011). "Axiomatic attribution for multilinear functions." *Proceedings of the 12th ACM conference on Electronic commerce*.

Sundararajan, Mukund, Kedar Dhamdhere, and Ashish Agarwal (2020). "The Shapley Taylor Interaction Index." *International Conference on Machine Learning*.

Sundararajan, Mukund and Amir Najmi (2020). "The many Shapley values for model explanation." *International Conference on Machine Learning*.

Sundararajan, Mukund, Ankur Taly, and Qiqi Yan (2017). "Axiomatic attribution for deep networks." *International Conference on Machine Learning*.

Ta, V.-T., R. Giraud, D. L. Collins, and P. Coupé (2014). "Optimized PatchMatch for Near Real Time and Accurate Label Fusion." *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2014*.

Talmi, I., R. Mechrez, and L. Zelnik-Manor (2017). "Template Matching with Deformable Diversity Similarity." *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Talton, J., L. Yang, R. Kumar, M. Lim, N. Goodman, and R. Měch (2012). "Learning Design Patterns with Bayesian Grammar Induction." *Proceedings of the 25th Annual ACM Symposium on User Interface Software and Technology*. URL: http://doi.acm.org/10.1145/2380116.2380127.

Talton, J. O., Y. Lou, S. Lesser, J. Duke, R. Měch, and V. Koltun (2011). "Metropolis Procedural Modeling." *ACM Trans. Graph.* ISSN: 0730-0301. URL: http://doi.acm.org/10.1145/1944846.1944851.

Tan, Mingxing, Bo Chen, Ruoming Pang, Vijay Vasudevan, Mark Sandler, Andrew Howard, and Quoc V Le (2019). "Mnasnet: Platform-aware neural architecture search for mobile." *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Tan, Sarah, Rich Caruana, Giles Hooker, Paul Koch, and Albert Gordo (2018). "Learning global additive explanations for neural nets using model distillation." *arXiv preprint arXiv:1801.08640*.

Tang, Yehui, Shan You, Chang Xu, Jin Han, Chen Qian, Boxin Shi, Chao Xu, and Changshui Zhang (2020). "Reborn filters: Pruning convolutional neural networks with limited data." *Proceedings of the AAAI Conference on Artificial Intelligence*.

Tao, Xiaoming (1996). "Mechanical properties of a migrating fiber." *Textile research journal*.

Tewari, A., Ohad Fried, J. Thies, V. Sitzmann, S. Lombardi, Kalyan Sunkavalli, Ricardo Martin Brualla, T. Simon, J. Saragih, M. Nießner, R. Pandey, Sean Fanello, G. Wetzstein, Jun-Yan Zhu, C. Theobalt, Maneesh Agrawala, E. Shechtman, Dan Goldman, and M. Zollhöfer (2020). "State of the Art on Neural Rendering." *Computer Graphics Forum*.

Tewari, A., J. Thies, B. Mildenhall, P. Srinivasan, E. Tretschk, W. Yifan, C. Lassner, V. Sitzmann, R. Martin-Brualla, S. Lombardi, T. Simon, C. Theobalt, M. Nießner, J. T. Barron, G. Wetzstein, M. Zollhöfer, and V. Golyanik (2022). "Advances in Neural Rendering." *Computer Graphics Forum (EG STAR 2022)*.

Theis, Lucas, Wenzhe Shi, Andrew Cunningham, and Ferenc Huszár (2017). "Lossy image compression with compressive autoencoders." *arXiv preprint arXiv:1703.00395*.

Tian, Y. and Srinivasa G. Narasimhan (2012). "Globally Optimal Estimation of Nonrigid Image Distortion." *International Journal of Computer Vision*. ISSN: 1573-1405. URL: https://doi.org/10.1007/s11263-011-0509-0.

Tipping, Michael E and Christopher M Bishop (1999). "Probabilistic principal component analysis." *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*.

Toderici, George, Sean M O'Malley, Sung Jin Hwang, Damien Vincent, David Minnen, Shumeet Baluja, Michele Covell, and Rahul Sukthankar (2015). "Variable rate image compression with recurrent neural networks." *arXiv preprint arXiv:1511.06085*.

Tripathi, Sandhya, N Hemachandra, and Prashant Trivedi (2020). "Interpretable feature subset selection: A Shapley value based approach,"" *Proceedings of 2020 IEEE International Conference on Big Data, Special Session on Explainable Artificial Intelligence in Safety Critical Systems*.

Trunz, Elena, Jonathan Klein, Jan Müller, Lukas Bode, Ralf Sarlette, Michael Weinmann, and Reinhard Klein (2023). "Neural Inverse Procedural Modeling of Knitting Yarns from Images." *arXiv:2303.00154 (under review), submitted to Computers & Graphics (CG)*.

Trunz, Elena, Sebastian Merzbach, Jonathan Klein, Thomas Schulze, Michael Weinmann, and Reinhard Klein (2019). "Inverse Procedural Modeling of Knitwear." *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Trunz, Elena, Michael Weinmann, Sebastian Merzbach, and Reinhard Klein (2022). "Efficient structuring of the latent space for controllable data reconstruction and compression." *Graphics and Visual Computing (GVC)*.

Tsai, D.-M. and C.-H. Chiang (2002). "Rotation-invariant pattern matching using wavelet decomposition." *Pattern Recognition Letters*. ISSN: 0167-8655. URL: http://www.sciencedirect.com/science/article/pii/S016786550100099X.

Tschannen, Michael, Eirikur Agustsson, and Mario Lucic (2018). "Deep generative models for distribution-preserving lossy compression." *arXiv preprint arXiv:1805.11057*.

Tucker, Ledyard R (1966). "Some mathematical notes on three-mode factor analysis." *Psychometrika*.

Ulyanov, Dmitry, Andrea Vedaldi, and Victor Lempitsky (2018). "Deep image prior." *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*.

Vanegas, Carlos A., Ignacio Garcia-Dorado, Daniel G. Aliaga, Bedrich Benes, and Paul Waddell (2012). "Inverse design of urban procedural models." *ACM Trans. Graph.* ISSN: 0730-0301. URL: http://doi.acm.org/10.1145/2366145.2366187.

Voborova, Jana, A Garg, Bohuslav Neckar, and Sayed Ibrahim (2004). "Yarn properties measurement: an optical approach." *2nd International textile, clothing and design conference*.

Wang, Hongcheng, Qing Wu, Lin Shi, Yizhou Yu, and Narendra Ahuja (2005). "Out-of-Core Tensor Approximation of Multi-Dimensional Matrices of Visual Data." *ACM Trans. Graph.* ISSN: 0730-0301. URL: https://doi.org/10.1145/1073204.1073224.

Wang, J., C. Liu, T. Shen, and L. Quan (2015). "Structure-driven facade parsing with irregular patterns." *2015 3rd IAPR Asian Conference on Pattern Recognition (ACPR)*.

Wang, Jiaping, Shuang Zhao, Xin Tong, John Snyder, and Baining Guo (2008). "Modeling anisotropic surface reflectance with example-based microfacet synthesis." *ACM SIGGRAPH 2008 papers*.

Wang, Jiaxuan, Jenna Wiens, and Scott Lundberg (2020). "Shapley Flow: A Graph-based Approach to Interpreting Model Predictions." *arXiv preprint arXiv:2010.14592*.

Wang, Rui, Xiaoqian Wang, and David I. Inouye (2021). "Shapley Explanation Networks." *International Conference on Learning Representations*. URL: https://openreview.net/forum?id=vsU0efpivw.

Wang, Xiaoqian, Hong Chen, Jingwen Yan, Kwangsik Nho, Shannon L Risacher, Andrew J Saykin, Li Shen, Heng Huang, and ADNI (2018). "Quantitative trait loci identification for brain endophenotypes via new additive model with random networks." *Bioinformatics*.

Weinmann, M. and R. Klein (2015). "Material Recognition for Efficient Acquisition of Geometry and Reflectance." *Computer Vision - ECCV 2014 Workshops*.

Weinmann, Michael, Juergen Gall, and Reinhard Klein (2014). "Material classification based on training data synthesized using a BTF database." *European Conference on Computer Vision*.

Weinmann, Michael, Fabian Langguth, Michael Goesele, and Reinhard Klein (2016). "Advances in Geometry and Reflectance Acquisition." *EG 2016 - Tutorials*. Edited by Augusto Sousa and Kadi Bouatouch.

Weissenberg, J., H. Riemenschneider, M. Prasad, and L. Van Gool (2013). "Is there a procedural logic to architecture?" *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*.

Wenzel, S. and W. Förstner (2016). "FACADE INTERPRETATION USING A MARKED POINT PROCESS." *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*. URL: https://www.isprs-ann-photogramm-remote-sens-spatial-inf-sci.net/III-3/363/2016/.

Westermann, R., Steinbach E., Seidel H.-P., Niemann H., Greiner G., Klein Reinhard, Meseth Jan, Girod B., Müller Gero, and Ertl T. (2003). "Compression and Real-Time Rendering of Measured BTFs Using Local PCA." *Vision, Modeling and Visualisation 2003*.

Westin, Stephen H., James R. Arvo, and Kenneth E. Torrance (1992). "Predicting Reflectance Functions from Complex Surfaces." *SIGGRAPH Comput. Graph.* ISSN: 0097-8930. URL: https://doi.org/10.1145/142920.134075.

Wonka, P., D. Aliaga, P. Müller, and C. Vanegas (2011). "Modeling 3D Urban Spaces Using Procedural and Simulation-based Techniques." *ACM SIGGRAPH 2011 Courses*. URL: http://doi.acm.org/10.1145/2037636.2037645.

Wu, F., D.-M. Yan, W. Dong, X. Zhang, and P. Wonka (2014). "Inverse Procedural Modeling of Facade Layouts." *ACM Trans. Graph.* ISSN: 0730-0301. URL: http://doi.acm.org/10.1145/2601097.2601162.

Wu, Hong-yu, Xiao-wu Chen, Chen-xu Zhang, Bin Zhou, and Qin-ping Zhao (2019). "Modeling yarn-level geometry from a single micro-image." *Frontiers of Information Technology & Electronic Engineering*.

Wu, Kui and Cem Yuksel (2017). "Real-Time Fiber-Level Cloth Rendering." URL: https://doi.org/10.1145/3023368.3023372.

Wu, Qing, Tian Xia, Chun Chen, Hsueh-Yi Sean Lin, Hongcheng Wang, and Yizhou Yu (2008). "Hierarchical Tensor Approximation of Multi-Dimensional Visual Data." *IEEE Transactions on Visualization and Computer Graphics*.

Xu, Ying-Qing, Yanyun Chen, Stephen Lin, Hua Zhong, Enhua Wu, Baining Guo, and Heung-Yeung Shum (2001). "Photorealistic rendering of knitwear using the lumislice." *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*.

Yang, Fei, Luis Herranz, Yongmei Cheng, and Mikhail G Mozerov (2021). "Slimmable compressive autoencoders for practical neural image compression." *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Yang, Fei, Luis Herranz, Joost Van De Weijer, José A Iglesias Guitián, Antonio M López, and Mikhail G Mozerov (2020). "Variable rate deep image compression with modulated autoencoder." *IEEE Signal Processing Letters*.

Yee, Y. L. H. (2000). "Spatiotemporal sensitivity and visual attention for efficient rendering of dynamic environments." *Master's thesis. Cornell University*.

Yu, Jiahui, Linjie Yang, Ning Xu, Jianchao Yang, and Thomas Huang (2018). "Slimmable neural networks." *arXiv preprint arXiv:1812.08928*.

Yu, Shipeng, Kai Yu, Volker Tresp, Hans-Peter Kriegel, and Mingrui Wu (2006). "Supervised probabilistic principal component analysis." *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*.

Yuksel, Cem, Jonathan M Kaldor, Doug L James, and Steve Marschner (2012). "Stitch meshes for modeling knitted clothing with yarn-level detail." *ACM Transactions on Graphics (TOG)*.

Zhang, Z., A. Ganesh, X. Liang, and Y. Ma (2012). "TILT: Transform Invariant Low-Rank Textures." *International Journal of Computer Vision*. ISSN: 1573-1405. URL: https://doi.org/10.1007/s11263-012-0515-x.

Zhao, S., F. Luan, and K. Bala (2016). "Fitting procedural yarn models for realistic cloth rendering." *ACM Transactions on Graphics (TOG)*.

Zhao, Shuang, Miloš Hašan, Ravi Ramamoorthi, and Kavita Bala (2013). "Modular flux transfer: efficient rendering of high-resolution volumes with repeated structures." *ACM Transactions on Graphics (TOG)*.

Zhao, Shuang, Wenzel Jakob, Steve Marschner, and Kavita Bala (2011). "Building volumetric appearance models of fabric using micro CT imaging." *ACM Transactions on Graphics (TOG)*.

Zhao, Shuang, Wenzel Jakob, Steve Marschner, and Kavita Bala (2012). "Structure-aware synthesis for predictive woven fabric appearance." *ACM Transactions on Graphics (TOG)*.

Zintgraf, Luisa M., Taco S. Cohen, Tameem Adel, and Max Welling (2017). "Visualizing Deep Neural Network Decisions: Prediction Difference Analysis." *CoRR*. arXiv: 1702.04595. URL: http://arxiv.org/abs/1702.04595.

List of Figures

List of Tables

Part IV

Appendix

Publication: "Efficient structuring of the latent space for controllable data reconstruction and compression"

Elena Trunz, Michael Weinmann, Sebastian Merzbach, and Reinhard Klein

Graphics and Visual Computing (GVC)

2022

DOI: 10.1016/j.gvc.2022.200059

Graphics and Visual Computing 7 (2022) 200059

Contents lists available at ScienceDirect

Graphics and Visual Computing

journal homepage: www.elsevier.com/locate/gvc

Technical section

Efficient structuring of the latent space for controllable data reconstruction and compression ☆

Elena Trunz [a,*], Michael Weinmann [b], Sebastian Merzbach [a,c], Reinhard Klein [a]

[a] University of Bonn, Germany
[b] Delft University of Technology, Netherlands
[c] X-Rite Europe GmbH, Switzerland

ARTICLE INFO

Article history:
Received 4 March 2022
Received in revised form 25 June 2022
Accepted 1 November 2022
Available online 5 November 2022

Keywords:
Autoencoder
Encoder–decoder architectures
Data compression
Data representation
Deep learning
Shapley values

ABSTRACT

Explainable neural models have gained a lot of attention in recent years. However, conventional encoder–decoder models do not capture information regarding the importance of the involved latent variables and rely on a heuristic a-priori specification of the dimensionality of the latent space or its selection based on multiple trainings. In this paper, we focus on the efficient structuring of the latent space of encoder–decoder approaches for explainable data reconstruction and compression. For this purpose, we leverage the concept of Shapley values to determine the contribution of the latent variables on the model's output and rank them according to decreasing importance. As a result, a truncation of the latent dimensions to those that contribute the most to the overall reconstruction allows a trade-off between model compactness (i.e. dimensionality of the latent space) and representational power (i.e. reconstruction quality). In contrast to other recent autoencoder variants that incorporate a PCA-based ordering of the latent variables, our approach does not require time-consuming training processes and does not introduce additional weights. This makes our approach particularly valuable for compact representation and compression. We validate our approach at the examples of representing and compressing images as well as high-dimensional reflectance data.

1. Introduction

The rapid progress in deep learning has led to huge improvements in numerous application areas including data reconstruction, compression and streaming, where explainability of the model's behavior and decisions is of particular importance. Respective approaches including autoencoders rely on the core idea of encoding the information extracted from the input data in another *latent* space, from where it can be decoded back to the original domain. Powerful and compact representations can then be obtained based on the combination of an information bottleneck, i.e. choosing the dimensionality of the latent space to be lower than the input dimensionality so that only the most salient features are preserved, and appropriate loss functions.

In this paper, we put our attention to the challenging problem of the efficient specification of a convenient dimensionality of the latent space that preserves model accuracy for respectively considered tasks.

So far, most approaches for designing encoder–decoder schemes have been based on a heuristic specification of the number of latent dimensions, i.e. without an actual explanation why the respective dimensionality has been chosen and without an analysis regarding the dimensionality that best suits the particular application. In contrast, determining a suitable number of latent variables has also been addressed for some encoder–decoder approaches (e.g. [1,2]) that are trained with different numbers of latent variables, where finally the dimensionality leading to the best trade-off between small dimensionality of the latent space and reconstruction quality is chosen. However, this procedure is very time-consuming, since the encoder–decoder needs to be trained anew for each number of latent variables, and, for high-dimensional latent spaces, such a repeated training may even be infeasible. Instead, we propose a novel approach for the specification of a suitable dimensionality of the latent space by analyzing the contribution of the individual latent dimensions and their respective ranking in encoder–decoder schemes based on their Shapley values [3]. The concept of Shapley values has originally been introduced in cooperative game theory for feature attribution, and we leverage this concept similar to the computation of a natural ordering of the components regarding their contribution based on principal component analysis (PCA) [4,5], but instead for the more general non-linear relationship that

☆ This article was recommended for publication by D. Bommes.

* Correspondence to: Visual Computing Department, FriedrichHirzebruch-Allee 5, 53115 Bonn, Germany.
E-mail addresses: trunz@cs.uni-bonn.de (E. Trunz), M.Weinmann@tudelft.nl (M. Weinmann), merzbach@cs.uni-bonn.de (S. Merzbach), rk@cs.uni-bonn.de (R. Klein).

https://doi.org/10.1016/j.gvc.2022.200059

typically allows autoencoders to find more flexible and more powerful latent spaces. While the latent space of an autoencoder exhibits compactness with respect to the original data domain, it does not allow gaining structural information of the latent space as in the case of a PCA. More sophisticated autoencoder variants [6,7] allow the ordering of the dimensions of the latent codes according a decreasing importance with respect to the input data while preserving statistically independent components. However, these approaches are based on progressively increasing the dimensionality of the latent space, i.e. learning one new dimension per step. Instead, our approach avoids such a progressive adaption of the required dimensionality by the direct use of the contribution of individual latent dimensions according to their Shapley values. The computation of the contributions of individual latent dimensionalities and their ordering in terms of the Shapley value based analysis can be applied at different times during training as long as the training loss does not significantly change over the epochs anymore. In the scope of our experiments, we will compare the results of applying the Shapley value based analysis in the middle of the training and at the end of the training. We investigate the beneficial combination of Shapley values and encoder–decoders regarding the choice of the dimensionality of the latent space, the ordering of the involved latent variables according their importance and the respective capability for reconstruction and compression. This is motivated by the fact proven in the paper that in case of a linear model the ordering based on Shapley values is the same as the one after the singular value decomposition and, hence, optimal. In summary, the key contributions of this paper are:

- We present novel method for ranking the latent variables in an encoder–decoder, based on their contribution to the result, and the subsequent specification of a suitable dimensionality of the latent space.
- We demonstrate the benefits of our approach by evaluations on various different application scenarios.
- We provide the theorem and the proof of the optimality of the Shapley ordering in the linear case.

2. Related work

The complexity of the concept of interpretability [8,9] makes a general interpretation regarding a model's behavior/decisions intractable. The key objective of feature attribution methods that focus on local interpretability is the identification of relevant features based on a scalar attribution score, relevance score [10] or contribution [11] that defines how much each input feature contributes to a model's behavior. However, a limited theoretical understanding as well as the lack of reliable quantitative metrics for evaluating explanations in case on ground truth is available [12] may lead to unreliable or even misleading results that may still appear visually appealing [12–15]. This problem has been addressed based on incorporating desirable axioms into the attribution method [13,16–19]. These axioms have to be fulfilled by any explanation obtained from the respective attribution methods and allow the design of attribution methods with theoretical guarantees [17]. In the context of deep learning, *backpropagation-based attribution methods* rely on the idea of computing the attribution based on backward passes through the network. Examples include the computation of attributions by exploiting gradients that carry the information regarding the local perturbations of features that mostly influence the output. Here, attributions can be obtained in terms of saliency maps [20], that refer to the gradient of the class score with respect to the input image, or by elementwise multiplications of input data and signed gradients (Gradient×Input) [21]. However, this approach only provides local information in case of highly nonlinear functions and, hence, not suitable to compute marginal contributions of features. Therefore, other approaches such as Layer-wise Relevance Propagation (LRP) [10,18], DeepLIFT variants [11,21] and Integrated Gradients [17] make use of different propagation rules in comparison to the use of the instant gradient in the Gradient×Input approach. However, perceptually similar inputs with the same predicted labels may be interpreted differently as even small random perturbations affect the feature importance and systematic perturbations may change the interpretation while keeping the label [14]. In contrast, perturbation-based approaches rely on the computation of the relevance of input features by analyzing the behavior of a neural network in case of feature removal or perturbation [22–24].

A related classical concept developed in the domain of cooperative game theory in order to distribute the contribution of individual players in a cooperative game while fulfilling desirable axioms is given by the Shapley values [3] and resulting feature attributions even seem to agree to human intuition [19]. As discussed in literature, computing exact Shapley values remains an NP-hard problem [25] and, in practice, can only be performed for less than 20 to 25 players (i.e. input features in our case respectively). For this reason, much effort was spent on finding adequate approximations of Shapley values such as in terms of sampling-based methods [26–29] as well as on investigating new classes of additive feature importance measures for particular predictions as denoted by SHapley Additive exPlanations (SHAP) [19]. To avoid the rapidly increasing number model evaluations for increasing numbers of input features, additional lasso regression has been used in KernelSHAP [19] and its respective extensions towards global interpretability [30], different importance metrics and feature packing [31], handling dependent features [32,33] and producing additional types of explanations [34] such as explaining whether samples are likely to a certain class, why prediction differ depending on the observations and when the model has a bad performance. Furthermore, approximations based on the assumption of model linearity have been proposed (such as DeepSHAP) [19] and extended to mixed model types [35] in terms of a layerwise propagation of Shapley values built upon DeepLIFT [11,21]. This also enables computational tractability for obtaining exact Shapley values for certain model types like tree-based models [36], such as random forests or gradient boosted trees, and allows attributing stacks of mixed models such as the feature extraction of neural networks into a tree model and also attributing loss functions. Polynomial approximations for specific games such as voting games [37] allow a polynomial-time approximation of Shapley values as shown with Deep Approximate Shapley Propagation (DASP) [38]. Further work includes extensions towards global explainability (by combining the Shapley value concept with Lorenz zonoids [39] to combine the advantages of the local Shapley value based approach with the properties of the Lorenz Zonoids), the generalization of Shapley values to the Shapley–Taylor index that reflects attributions of subsets of features [40] and the exploitation of assumptions regarding the underlying data structures [41]. As a counterpart, Shapely Residuals [42] have been introduced to capture the information not preserved by Shapley values.

However, most of the approaches for calculating Shapley values rely on post-hoc explanations. Therefore, the explanation approach cannot be used for designing and training models. Generalized Additive Models (GAM) based on tree boosting [43,44] or neural networks [45] allow the simultaneous prediction and computation of the corresponding exact SHAP explanation, but their representational is power inherently limited. Instead, Shapley Explanation Networks [46] rely on directly incorporating Shapley values as the learned latent representations in deep neural

networks. SHapley Additive exPlanations connect local explanations with optimal credit allocation and has been used to rank input variables for identification and prediction of failure modes [47]. Furthermore, Shapley value based error apportioning (SVEA) [48] has been introduced where the key idea is the apportioning of the total training error among the features and Ghorbani and Zou [49] focus on equitable data valuation in the context of supervised learning, where data Shapley values are used to rate the contribution of each training sample to the prediction performance. Covert et al. [50] focused on explainability in terms of simulating the effect of feature removal to determine the influence of individual features. Their framework analyzes how features are removed for different methods, the respectively explained model behavior, and how methods summarize the features' contributions. In addition, by assigning contribution scores to edges instead of nodes within a causal graph structure, Shapley Flow [51] generalizes the Shapley value axioms to directed acyclic graphs. Ghorbani and Zou [52] used Shapley values for quantifying the importance of individual neurons for network predictions and performance. This allows a more efficient identification of important filters in comparison to the use of activation patterns. Furthermore, Ma et al. [53] considered Shapley values in the scope of Bayesian networks and showed a relation between Shapley values and conditional independence.

We exploit feature attribution based on Shapley values in encoder–decoder frameworks to efficiently structure the latent space by ordering the latent variables according to their importance and exploit this strategy for explainable data reconstruction and compression.

Aside from feature attribution, several works specifically focus on data compression to allow efficient storage and transmission of contents through constrained channels, where in particular neural image compression has gained a lot of attention in recent years. The targeted tradeoff of determining an as-compact-as-possible binary representation (i.e. lowest rate bitstream) while preserving a certain level of fidelity (i.e. minimum distortion) of the data has been investigated in terms of autoencoder architectures with quantization and entropy coding. Such compressive autoencoders [54–56] rely on also minimizing the combination of rate and distortion during training and have been improved by multi-scale extensions of the encoder and/or decoder [57–59] or adding generalized divisive normalization (GDN) layers [56,60]. Furthermore, end-to-end training can be achieved based on replacing the non-differentiable quantization by differentiable proxies [55,61,62]. In addition, hyperpriors [63] and contextual models [64–68] have been used to improve entropy coding. Several works also focus on adversarial training schemes to achieve very low rates [58,69,70].

Targeting variable rate image compression, traditional compression methods were based on quantizing Discrete Cosine Transform (DCT) coefficients according to the target rate. Further techniques include the learning of rate-specific bottleneck scaling (i.e. scaling the bottleneck features before quantization) [55], the modulation of intermediate features based on modulated autoencoders (MAEs) [71] and conditional autoencoders (cAEs) [72], the use of recurrent neural networks [54] and the use of a multi-scale decomposition network where each scale targets a different rate. Furthermore, increasing the efficiency of deep learning for resource-limited scenarios as occurring for tablets or smartphones has been generally addressed in terms of searching lightweight architectures [73–75], integer and binary networks [76–78], automatic architecture search [79], or adjusting the width of layers to achieve a trade-off between computational efficiency and accuracy based on slimmable neural networks [80]. In the scope of neural image compression, network architecture search [81] or progressive ecoding [82] have been used to address runtime and latency respectively. However, memory requirements and the computational burden do not change significantly and only a single rate–distortion tradeoff is considered which prohibits flexibility regarding rate, memory or the computational burden. To increase the practicality of neural image compression, slimmable compressive autoencoders [83] also allow controlling computation, memory and rate. While these approaches have shown great potential in the context of image compression, our approach represents a powerful alternative that can be combined with these approaches and can be applied to any compression method, which utilizes an encoder–decoder.

3. Structuring and pruning of latent space representations

Data reconstruction and compression techniques often rely on the transformation of the input data into a lower-dimensional latent space that describes the most salient features. Here, the dimensionality of the latent space has to be chosen to adequately represent the distribution of the input data while allowing an as-compact-as possible latent representation. This trade-off between reconstruction accuracy and compression rate has to be carefully considered depending on the underlying task and its implications. For this purpose, we have to focus on the following central questions:

1. What is a suitable choice for the dimensionality of the latent space, i.e. how many latent variables are required?
2. Can we relate the lossy compression induced by discarding dimensions to individual features' importance?
3. Does the structure of the latent space exhibit insights on how much we gain by taking less or more latent variables, thereby allowing the efficient control for the specification of a suitable dimensionality of the latent space without the need of having to train different autoencoders for each dimensionality like in the approach by Rainer et al. [1]?

Gaining control over latent variables and the ability to detect the contribution of each dimension to the overall performance of the model is an important step towards explainable models. In fact, we would even wish to determine the contribution of sets of important dimensions, i.e. we would like to get insights regarding which subsets of k dimensions exhibit the largest importance instead of taking the k individually most contributing features. That way, we can order the dimensions according their contribution and, hence, structure the latent space, which, in turn, allows taking the first most important k dimensions to get a rank-k approximation of the data for lossy compression.

In the following, we first provide an overview on how these questions have been handled in the scope of linear approaches. In this regard, we will demonstrate that, in the linear case, where the Eckart–Young–Mirsky theorem states how low-rank approximation can be approached, the ordering of the eigenvectors according to their Shapley values is equal to the ordering of the corresponding singular values. Then, we motivate why these questions become much more challenging in the case of non-linear models such as autoencoders and finally devise a strategy towards an efficient model that helps answering the aforementioned questions. Due to the properties of Shapley values that directly measure contributions of individual components, we aim at analyzing whether these could also serve in these non-linear scenarios (where the Eckart–Young–Mirsky theorem would not be applicable) and experimentally address these questions.

3.1. Latent space representations in linear models

Before analyzing whether the properties of Shapely values regarding their capability to directly measure contributions of

individual components make them suitable for non-linear scenarios as given for encoder–decoder architectures, we first demonstrate that in the linear case, where the Eckart–Young–Mirsky theorem states how low-rank approximation can be approached, the ordering of the eigenvectors according to their Shapley values is equal to the ordering of the corresponding singular values.

In case of a linear relationship $Ax = y$ between inputs x and model outputs y, we can compute the best rank-k approximation A_k to A in the L_2-norm in terms of an analytical solution, i.e. by performing a singular value decomposition of $A = U\Sigma V^*$, as postulated by the Eckart–Young–Mirsky theorem [84]. This low-rank approximation A_k is given by

$$A_k = \sum_{i=1}^{k} (\sigma_i u_i v_i^*), \tag{1}$$

where σ_i are the singular values, u_i are the columns of U and v_i are the columns of V. Classical low-rank approximations such as SVD-based approaches and respective robust variants [4,5,85] have been proven to work for matrix-based data. However, the extension of these methods to higher-dimensional data is not straight-forward and comes at the loss of some of their underlying unique properties. For this reason, higher-order tensor decomposition models such as multilinear SVD [86], higher-order orthogonal iteration (HOOI) and the higher order power-method (HOPM) [87] as well as the CANDECOMP/PARAFAC (CP) model [88,89] and the Tucker tensor model [90] have been widely applied. Tensor decomposition can be interpreted as a generalization of the SVD approach to higher-order tensor data and the original data can be approximated based on a rank-reduced tensor decomposition [91,92]. Furthermore, latent variable models such as factor analysis [93] and probabilistic principal component analysis [94–96] exploit low-dimensional features to define powerful generative models.

3.2. Latent space representations in non-linear models

In contrast to linear models, deep neural networks enable non-linear mappings [97,98] and have demonstrated their power in modeling high-dimensional data. Autoencoders focus on the reconstruction of the inputs by first using an encoder e to project the input data to a latent space, which is followed by a decoder d that transfers the latent code back into the original domain to get a reconstruction of the input.

Therefore, their objective consists of minimizing reconstruction errors given by the reconstruction loss, where specific performance metrics v such as the L_2-norm are widely used. To prevent autoencoders from directly copying the inputs and instead force the encoder to learn useful properties of the data, autoencoders typically constrain the latent representation to be lower-dimensional than the input, thereby enforcing them to instead capture the most salient features of the input. In addition, if the latent space has lower dimensionality than the input space $\mathcal{X}$, the latent vector $e(x)$ can be regarded as a compressed representation of the input $x \in \mathcal{X}$. Ideally, the dimension of the latent space should be chosen according to the complexity of the considered problem.

Unfortunately, the latent space is not represented in terms of independent components that can be ordered according to a decreasing relevance for the data as in the case of the PCA. Hence, there is no analytical solution for rank-k approximation problems equivalent to the case of PCA that provides insights regarding the top-k latent components that together most contribute to the reconstruction quality. This induces the initial prerequisite to design a neural network architecture that suits the requirements of a particular task by manually selecting the number of latent variables which, a-priori, is non-trivial. A reasonable and widely followed approach is choosing a sufficiently high number of dimensions of the latent space and adding a respective regularization term.

In fact, we would instead wish to rate the contributions of latent dimensions based on a function ϕ that fulfills the following three axioms:

1. **Zero-player**: Latent variables that do not contribute to the resulting output should be assigned the weight 0 (or a *baseline value*).
2. **Symmetry**: The loss should not depend on the ordering of the latent variables but instead only on their presence.
3. **Efficiency**: Contributions of individual latent variables sum up to the contribution of all latent variables.

For the sake of explainability in terms of getting insights on the structure of the latent space, it would be useful to have a respective ordering of components according to an importance score and the respectively resulting error induced by rank-k approximation similar to the ordering in terms of singular values in the linear case. A naive way to approach this goal is the training of several autoencoders with different numbers of latent variables [2]. As a result, the corresponding error for each number of dimensions is obtained and, hence, the dimensionality that results in the best trade-off between dimensionality of the latent space and reconstruction quality can be selected. However, the multitude of involved training procedures make this procedure time- and resource-demanding.

In contrast, the PCA-like autoencoder [6] and the principal Component Analysis Autoencoder (PCAAE) [7] organize the dimensions of the latent space in decreasing importance with respect to the input data while preserving statistically independent components. For this purpose, these approaches rely on progressively increasing the dimensionality of the latent space and learning one new dimension per step as well as extending the standard autoencoder reconstruction loss by an additional covariance loss applied to the latent codes to enforce statistically independent latent space components. However, this procedure is very time-consuming, as the decoder needs to be re-trained with every additional dimension. Instead, we propose to leverage the ordering of the latent variables according to their contribution defined in terms of Shapley values [3], a concept of game theory for computing the contribution of players in a cooperative game.

3.3. Shapley value guided latent representation

In the scope of a cooperative game, Shapley values [3] assign the participating players, in our case latent variables, their respective contribution to the overall task, in our case the reconstruction of the latent code through the decoder. More formally, the Shapely value $\phi_i(v, N)$ of a latent dimension i is calculated as

$$\phi_i(v, N) = \sum_{S \subseteq N \setminus \{i\}} \frac{|S|!(n - |S| - 1)!}{n!} (v(S \cup \{i\}) - v(S)), \tag{2}$$

where v is a coalition function that maps each subset $S \subseteq N$ of the players to real numbers, which represent the outcome of the game when players in S participate in it. In our scenario, N is the set of $n = |N|$ latent dimensions and the function v can be adapted for a decoder function d by defining the baseline as discussed by Ancona et al. [38]. This way we replaced $v(S)$ in (2) by $d(\mathbf{z}_S)$ and $\mathbf{z}_S$ denotes the original latent vector $\mathbf{z}$ where all entries not included in S are replaced with the baseline value, which is zero in our case. Since we have to process more than one latent vector in order to calculate the contribution of one latent dimension, we randomly choose a set of m example latent vectors

from the space of all possible ones and average the resulting contributions.

Shapely values fulfill several desirable axioms and in particular the three axioms described in Section 3.2, which are of great relevance for our envisioned objective of ranking individual latent variables according to their contributions and specifying which set of them to keep for an as-compact-as-possible but still powerful latent representation. Since we are interested in the correct ordering of the latent dimensions according their descending contribution, we need to validate that the ordering according to the Shapley values exhibits this property. Therefore, we first prove that in the case of a linear model the ordering according to the Shapley values is optimal:

Theorem 1. *Let $A \in \mathbb{R}^{m\times n}$ be a real matrix with $m \geq n$ and $A = U\Sigma V^T$ be the singular value decomposition of A, where Σ is an $m\times n$ diagonal matrix with entries $\sigma_1, \ldots, \sigma_n$, such that $\sigma_1 \geq \cdots \geq \sigma_n$. Furthermore, let $N = \{u_1, \ldots, u_n\}$ be the set of left-singular vectors, i.e. the columns of U. Let v be a function, which assigns a set $S \subseteq N$ the corresponding reconstruction error, i.e. $v(S) = \|A - A_S\|_2$, where $A_S = \sum_{i\in S} \sigma_i u_i v_i^T$ denotes the approximation of the matrix A with the vectors from the set S. Then for all pairs i, j with $i \neq j$ the following holds: $\sigma_i \geq \sigma_j \Leftrightarrow \phi_i(v, N) \leq \phi_j(v, N)$, where $\phi_i(v, N)$ is defined according to (2).*

We provide the proof of this theorem in the supplemental. Note that in our case v is not a function of the reconstruction error but a function of the actual reconstruction, i.e. the decoder. We can still apply the theorem, if we change $(v(S\cup\{i\}) - v(S))$ in (2) to the absolute value, as used in the remainder of our paper. As a direct corollary to this theorem we conclude that the first k elements according to the Shapley values constitute the best rank-k approximation of a linear model A.

In other words, the aforementioned Theorem 1 demonstrates that, in the linear case, where the Eckart–Young–Mirsky theorem states how low-rank approximation problem can be approached, the ordering of the eigenvectors according to their Shapley values is equal to the ordering of the corresponding singular values. In our work, we additionally (experimentally) investigate whether the properties of Shapley values in terms of being a measure for the contribution of individual components also brings benefits for non-linear scenarios, where the Eckart–Young–Mirsky theorem would not be applicable.

Unfortunately, the computation of exact Shapley values remains an NP-hard problem [25] and is feasible only for a very limited number of less than 20 to 25 players or, in our case, latent dimensions respectively. Recently, Deep Approximate Shapley Propagation (DASP) [38] has been introduced as an approach that allows incorporating desirable axioms in the scope of a polynomial-time approximation of Shapley values which makes them suitable for being used in deep neural networks. We use this approach to approximate the Shapley values of the latent dimension for our purposes. The Shapley values are then approximated by the average of the expected contribution to a random coalition according to

$$\mathbf{E}[\phi_i] = \frac{1}{n}\sum_{j=0}^{n-1} \mathbf{E}_j[\phi_{i,j}]. \tag{3}$$

Here the expectations $\mathbf{E}_j$ are calculated over the distribution of sets of size j and $\mathbf{E}_j[\phi_{i,j}]$ denotes the contribution of the latent entry z_i to any random coalition of size j. $\mathbf{E}_j[\phi_{i,j}]$ is then calculated as follows:

$$\mathbf{E}_j[\phi_{i,j}] = |\mathop{\mathbf{E}}_{S\subseteq N\setminus\{i\},|S|=j}[d(\mathbf{z}_{S\cup\{i\}})] - \mathop{\mathbf{E}}_{S\subseteq N\setminus\{i\},|S|=j}[d(\mathbf{z}_S)]| \tag{4}$$

As already described, we use the absolute value instead of the difference. If our decoder function outputs values with more than one dimension, as it does for example in the case of RGB values, then the difference in (4) is computed componentwise and then we sum over all dimensions of the output.

3.4. Choice of the latent space dimensionality

In order to choose an appropriate dimensionality for the latent space that allows capturing the most salient features in an as-compact-as-possible latent representation, we start the training of the network with a sufficiently (i.e. typically too) large number of dimensions of the latent space, thereby following the intuition that a too large size of the latent space can be determined quite easily. To be sure that the initial "big drop-off" of the loss is passed and we reach a plateau-like behavior, we let the training progress for half the number of the epochs of a full training. We then compute the contribution of each latent dimension in terms of their Shapley values and order the dimensions in a descending order according to these contributions. Note that in earlier epochs of the training the adaptions to the network weights, and hence also the adaptions of the distribution of data in the latent space, are strongly changing. Therefore, Shapley value based contribution assignments to individual dimensions of the latent space would provide less insights there while requiring a computational overhead, and, hence, we apply the Shapley value based analysis only when approaching a plateau-like behavior of the loss. Subsequently, we compute the loss for each set of the h first dimensions. Based on the cumulative contribution and cumulative loss we choose the dimensionality k of the latent space and also specify which of the latent variables to take for the continuation of the training by taking a reference of the coverage percentage of the contributions, i.e. according to the percentage of contribution that should be covered. As a result, we prune the latent space custom-tailored according to the complexity of the considered application scenario. The training continues with the same autoencoder as before but without the discarded latent dimensions and the corresponding neurons in the input layer of the decoder. The overall training is finished in the same total amount of epochs as the full training with the only overhead of the computation of Shapley values. Hence, this strategy does not require a time-consuming training process based on successively adding one more dimensions for iteratively conducted trainings. Instead, it only requires a single training to identify less relevant latent variables, and several of these latent variables with low importance can be discarded in a single step. The individual steps of our approach are presented in Algorithm 1.

Algorithm 1 Shapley value based pruning of latent dimensions

1: Train model for a certain number of epochs with initially specified number of latent dimensions
2: Select a subset of m samples of the latent codes (obtained for training examples)
3: Compute approximate Shapley values for the latent variables based on the selected m samples and the decoder function (i.e., coalition function), e.g. based on DASP
4: Order the latent variables in descending order w.r.t. the Shapley values and compute the cumulative contribution and visualizations
5: Decide from the orderings, cumulative contribution and visualizations how many dimensions k will be kept
6: Modify the last layer encoder layer and the first decoder layer by only keeping the connections/neurons belonging to the first k latent variables
7: Resume the training

Note that the Shapley value based analysis to compute the contributions of individual latent dimensionalities and their ordering can be applied at different times during training. The

only requirement is that the training loss does not significantly change over the epochs, but instead is (almost) approaching a flat decreasing behavior. In the scope of our experiments, we will compare the effects of conducting the Shapley value based analysis in the middle of the training as well as at the end of the training. We observed that the importance of latent dimensions still changes until the end of the training and performing the Shapley analysis early might overestimate the number of dimensions that are needed. Instead, to get the best results we should perform the analysis in the end. However, in the scope of our experiments we show that for a good estimation it is sufficient to perform the analysis in the middle of the training to save time, since the changes of the importance of latent dimensions are only small afterwards.

4. Experiments

In the following, we validate our approach for determining a suitable dimensionality of the latent space and the compression of models. To demonstrate the versatile applicability of our approach, we focus on a set of different exemplary application scenarios that differ in the type of data and their respective complexity. We validate the benefits of incorporating Shapley values within autoencoder frameworks regarding the choice of the latent code size and the respective ordering of the dimensions according their importance at the examples of representing and compressing images as well as high-dimensional reflectance data. All experiments were performed on a desktop computer with an Intel(R) Xeon(R) CPU E5-2640 v4 @ 2.40 GHz and an Nvidia Titan XP GPU with 12 GB of RAM.

4.1. Reflectance representation and compression

First, we demonstrate the potential of our approach for the task of representing and compressing reflectance data. Bidirectional texture functions (BTFs) $f(\mathbf{x}, \lambda, \omega_i, \omega_o)$ have been proven to accurately capture local material appearance at surface positions $\mathbf{x}$ of a material sample under varying viewing conditions ω_o and lighting conditions ω_i and possibly also depending on the wavelength λ [99], however, at the cost of massive memory consumption. In the scope of our experiments, we used publicly available BTF datasets provided by Weinmann et al. [100] and particularly focused on leather, carpet and fabric materials due to their complex reflectance behavior. These measurements come at a high angular resolution (i.e. $151 \times 151 = 22801$ light/view configurations with approximately identical samplings of the light and view configurations) and a spatial resolution of 400×400 texels.

The measurements for individual surface positions $\mathbf{x}$ are stored as 4D reflectance functions $f_{\mathbf{x},\lambda}(\omega_i, \omega_o)$ that are denoted as *apparent bidirectional reflectance distribution functions (ABRDFs)*. In contrast to bidirectional reflectance distribution functions (BRDFs), ABRDFs also capture non-local effects of light exchange at the surface such as local subsurface scattering, self-masking or self-shadowing. Finally, material samples are represented in terms of a matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$, where the columns represent the ABRDFs of the m considered surface texels. Recent work on BTF compression and interpolation, which is particularly required for efficiently storing and rendering such data, includes the neural approach by Rainer et al. [1]. In contrast to matrix factorization techniques, that may cause blurring or ghosting artefacts in case of coarse angular resolution, and the fitting of analytic models with a reduced representation capability regarding complex non-local lighting effects, Rainer et al. leveraged the concept of autoencoders to introduce a neural network-based BTF representation. Here, the local surface appearance under different viewing and lighting conditions is first compressed to a latent representation by an encoder component, and the decoder decodes the latter with additional light and view specifications to render the color of a particular surface point, thereby overcoming limitations based on a linear interpolation between measurements. However, for each BTF a new autoencoder needs to be trained. To make a reasonable choice regarding the dimensionality of the latent space, Rainer et al. [1] separately trained autoencoders for different numbers of dimensions of the latent space up to $n = 32$ and finally selected 8 latent variables, based on the trade-off between the compression rate and the reconstruction error. Besides the resulting high computational burden for all these trainings, the selection of an actually optimal trade-off can only be reached for the single, considered BTF as the network has been trained for each BTF separately. From the observations for a single BTF or a very limited set of BTFs, Rainer et al. concluded 8 dimensions to be a suitable size of the latent space. However, this chosen dimensionality may not be adequate for other materials, e.g. with different or more complex appearance characteristics that had not been investigated. Indeed, the resulting mean squared error for the fully trained networks for different numbers of latent dimensions for 3 BTFs depicted in Fig. 1 reveal that the curves deviate, i.e. the reconstruction based on the same number of latent dimensions will result in different quality levels for different materials/BTFs. Instead, a separate analysis of a suitable trade-off choice of dimensions even further increases the computational effort to also address these materials. This demonstrates that the provided trade-off value is not an optimal choice in general, despite the high computational burden.

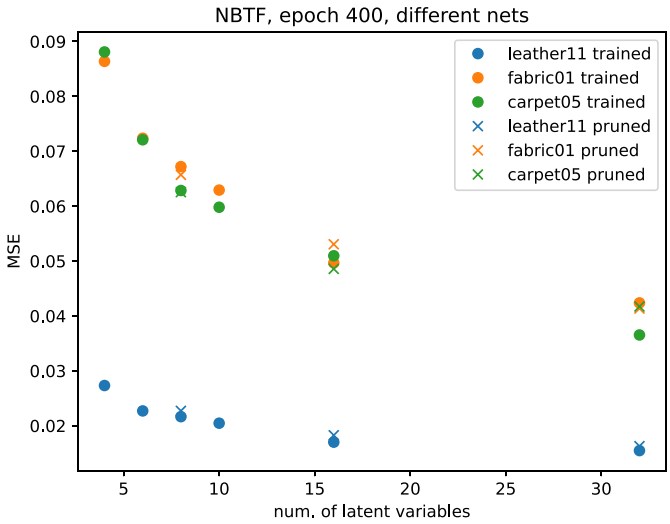

Fig. 1. MSE errors observed for different BTFs when training with different numbers of latent variables from scratch (points) vs. when starting training with 64 dimensions and pruning after 200 epochs to a different number of (most important) latent variables (crosses). We observe that both methods converge to nearly the same results. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

In the scope of our experiments, as suggested by Rainer et al. [1], we applied a log transform as well as a whitening to the input ABRDFs and used 400 epochs for the full training. On one GPU, one full training took approximately 4 h. We used DASP to approximate Shapley values after training for the first 200 epochs. In order to calculate the Shapley values of the latent dimensions according to the coalition function, which is the decoder function d in our case, we need to generate some latent vector examples $Z = \{\mathbf{z}_1, \ldots, \mathbf{z}_m\}$. For this we randomly sampled 2000 pairs of view and light directions (out of the 151×151 view-light sampling) for 100 randomly pixels (within the 100×100 available ones), resulting in 200 000 latent vectors in total. The

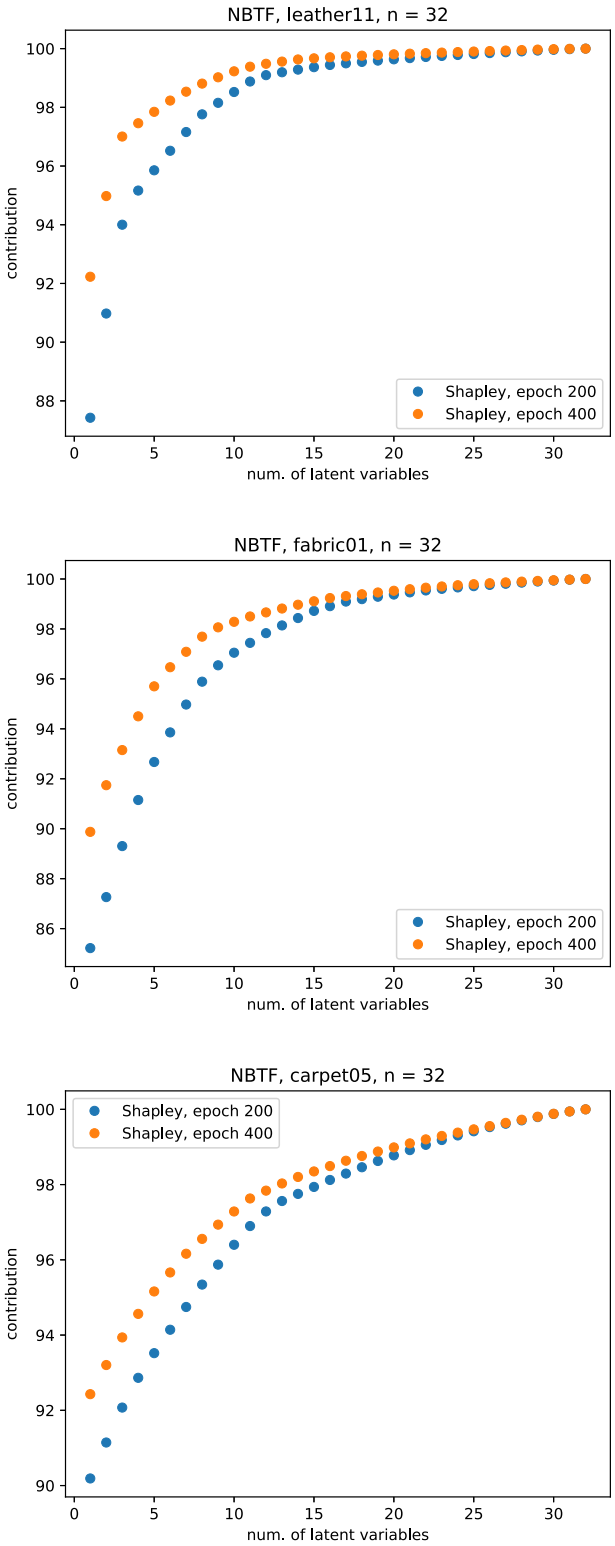

Fig. 2. Cumulative contributions observed when performing Shapley analysis after half of the training vs. after the full training for leather11 BTF (top), fabric01 BTF (middle) and carpet01 (bottom).

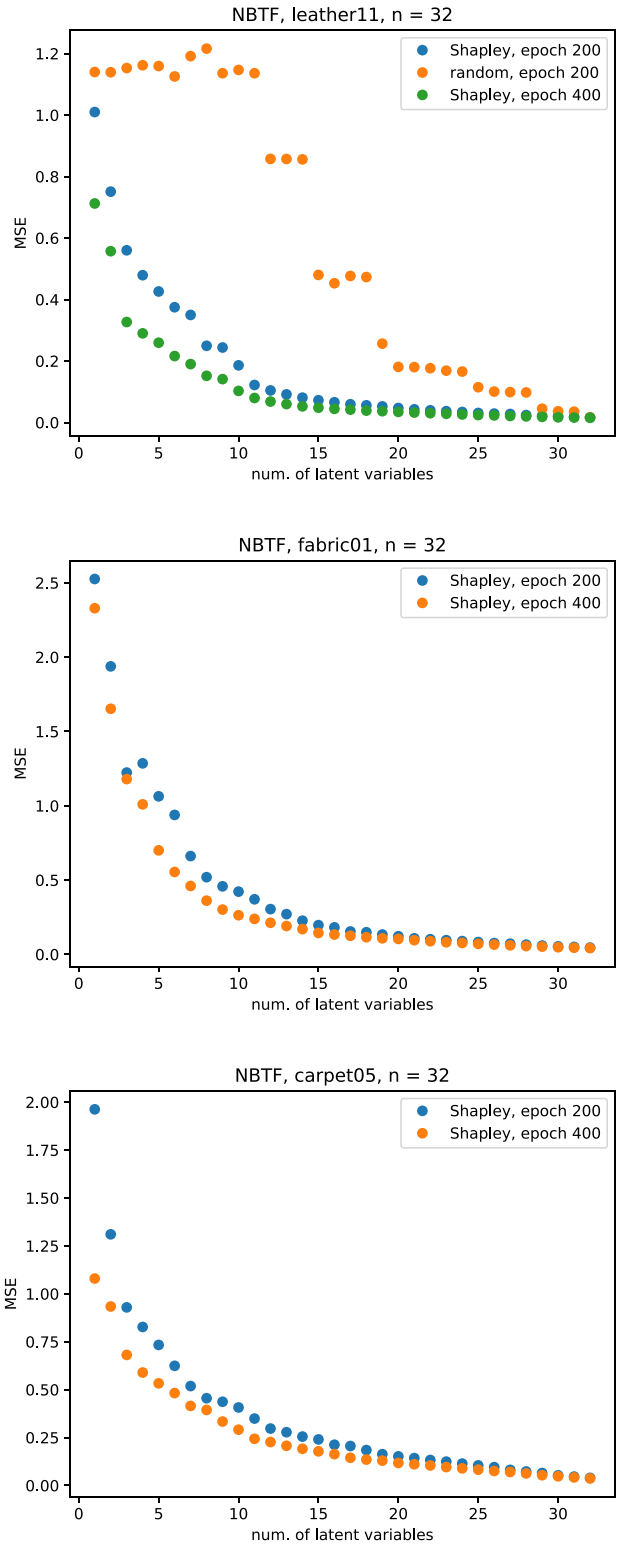

Fig. 3. Cumulative errors with respect to the full net, observed when performing Shapley analysis after half of the training vs. after the full training for leather11 BTF (top), fabric01 BTF (middle) and carpet01 (bottom).

Shapley value based analysis took about 8 min and Figs. 2 and 3 depicts the resulting contributions and MSE plots, whereas Figs. 4, 5 and Fig. 1 in supplemental provide respective visualizations. As a reference, we also show plots and visualizations obtained for a full training. In Table 1, we present a detailed analysis of the relationship between the number of latent variables and the time required to compute the Shapley values with the DASP method. In Fig. 2, we observe that different BTFs need different numbers of latent dimensions to achieve the same cumulative contribution percentage.

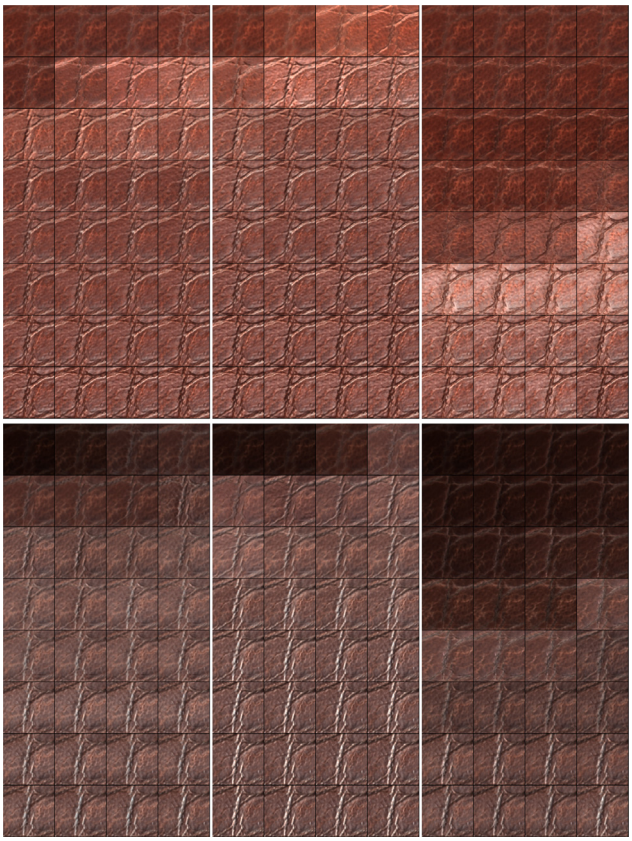

Fig. 4. Visualizations of the material appearance reconstructed based on different numbers of latent variables (for $h = 1, \ldots, 32$) for two different pairs of light and view directions ($L = 3, V = 3$ (top) and $L = 35, V = 100$ (bottom), see Fig. 8) for the material leather11. On the left we see the visualizations after 200 epochs, in the middle the visualizations after 400 epochs and on the right the visualization according to a random ordering after 200 epochs.

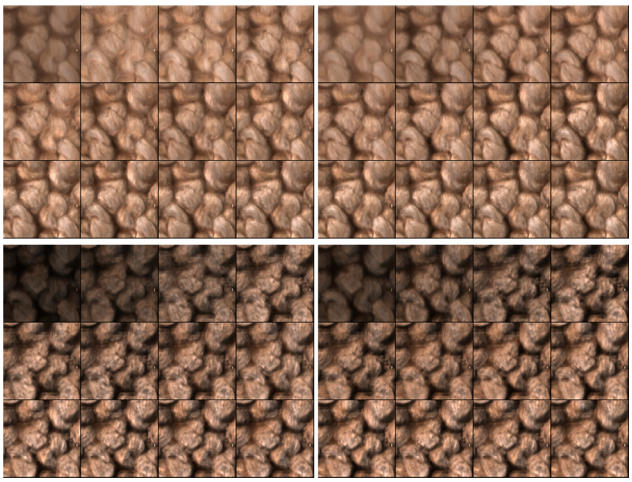

Fig. 5. Visualizations of the material appearance reconstructed based on different numbers of latent variables for two different pairs of light and view directions for carpet05. Top: $L = 3, V = 3$ and bottom $L = 35, V = 100$ (see Fig. 8). On the left we see the visualizations after 200 epochs, on the right the visualization after 400 epochs, each with the first $h = 1, 4, 7, 9, 10, 13, 16, 19, 22, 25, 28, 32$ latent variables according to the Shapley ordering.

For instance, taking 8 latent variables for the material leather11 results in capturing about 98% of the contribution of all latent variables. In contrast, matching this reconstruction quality

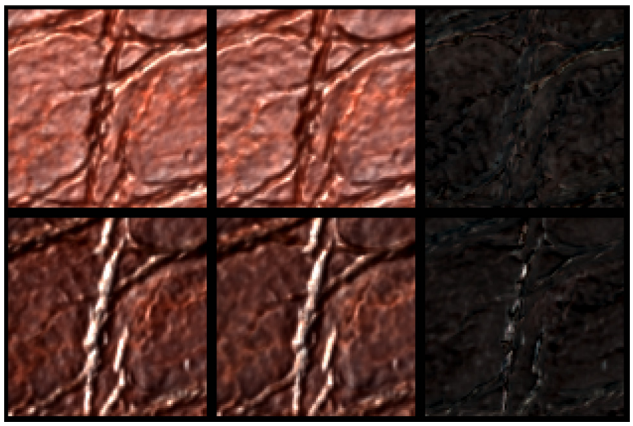

Fig. 6. Visualizations of the material appearance reconstructed based on different numbers of latent variables for two different pairs of light and view directions for leather11 after training with 32 latent variables (left) and 5 latent variables (right) and their difference. Top: $L = 3, V = 3$ and bottom $L = 35, V = 100$ (see Fig. 8).

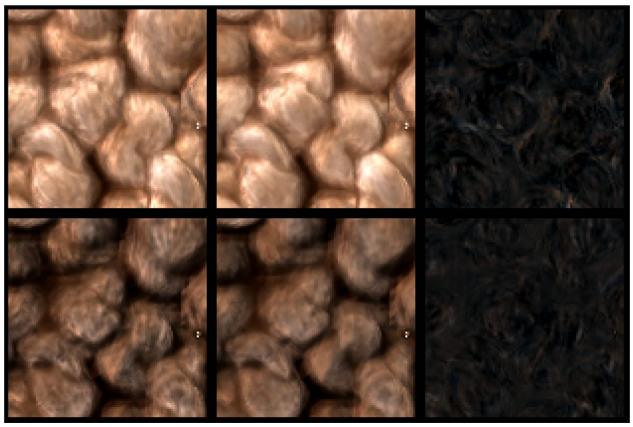

Fig. 7. Visualizations of the material appearance reconstructed based on different numbers of latent variables for two different pairs of light and view directions for carpet05 after training with 32 latent variables (left) and 9 latent variables (right) and their difference. Top: $L = 3, V = 3$ and bottom $L = 35, V = 100$ (see Fig. 8).

Table 1
Relation between the time (in minutes) required to compute the Shapley values and the number of latent variables. When the number of latent variables doubles, the time for the computation of Shapley values with DASP increases by a factor of about four.

	8	16	32	64	128
BTF	0,43	1,73	7,2	30,3	141,6
IC	2,5	10,5	41	171	695

in terms of 98% of the overall contributions requires a dimensionality of 12 for the latent space for the fabric01 BTF and a dimensionality of 17 for the latent space for the carpet05 BTF. Besides these significant variations of the cumulative contributions over the number of latent variables, we get evidence that the fixed choice of 8 latent dimensions independent of the considered material as used by Rainer et al. [1] can be suboptimal for different materials.

When analyzing the accumulated contributions (see Fig. 2), the MSE behavior depending on the number of used dimensions (see Fig. 3) as well as the corresponding visual depictions (see Figs. 4, 5 and Fig. 1 in supplemental), we observe that we can take the first ordered latent variables that contribute to 95%–96% of the overall reconstruction and continue training with

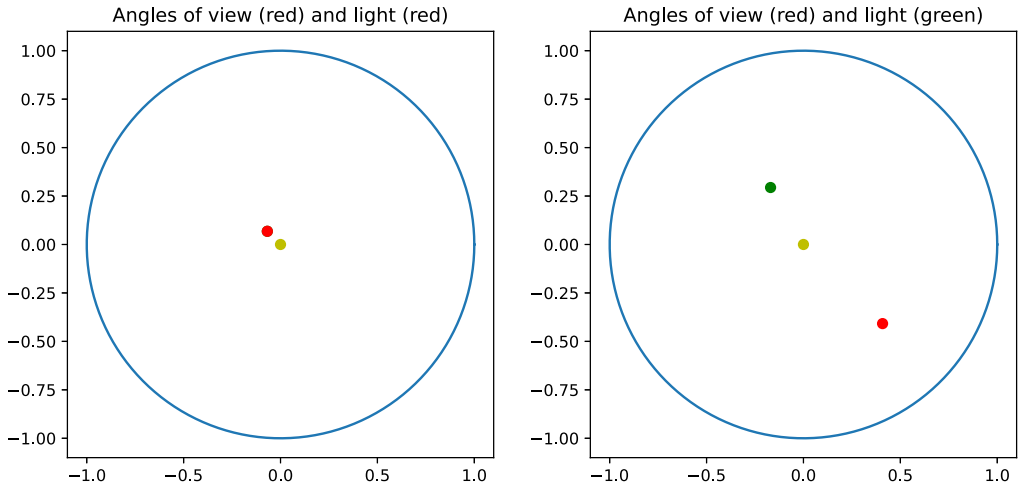

Fig. 8. Angles displayed on the unit disk: view direction V (green), light direction L (red). LE eft: $L = 3, V = 3$ and right: $L = 35, V = 100$. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

the dimensions 5, 7 and 9 for leather11, fabric01 and carpet05 respectively. Figs. 6, 7 and Fig. 2 in supplemental in show the visualizations of these pruned trainings in comparison to the full training with the initial 32 dimensions. Hence, we achieve a guidance of the compression that allows using different dimensionalities of the latent spaces to represent different materials depending on the complexity of their appearance characteristics which allows a more suitable reconstruction and compression than taking a single fixed number of dimensions for all materials as done by Rainer et al.. To analyze whether the ordering of latent variables according to their Shapley values is reasonable, we provide a comparison of the ordering of the latent dimensions according to their Shapley values to a random ordering (Figs. 3, 4). We observe that the random ordering does not allow insights regarding the choice of a plausible dimensionality of the latent space in contrast to our approach based on Shapley values.

4.2. Image compression

As already discussed in Section 2, there has been significant progress in neural image compression and in particular advanced autoencoder architectures have been demonstrated to be promising. The targeted trade-off of determining an as-compact-as-possible binary representation (i.e. lowest rate bitstream) while preserving a certain level of fidelity (i.e. minimum distortion) of the data has been investigated in terms of autoencoder architectures with quantization and entropy coding. One popular approach for image compression is the lossy compression approach based on compressive autoencoders by Theis et al. [55]. We used the publicly available implementation of this approach [https://github.com/alexandru-dinu/cae], which combines the proposed network architecture with the idea of compressing the code with stochastic binarization [54] instead of rounding it. The benefits of such stochastic binarization for image compression as described by Toderici et al. [54] also include the advantage that bit vectors are trivially serializable/deserializable which helps in efficient data transmission. The latent code then exhibits the form of a binary matrix, where the matrix dimensions directly correspond to the number of bits stored per image and the MSE loss is used for the optimization. Each image is pre-processed in terms of an arrangement of 10×6 non-overlapping patches of size 128×128 so that an image is represented based on 60 patches and, hence, based on 60 latent codes. For the subsequent processing, we took one of the models provided with the implementation with dimensions of $32 \times 32 \times 32$ (i.e. 32 channels of size 32×32 resulting from the used encoder architecture) and analyzed, whether and how much we can reduce the dimension of channels based on the Shapley value based analysis.

Furthermore, for our respective experiments, we used the YouTube-8M dataset that was taken from [https://research.google.com/youtube8m/] for training. Due to the missing information regarding a suitable explicit specification of the number of epochs during training by Theis et al. [55], we analyzed the behavior of training and validation losses and concluded 50 epochs to be a suitable choice for a complete training process. After training for 25 epochs, i.e. after half of the overall training, we perform an analysis of the latent space based on Shapley values. For this purpose, we used the code provided along the DASP approach [38] and extended their implementation of Lightweight Probabilistic Deep Networks [101] by the remaining probability layers required for the underlying architecture that have not been included in the DASP framework.

Even though using DASP for the approximation of the Shapley values is quite fast, it still depends on the number m of examples used for the computation and the time which is used to process a batch, i.e. forward pass of the decoder. So since the decoder of this network is a lot more complex than the one used for the BTF compression, we took less latent codes for Shapley computations for this application. When using sample sizes of six randomly selected patches from 21 randomly chosen images, i.e. 126 latent codes in total, the DASP computation took about 43 min, while the overall training took about 12 h, i.e. the total time is only moderately influenced by the DASP computation.

Figs. 9 and 10 show the contributions and MSE curves obtained by our approach, while Fig. 11 provides a depiction of qualitative results. We can observe that the first 25 ordered latent variables contribute to 95% of the reconstruction but after observing the error plot and some visualizations, we conclude that for this application taking 22 dimensions, which correspond to 92% of contribution, is still a reasonable choice for the subsequent training. Table 2 shows the test error we get if we continue training with different numbers of latent dimensions. Fig. 12 shows the results after the continued training with 22 latent dimensions vs. the full training with 32 channels. Furthermore, Figs. 9 and 10 provide the plots we obtain if we perform the Shapley analysis

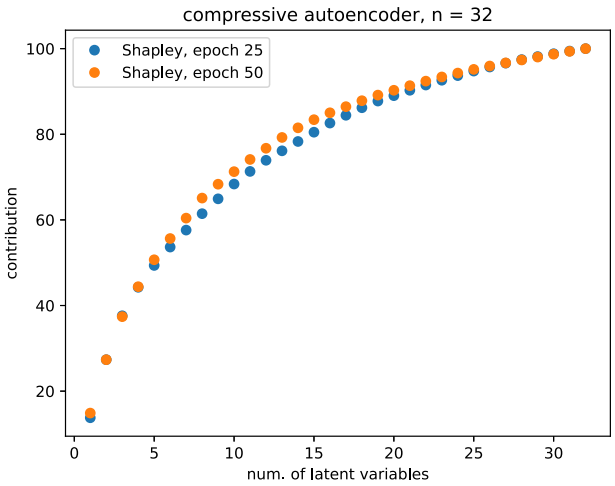

Fig. 9. Cumulative contributions observed when performing Shapley analysis after half of the training vs. after the full training for image compression.

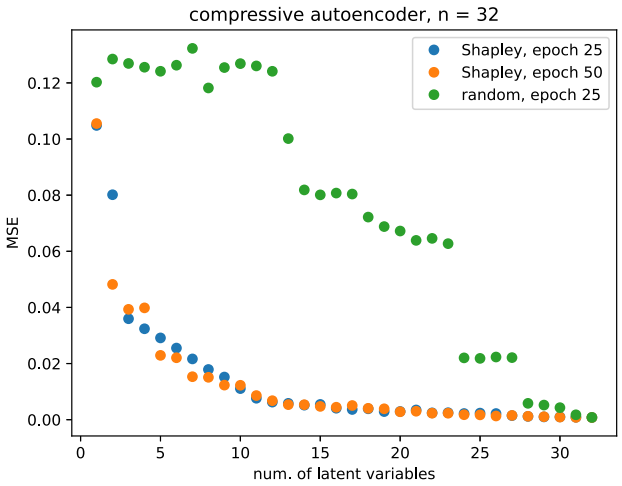

Fig. 10. Illustration of the reconstruction error depending on the number of latent variables that are used for the reconstruction. Note, how the error decays much quicker when variables are added in the order of their importance compared to a random selection of the same number of variables (blue vs. green plot). At the same time, there is no significant difference between applying the Shapley value based analysis after the full or after half of the training (blue vs. orange plot). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

after a full training. As demonstrated, the results do not significantly change already after 25 epochs, which indicates that applying the analysis regarding relevant dimensions to be used for the subsequent training based on Shapley values after half of the overall training epochs seems a reasonable choice.

To validate that the ordering of latent variables according to their Shapley values is a good choice, we again compared it to the random ordering. Fig. 10 depicts the error for the ordering of the dimensions according to their contribution as computed by the Shapley values in comparison to a random ordering and respective qualitative results are provided in Fig. 11.

Since the main effect of our method is the compression of the latent space, in order to compare to other methods, we studied the effect of alternatively compressing the latent space of the autoencoder based on PCA. Fig. 13 shows the error plot which results if we perform the PCA on the latent matrix and then reconstruct the latent matrix using different numbers p of principal components. From this plot and some image series like

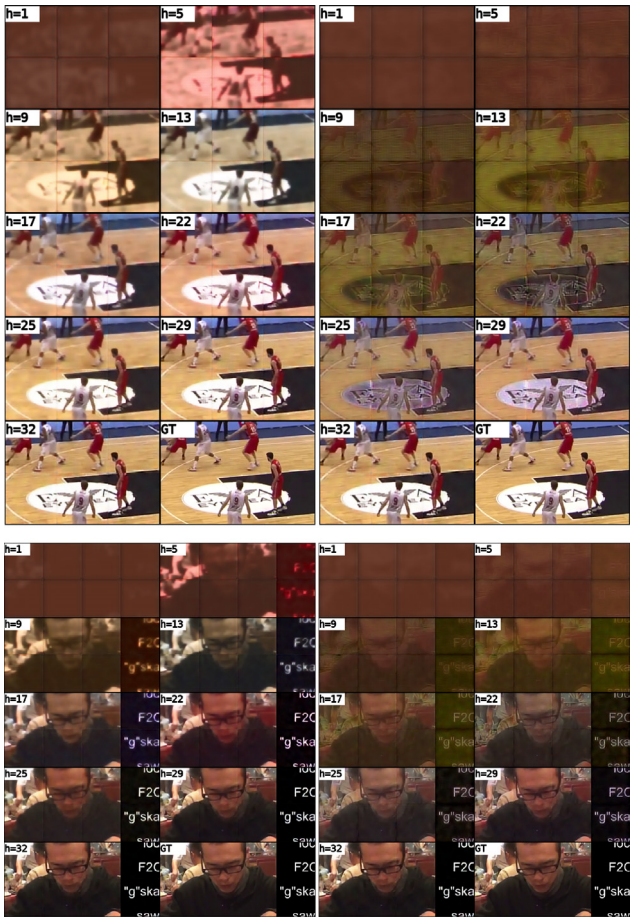

Fig. 11. Two different images reconstructed with different numbers of latent variables and applying the Shapley value based analysis after half of the training. Left: The latent variables are added in accordance to their estimated importance. Right: The latent variables are selected randomly. Note how the random selection leads to a significantly slower convergence. (The blocking artefacts result from the implementation of tiling the original image into patches and are not originating from our approach.)

Table 2
Correspondence between the estimated importance from the Shapley analysis to the final image reconstruction error. While Fig. 13 suggest, that 7 latent variables are sufficient for good quality (a-priori estimation), this analysis shows that more variables should be used (a-posteriori analysis).

% contribution	Dim of net	MSE	MSE increase compared to full net (in %)	MSE increase per pruned dimension compared to full net (in %)
58	7	0.00119	75	3
92	22	0.00075	10,3	1,3
95	25	0.00073	7,35	1,5
98	29	0.00070	3	1
100	32	0.00068	–	–

in 14 we observed that 7 principal components already suffice to reconstruct the image quite well. So if we link the principal components to the idea of the latent variables and conclude that we only need to train with 7 latent dimensions, we would see that PCA under-estimates the required number of latent dimensions. Fig. 12 shows the reconstructions obtained with the full network, with a pruned network with 22 channels resulting from our choice after the Shapley analysis and with a pruned network with 7 channels, as chosen after a PCA analysis. Moreover, Table 2 also reports the procentual increase of the error. We observe that

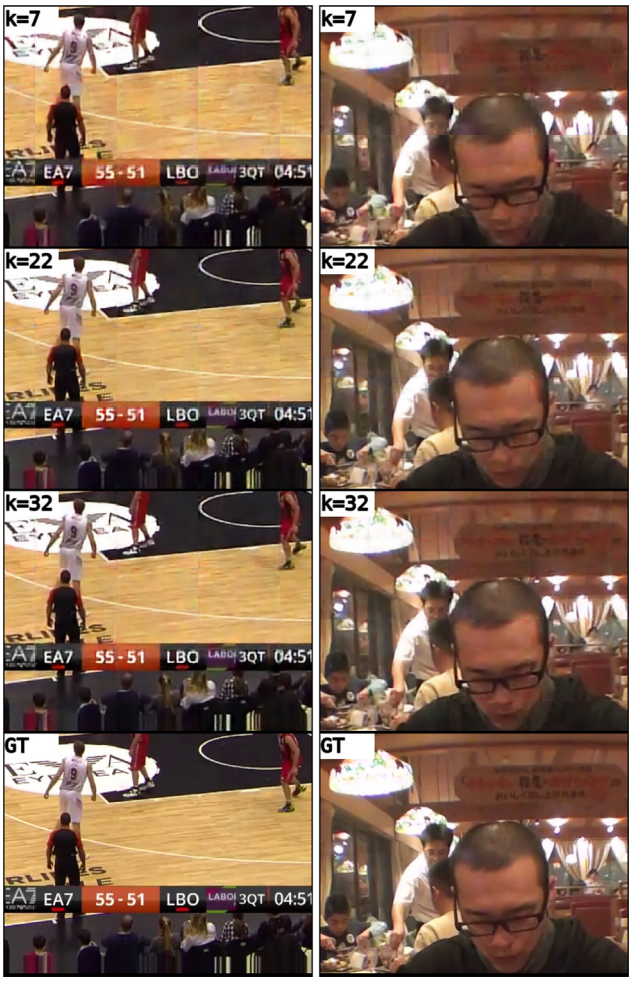

Fig. 12. Final image reconstructions achieved after the full training for different numbers of latent variables.

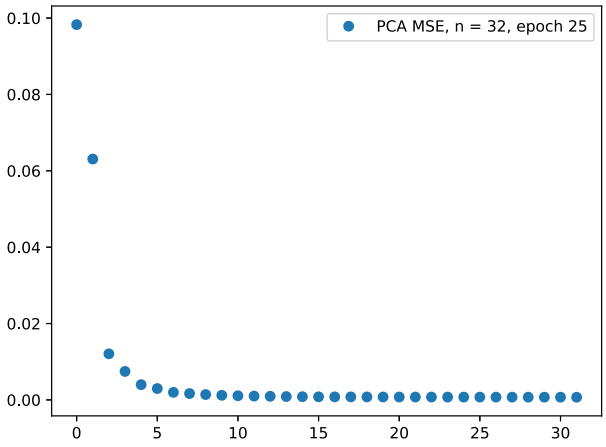

Fig. 13. Resulting error if we perform the PCA on the latent matrix and then reconstruct the latent matrix using different numbers of principal components. While this plot suggests that a low number of variables may be sufficient for the final reconstruction, we find that this does not translate to the number of latent variables required during the training.

with 7 dimensions the quality of the reconstruction is significantly reduced as opposed to what we expected when performing the reconstruction using the PCA. The explanation is that even though we can use only 7 components, the PCA was performed

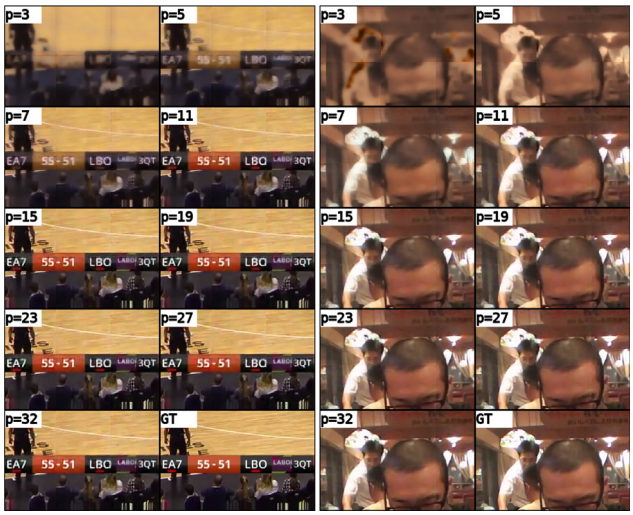

Fig. 14. Images corresponding to the PCA selection strategy shown in Fig. 13.

on the whole set of 32 dimensions which means that this full dimensionality is still indirectly included and not reduced as in the case of retraining the network with less dimensions.

Discussion. We experimented with leather11, fabric01 and carpet05 by training with different initial number of latent variables (32, 64 and 128), performing Shapley analysis in the middle of the training and observed that we get the same results if we prune the latent space of a model with 32 and 64 dimensions. If we prune a model with 128 dimensions, it tends to overestimate the number of latent dimensions (12 in case of leather, 15 in the case of fabric and 17 in the case of carpet). One way to overcome this, when choosing a very large number of dimensions beforehand, would be to perform the Shapley analysis again in the end of the training, prune again and continue training (in our case, training for another 100 epochs was sufficient) to converge to the same result as the one when starting with 64 or 32 dimensions.

Limitations. While DASP allows a better handling of larger numbers of dimensions, the computation of the Shapley values directly involves the decoder function. As a result, the computational burden increases with the complexity of the decoder structure, which results in increasing processing times.

5. Conclusions

We presented a novel approach for efficiently structuring the latent space for explainable data reconstruction and compression in a single training process. In particular, we have demonstrated that leveraging Shapley values to determine the contribution of the latent variables on the model's output which, in turn, allows organizing the latent variables according to a decreasing importance, discarding several latent variables at the same step and, finally, specifying a reasonable size of the latent codes. The truncation obtained when discarding latent variables after the first k latent variables with most importance results in an effect similar to a rank-k approximation as achieved when applying PCA in the linear case. We have demonstrated the relevance of this approach for compact representation and compression for images and high-dimensional material appearance.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Appendix A. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.gvc.2022.200059.

References

[1] Rainer G, Jakob W, Ghosh A, Weyrich T. Neural btf compression and interpolation. In: Computer graphics forum, Vol. 38. Wiley Online Library; 2019, p. 235–44.

[2] Rainer G, Ghosh A, Jakob W, Weyrich T. Unified neural encoding of btfs. In: Computer graphics forum, Vol. 39. Wiley Online Library; 2020, p. 167–78.

[3] Shapley LS. A value for n-person games. Contrib Theory Games 1953;2(28):307–17.

[4] Karl Pearson F. LIII. On lines and planes of closest fit to systems of points in space. Lond Edinb Dublin Philos Mag J Sci 1901;2(11):559–72. http://dx.doi.org/10.1080/14786440109462720, arXiv:https://doi.org/10.1080/14786440109462720.

[5] Hotelling H. Relations between two sets of variates. Biometrika 1936;28(3/4):321–77, URL: http://www.jstor.org/stable/2333955.

[6] Ladjal S, Newson A, Pham C-H. A PCA-like autoencoder. 2019, arXiv preprint arXiv:1904.01277.

[7] Pham C-H, Ladjal S, Newson A. PCAAE: Principal component analysis autoencoder for organising the latent space of generative networks. 2020, arXiv preprint arXiv:2006.07827.

[8] Doshi-Velez F, Kim B. Towards a rigorous science of interpretable machine learning. 2017, arXiv preprint arXiv:1702.08608.

[9] Lipton ZC. The mythos of model interpretability. In: ICML workshop on human interpretability in machine learning (WHI). 2016.

[10] Bach S, Binder A, Montavon G, Klauschen F, Müller K-R, Samek W. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. PLoS One 2015;10(7):e0130140.

[11] Shrikumar A, Greenside P, Kundaje A. Learning important features through propagating activation differences. In: International conference on machine learning. PMLR; 2017, p. 3145–53.

[12] Adebayo J, Gilmer J, Muelly M, Goodfellow I, Hardt M, Kim B. Sanity checks for saliency maps. Adv Neural Inf Process Syst 2018;31:9505–15.

[13] Kindermans P-J, Hooker S, Adebayo J, Alber M, Schütt KT, Dähne S, Erhan D, Kim B. The (un)reliability of saliency methods. In: Explainable AI: Interpreting, explaining and visualizing deep learning. Cham: Springer International Publishing; 2019, p. 267–80.

[14] Ghorbani A, Abid A, Zou J. Interpretation of neural networks is fragile. In: Proceedings of the AAAI conference on artificial intelligence, Vol. 33. 2019, p. 3681–8.

[15] Nie W, Zhang Y, Patel A. A theoretical explanation for perplexing behaviors of backpropagation-based visualizations. In: International conference on machine learning. PMLR; 2018, p. 3809–18.

[16] Sun Y, Sundararajan M. Axiomatic attribution for multilinear functions. In: Proceedings of the 12th ACM conference on electronic commerce. 2011, p. 177–8.

[17] Sundararajan M, Taly A, Yan Q. Axiomatic attribution for deep networks. In: International conference on machine learning. PMLR; 2017, p. 3319–28.

[18] Montavon G, Lapuschkin S, Binder A, Samek W, Müller K-R. Explaining nonlinear classification decisions with deep taylor decomposition. Pattern Recognit 2017;65:211–22.

[19] Lundberg SM, Lee S-I. A unified approach to interpreting model predictions. Adv Neural Inf Process Syst 2017;30:4765–74.

[20] Simonyan K, Vedaldi A, Zisserman A. Deep inside convolutional networks: Visualising image classification models and saliency maps. In: Bengio Y, LeCun Y, editors. 2nd international conference on learning representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, workshop track proceedings. 2014, URL: http://arxiv.org/abs/1312.6034.

[21] Shrikumar A, Greenside P, Kundaje A. Learning important features through propagating activation differences. In: Precup D, Teh YW, editors. Proceedings of the 34th international conference on machine learning. Proceedings of machine learning research, Vol. 70, International Convention Centre, Sydney, Australia: PMLR; 2017, p. 3145–53, URL: http://proceedings.mlr.press/v70/shrikumar17a.html.

[22] Ribeiro MT, Singh S, Guestrin C. "Why should i trust you?" Explaining the predictions of any classifier. In: Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining. 2016, p. 1135–44.

[23] Zintgraf LM, Cohen TS, Adel T, Welling M. Visualizing deep neural network decisions: Prediction difference analysis. 2017, CoRR, abs/1702.04595 arXiv:1702.04595 URL: http://arxiv.org/abs/1702.04595.

[24] Fong RC, Vedaldi A. Interpretable explanations of black boxes by meaningful perturbation. In: IEEE international conference on computer vision, ICCV 2017, Venice, Italy, October 22-29, 2017. IEEE Computer Society; 2017, p. 3449–57. http://dx.doi.org/10.1109/ICCV.2017.371.

[25] Matsui Y, Matsui T. NP-completeness for calculating power indices of weighted majority games. Theoret Comput Sci 2001;263(1):305–10. http://dx.doi.org/10.1016/S0304-3975(00)00251-6, URL: https://www.sciencedirect.com/science/article/pii/S0304397500002516, Combinatorics and Computer Science.

[26] Castro J, Gómez D, Tejada J. Polynomial calculation of the Shapley value based on sampling. Comput Oper Res 2009;36(5):1726–30.

[27] Strumbelj E, Kononenko I. An efficient explanation of individual classifications using game theory. J Mach Learn Res 2010;11:1–18.

[28] Maleki S, Tran-Thanh L, Hines G, Rahwan T, Rogers A. Bounding the estimation error of sampling-based Shapley value approximation. 2013, arXiv preprint arXiv:1306.4265.

[29] Datta A, Sen S, Zick Y. Algorithmic transparency via quantitative input influence: Theory and experiments with learning systems. In: 2016 IEEE symposium on security and privacy (SP). IEEE; 2016, p. 598–617.

[30] Tan S, Caruana R, Hooker G, Koch P, Gordo A. Learning global additive explanations for neural nets using model distillation. 2018, arXiv preprint arXiv:1801.08640.

[31] Nohara Y, Matsumoto K, Soejima H, Nakashima N. Explanation of machine learning models using improved Shapley Additive Explanation. In: Proceedings of the 10th ACM international conference on bioinformatics, computational biology and health informatics. 2019, p. 546.

[32] Aas K, Jullum M, Løland A. Explaining individual predictions when features are dependent: More accurate approximations to Shapley values. 2019, arXiv preprint arXiv:1903.10464.

[33] Sellereite N, Jullum M. Shapr: An R-package for explaining machine learning models with dependence-aware Shapley values. J Open Source Softw 2019;5(46):2027. http://dx.doi.org/10.21105/joss.02027.

[34] Bowen D, Ungar L. Generalized SHAP: Generating multiple types of explanations in machine learning. 2020, arXiv preprint arXiv:2006.07155.

[35] Chen H, Lundberg S, Lee S-I. Explaining models by propagating Shapley values of local components. In: Explainable AI in healthcare and medicine. Springer; 2021, p. 261–70.

[36] Lundberg SM, Erion G, Chen H, DeGrave A, Prutkin JM, Nair B, Katz R, Himmelfarb J, Bansal N, Lee S-I. Explainable AI for trees: From local explanations to global understanding. 2019, arXiv preprint arXiv:1905.04610.

[37] Fatima SS, Wooldridge M, Jennings NR. A linear approximation method for the Shapley value. Artificial Intelligence 2008;172(14):1673–99.

[38] Ancona M, Oztireli C, Gross M. Explaining deep neural networks with a polynomial time algorithm for shapley value approximation. In: International conference on machine learning. PMLR; 2019, p. 272–81.

[39] Giudici P, Raffinetti E. Shapley-Lorenz explainable artificial intelligence. Expert Syst Appl 2020;114104.

[40] Sundararajan M, Dhamdhere K, Agarwal A. The Shapley taylor interaction index. In: International conference on machine learning. PMLR; 2020, p. 9259–68.

[41] Chen J, Song L, Wainwright MJ, Jordan MI. L-Shapley and C-Shapley: Efficient model interpretation for structured data. In: International conference on learning representations. 2019.

[42] Kumar IE, Scheidegger C, Venkatasubramanian S, Friedler S. Shapley residuals: Quantifying the limits of the Shapley value for explanations. In: ICML workshop on workshop on human interpretability in machine learning (WHI). 2020.

[43] Lou Y, Caruana R, Gehrke J. Intelligible models for classification and regression. In: Proceedings of the 18th ACM SIGKDD international conference on knowledge discovery and data mining. 2012, p. 150–8.

[44] Lou Y, Caruana R, Gehrke J, Hooker G. Accurate intelligible models with pairwise interactions. In: Proceedings of the 19th ACM SIGKDD international conference on knowledge discovery and data mining. 2013, p. 623–31.

[45] Wang X, Chen H, Yan J, Nho K, Risacher SL, Saykin AJ, Shen L, Huang H, ADNI. Quantitative trait loci identification for brain endophenotypes via new additive model with random networks. Bioinformatics 2018;34(17):i866–74.

[46] Wang R, Wang X, Inouye DI. Shapley explanation networks. In: International conference on learning representations. 2021, URL: https://openreview.net/forum?id=vsU0efpivw.

[47] Mangalathu S, Hwang S-H, Jeon J-S. Failure mode and effects analysis of RC members based on machine-learning-based Shapley Additive explanations (SHAP) approach. Eng Struct 2020;219:110927.

[48] Tripathi S, Hemachandra N, Trivedi P. Interpretable feature subset selection: A Shapley value based approach,". In: Proceedings of 2020 IEEE international conference on big data, special session on explainable artificial intelligence in safety critical systems. 2020.

[49] Ghorbani A, Zou J. Data shapley: Equitable valuation of data for machine learning. In: International conference on machine learning. PMLR; 2019, p. 2242–51.
[50] Covert I, Lundberg S, Lee S-I. Explaining by removing: A unified framework for model explanation. 2020, arXiv preprint arXiv:2011.14878.
[51] Wang J, Wiens J, Lundberg S. Shapley flow: A graph-based approach to interpreting model predictions. 2020, arXiv preprint arXiv:2010.14592.
[52] Ghorbani A, Zou J. Neuron shapley: Discovering the responsible neurons. 2020, arXiv preprint arXiv:2002.09815.
[53] Ma S, Tourani R. Predictive and causal implications of using shapley value for model interpretation. In: Proceedings of the 2020 KDD workshop on causal discovery. PMLR; 2020, p. 23–38.
[54] Toderici G, O'Malley SM, Hwang SJ, Vincent D, Minnen D, Baluja S, Covell M, Sukthankar R. Variable rate image compression with recurrent neural networks. 2015, arXiv preprint arXiv:1511.06085.
[55] Theis L, Shi W, Cunningham A, Huszár F. Lossy image compression with compressive autoencoders. 2017, arXiv preprint arXiv:1703.00395.
[56] Ballé J, Laparra V, Simoncelli EP. End-to-end optimized image compression. In: 5th international conference on learning representations, ICLR 2017, Toulon, France, April 24-26, 2017, conference track proceedings. 2017.
[57] Cai C, Chen L, Zhang X, Gao Z. Efficient variable rate image compression with multi-scale decomposition network. IEEE Trans Circuits Syst Video Technol 2018;29(12):3687–700.
[58] Rippel O, Bourdev L. Real-time adaptive image compression. In: International conference on machine learning. PMLR; 2017, p. 2922–30.
[59] Nakanishi KM, Maeda S-i, Miyato T, Okanohara D. Neural multi-scale image compression. In: Asian conference on computer vision. Springer; 2018, p. 718–32.
[60] Ballé J, Laparra V, Simoncelli EP. Density modeling of images using a generalized normalization transformation. In: 4th international conference on learning representations, ICLR 2016. 2016.
[61] Ballé J, Laparra V, Simoncelli EP. End-to-end optimized image compression. 2016, arXiv preprint arXiv:1611.01704.
[62] Agustsson E, Mentzer F, Tschannen M, Cavigelli L, Timofte R, Benini L, Van Gool L. Soft-to-hard vector quantization for end-to-end learning compressible representations. 2017, arXiv preprint arXiv:1704.00648.
[63] Ballé J, Minnen D, Singh S, Hwang SJ, Johnston N. Variational image compression with a scale hyperprior. 2018, arXiv preprint arXiv:1802.01436.
[64] Mentzer F, Agustsson E, Tschannen M, Timofte R, Van Gool L. Conditional probability models for deep image compression. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2018, p. 4394–402.
[65] Lee J, Cho S, Beack S-K. Context-adaptive entropy model for end-to-end optimized image compression. 2018, arXiv preprint arXiv:1809.10452.
[66] Li M, Ma K, You J, Zhang D, Zuo W. Efficient and effective context-based convolutional entropy modeling for image compression. IEEE Trans Image Process 2020;29:5900–11.
[67] Minnen D, Ballé J, Toderici G. Joint autoregressive and hierarchical priors for learned image compression. 2018, arXiv preprint arXiv:1809.02736.
[68] Minnen D, Singh S. Channel-wise autoregressive entropy models for learned image compression. In: 2020 IEEE international conference on image processing (ICIP). IEEE; 2020, p. 3339–43.
[69] Tschannen M, Agustsson E, Lucic M. Deep generative models for distribution-preserving lossy compression. 2018, arXiv preprint arXiv:1805.11057.
[70] Agustsson E, Tschannen M, Mentzer F, Timofte R, Gool LV. Generative adversarial networks for extreme learned image compression. In: Proceedings of the IEEE/CVF international conference on computer vision. 2019, p. 221–31.
[71] Yang F, Herranz L, Van De Weijer J, Guitián JAI, López AM, Mozerov MG. Variable rate deep image compression with modulated autoencoder. IEEE Signal Process Lett 2020;27:331–5.
[72] Choi Y, El-Khamy M, Lee J. Variable rate deep image compression with a conditional autoencoder. In: Proceedings of the IEEE/CVF international conference on computer vision. 2019, p. 3146–54.
[73] Howard AG, Zhu M, Chen B, Kalenichenko D, Wang W, Weyand T, Andreetto M, Adam H. Mobilenets: Efficient convolutional neural networks for mobile vision applications. 2017, arXiv preprint arXiv:1704.04861.
[74] Sandler M, Howard A, Zhu M, Zhmoginov A, Chen L-C. Mobilenetv2: Inverted residuals and linear bottlenecks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. 2018, p. 4510–20.
[75] Tang Y, You S, Xu C, Han J, Qian C, Shi B, Xu C, Zhang C. Reborn filters: Pruning convolutional neural networks with limited data. In: Proceedings of the AAAI conference on artificial intelligence, Vol. 34. 2020, p. 5972–80.
[76] Khan A, Hines E. Integer-weight neural nets. Electron Lett 1994;30(15):1237–8.
[77] Rastegari M, Ordonez V, Redmon J, Farhadi A. Xnor-net: Imagenet classification using binary convolutional neural networks. In: European conference on computer vision. Springer; 2016, p. 525–42.
[78] Jacob B, Kligys S, Chen B, Zhu M, Tang M, Howard A, Adam H, Kalenichenko D. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In: Proceedings of the IEEE conference on computer vision and pattern recognition. 2018, p. 2704–13.
[79] Tan M, Chen B, Pang R, Vasudevan V, Sandler M, Howard A, Le QV. Mnasnet: Platform-aware neural architecture search for mobile. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2019, p. 2820–8.
[80] Yu J, Yang L, Xu N, Yang J, Huang T. Slimmable neural networks. 2018, arXiv preprint arXiv:1812.08928.
[81] Johnston N, Eban E, Gordon A, Ballé J. Computationally efficient neural image compression. 2019, arXiv preprint arXiv:1912.08771.
[82] Cai C, Chen L, Zhang X, Lu G, Gao Z. A novel deep progressive image compression framework. In: 2019 picture coding symposium (PCS). IEEE; 2019, p. 1–5.
[83] Yang F, Herranz L, Cheng Y, Mozerov MG. Slimmable compressive autoencoders for practical neural image compression. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2021, p. 4998–5007.
[84] Eckart C, Young G. The approximation of one matrix by another of lower rank. Psychometrika 1936;1(3):211–8.
[85] Jolliffe IT, Cadima J. Principal component analysis: a review and recent developments. Phil Trans R Soc A 2016;374(2065):20150202.
[86] De Lathauwer L, De Moor B, Vandewalle J. A multilinear singular value decomposition. SIAM J Matrix Anal Appl 2000;21(4):1253–78.
[87] De Lathauwer L, De Moor B, Vandewalle J. On the best rank-1 and rank-(r 1, r 2,..., rn) approximation of higher-order tensors. SIAM J Matrix Anal Appl 2000;21(4):1324–42.
[88] Carroll JD, Chang J-J. Analysis of individual differences in multidimensional scaling via an N-way generalization of "Eckart-Young" decomposition. Psychometrika 1970;35(3):283–319.
[89] Harshman RA, Lundy ME. PARAFAC: Parallel factor analysis. Comput Statist Data Anal 1994;18(1):39–72.
[90] Tucker LR. Some mathematical notes on three-mode factor analysis. Psychometrika 1966;31(3):279–311.
[91] Kolda TG, Bader BW. Tensor decompositions and applications. SIAM Rev 2009;51(3):455–500.
[92] Pajarola R, Suter SK, Ruiters R. Tensor approximation in visualization and computer graphics. In: Eurographics 2013 - tutorials, Vol. t6. Girona, Spain: Eurographics Association; 2013, http://dx.doi.org/10.2312/conf/eg2013/tutorials/t6, URL: http://diglib.eg.org/EG/DL/conf/EG2013/tutorials/t6.pdf.
[93] Bartholomew DJ, Knott M, Moustaki I. Latent variable models and factor analysis: a unified approach, Vol. 904. John Wiley & Sons; 2011.
[94] Tipping ME, Bishop CM. Probabilistic principal component analysis. J R Stat Soc Ser B Stat Methodol 1999;61(3):611–22.
[95] Yu S, Yu K, Tresp V, Kriegel H-P, Wu M. Supervised probabilistic principal component analysis. In: Proceedings of the 12th ACM SIGKDD international conference on knowledge discovery and data mining. 2006, p. 464–73.
[96] Guan Y, Dy J. Sparse probabilistic principal component analysis. In: Artificial intelligence and statistics. PMLR; 2009, p. 185–92.
[97] Rezende DJ, Mohamed S, Wierstra D. Stochastic backpropagation and approximate inference in deep generative models. In: International conference on machine learning. PMLR; 2014, p. 1278–86.
[98] Kingma DP, Welling M. Auto-encoding variational bayes. 2013, arXiv preprint arXiv:1312.6114.
[99] Weinmann M, Langguth F, Goesele M, Klein R. Advances in geometry and reflectance acquisition. In: Sousa A, Bouatouch K, editors. EG 2016 - tutorials. The Eurographics Association; 2016, http://dx.doi.org/10.2312/egt.20161032.
[100] Weinmann M, Gall J, Klein R. Material classification based on training data synthesized using a BTF database. In: European conference on computer vision. Springer; 2014, p. 156–71.
[101] Gast J, Roth S. Lightweight probabilistic deep networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. 2018, p. 3369–78.

Publication:
“Inverse Procedural Modeling of Knitwear”

Elena Trunz, Sebastian Merzbach, Jonathan Klein, Thomas Schulze,
Michael Weinmann, and Reinhard Klein

2019 IEEE/CVF Conference on Computer Vision and Pattern
Recognition (CVPR)

2019

DOI: 10.1109/CVPR.2019.00883

Inverse Procedural Modeling of Knitwear

Elena Trunz, Sebastian Merzbach, Jonathan Klein, Thomas Schulze
Michael Weinmann, Reinhard Klein
Institute of Computer Science II, University of Bonn, Germany
{trunz,merzbach,kleinj,mw,rk}@cs.uni-bonn.de, s6tsschu@uni-bonn.de

Abstract

The analysis and modeling of cloth has received a lot of attention in recent years. While recent approaches are focused on woven cloth, we present a novel practical approach for the inference of more complex knitwear structures as well as the respective knitting instructions from only a single image without attached annotations. Knitwear is produced by repeating instances of the same pattern, consisting of grid-like arrangements of a small set of basic stitch types. Our framework addresses the identification and localization of the occurring stitch types, which is challenging due to huge appearance variations. The resulting coarsely localized stitch types are used to infer the underlying grid structure as well as for the extraction of the knitting instruction of pattern repeats, taking into account principles of Gestalt theory. Finally, the derived instructions allow the reproduction of the knitting structures, either as renderings or by actual knitting, as demonstrated in several examples.

1. Introduction

Fabrics are an essential matter in our daily life. In contrast to their woven counterparts, knitted clothing visually sticks out due to complicated underlying stitch structures formed by various knitting operations, each inducing a characteristic appearance. Furthermore, knitting clothing of various types still belongs to a handcraft mastered by a rather large group of our society covering all ages. One reason for this may be the interest in manufacturing one's own clothing with knitting patterns following the individual subjective preferences. While there are books and websites that provide a wide range of patterns together with their respective construction instructions, it would be desirable to be able to reproduce patterns from images provided e.g. by standard search engines such as *Google*, which lack the corresponding knitting instruction.

Unfortunately, inferring the underlying knitting patterns by only "reading" a single image is particularly challenging, even for experts. The visual appearance of stitches exhibits a large variety as neighboring stitches may occlude them or cast shadows. The variability in the appearance of the basic stitch types is further increased by the properties of the used yarns, such as their thickness, type of yarns, etc., or the individual knitting styles of different people leading to various deformations, such as stretchings, holes and tight or loose stitches. Therefore, even experts often prefer analyzing the respective physical clothing pieces by stretching it and performing the inspection from both sides, in order to reliably infer the knitting instructions, including otherwise covered stitches. These manipulations are not possible when analyzing knitted fabrics in single photos.

Inverse procedural modeling of objects from only a few or even single examples has received a lot of attention in the last decade. Corresponding applications encompass the derivation of production rules for plants, woven fabrics, buildings and facades. However, the developed approaches are custom tailored for the corresponding applications and cannot be easily transformed to infer knitting patterns.

In this paper, we direct our attention to the inference of the complicated structures of knitwear and the derivation of the respective knitting instructions, to the best of our knowledge, for the first time from only a single image without annotations. This implies solving the labeling problem, i.e. the identification of the occurring stitch types as well as their proper localization from the visually complex appearance depicted in the input photographs. For this purpose, we introduce a novel pipeline that involves four major components represented by (1) the search for the individual stitch types across the image, (2) the inference of the underlying grid structure from the coarsely localized stitches from the previous step, (3) an error correction and pattern size detection step that determines the size of the desired pattern and corrects the labeling errors from the first step, and, finally, (4) the derivation of the final knitting instruction (in analogy to instructions in knitting books), based on the found pattern size and corrected underlying grid structure, taking into account the intuition of human perception by applying the Law of Symmetry and the Law of Prägnanz [5]. These

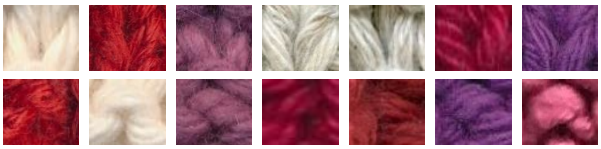

Figure 1: Exemplary appearance variations of knits (top) and purls (bottom).

derived instructions allow the reproduction of the knitting patterns with possibly different yarn types as demonstrated in several examples.

In summary, the key contributions of this paper are:

- A novel method for inverse procedural modeling of knitwear from a single image.
- The derivation of the underlying optimal regular grid structure from initially determined hypotheses regarding the coarse localization of stitch types.
- An error correction technique that determines the correct size of the knitting pattern and corrects possible recognition errors.
- A final induction of the knitting instruction from the derived grid structure following human intuition.

2. Background

Knitting relies only on a relatively small set of basic actions, and their combinations allow to generate the underlying knit structures for various patterns [34]. Therefore, even children can learn to produce caps, scarfs or potholders. These basic actions and their combinations result in a number of stitch types, which are used to generate a wide range of various knitting designs by repetitions and different orderings. In this paper, we focus on the two fundamental stitch types: *knit* and *purl* (see Figure 1).

The usual shape of a knit resembles the structure of a "v". If there is a purl stitch above or below (or both), a knit stitch becomes partially covered by the purl stitch(es). Additionally, the width of the stitch gets smaller than it would be if it had other knits as an upper and/or lower neighbor. The purl stitch usually resembles a wave structure. This wave becomes wider if there are knit stitches underneath and above the purl. However, in case there is a knit stitch on the right or the left side (or both) of a purl, then the purl stitch gets partially covered. Figure 1 illustrates some of the appearance/shape variations of purls and knits. Note that the shown stitch variations are solely induced by arranging the basic stitch types in different orderings. Additionally, these appearance variations heavily depend on the properties of the used yarn, as well as stitch deformations resulting from subjective knitting styles, making almost every stitch of a hand-made piece of knitted fabric individual.

3. Related Work

The major components of our framework for inverse modeling of knitwear include the search for occurrences of basic stitch types as marked by the user within the image, the inference of a grid structure based on the stitch candidates and underlying repeating patterns. As a consequence, we briefly review the developments in the areas of template matching and inverse procedural modeling. We refrain from a detailed discussion regarding the visualization of knitted fabrics, but only refer to the work by Yuksel et al. [34].

Template matching: Traditional techniques for efficiently searching a query patch within an image are usually based on using the Sum-of-Squared-Distances (SSD), the Sum-of-Absolute-Distances (SAD) or Normalized Cross-Correlation (NCC). Subsequent works addressed their lacking robustness towards handling noise [10] and illumination changes [13]. Further improvements came with the use of robust error functions [6, 22, 21, 18]. Later, Barnes et al. [2, 3] introduced the PatchMatch algorithm for nearest neighbor matching across translations, rotations and scales. However, all of these techniques only allow a one-to-one mapping between a template and the query region and rely on a strict rigid geometric deformation between the template patch and the target patch. As a consequence, they are not capable of dealing with the geometric deformations we expect for patches containing knitting primitives (knits and purls). Towards handling appearance variations for material recognition, other approaches rely on the matching of histograms extracted for different images by considering various descriptors (e.g. [20]) within classification frameworks. Furthermore, set-based matching has been explored to allow a more robust matching of textures based on the consideration of the appearance space of textures [14, 31].

Other approaches have been designed to explicitly handle parametric deformations such as 2D affine transformations [16] or more general non-rigid distortions [29]. However, despite the requirement of a parametric distortion model for the underlying geometry, these techniques also rely on the assumption of a one-to-one mapping between the query and target patch, which is susceptible to errors in the presence of occlusions or background clutter.

Further work explores the bi-directional similarity between target and query patch. Simakov et al. [23] represent images in terms of a set of patches and the considered bi-directional similarity (BDS) measure considers the sum of distances between a patch in the first image and its nearest neighbor in the second image and vice versa. To also distinguish between inliers and outliers arising from foreground/background parts of the considered patches, the Best Buddies Similarity (BBS) has been proposed [7], based on counting the Best Buddy Pairs and, hence, using the actual distances only implicitly. Therefore, an increased robustness in comparison to BDS has been

achieved. Talmi et al. [26] extended this work by enforcing diversity in the mutual nearest-neighbor matching and explicitly considering the deformation of the nearest-neighbor field. To achieve a speed up of the matching process, they use an approximate nearest neighbor search.

While any of the template matching techniques could be applied to derive probability maps for the localization of certain basic primitives required in our approach, we use template matching based on BBS due to its proven robustness to deformations that are expected to occur for the primitives of knitting. We improved the BBS technique by the use of additional gradient information. In the evaluation, we compare several template matching techniques and show that the extended BBS approach outperforms the other techniques in the context of our particular problem.

Image-based detection of weave patterns: Cloth modeling has received a lot of attention so far. Especially approaches for detecting weave patterns from images are closely related to our work. In particular, the complete reverse-engineering of woven cloth at the yarn level as approached by Schröder et al. [19] and Guarnera et al. [11] has been demonstrated to be the current state-of-the-art technique. While these approaches are powerful for the analysis of woven cloth, they are not designed to handle knitted textiles. Knitted clothing is inherently 3D and the final shapes of stitches, especially hand-made stitches, do not possess the similarity and regularity of warp and weft of woven cloth, where occlusions and non-rigid deformations of the yarn have to be taken into account in order to be able to find the actual position and the type of the stitches in the image. To the best of our knowledge, we are the first to tackle the problem of detecting knitting patterns and the respective knitting instructions from a single image.

Inverse procedural modeling: Inverse procedural modeling (IPM) is the problem of inferring a set of parameters [28, 30] or even a whole procedural description for a given model. Early investigations on applying inverse procedural modeling for graphics applications include the works on 3D meshes [4] and 2D vector designs [24], but there has been a lot of progress in this area of research. Meanwhile, inverse procedural modeling is widely used and has been applied for varying purposes ranging from the inference of 3D design patterns [27] over the modeling of plants [25, 17] to editing of building point clouds [8] as well as inferring procedural descriptions of building facades [32, 33, 9, 17] and reverse engineering of woven cloth [19]. For a detailed and extensive survey on inverse procedural modeling, we refer to the report by Aliaga et al. [1].

4. Stitch Pattern Inference Approach

In this section, we introduce our approach to infer knitting patterns and the respective instructions for their generation from single input images. An initial pre-processing step compensates for non-axis-alignment of the depicted knit patterns and, hence, makes our framework capable of handling tilted images of knitwear. In the next step, the user provides exemplars of particular basic stitch types, such as knits or purls within the image via an intuitive interface. Subsequently, image patches containing these stitch types are searched within the whole image and the resulting coarse localization of the individual stitch types is used to infer the underlying grid structure. Furthermore, an error correction procedure allows to compensate for possible misclassifications of the stitch types in the grid and detects the size of the repeating pattern. The found size and the optimized grid structure are then used to find the starting position of the pattern, thus finalizing the process of stitch pattern inference. Finally, we derive the underlying production rules and convert them into corresponding knitting instructions that allow the reproduction of the knitting pattern depicted in the input image. Details regarding the involved components are described in the following sections.

4.1. Pre-Processing

Before allowing the user to specify templates for the relevant stitch types in the image, we perform a pre-processing step to facilitate the annotation process. To compensate for deviations from axis-alignment, we use Histograms of Oriented Gradients (HOG) to determine the most dominant directions in the photo, which is justified due to the inherent grid structure resulting from the production process. This allows the reversal of rotations to align the grid structure with the axes and, hence, makes our algorithm capable of also handling non-axis-aligned input patterns. Respective examples are shown in the supplemental material.

4.2. Interactive Selection of Relevant Stitch Types

The detection of stitch types could possibly be approached with a completely automatic pipeline. However, this would require huge annotated databases depicting the possibly occurring stitch types with various stitch neighborhoods and distortions with yarns of different properties (yarn thickness, reflectance behavior, etc.) under different illumination conditions. As such databases, to the best of our knowledge, are not yet publicly available, we refrain from relying on a completely automatic approach to detect stitch types across the image based on machine learning techniques. Instead we let the user guide the search for stitch types by providing a single template for the individual stitch types occurring in the input image, in order to keep user interaction as minimal as possible. For this purpose, we implemented an easy-to-use interface that allows the user to choose a sample for each stitch type by simply drawing a rectangle over a stitch. In turn, considering the possibly strong variations of the occurring stitches requires a subsequently applied robust template matching technique.

4.3. Derivation of Stitch Localization Hypotheses

Finding certain stitch types in an image is complicated because their appearance may significantly vary due to partial occlusions by neighboring stitches, variances in the used yarn types including their reflectance behavior, thickness and hairiness, as well as variations induced by the individual knitting style during manufacturing, manifested in deformations like tight or loose stitches. To be able to find stitch types across the image based on a given template, handling distortions and partial matches becomes an essential prerequisite for the derivation of hypotheses of where the respective stitch types are found.

Best Buddies Similarity (BBS) based template matching [7] has been designed towards these goals of matching distorted and partially occluded patterns and proven to outperform most previous techniques in this regard. We therefore apply this technique for the detection of hypotheses for the individual stitch types such as knits or purls. Following Dekel et al. [7], the BBS between two point sets $P = \{p_i\}_{i=1}^{N}$ and $Q = \{q_i\}_{i=1}^{M}$ extracted from a local image region and a template is defined according to

$$BBS(P,Q) = \frac{1}{\min(M,N)} \sum_{i=1}^{N} \sum_{j=1}^{M} bb(p_i, q_j, P, Q). \quad (1)$$

Here,

$$bb(p_i, q_j, P, Q) = \begin{cases} 1, & \text{if } \mathrm{NN}(p_i, Q) = q_j \\ & \quad \text{and } \mathrm{NN}(q_j, P) = p_i \\ 0, & \text{otherwise} \end{cases} \quad (2)$$

acts as an indicator function influenced by the nearest neighbor definition

$$\mathrm{NN}(p_i, Q) = \mathrm{argmin}_{q \in Q} d(p_i, q) \quad (3)$$

and the distance measure

$$\begin{aligned} d(p,q) = \quad & ||p_i^{(A)} - q_j^{(A)}||_2^2 + \lambda_G ||p_i^{(G)} - q_j^{(G)}||_2^2 \\ & + \lambda_L \, ||p_i^{(L)} - q_j^{(L)}||_2^2. \end{aligned} \quad (4)$$

In comparison to the original implementation [7], we extend the RGB-based appearance (A) and spatial distance (L) within the patch with an additional gradient constraint (G), that enforces similar gradients within the patches. Based on several examples, we determined $\lambda_G = 100$ to be suitable for our purpose, and otherwise follow the original implementation in using $\lambda_L = 2$ and a decomposition of image and template into $k \times k$ patches with $k = 3$.

As a result, we obtain BBS likelihood maps that indicate where the respective stitch types are (coarsely) localized. Finally, we merge the likelihood maps obtained for the different stitch types to a resulting likelihood map that contains the maximum likelihood of the individual stitch types obtained per pixel as well as the corresponding most likely stitch type. An example of these maps is shown in Figure 2.

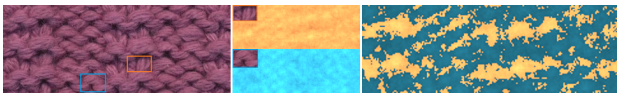

Figure 2: User-specified stitch templates (left) and corresponding likelihood maps (middle). The likelihood value is indicated with the colorization, i.e. the lighter the spot the higher the probability. The image on the right depicts the maximum likelihood map including the assignments to the stitch types (knit = orange, purl = blue).

4.4. Inference of Grid Structure of Stitches

The maximum likelihood map retrieved in the last step contains the coarse per-pixel likelihood regarding the local presence of respective stitch types. From this coarse localization we need to determine the fine-grained arrangement and the corresponding classes of the individual stitches. For this purpose, we exploit the presence of an underlying grid-like structure induced by the knitting process to account for the fact that the spatial extension of the individual stitches constrains their locations. In general, the latter will not be equidistant and the grid may exhibit significant distortions due to the non-ideal man-made manufacturing process or the respective treatment of the fabric. To model this behavior, we associate the centers of the stitches with a set of labeled points arranged in a 2D grid-like structure. These points have to fulfill the following properties:

- Each point is assigned a high likelihood of representing a certain stitch type such as knits or purls (P1).
- Neighboring points must preserve a minimal distance (on the order of magnitude of a stitch prototype) to each other (P2).
- Adjacent points cannot be further apart than the maximal extension of a stitch type (P3).
- The set of points has the structure of a regular approximately rectangular grid (P4).

Finding the optimal set of points fulfilling the above stated properties can then be formulated in terms of a point selection problem.

4.4.1 Stitch Localization as Point Selection Problem

To infer the positions of the centers of the individual stitches, we solve a point selection problem that can be formulated in terms of an integer linear program (ILP). Let P denote the set of all possible points (pixels) of the input image. Furthermore, let P_{opt} be the point set corresponding to the solution of our optimization problem. Denoting the

likelihood value of a pixel $p_i \in P$ to be assigned to a certain stitch type according to the likelihood map from the previous step with $score(i)$ and using the binary variables

$$o_i = \begin{cases} 0 & \text{if } p_i \notin P_{opt} \\ 1 & \text{if } p_i \in P_{opt} \end{cases} \tag{5}$$

that determine whether a point p_i is assigned to the optimal solution, we maximize the functional

$$\sum_{p_i \in P} score(i)\, o_i \tag{6}$$

subject to the constraints following from the aforementioned properties P2, P3 and P4. To ensure the properties P2 and P3, we determine for each pixel $p_j \in P$ two corresponding rectangular regions $P_j^{min} \subset P$ and $P_j^{max} \subset P$ that represent the uncertainty in the location of neighboring points in the grid structure . P_j^{min} has the width w_{min} and the height h_{min} of the estimated minimal extension of the stitch and P_j^{max} has the width w_{max} and the height h_{max} of the estimated maximal extension. The computation of the corresponding extension size values and the uncertainties is discussed in Section 4.4.3. To account for the property P2, we constrain each region P_j^{min} to contain *at most* one optimal point $p \in P_{opt}$ and, to account for the property P3, each region P_j^{max} is constrained to have *at least* one point $p \in P_{opt}$, i.e.:

$$\sum_{\forall i: p_i \in P_j^{min}} o_i \leq 1 \quad \text{and} \quad \sum_{\forall i: p_i \in P_j^{max}} o_i \geq 1 \quad \forall p_j \in P. \tag{7}$$

In order to force the points of the optimal solution to have a grid-like structure (P4), we subdivide the input image into $r \cdot c$ grid cells $G_k \subset P$ with the width $\frac{w_i}{c}(1 - u_w)$ and the height $\frac{h_i}{r}(1 - u_h)$, where w_i and h_i denote the width and the height of the image respectively and c and r denote the number of rows and columns of the grid. These are precomputed, as described in Section 4.4.2. With u_w and u_h, we denote the uncertainties in the spatial extension of the stitches in x and y direction respectively, which are used here to allow the overlap of the cells. The values of u_w and u_h are computed as described in Section 4.4.3. We constrain the optimal solution to contain at least one point in each grid cell. Furthermore, the number of points in the solution is constrained to be equal to $r \cdot c$. Both constraints ensure (P4) while allowing for overlapping cells:

$$\sum_{\forall i: p_i \in G_j} o_i \geq 1 \qquad \forall G_j \in G, \tag{8}$$

$$\sum_{\forall i: p_i \in P} o_i = r \cdot c. \tag{9}$$

G denotes the set of all grid cells. To solve this ILP, we use the Gurobi solver [12]. From the points contained in the resulting optimal solution we construct the grid in the following manner: We sort all points according to the x coordinates of the pixels and assign to each row of the grid c points. The stitch types assigned to the individual points are stored in a matrix $M^{r \times c}$.

4.4.2 Computing the Number of Rows and Columns

To determine the number of columns in the grid, we use the position of one of the stitch samples selected by the user during the initial step of the inference approach and select the region around the stitch position within the likelihood map. The height of the selected region corresponds to the height of the selected template with some additional tolerance and the respective width is given by the image width. To account for possible distortions, the height is allowed to deviate up to u_y in each direction from the center of the chosen stitch sample (in our experiments, we use $u_y = 25\%$). Using the data of this truncated map, we apply a similar ILP formulation as before with slight changes. In contrast to the optimization described before, the point variables consist of the pixels from the chosen strip. The objective functional and all constraints except for the row and the column number constraints remain unmodified. The number r of rows is set to 1. Now we compute the possible minimal value of the number of columns c as $c_{min} = \frac{w_i}{W_{max}}$, where W_{max} denotes the maximal width of both templates, since we do not yet know the correct stitch labelings of this strip. To account for possible stitch occlusions, we compute the maximal value of c as $c_{max} = 2\frac{w_i}{W_{min}}$, where W_{min} denotes the minimal width of both templates. We iterate through the possible numbers of columns from c_{min} to c_{max} and divide the resulting objective function of each optimal solution by the current number of columns. Finally, we obtain the number c of columns corresponding to the largest value of the normalized objective function as the optimal solution. The number of rows is determined accordingly.

4.4.3 Uncertainty in the Locations of Adjacent Stitches

Because of occlusions and different deformations the stitches vary from each other in size. Additional variations are induced by the use of different yarn types and inconsistencies of the knitter. We implicitly take these aspects into account by analyzing the strips extracted from the previous step to estimate uncertainties in the spatial extensions of the stitches. First, we compute the average width w_a and height h_a of the stitches taken from the four strips (two for each sample). By computing the maximum absolute deviation for the width (d_w) and height (d_h) separately, we get the uncertainties $u_w = \frac{d_w}{w_a}$ and $u_h = \frac{d_h}{h_a}$, yielding the values $w_{min} = w_a - u_w$ and $w_{max} = w_a + u_w$ for the width and analogous values h_{min} and h_{max} for the height, which are then used for the optimization.

For the computation of the number of rows and columns we use the average of the actual sizes of the templates selected by the user and set the values of uncertainties u_x and u_y to be equal to 25% of the average template size each. Note that large uncertainty values result in an increasing number of variables during the optimization, hence significantly increasing computational time.

4.5. Error Correction and Repeat Size Detection

After the inference of the underlying grid structure M from the previous step, we aim at finding an intuitive repeating pattern of minimal size. For this purpose, we assume that the knitting pattern of interest is at least twice contained completely within the image, but unfinished repeats may occur as well. In order to find the pattern, we first find the correct size and subsequently determine the starting position of the pattern. As the extracted matrix M of stitch types resulting from the grid optimization step might still contain some wrongly recognized stitch types, the identification of the pattern size in the matrix M as well as the potentially required error correction have to be conducted simultaneously. While the size of the repeating structure may be derived from the matrix M without labeling errors of the stitch types using region growing procedures as proposed by Wu et al. [33], the occurrence of errors in M instead forces us to perform an exhaustive search over all possible repeat sizes and to compute an error score for each possible repeat size. Finally, the size with the least error score is assumed to be the correct one.

Let r and c denote the number of the rows and columns of a possible repeat. Assuming the presence of at least two occurrences of the pattern in the image, we only consider repeat sizes $s = (r, c)$ that satisfy at least one of conditions $r \leq \frac{R}{2}$ and $c \leq \frac{C}{2}$. In more detail, we fully partition M into a set of non-overlapping submatrices M^s for each possible repeat size $s = (r, c)$, where each $M_i^s \in M^s$ has the size s (or smaller if depicting an unfinished repeat on a boundary). The partitions are evaluated at different positions (m, n) of M with $m = r\ k$ and $n = c\ h$ with $k = 1, \ldots, \lfloor \frac{R}{r} \rfloor$ and $h = 1, \ldots, \lfloor \frac{C}{c} \rfloor$, respectively. Subsequently, we align all $M_i^s \in M^s$ according to their indices and compute the matrix M_{max}^s, which contains the stitch type with the maximal occurrence for each equal index of the submatrices. Then, we compute the Hamming distance D_i between each M_i^s and M_{max}^s. The sum of all Hamming distances yields the overall distance D_s of the current repeat size s.

We consider the size and the underlying stitch type matrix M_{max}^s with the minimal distance as the resulting pattern size. If there are several sizes with the same edit distance, we take the one with the smallest value $r + c$, since otherwise there is evidence for having another repeat withing the repeat. In the case that we cannot determine the type with maximal occurrence for an index position due to an equal number of the stitch types at this position, we compare the corresponding likelihood values of the pixels from which these types were derived to determine the final type.

If the distance of the resulting optimal repeat deviates from zero, errors have been detected in the matrix M. In this case, the corresponding M_{max}^s is determined to have the correct labelings of the stitches and all the submatrices are corrected according to M_{max}^s.

4.6. Repeat Position Determination

Finally, the localization of an as intuitive as possible repeating pattern from the underlying grid structure and the size of the repeating pattern has to be computed. In order to select an intuitive pattern repeat, we take inspiration from human perception and make use of two of the basic laws in Gestalt theory [5]. The Law of Symmetry states that symmetrical elements tend to be perceived as a unified group. Taking this into consideration, we search for symmetry along the x-direction of the pattern. If there is a symmetry, we take it into account when selecting the starting position of the repeat. If there is no symmetry in the structure of the repeat, we apply the Law of Prägnanz. According to this law, humans prefer simpler and ordered states that require less cognitive effort and, hence, can be faster processed than complex structures that, in turn, might have to be reorganized or even further decomposed. In our case, this corresponds to selecting the starting position of the pattern from all the possible positions that leads to the least amount of changes from one type of stitch to another when computing the sum of the type changes from each two adjacent rows and columns of the pattern in question. This ensures that individual structures such as squares or circles appearing in the pattern will not be broken.

5. Results and Discussion

Sample selection: In order to test our approach, we have chosen 25 photos and scans that depict knitting samples with different patterns and were produced with yarns of various types and colors. Eight of the photos were taken from the internet, one photo depicts a machine knitted piece and sixteen photos depict hand-made knitting fabrics. The focus on hand-made samples results from the fact that these exhibit a higher degree of variation and, hence, are more challenging than machine-knitted samples.

Performance analysis: Table 1 provides an overview over the computation times as well as the problem sizes for the four examples selected for this paper. More examples with corresponding running times are shown in the supplemental material. Since the most time-consuming operations were computations of the similarity maps with BBS and solving for the optimum with the Gurobi solver [12], we report the computation times only for these steps. The other steps required only a negligible amount of time. All com-

putations were performed with an unoptimized implementation on an Intel(R) Core(TM) i7-5820K CPU with 3.30 GHz.

Table 1: The columns contain the image size (IS), the runtimes (in seconds) of BBS and ILP, as well as the size of both the grid (GS) and the pattern (PS). e_G and e_P denote the fraction of misclassified stitch types for the overall grid and the pattern after error correction.

ID	IS	BBS	ILP	GS	PS	e_G	e_P
1	673× 257	102.53	11.02	9×5	4×2	$1/45$	0
2	690×370	31.93	8.13	15×11	7×6	$1/165$	0
3	803×844	219.79	62.02	11×17	8×8	$4/187$	0
4	516×347	137.74	4.78	7×6	3×4	0	0

Visual quality: Figure 3 demonstrates the results of the individual steps of the pipeline for four of the example textiles. For the example in the second row one stitch in the input image (the sixth stitch from the left in the bottom row) has actually been wrongly knitted (knit instead of purl). This error was recognized and corrected by our method. To obtain the shown realistic renderings (last column), we synthesize yarns using the procedural model of Zhao et al. [35]. We then deform the yarn according to the discovered knitting instructions and discretize the resulting fiber geometry into a voxel grid, storing averaged densities and fiber orientations per voxel [15]. This voxel-based representation is then rendered using a volumetric path tracer [15]. Furthermore, Figure 4 shows results for the inferred grid structure obtained when applying our method on worn clothing.

Susceptibility to template selection: To evaluate the robustness regarding the selection of templates for the individual stitch types, we performed a study where 10 people aged from 10 to 67 years were asked to provide respective annotations. The results do not exhibit significant differences, as long as the testers follow the simple instruction of selecting two templates, which look similar to other stitches of the same type (Figure 1) (see supplemental material).

Table 2: Performance comparison of several template matching techniques: The rows contain the fractions of misclassified pixels e_{Px}, stitch types for the overall grid e_G and the pattern after error correction e_P. For the computation of the first measure we excluded pixels within a small band at the transitions between different stitch types. In 32% of the tests, the optimization based on the SAD likelihood maps did not succeed in detecting the correct number of rows and/or columns of the grid. These cases are excluded from the reported values e_G and e_P for SAD.

	SAD	NCC	DDIS	BBS	BBSg
e_{Px}	0.422	0.453	0.216	0.342	**0.194**
e_G	0.633	0.087	0.062	0.071	**0.040**
e_P	0.268	0.056	0.051	0.046	**0.029**

Suitability of different template matching schemes: We evaluated the suitability of different template matching schemes for the generation of likelihood maps for the individual stitch types. For this purpose, we compare our extended version of the BBS technique with additional gradient information (BBSg) to the normalized cross-correlation (NCC), the sum of absolute differences (SAD), the original BBS approach [7] without the proposed extension and the deformable diverse similarity (DDIS) approach [26]. Table 2 summarizes the respective results. In order to achieve meaningful results, the resolution of the input image is required to be sufficient so that the minimal template size is not smaller that 30×30 pixels.

Computational efficiency: In order to find the final center positions of stitches, we apply a global optimization that is formulated as an ILP problem. As ILP problems are known to be NP-hard, the computational times may be impractical. In order to speed-up the inference of optimal grids, which is particularly required for larger images, we downsize the corresponding likelihood maps by the factor of 0.5. The downsizing significantly decreases the computational time of the optimization, while still yielding similar results as without downsizing (for the evaluation of scaling we refer to the supplemental material).

Another possibility to decrease the computational time is to choose some iterative locally optimal approach instead of global optimization. For comparison, we use the likelihood and the stitch type assignments from the template matching step as the starting point for a greedy strategy to select neighboring stitch centers, where we also exploit uncertainty of the template sizes. In a first step, we take the maximum of the likelihood to find the the most likely location of a stitch and define a minimum distance within which no other stitch is allowed to occur depending on the template uncertainty. After discarding the respective area in the likelihood, we continue to search for the next highest likelihood, place a stitch center and again remove the region from the likelihood. This process is iterated until no further stitch center can be placed or the remaining likelihoods are lower than a certain threshold t (we used $t = 0.2$). As shown in Figure 3, this approach does not result in acceptable stitch center hypotheses, due to the iterative local optimization. Furthermore, this method does not compute the uncertainties automatically but requires their manual specification for each fabric sample individually. In contrast, our global optimization technique yields stitch center hypotheses at a higher quality.

Pattern search: In principle, once the size and correct labeling of the repeated pattern is found, one could reproduce the initial knitted example, since the knitting is done periodically. However, when knitting whole clothing pieces, the borders of the piece should be appealing. Hence, we need to identify the starting position of the correct or at

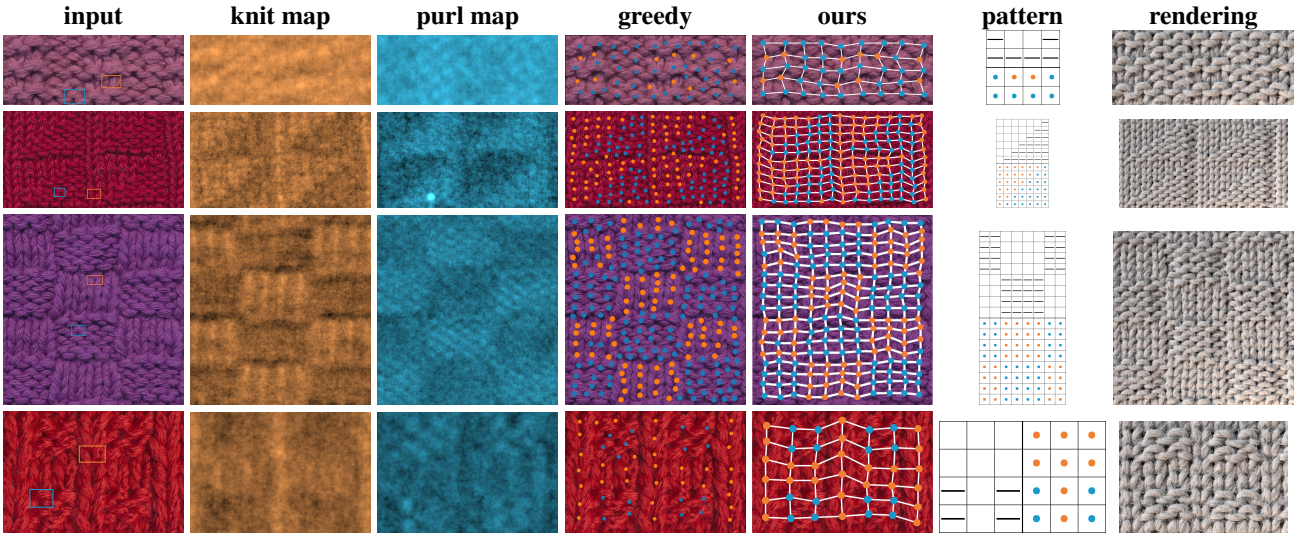

Figure 3: From left to right: input image, likelihoods for both stitch types, stitch center hypotheses derived via a greedy approach and grid structure inferred via our approach, corresponding knitting instruction (counting the rows from bottom to top, empty cells correspond to knits in odd rows and purls in even rows while cells containing bars correspond to purls in odd rows and knits in even rows) and rendering.

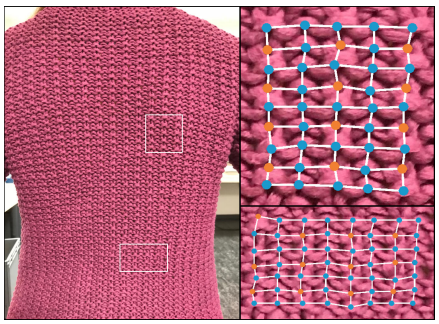

Figure 4: General, unrestricted setup of knitwear worn by a person (left) with detected grid structures for some regions of interest (right).

least of a nice pattern. In the supplementary material, we illustrate the problem of choosing an intuitive pattern. With our pattern search procedure we try to avoid breaking existing structures of the pattern, such as triangles or checkerboards, thereby following the Gestalt principles.

Limitations: In this paper, we limit our approach to the two fundamental stitch types: knit and purl. However, the number of stitch types is not strictly limited to two. In the supplemental material we also provide an example with three stitch types. However, including stitch types (e.g. holes) that deform the grid-like structure of the patten, requires including additional constraints, which we want to pursue in future work. Furthermore, if the input image is of low quality or contains almost completely occluded stitches, so that already the coarse localization does not yield meaningful results, the optimization technique will not produce the correct labeling.

6. Conclusion and Future Work

We have presented a novel practical framework for the inference of the complicated structures of knitwear as well as the corresponding knitting instructions from a single image. Templates for individual stitch types, as provided by the user, are roughly localized across the complete image and the resulting stitch positions are subsequently refined by optimizing the underlying grid structure within an integer linear program. The size of the repeating pattern is computed from the derived stitch labeling at the vertices of the resulting grid. Subsequently, we apply the Law of Symmetry and the Law of Prägnanz from Gestalt theory to find an intuitive pattern repeat and derive the corresponding knitting instruction. While our approach was demonstrated to allow the derivation of the knitting instructions for several different knitwears, there are still some open challenges to be addressed by future research. Including further stitch types into the framework as well as further reducing the degree of user interaction based on the combination of a large database of stitch types with their respective appearance variations and machine learning techniques is a promising avenue of research that we plan to pursue in future work.

References

[1] D. G. Aliaga, İ. Demir, B. Benes, and M. Wand. Inverse procedural modeling of 3D models for virtual worlds. In *ACM*

SIGGRAPH 2016 Courses, SIGGRAPH '16, pages 16:1–16:316, New York, NY, USA, 2016. ACM.

[2] C. Barnes, E. Shechtman, A. Finkelstein, and D. B. Goldman. PatchMatch: A randomized correspondence algorithm for structural image editing. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 28(3), Aug. 2009.

[3] C. Barnes, E. Shechtman, D. B. Goldman, and A. Finkelstein. The generalized PatchMatch correspondence algorithm. In *Proceedings of the 11th European Conference on Computer Vision Conference on Computer Vision: Part III*, ECCV'10, pages 29–43, Berlin, Heidelberg, 2010. Springer-Verlag.

[4] M. Bokeloh, M. Wand, and H.-P. Seidel. A connection between partial symmetry and inverse procedural modeling. *ACM Trans. Graph.*, 29(4):104:1–104:10, July 2010.

[5] S. Bradley. Design principles: Visual perception and the principles of Gestalt. https://www.smashingmagazine.com/2014/03/design-principles-visual-perception-and-the-principles-of-gestalt/, 2014.

[6] J.-H. Chen, C.-S. Chen, and Y.-S. Chen. Fast algorithm for robust template matching with M-estimators. *IEEE Transactions on Signal Processing*, 51(1):230–243, Jan 2003.

[7] T. Dekel, S. Oron, M. Rubinstein, S. Avidan, and W. T. Freeman. Best-buddies similarity for robust template matching. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2021–2029, June 2015.

[8] I. Demir, D. G. Aliaga, and B. Benes. Procedural editing of 3D building point clouds. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2147–2155, 2015.

[9] I. Demir, D. G. Aliaga, and B. Benes. Proceduralization for editing 3D architectural models. In *2016 Fourth International Conference on 3D Vision (3DV)*, pages 194–202, Oct 2016.

[10] E. Elboher and M. Werman. Asymmetric correlation: A noise robust similarity measure for template matching. *IEEE Transactions on Image Processing*, 22(8):3062–3073, Aug 2013.

[11] G. C. Guarnera, P. Hall, A. Chesnais, and M. Glencross. Woven fabric model creation from a single image. *ACM Trans. Graph.*, 36(5):165:1–165:13, Oct. 2017.

[12] Inc. Gurobi Optimization. Gurobi optimizer reference manual, 2016.

[13] Y. Hel-Or, H. Hel-Or, and E. David. Matching by tone mapping: Photometric invariant template matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(2):317–330, Feb 2014.

[14] D. P. Huttenlocher, G. A. Klanderman, and W. J. Rucklidge. Comparing images using the Hausdorff distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(9):850–863, Sep 1993.

[15] W. Jakob, A. Arbree, J. T. Moon, K. Bala, and S. Marschner. A radiative transfer framework for rendering materials with anisotropic structure. In *ACM Transactions on Graphics (TOG)*, volume 29, page 53. ACM, 2010.

[16] S. Korman, D. Reichman, G. Tsur, and S. Avidan. FasT-Match: Fast affine template matching. In *2013 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2331–2338, June 2013.

[17] S. Lienhard, C. Lau, P. Müller, P. Wonka, and M. Pauly. Design transformations for rule-based procedural modeling. *Comput. Graph. Forum*, 36(2):39–48, May 2017.

[18] O. Pele and M. Werman. Robust real-time pattern matching using bayesian sequential hypothesis testing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(8):1427–1443, Aug 2008.

[19] K. Schröder, A. Zinke, and R. Klein. Image-based reverse engineering and visual prototyping of woven cloth. *IEEE Transactions on Visualization and Computer Graphics*, 21:188–200, 2015.

[20] L. Sharan, C. Liu, R. Rosenholtz, and E. H. Adelson. Recognizing materials using perceptually inspired features. *International Journal of Computer Vision*, 103(3):348–371, Jul 2013.

[21] B. G. Shin, S.-Y. Park, and J. J. Lee. Fast and robust template matching algorithm in noisy image. In *2007 International Conference on Control, Automation and Systems*, pages 6–9, Oct 2007.

[22] A. Sibiryakov. Fast and high-performance template matching method. In *CVPR 2011*, pages 1417–1424, June 2011.

[23] D. Simakov, Y. Caspi, E. Shechtman, and M. Irani. Summarizing visual data using bidirectional similarity. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, June 2008.

[24] O. Stava, B. Benes, R. Mech, D. G. Aliaga, and P. Kristof. Inverse procedural modeling by automatic generation of L-systems. *Comput. Graph. Forum*, 29(2):665–674, 2010.

[25] O. Stava, S. Pirk, J. Kratt, B. Chen, R. Mźch, O. Deussen, and B. Benes. Inverse procedural modelling of trees. *Comput. Graph. Forum*, 33(6):118–131, Sept. 2014.

[26] I. Talmi, R. Mechrez, and L. Zelnik-Manor. Template matching with deformable diversity similarity. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1311–1319, 2017.

[27] J. Talton, L. Yang, R. Kumar, M. Lim, N. Goodman, and R. Měch. Learning design patterns with bayesian grammar induction. In *Proceedings of the 25th Annual ACM Symposium on User Interface Software and Technology*, UIST '12, pages 63–74, New York, NY, USA, 2012. ACM.

[28] J. O. Talton, Y. Lou, S. Lesser, J. Duke, R. Měch, and V. Koltun. Metropolis procedural modeling. *ACM Trans. Graph.*, 30(2):11:1–11:14, Apr. 2011.

[29] Y. Tian and Srinivasa G. Narasimhan. Globally optimal estimation of nonrigid image distortion. *International Journal of Computer Vision*, 98(3):279–302, Jul 2012.

[30] C. A. Vanegas, I. Garcia-Dorado, D. G. Aliaga, B. Benes, and P. Waddell. Inverse design of urban procedural models. *ACM Trans. Graph.*, 31(6):168:1–168:11, Nov. 2012.

[31] M. Weinmann and R. Klein. Material recognition for efficient acquisition of geometry and reflectance. In *Computer Vision - ECCV 2014 Workshops*, pages 321–333. Springer International Publishing, 2015.

[32] J. Weissenberg, H. Riemenschneider, M. Prasad, and L. Van Gool. Is there a procedural logic to architecture? In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 185–192. IEEE, 2013.

[33] F. Wu, D.-M. Yan, W. Dong, X. Zhang, and P. Wonka. Inverse procedural modeling of facade layouts. *ACM Trans. Graph.*, 33(4):121:1–121:10, July 2014.

[34] C. Yuksel, J. M. Kaldor, D. L. James, and S. Marschner. Stitch meshes for modeling knitted clothing with yarn-level detail. *ACM Trans. Graph.*, 31(4):37:1–37:12, July 2012.

[35] S. Zhao, F. Luan, and K. Bala. Fitting procedural yarn models for realistic cloth rendering. *ACM Transactions on Graphics (TOG)*, 35(4):51, 2016.