



Delft University of Technology

Reconstruction, Generation and Exploration of Digital Content with Deep Neural Features

Luo, X.

DOI

[10.4233/uuid:e74e07a8-397a-43fd-978b-c167da02a15a](https://doi.org/10.4233/uuid:e74e07a8-397a-43fd-978b-c167da02a15a)

Publication date

2025

Document Version

Final published version

Citation (APA)

Luo, X. (2025). *Reconstruction, Generation and Exploration of Digital Content with Deep Neural Features*. [Dissertation (TU Delft), Delft University of Technology]. <https://doi.org/10.4233/uuid:e74e07a8-397a-43fd-978b-c167da02a15a>

Important note

To cite this publication, please use the final published version (if applicable).
Please check the document version above.

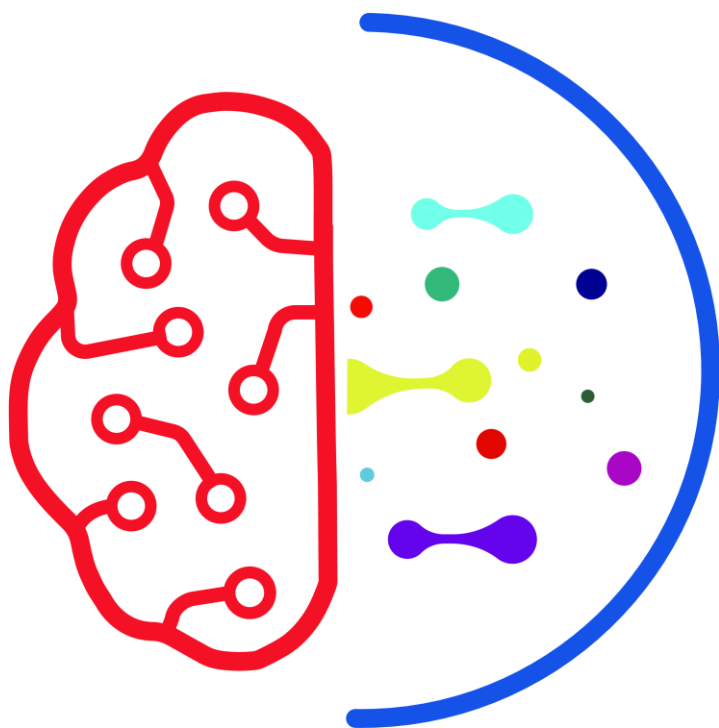
Copyright

Other than for strictly personal use, it is not permitted to download, forward or distribute the text or part of it, without the consent of the author(s) and/or copyright holder(s), unless the work is under an open content license such as Creative Commons.

Takedown policy

Please contact us and provide details if you believe this document breaches copyrights.
We will remove access to the work immediately and investigate your claim.

Reconstruction, Generation and Exploration of Digital Content with Deep Neural Features



Xuejiao Luo | 罗雪娇

**RECONSTRUCTION, GENERATION AND
EXPLORATION OF DIGITAL CONTENT WITH
DEEP NEURAL FEATURES**

RECONSTRUCTION, GENERATION AND EXPLORATION OF DIGITAL CONTENT WITH DEEP NEURAL FEATURES

Dissertation

for the purpose of obtaining the degree of doctor
at Delft University of Technology
by the authority of the Rector Magnificus, prof. dr. ir. T.H.J.J. van der Hagen,
chair of the Board for Doctorates
to be defended publicly on
Thursday 13, March 2025 at 12:30 o'clock

by

Xuejiao LUO

Master of Science in Multimedia Networking, Université Paris-Saclay, France
born in Guangdong, China

This dissertation has been approved by the promotor.

Composition of the doctoral committee:

Rector Magnificus,	chairperson
Prof.dr. E. Eisemann,	Delft University of Technology, <i>promotor</i>
Dr. R. Marroquim,	Delft University of Technology, <i>copromotor</i>

Independent members:

Prof.dr. A. Vilanova Bartoli	Delft University of Technology/ Eindhoven University of Technology
Prof.dr. M.A. Neerincx	Delft University of Technology
Dr.ir. C.C.S. Liem	Delft University of Technology
Dr. P.K. Didyk	Università della Svizzera italiana, Switzerland
Dr. K. Hildebrandt	Delft University of Technology, reserve member

Other members:

Dr. A. Bousseau	Inria, France
-----------------	---------------



This work was partially funded by the NWO VIDI grant NextView.

This work was carried out in the ASCI graduate school. ASCI dissertation series number: 465.

Printed by: ProefschriftMaken

Copyright © 2025 by X. Luo

ISBN 978-94-6510-533-8

An electronic copy of this dissertation is available at
<https://repository.tudelft.nl/>.

CONTENTS

Summary	ix
Samenvatting	xi
1 Introduction	1
1.1 Deep Neural Features in Digital Content	2
1.2 Our Contributions	4
2 Single-Image SVBRDF Estimation with Learned Gradient Descent	7
2.1 Introduction	8
2.2 Background and Related Work	8
2.3 Appearance capture with learned gradient descent	11
2.3.1 Problem formulation	11
2.3.2 Implementation	12
2.3.3 Data and training	13
2.4 Ablation studies	14
2.4.1 Number of iterations	14
2.4.2 Test-time gradient information	15
2.5 Results	16
2.5.1 Comparison on synthetic images	16
2.5.2 Comparison on real images	18
2.6 Limitations, extensions and future work	19
2.7 Conclusion	24
2.8 Acknowledgements	24
3 Perceptually Consistent Interpolation using MaterialGAN	25
3.1 Introduction	26
3.2 Methodology	26
3.2.1 Problem Formulation	26
3.2.2 Latent Space Interpolation	27
3.2.3 Regularization on Interpolation Weights	27
3.2.4 Optimization Framework	27
3.2.5 Gradient Descent Optimization	28
3.3 Results	28
3.4 Conclusion	31
4 Texture Browser: Feature-based Texture Exploration	35
4.1 Introduction	36
4.2 Related Work	37

4.3	Our Approach	39
4.3.1	Embedding	39
4.3.2	Prioritized t-SNE	41
4.3.3	Multi-scale Replication	41
4.3.4	Clustering	42
4.4	Interface	43
4.5	Evaluation	46
4.5.1	Comparison with alternative methods	47
4.5.2	Prioritized t-SNE and image-based search tool evaluation	50
4.6	Conclusion	52
4.7	Acknowledgments	52
5	Musicon: Glyph-Based Design for Music Visualization and Retrieval	53
5.1	Introduction	54
5.2	Related Work	54
5.3	Our Approach	55
5.3.1	Feature extraction	56
5.3.2	Feature Dimensionality Reduction	58
5.3.3	Glyph design	58
5.4	Interface	62
5.4.1	Customized icon	63
5.4.2	Search-by-icon	63
5.4.3	Playlist sorting	63
5.4.4	Enhancing icon contrast	64
5.5	Evaluation	66
5.5.1	Customized icon	67
5.5.2	Search by icon	69
5.5.3	Search by playlist	70
5.6	Conclusion	73
6	Conclusion	75
A	Supplementary material of Chapter 4	93
A.1	Introduction	93
A.2	Comparison with alternative methods	93
A.2.1	Tasks	93
A.2.2	Procedure	94
A.2.3	Results and discussion	94
A.3	Prioritized t-SNE and image-based search tool evaluation	98
A.3.1	Tasks	99
A.3.2	Procedure	99
A.3.3	Results and discussion	100
B	Supplementary material of Chapter 5	103
B.1	Introduction	103

B.2	Test one: Visual clustering	104
B.2.1	Task	104
B.2.2	Results and discussion	104
B.3	Test two: Outlier detection	105
B.3.1	Task	105
B.3.2	Results and discussion	105
B.4	Test three: Generalisation, Contrast and CVD Robustness	106
B.4.1	Task	106
B.4.2	Results and discussion	107
B.5	Test four: Search-by-icon	110
B.5.1	Task	110
B.5.2	Results and discussion	111
B.6	Test five: Search-in-playlist	112
B.6.1	Task	112
B.6.2	Results and discussion	112
Acknowledgements		119
Curriculum Vitæ		121
List of Publications		123

SUMMARY

This dissertation investigates the use of deep-learning methods for user-guided content creation, manipulation, and exploration. It illustrates the potential of neural techniques to support working with large data collections and we illustrate our solutions through several applications. Regarding content generation, we propose an algorithm to produce material representations from a single image. We illustrate content manipulation with an approach to perform perceptually plausible interpolation and examine exploration in the context of interactive retrieval. For the latter, we show that features spaces are of high relevance to organize data and show the generality of this concept by proposing novel exploration methods for image and music collections.

In **Chapter 2** we discuss the challenges in estimating Spatially-Varying Bidirectional Reflectance Distribution Functions (SVBRDFs) from a single image. It proposes a learned gradient descent method that combines neural network predictions with test-time optimization for robust SVBRDF estimation. This chapter presents experimental results demonstrating the effectiveness of the proposed method on both synthetic and real images.

In **Chapter 3** we explore the problem of generating smooth and perceptually uniform interpolations between images, and propose a method that maps linear latent space changes to perceptually consistent steps between consecutive images. Examples and evaluations of the generated interpolated images are provided, showcasing their visual consistency and quality.

While image retrieval is a well-researched field, many algorithms focus on the image content in terms of objects and actions rather than abstract semantic features. In **Chapter 4** We illustrate that feature-based approaches can be used to pursue more abstract image content. Specifically, we address the difficulty of exploring large collections of texture images, which focus on patterns and structures. We utilize high-level semantic features and dimensionality reduction techniques for efficient texture navigation and retrieval. The chapter describes the system's user interface and tools designed to enhance the exploration experience, and presents a user evaluation demonstrating the system's effectiveness.

In **Chapter 5** we illustrate that visual navigation linked to neural features can also be highly beneficial for non-visual data types, as we focus on the visualization and retrieval of music using glyph-based designs. It proposes a system that extracts deep latent features from music tracks and maps them to visual glyphs for intuitive search and exploration. The chapter evaluates the system through a user study, highlighting its benefits in enhancing music discovery and user engagement.

To conclude, we summarize the key contributions and findings of the thesis, discuss the research implications, and outline potential future directions, providing a comprehensive closure to the work.

SAMENVATTING

Deze dissertatie onderzoekt het gebruik van deep-learning-methoden voor gebruikersgestuurde contentcreatie, manipulatie en verkenning. Het illustreert het potentieel van neurale technieken om het werken met grote datacollecties te ondersteunen en we illustreren onze oplossingen aan de hand van verschillende toepassingen. Wat betreft contentgeneratie, stellen we een algoritme voor om materiaalrepresentaties te produceren op basis van een enkele afbeelding. We illustreren contentmanipulatie met een benadering om perceptueel plausibele interpolatie uit te voeren en onderzoeken verkenning in de context van interactieve retrieval. Voor dit laatste tonen we aan dat featurespaces van groot belang zijn voor het organiseren van data en tonen we de generaliteit van dit concept door nieuwe verkenningmethoden voor beeld- en muziekcollecties voor te stellen.

In **Hoofdstuk 2** bespreken we de uitdagingen bij het schatten van Spatially-Varying Bidirectional Reflectance Distribution Functions (SVBRDFs) uit een enkele afbeelding. Het stelt een geleerde gradient descent-methode voor die voorspellingen van een neuraal netwerk combineert met test-tijd optimalisatie voor robuuste SVBRDF-schatting. Dit hoofdstuk presenteert experimentele resultaten die de effectiviteit van de voorgestelde methode aantonen op zowel synthetische als echte afbeeldingen.

In **Hoofdstuk 3** verkennen we het probleem van het genereren van vloeiende en perceptueel uniforme interpolaties tussen afbeeldingen, en stellen we een methode voor die lineaire veranderingen in de latente ruimte omzet naar perceptueel consistente stappen tussen opeenvolgende afbeeldingen. Voorbeelden en evaluaties van de gegenereerde geïnterpoleerde afbeeldingen worden gepresenteerd, waarbij hun visuele consistentie en kwaliteit worden aangetoond.

Hoewel het ophalen van afbeeldingen een goed onderzocht gebied is, richten veel algoritmen zich op de inhoud van de afbeelding in termen van objecten en acties in plaats van op abstracte semantische kenmerken. In **Hoofdstuk 4** illustreren we dat op kenmerken gebaseerde benaderingen kunnen worden gebruikt om meer abstracte beeldinhoud na te streven. We richten ons specifiek op de uitdaging van het verkennen van grote collecties textuurafbeeldingen, die zich concentreren op patronen en structuren. We maken gebruik van semantische kenmerken op hoog niveau en technieken voor dimensionaliteitsreductie voor efficiënte textuurnavigatie en -ophaling. Het hoofdstuk beschrijft de gebruikersinterface van het systeem en de hulpmiddelen die zijn ontworpen om de verkenningservaring te verbeteren, en presenteert een gebruikersbeoordeling die de effectiviteit van het systeem aantoont.

In **Hoofdstuk 5** tonen we aan dat visuele navigatie gekoppeld aan neurale features ook zeer nuttig kan zijn voor niet-visuele datatypen, aangezien we ons richten op de visualisatie en het ophalen van muziek met behulp van op glyphs gebaseerde ontwerpen. Het stelt een systeem voor dat diepe latente kenmerken uit muziekstukken extraheert en deze omzet naar visuele glyphs voor intuïtieve zoek- en verkenning. Het hoofdstuk

evalueert het systeem door middel van een gebruikersstudie en benadrukt de voordelen ervan bij het verbeteren van muziekontdekking en gebruikersbetrokkenheid.

Ter afsluiting vatten we de belangrijkste bijdragen en bevindingen van de thesis samen, bespreken we de implicaties van het onderzoek en schetsen we mogelijke toekomstige richtingen, waarmee we een uitgebreide afsluiting van het werk bieden.

1

INTRODUCTION

The rapid advancement in computer graphics has introduced a wide range of techniques and tools for creating, manipulating, and exploring digital content. The AI revolution in computer vision and natural language processing was largely enabled by deep learning, particularly deep neural networks that led to sophisticated data-driven approaches [1]. This thesis explores the use of deep-learning as a user-oriented technique in the context of Computer Graphics, focusing on the reconstruction, generation, and exploration of digital content using deep neural features. Specifically, Most of our contributions in this dissertation are targeting material representations, which play an important role in Computer Graphics to be able to generate realistic imagery from a virtual scene. Still, our findings are more general and describe concepts that are applicable in a broader context. We illustrate this latter point with our work on music exploration.

Materials are indeed a crucial component in the realm of digital content creation. One such representation is the Spatially-Varying Bidirectional Reflectance Distribution Function (SVBRDF), which describes how light interacts with pixel-level elements of a material surface. The accuracy and efficiency of material reconstruction and retrieval directly influence the appearance of objects in 3D modeling, rendering, and visualization, as well as in photography, cinematography, and game development, all of which are essential for creating immersive experiences. Traditional methods of measuring SVBRDF typically involve capturing multiple images under various lighting conditions or using complex setups with specialized equipment, as shown on the left in Fig. 1.1. This process is often time-consuming and requires precise calibration to ensure accurate results [2, 3]. For exploring and retrieving material images, metadata-based approaches are commonly used, where images are categorized and stored based on manually added characteristics. In our work, we will show solutions to ease creating, manipulating, and exploring material definitions.

Recent advancements in the estimation of SVBRDF have been driven by the integration of deep learning techniques and hybrid approaches, largely enhancing the accuracy, efficiency, and accessibility of material property estimation. Convolutional Neural Networks (CNNs) and Generative Adversarial Networks (GANs) have been particularly impactful, enabling the estimation of complex, spatially varying material properties from minimal input data, such as a single image (Fig. 1.1, right) or a limited set of images [4–8]. These models have been trained on large datasets to learn intricate material characteristics,

enabling a detailed and physically plausible SVBRDF estimation.

Nevertheless, while generating SVBRDFs from a single image has become an option with deep learning and we show that novel insights in the optimization procedure can improve the outcome of this estimation process (**Chapter 2**), assisting users in finding the image that they need is challenging. Textures can be hard to describe and, as a result, difficult to retrieve using labels in natural language. When searching for the example images or the desired texture, efficient navigation and retrieval systems become essential. Many algorithms focus on the image content in terms of objects and actions. We illustrate that feature-based approaches, particularly the high-level semantic neural features, can be used to explore more abstract image content. This is especially relevant in large collections of texture images where patterns and structures dominate. Interestingly, the principles applied to image navigation and retrieval can be extended to other types of digital content, such as music files, demonstrating the versatility and broad applicability of deep neural features across various data domains. This is an aspect that we will investigate (**Chapter 4**) and also illustrate the breadth of this idea by showing this concept being applied to music exploration (**Chapter 5**).

Further, if an image is not available, one might want to describe a material as a mix between different inputs. Here, interpolation methods enable us to use example images to generate the desired result that meets specific criteria. We show that seamless blending of images, which ensures smooth transitions and realistic effects, for example, a specified perceptually uniform sequence (examples in Fig. 1.2) can be achieved using the deep neuron interpolation with the generative models (**Chapter 3**), making the interpolation process much easier to control.

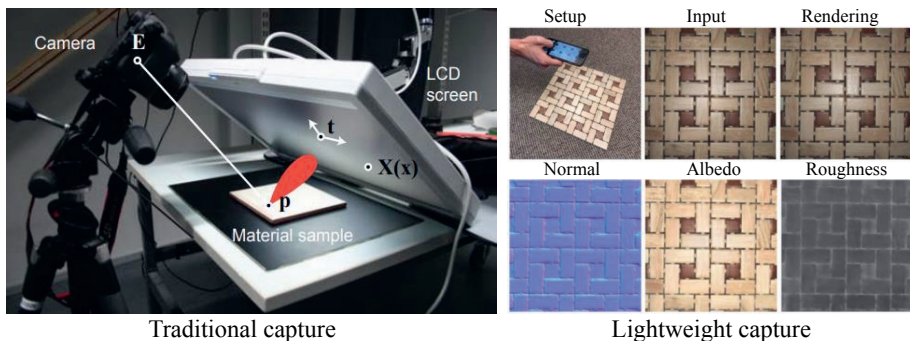


Figure 1.1: Traditional SVBRDF capture using complex setups (left, adapted from [2]) and streamlined SVBRDF capture using a single mobile phone photo (right, adapted from [5]).

1.1. DEEP NEURAL FEATURES IN DIGITAL CONTENT

The fulfillment of our vision depends on recent advancements in deep neural networks, which is why we revisit a few of the key aspects in this section, which have revolutionized

the analysis and manipulation of digital content.

Neural networks can automatically learn and identify intricate patterns and structures within the data, enabling more specific and efficient tasks. The representational power of neurons in learned deep networks provides opportunities to guide the exploration of large datasets across various media, such as images or music files. By explicitly manipulating neurons as latent vectors in generative models, it is even possible to alter the appearance of images in a controlled manner. We explore their potential in our tasks on reconstruction, generation, and exploration.

THE EXPRESSIVE CAPACITY OF VGG NETWORKS

Among various prominent works in AI from the last decade, convolutional neural networks (CNNs), particularly VGG16 and VGG19 [9], have significantly advanced the field of computer vision through its deep architecture and effective feature representation. VGG16 and VGG19 are renowned for their architecture with 16 and 19 layers, respectively, which allows them to capture intricate patterns in image data [9, 10]. Coupled with small 3×3 convolution filters, VGG networks perform tasks like image classification and object detection effectively, while maintaining computational efficiency [11]. The hierarchical feature extraction capability of VGG networks, from simple edges in early layers to complex texture information and object parts in deeper layers, is crucial for understanding and interpreting visual data [10–12]. It leads to highly discriminative and robust features. These learned features from pre-trained VGG models can be adapted to new problems with relatively little additional training [13, 14], which is employed in our exploration of large texture image datasets.

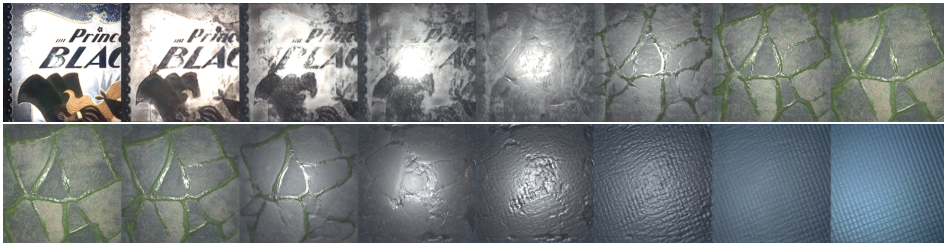


Figure 1.2: Perceptually consistent transitions between consecutive images in the interpolation sequence generated with GAN.

DIMENSIONALITY REDUCTION

Towards a practical and intuitive exploration and retrieval, the representative deep neural features are typically mapped to a lower-dimensional space. Dimensionality reduction techniques, such as t-SNE (t-distributed Stochastic Neighbor Embedding) [15] and UMAP [16], are commonly used to project high-dimensional neural features into a low-dimensional space, preserving the similarity structure of the data.

The integration of dimensionality reduction techniques with deep neural networks allows for more interpretable mapping of the features, helping users navigate large data

collections by visually clustering the data, a principle that we also rely on in our exploration solutions, for example in the music visualization and retrieval in **Chapter 5**, as illustrated in Fig. 1.3.

LATENT VECTORS IN GENERATIVE MODELS

The representational capacity of neurons is powerful not only in discriminative tasks, but also in generative tasks. Generative models, especially Generative Adversarial Networks (GANs), use latent vectors as a core component to generate new data samples [17–20]. The latent vector is a compact representation of a data point in a high-dimensional space learned by the model from the given training database. In the context of GANs, the generator network takes the latent vector as input and transforms it into a synthetic data sample, such as an image [21].

Latent vectors capture the underlying factors of variation in the training data, enabling the generation of diverse and realistic samples. By interpolating between latent vectors, it is possible to create smooth transitions between different generated samples, which is useful for tasks such as image interpolation [21], as shown in Fig. 1.2. The latent space of a GAN is typically structured such that similar vectors produce similar outputs [17], which inspired us to make it a powerful tool to generate controlled variations of texture images as the interpolations from existing examples.

1.2. OUR CONTRIBUTIONS

Digital reconstruction, generation, and exploration in computer graphics can traditionally demand highly professional equipment or significant manual effort. Meanwhile, deep neural networks offer powerful representative neural features that can describe digital content with minimal information. In this dissertation, we aim to utilize advanced deep learning techniques, including CNNs, RNNs, and GANs, to achieve more intuitive, efficient, and creative processes with the digital content using minimal manual input. Recent advancements, such as the deep learning-powered estimation of SVBRDFs and image interpolation, demonstrate improvements over traditional methods. However, these advancements are still limited due to the lack of proper target-specific optimization. Digital content visualization and exploration have traditionally relied on older methods and suffer from insufficient perceptual representations and unintuitive interaction. Below, we detail the challenges within these fields and outline the innovations introduced in each chapter of this thesis.

In the context of *single-image SVBRDF estimation*, current methods often rely on feed-forward neural networks that predict reflectance parameters based purely on data-driven priors, neglecting the output quality with respect to the input image. Methods utilizing differentiable rendering for optimization are prone to poor local minima, especially when the input images differ significantly from the training data. Moreover, optimization in lower-dimensional latent spaces can hinder the accurate reconstruction of SVBRDFs for images outside the training distribution. In **Chapter 2**, we introduce a learned gradient descent approach that combines the rapid prediction capabilities of feed-forward neural networks with the precision of test-time optimization. By leveraging a recurrent neural network, our method dynamically updates reflectance parameters based on the gradient

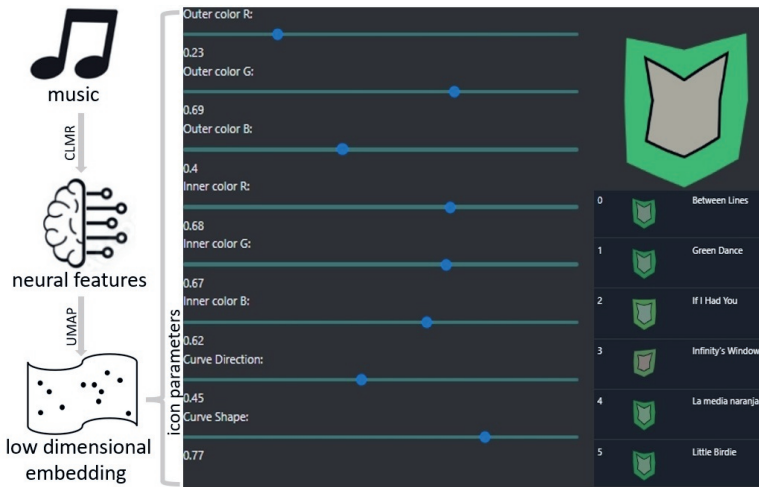


Figure 1.3: Informed by dimensionality reduced deep neural features, the glyph-based designs enhance the user interface for identifying and exploring music through visual cues.

of the reconstruction likelihood, achieving a maximum a posteriori (MAP) estimate. This hybrid approach overcomes the limitations of both purely data-driven and optimization-based methods, providing robust SVBRDF estimation even for challenging real-world images.

For *material image interpolation*, traditional interpolation methods often fail to maintain perceptual uniformity when the interpolation parameters are adjusted linearly. Linear interpolation in the latent space of GANs may not ensure smooth transitions in the output sequence, leading to inconsistencies. Our research in **Chapter 3** addresses this challenge by focusing on perceptually consistent interpolation of material appearance. We propose a method that maps linear changes in the latent space of GAN to consistent perceptual steps, ensuring a constant Structural Similarity Index (SSIM) between consecutive rendered material images. An optimization framework adjusts interpolation weights to maintain uniform SSIM across the interpolation sequence. Experiments on various material datasets show that our SSIM-optimized interpolation significantly improves transition smoothness and visual consistency compared to direct linear interpolation. Quantitative evaluations confirm a lower variance in SSIM, ensuring both visually appealing results and perceptual uniformity.

In the area of *texture exploration*, existing retrieval systems rely heavily on manual annotations, which are time-consuming and often lack of standardization. Traditional methods do not provide intuitive navigation tools, making it difficult for users to explore large, unlabeled texture collections effectively. In **Chapter 4**, we propose an automated exploration system that organizes textures based on high-level semantic features extracted from a pre-trained convolutional neural network. Our use of prioritized t-SNE for dimensionality reduction, along with enhanced navigation tools such as scalable clustering and

flip zooming, noticeably improves the efficiency and intuitiveness of texture exploration and retrieval.

For *music visualization and retrieval*, current music discovery platforms offer limited visual representations, making it challenging for users to assess music characteristics without listening. The absence of effective visual search tools can hinder user interaction and exploration of large and unfamiliar music libraries. In **Chapter 5**, we propose a glyph-based visualization system that translates deep latent features of music into intuitive visual icons, enhancing the user-directed search process. By combining deep learning-based feature extraction with dimensionality reduction, our system offers a novel interface that improves user engagement and efficiency in music exploration.

Through this research, we show that working with deep-learning features can reduce the need for costly equipment and extensive manual labor. Our innovations promise to offer solutions that improve the accuracy and efficiency in the digital material property reconstruction, and simplify and enhance the workflows involved in the generation, and exploration of data content.

2

SINGLE-IMAGE SVBRDF ESTIMATION WITH LEARNED GRADIENT DESCENT

Recovering spatially-varying materials from a single photograph of a surface is inherently ill-posed, making the direct application of a gradient descent on the reflectance parameters prone to poor minima. Recent methods leverage deep learning either by directly regressing reflectance parameters using feed-forward neural networks or by learning a latent space of SVBRDFs using encoder-decoder or generative adversarial networks followed by a gradient-based optimization in latent space. The former is fast but does not account for the likelihood of the prediction, i.e., how well the resulting reflectance explains the input image. The latter provides a strong prior on the space of spatially-varying materials, but this prior can hinder the reconstruction of images that are too different from the training data. Our method combines the strengths of both approaches. We optimize reflectance parameters to best reconstruct the input image using a recurrent neural network, which iteratively predicts how to update the reflectance parameters given the gradient of the reconstruction likelihood. By combining a learned prior with a likelihood measure, our approach provides a maximum a posteriori estimate of the SVBRDF. Our evaluation shows that this learned gradient-descent method achieves state-of-the-art performance for SVBRDF estimation on synthetic and real images.

2.1. INTRODUCTION

Real-world objects have a rich visual appearance due to spatially-varying material properties, which can be represented by Spatially-Varying Bidirectional Reflectance Distribution Functions (SVBRDFs). This paper presents a lightweight method to capture the appearance of real surfaces with only a single photo.

Since few measurements are insufficient to ensure a unique interpretation of the many reflectance parameters, recent research leveraged deep learning to automatically build priors based on the distribution of plausible SVBRDFs. A first family of methods trains feed-forward neural networks to predict spatially-varying reflectance parameters from as little as a single flash picture of a flat surface [4–7, 22–24]. While fast, such neural networks mostly rely on data-driven priors to make their prediction. During use, they never evaluate the actual quality of their output with respect to the input image.

A second family of methods achieves higher accuracy by using differentiable rendering for online optimization, where the estimated SVBRDF is rendered under the same viewing and lighting conditions as for capture. Gradient descent is used to minimize the difference between the rendering and the input image. Yet, relying solely on differentiable rendering to optimize reflectance parameters is prone to bad minima, which is why several groups of authors proposed to perform the optimization in a lower-dimensional latent space learned from a dataset of representative SVBRDFs [25, 26]. Nevertheless, gradient descent typically requires many iterations to converge and latent-space regularization can limit the quality of the estimation when the input differs too much from the images used to build the latent space.

Recent work proposed to combine these two strategies by training a neural network on a large dataset of SVBRDFs, and then fine-tuning the network weights at test time such that its prediction best reproduces the input image [27, 28]. A key challenge with this new strategy is to prevent the fine-tuning phase to forget the priors learned during the training phase.

Our algorithm combines the speed of neural network prediction with the accuracy of test-time optimization. It is inspired by *learned gradient descent* [29, 30], which replaces the analytic gradient update rule of standard optimization by a recurrent neural network. This network is trained to predict the best updates given the current state of the estimation and the gradient of the cost function to be minimized. Importantly, the neural network weights are not updated at test-time, avoiding the risk of forgetting its priors. In our context, the cost function captures the *likelihood* of the SVBRDF to reproduce the input image when rendered under the same light and view, while the neural network learns a *prior* over the distribution of SVBRDFs. By combining likelihood and prior information, our method effectively solves for a *maximum a posteriori* estimate of the SVBRDF. While trained on a synthetic dataset of SVBRDFs, our method generalizes well to real data, outperforming both feed-forward and optimization-based prior work, as demonstrated on a large set of photographs.

2.2. BACKGROUND AND RELATED WORK

We focus our discussion on recent deep learning methods for lightweight SVBRDF capture, and refer to surveys for a comprehensive overview of the vast domain of appearance

acquisition [31–33]. We first introduce general concepts on which our approach relies, before diving into recent methods, which combine deep-learning and gradient-based optimization to recover SVBRDF parameters from one or a few flash images of a planar surface.

Appearance capture as an inverse problem. Formally, the image \mathbf{I} of a surface depends on its reflectance properties \mathbf{R} , as well as the viewing conditions \mathbf{V} and lighting conditions \mathbf{L} under which the surface is captured:

$$\mathbf{I} = f(\mathbf{R}, \mathbf{L}, \mathbf{V}) + \mathbf{n}, \quad (2.1)$$

where f is the image formation model and \mathbf{n} is measurement noise.

Appearance capture aims at inverting the image formation to recover \mathbf{R} from observations \mathbf{I} , typically under known viewing and lighting conditions:

$$\hat{\mathbf{R}} = \arg \min_{\mathbf{R}} \mathcal{L}_{\text{reconstruct}}(\mathbf{I}, f(\mathbf{R}, \mathbf{L}, \mathbf{V})), \quad (2.2)$$

where $\mathcal{L}_{\text{reconstruct}}$ is a cost function measuring the difference between the observations and renderings of the estimated reflectance. Assuming that $\mathcal{L}_{\text{reconstruct}}$ is differentiable, gradient descent can be employed for the minimization:

$$\mathbf{R}_{t+1} = \mathbf{R}_t - \gamma_t \left. \frac{\partial \mathcal{L}_{\text{reconstruct}}(\mathbf{I}, f(\mathbf{R}, \mathbf{L}, \mathbf{V}))}{\partial \mathbf{R}} \right|_{\mathbf{R}=\mathbf{R}_t}, \quad (2.3)$$

where γ_t is the step size at iteration t .

To make this inverse problem well-posed, early work relied on dedicated gantries to capture many images of the target surface under different light and view configurations [34–37]. Despite progress in hardware setups and optimization algorithms [2, 38, 39], precise acquisition of spatially-varying materials remains a costly and time-consuming process. Moreover, gradient-based optimization often requires a large number of iterations and is subject to bad local minima, especially using few measurements.

Lightweight capture methods trade accuracy for simplicity to enable SVBRDF capture with as few as a single photograph of a surface – typically planar. Such methods compensate for the measurement scarcity by making various assumptions on the materials to be acquired, such as the existence of a low-dimensional basis of BRDFs [40–43], or the presence of repetitive or stochastic patterns [44, 45].

Feed-forward SVBRDF prediction. Recent work shifted from hand-crafted assumptions towards priors learned from large datasets of (synthetic) SVBRDFs. A first family of methods cast SVBRDF acquisition as a regression task, for which they train a feed-forward neural network g_ω to directly predict reflectance properties from an input image [4–7, 22, 24]. Denoting $\{\tilde{\mathbf{R}}, \tilde{\mathbf{I}}\}$ a large set of SVBRDFs and their renderings, training the neural network with supervised learning amounts to solving for parameters ω , minimizing a loss function $\mathcal{L}_{\text{reflectance}}$, which compares the predicted SVBRDFs with the ground truth:

$$\hat{\omega} = \arg \min_{\omega} \sum_{\{\tilde{\mathbf{R}}, \tilde{\mathbf{I}}\}} \mathcal{L}_{\text{reflectance}}(g_\omega(\tilde{\mathbf{I}}), \tilde{\mathbf{R}}). \quad (2.4)$$

Further developments of such methods include the use of a rendering loss function $\mathcal{L}_{\text{rendering}}(f(g_\omega(\tilde{\mathbf{I}}), \{\mathbf{L}, \mathbf{V}\}), f(\tilde{\mathbf{R}}, \{\mathbf{L}, \mathbf{V}\}))$ to evaluate whether the predicted SVBRDF has the same appearance as the ground truth under varying viewing and lighting conditions [6], or an adversarial loss $\mathcal{L}_{\text{adv}}(g_\omega(\tilde{\mathbf{I}}), \{\tilde{\mathbf{R}}\})$ to evaluate whether the predicted SVBRDF resembles the ones in the dataset [24], or $\mathcal{L}_{\text{adv}}(f(g_\omega(\tilde{\mathbf{I}}), \mathbf{L}, \mathbf{V}), \{\tilde{\mathbf{I}}\})$ to evaluate whether the re-rendered image resembles synthetic and real images [7, 23, 46].

Latent-space optimization. While feed-forward neural networks are fast to evaluate, the SVBRDF parameters they produce are entirely defined by the SVBRDF dataset $\{\tilde{\mathbf{R}}\}$ they are trained on, not by how well these parameters reproduce the input image \mathbf{I} at test time. In other words, feed-forward networks only provide an approximate solution to the inverse problem formulated in Eq. 2.2, and the severity of this approximation tends to increase for input images that deviate from the distribution of the training images $\{\tilde{\mathbf{I}}\}$. This discrepancy has motivated the development of test-time optimization methods that use gradient descent (Eq. 2.3) to refine neural-network predictions to better fit the input images. Since SVBRDF recovery from few input images is ill-posed, several papers propose to regularize the problem by performing gradient descent in a low-dimensional SVBRDF latent space, instead of the original high-dimensional parameter space of \mathbf{R} [25, 26]:

$$\mathbf{z}_{t+1} = \mathbf{z}_t - \gamma_t \left. \frac{\partial \mathcal{L}_{\text{reconstruct}}(\mathbf{I}, f(d_\psi(\mathbf{z}), \mathbf{L}, \mathbf{V}))}{\partial \mathbf{z}} \right|_{\mathbf{z}=\mathbf{z}_t}, \quad (2.5)$$

where a network d_ψ decodes the latent code \mathbf{z} into an SVBRDF. Learning the latent space from a large dataset of SVBRDFs $\{\tilde{\mathbf{R}}\}$ ensures that the optimization produces plausible solutions. However, the optimization might struggle to find a latent code, which reproduces the input image well if it differs too much from the training data. Further, the many iterations required by gradient-based optimizations induce a significant overhead compared to direct prediction.

Network fine-tuning. Several authors proposed to fine-tune a feed-forward network g_ω at test time such that its prediction better reproduces the input [27, 28, 47], which amounts to performing gradient descent on the neural-network parameters rather than on the reflectance parameters or latent code:

$$\hat{\omega} = \arg \min_{\omega} \mathcal{L}_{\text{reconstruct}}(\mathbf{I}, f(g_\omega(\mathbf{I}), \mathbf{L}, \mathbf{V})). \quad (2.6)$$

This strategy enables adjusting the prediction to the input, while still benefiting from the priors learned by the network during pre-training on a large dataset. Fischer and Ritschel [27] build on the concept of *meta-learning* to optimize the initialization of the network parameters and the gradient descent step sizes such that fine tuning converges quickly to good solutions. However, test-time fine-tuning runs the risk of forgetting the learned priors since it updates the weights by minimizing only the reconstruction error. A critical difference of our approach is to perform test-time optimization on the SVBRDF maps themselves, not on network weights, which ensures that the learned priors encoded by our recurrent neural network are preserved.

Similarly to meta-learning, Zhou and Kalantari [28] propose to include fine-tuning steps during pre-training of the network, an algorithm they call *look-ahead training*. Yet, their approach also includes a secondary network that is trained to predict reflectance maps, which serves as a data-driven prior during test-time fine-tuning. Nevertheless, this prior is combined with the reconstruction error as a linear combination (Eq.7 in their paper) and is only used for the first iteration of the optimization (Section 4.4 in their paper). In contrast, we provide the gradient of $\mathcal{L}_{\text{reconstruct}}$ to a recurrent network that learns to best combine this test-time information with its priors to iteratively improve the prediction.

Importantly, while [27] and [28] rely on hand-tuned step sizes for the gradient descent optimization, our method predicts the magnitude of the steps and yields results of similar quality in much fewer steps, making it 10x faster than [28] (Table 2.2).

2.3. APPEARANCE CAPTURE WITH LEARNED GRADIENT DESCENT

2.3.1. PROBLEM FORMULATION

Our approach combines the respective strengths of optimization-based and regression-based methods. We cast appearance capture as the minimization problem of Eq. 2.2, using a single flash image \mathbf{I} as observation of the planar surface to acquire. Yet, we replace the brittle and costly analytic gradient descent of Eq. 2.3 by a *learned* gradient descent [29, 30], where we train a recurrent neural network h_θ to predict how to progressively update an estimate \mathbf{R}_t of the SVBRDF:

$$\mathbf{R}_{t+1} = \mathbf{R}_t - h_\theta \left(\frac{\partial \mathcal{L}_{\text{reconstruct}}(\mathbf{I}, f(\mathbf{R}, \mathbf{L}, \mathbf{V}))}{\partial \mathbf{R}} \bigg|_{\mathbf{R}=\mathbf{R}_t}, \mathbf{R}_t \right). \quad (2.7)$$

This formulation corresponds to a *maximum a posteriori estimation*, where the cost function $\mathcal{L}_{\text{reconstruct}}$ is proportional to the *likelihood* of the solution with respect to the input, while the neural network h_θ captures a *prior* on the distribution of SVBRDFs. Intuitively, the likelihood term encourages fidelity to the input, while the prior helps resolving ambiguities and prevents overfitting. This formulation has several advantages over existing work:

- In the absence of a prior, standard gradient descent (Eq. 2.3) corresponds to *maximum likelihood estimation*, which is ill-posed when only a single input image is available. While network fine-tuning makes the problem better posed by initializing the optimization with a data-driven prediction, it runs the risk of forgetting the prior learned by the network if too many optimization steps are performed. In contrast, by combining the neural-network prior with test-time gradients of $\mathcal{L}_{\text{reconstruct}}$, our approach converges to a good solution in only a few steps. In addition, our approach does not require specifying a step size γ_t , as the magnitude of the update is implicitly predicted by h_θ .
- In the absence of a test-time likelihood term, feed-forward networks rely mostly on priors learned from the training data distribution (Eq. 2.4) to predict SVBRDFs in

a single step. In contrast, our network performs the simpler task of progressively improving a running estimate of the SVBRDF given gradient information about its likelihood. In practice, this online optimization scheme allows us to produce much more accurate results than feed-forward methods.

- While latent-space optimization methods benefit from data-driven priors, these priors are learned in a pre-process via auto-encoders [25] or generative-adversarial networks [26] trained on synthetic SVBRDFs. In contrast, our neural network learns priors by being trained specifically to perform maximum a posteriori estimation. As such, it accounts for the availability of the test-time likelihood. Importantly, our optimization happens in the original reflectance parameter space and is, thus, not limited to a pre-defined latent space.

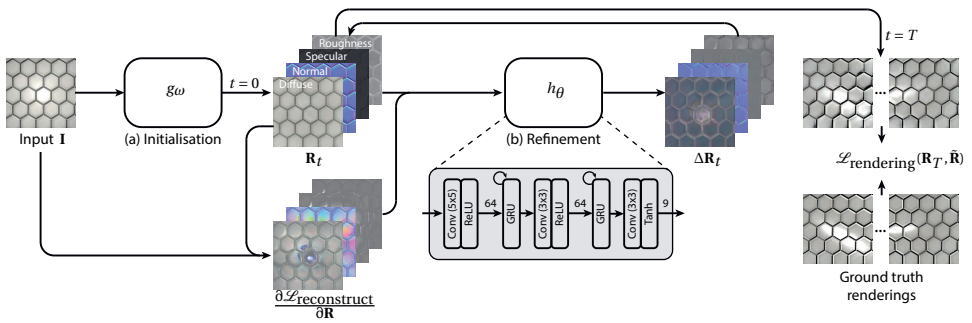


Figure 2.1: Overview of our approach. The input image \mathbf{I} is first fed to an existing feed-forward neural network g_ω to predict an initialization \mathbf{R}_0 of the SVBRDF maps (a). This prediction is then iteratively refined by our recurrent neural network h_θ (b). At each iteration, the current estimate \mathbf{R}_t is compared to the input using a differentiable renderer. The gradient of this reconstruction loss, $\frac{\partial \mathcal{L}_{\text{reconstruct}}}{\partial \mathbf{R}}$, is fed to h_θ along with \mathbf{R}_t . The recurrent network predicts an update $\Delta \mathbf{R}_t$ of the SVBRDF, which is added to \mathbf{R}_t to form the estimate \mathbf{R}_{t+1} for the next iteration. Our algorithm performs $T = 6$ such iterations in practice. We train g_ω and h_θ jointly to minimize the difference between renderings of the final prediction \mathbf{R}_T and renderings of the ground truth material maps under various view and light conditions. Note that the gradient and update images were scaled for visualization purpose.

2.3.2. IMPLEMENTATION

Our method belongs to the family of learned gradient-descent algorithms [29, 30] that rely on recurrent neural networks to implement update rules that automatically leverage the inherent structure of the optimization problem at hand. While learned gradient descent has been successfully used to solve inverse imaging problems, such as novel-view synthesis [48] and MRI reconstruction [49, 50], we make specific adaptations to apply this approach to single-image SVBRDF capture (see Fig. 2.1).

The core of our approach is a lightweight recurrent neural network h_θ that takes as input the current estimate of the SVBRDF \mathbf{R}_t along with the gradient of the cost function $\mathcal{L}_{\text{reconstruct}}$ with respect to \mathbf{R}_t , which we obtain via automatic differentiation. The network outputs an update $\Delta\mathbf{R}_t$, which is summed with \mathbf{R}_t to produce \mathbf{R}_{t+1} . In our implementation, we formulate $\mathcal{L}_{\text{reconstruct}}$ as the image difference between the input \mathbf{I} and a rendering of \mathbf{R}_t under a view and light setup that corresponds to a flash picture taken perpendicularly to the surface at a fixed distance. We use the L_2 norm to compute this difference, which corresponds to the log-likelihood under a Gaussian distribution assumption.

We initialize the SVBRDF estimate \mathbf{R}_0 by processing the input image \mathbf{I} with the feed-forward network g_ω of Deschaintre et al. [6]. While we experimented with the pre-trained weights provided by the authors, we achieved better results by re-training this initialization network jointly with our recurrent updating network. We hypothesize that joint training enables the initialization network to account for the subsequent optimization performed by the recurrent network, similarly to the *meta-learning* and *look-ahead* strategies recently proposed by [27] and [28] in the context of test-time network fine tuning.

Internally, h_θ is composed of three convolutional layers interleaved with Gated Recurrent Units (GRUs) [51]. The first two convolutional layers are activated with leaky ReLU functions and output feature maps of 64 channels, while the third convolutional layer is activated with a hyperbolic tangent to produce values between -1 and 1, which represent the update of the 9 SVBRDF channels, where 3 channels correspond to the diffuse albedo, 3 channels to the specular albedo, 2 channels to the normal, and 1 channel to the specular roughness. We used convolutional kernels of size 5×5 for the first layer and 3×3 for the second and third layer, resulting in 405,376 parameters in total for h_θ , much less than the 159,741,922 parameters of the initialization network g_ω . We voluntarily built on the classical UNet of Deschaintre et al. and on a lightweight recurrent network to demonstrate that the boost in performance achieved by our approach is due to methodological rather than architectural novelty.

An important hyper-parameter of our method is the number of iterations (or updates) T performed by the recurrent network. While several iterations are necessary to improve the prediction, performing too many iterations can be expensive in terms of GPU memory and time. Specifically, the GPU memory consumption of the recurrent network increases linearly with the number of time steps. Every iteration adds a forward pass through the CNN layers and the computation of the gradient of the reconstruction error. Therefore, the training/testing time of the recurrent network also increases linearly with the number of time steps. We empirically found that $T = 6$ iterations offer a good trade-of, as detailed in Section 2.4.1.

2.3.3. DATA AND TRAINING

Similarly to prior work [6, 7, 24–26, 28], we adopt a Cook-Torrance SVBRDF model [52] with the GGX distribution [53], which is parameterized by four material maps, corresponding to the diffuse/specular albedo, specular roughness, and surface normal. We visualize all inputs and results in gamma space, except normals and roughness, which we keep in linear space.

We train the initialization network g_ω and our recurrent update network h_θ jointly on the dataset of [6], which contains 99,533 synthetic SVBRDFs $\{\hat{\mathbf{R}}\}$. We render the

images $\{\tilde{\mathbf{I}}\}$ of these SVBRDFs under view \mathbf{V} and light \mathbf{L} that emulate a camera positioned perpendicularly and at a fixed distance to the planar surface, with a co-located flash of fixed intensity. We adjusted these parameters by hand to best reproduce the appearance of the renderings provided by [6]. We assume that the test-time input images are captured under similar view and light conditions, and thus use the same parameters to compute the gradient of $\mathcal{L}_{\text{reconstruct}}$ fed to h_θ . We train our method to minimize the rendering loss proposed by [6], which compares renderings of the material maps \mathbf{R}_T predicted at the last iteration of our recurrent network with renderings of the ground-truth maps $\tilde{\mathbf{R}}$, under 9 random lighting and viewing conditions $\{\mathbf{L}, \mathbf{V}\}$. Following [6], we use the $L1$ norm and compare the logarithmic values of the renderings:

$$\mathcal{L}_{\text{rendering}}(\mathbf{R}_T, \tilde{\mathbf{R}}) = \sum_{\{\mathbf{L}, \mathbf{V}\}} |\log f(\mathbf{R}_T, \mathbf{L}, \mathbf{V}) - \log f(\tilde{\mathbf{R}}, \mathbf{L}, \mathbf{V})|. \quad (2.8)$$

We used the Adam optimizer with a learning rate set to 0.00002, betas set to (0.9, 0.999), and the weight decay set to 0. We trained our method until convergence (80 epochs with a batch size of 4), which took three weeks on an NVIDIA A40 GPU. Once trained, our method infers SVBRDF maps from an image in around 0.1 seconds on the same NVIDIA A40 GPU.

2.4. ABLATION STUDIES

We conducted several ablation studies to assess the impact of the number of iterations performed by our recurrent network, as well as the benefit of providing gradient information to this network at test time. Similarly to [28], we performed all studies on a set of 61 synthetic SVBRDF, 22 being provided by [54] and 39 by [26]. Importantly, none of these SVBRDFs were used to generate the training data. We created synthetic flash inputs for this test set by rendering each SVBRDF under the same light and view conditions as the ones used for training our method.

We evaluate the quality of the prediction by comparing re-renderings of the SVBRDFs to ground truth in terms of root mean squared error (RMSE) and learned perceptual image patch similarity (LPIPS) [55], averaged over 20 random light and view configurations that differ from the colocated flash configuration used to render the input.

2.4.1. NUMBER OF ITERATIONS

We first evaluate the performance of our recurrent network related to the number of iterations T . We trained different models with $T = 2$ to $T = 10$. Fig. 2.2 (top left) plots the RMSE and LPIPS achieved by these models on the test set. This experiment reveals that while the RMSE saturates after 6 iterations, LPIPS increases slightly when more iterations are performed, even though it remains lower than the LPIPS achieved by previous methods (see Table 2.2). We thus fix the total number of iterations to 6, which offers a good trade-off between accuracy and complexity of the model. Fig. 2.2 (top right) plots the evolution of the RMSE and LPIPS of the test set over the iterations of the model trained for 6 iterations, showing that quality improves as the optimization progresses. In practice, the magnitude of improvement varies between materials. Fig. 2.3 shows two typical SVBRDFs where the initial prediction is either too shiny, or not enough, and gets corrected by subsequent

iterations. Finally, Fig. 2.2 (bottom) plots the evolution of the same metric when we let the model trained on 6 iterations run for more iterations. While the error remains stable for up to 24 iterations, it does not decrease significantly, and it eventually increases if too many iterations are performed.

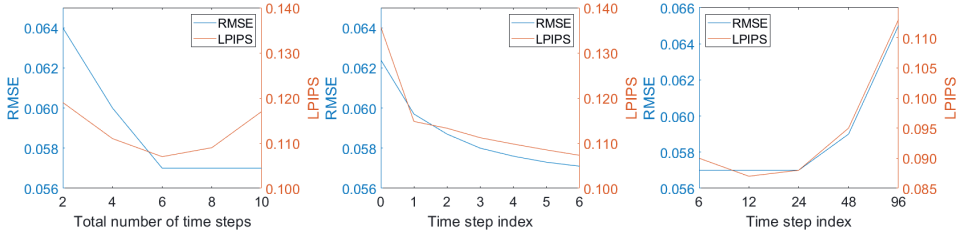


Figure 2.2: Impact of the number of iterations performed by the recurrent network. Upper left: comparison between models trained with an increasing total number of iterations. Upper right: evolution of the accuracy achieved by a model trained for a total of 6 iterations. Lower middle: evolution of the accuracy in further inference steps with the same model trained for a total of 6 iterations.

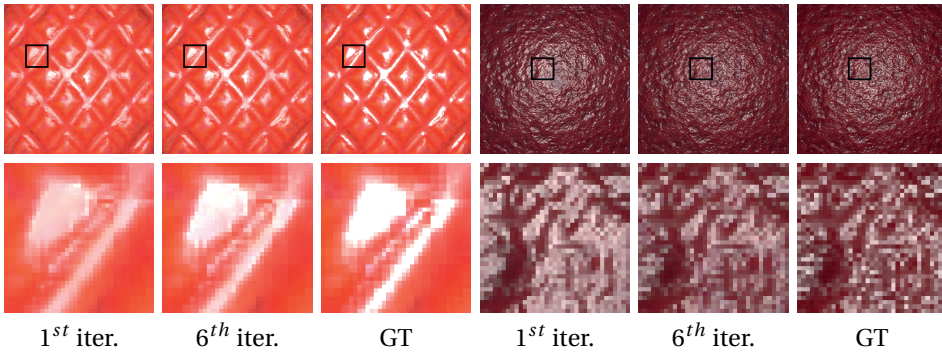


Figure 2.3: Starting with the initial prediction (left), the recurrent network refines the result (middle), bringing it closer to GT (right). In these examples, the refinement mostly affects the intensity, spread and sharpness of the highlights to make the material more (left side) or less (right side) shiny.

2.4.2. TEST-TIME GRADIENT INFORMATION

Our architecture improves upon the one proposed by [6] by complementing it with a recurrent network, and by providing test-time gradient information to that recurrent network. We now evaluate the impact of these two additional components. To do so, we compare the pre-trained model g_ω by [6] to two versions of our architecture.

The first version augments g_ω with the recurrent network h_θ , but only feeds this network with the intermediate prediction \mathbf{R}_t at each iteration. The second and complete version feeds the recurrent network with \mathbf{R}_t and the gradient $\frac{\partial \mathcal{L}_{\text{reconstruct}}}{\partial \mathbf{R}}$.

Table 2.1 summarizes the experiment’s outcome. Complementing the architecture of [6] with a recurrent network already yields a significant increase in accuracy, which we attribute to the additional capacity that each iteration provides. Providing test-time gradient information to this recurrent network improves accuracy further, reducing RMSE by 31% and LPIPS by 52% over the baseline g_ω .

	RMSE	LPIPS
without h_θ	0.083	0.223
without gradient	0.069	0.119
Ours	0.057	0.107

Table 2.1: Ablation study to compare our complete method to the baseline architecture by [6], which does not include the recurrent network h_θ , and to a version that includes the recurrent network but no test-time gradient. RMSE and LPIPS of re-renderings are averaged over 20 random light/view configurations.

2.5. RESULTS

We compare our approach to recent methods for lightweight SVBRDF capture, either based on feed-forward networks [6, 7, 24] or on test-time optimization [25, 26, 28]. We used the code and pre-trained weights provided by the authors of each method, except for [24] for which we sent our testing data to the authors, who kindly agreed to run their method and send back their results. We ran all methods on a single input image, even for methods that can process multiple images. We provide additional results, including animations under moving lights, as supplemental materials.

2.5.1. COMPARISON ON SYNTHETIC IMAGES

We first focus on the synthetic test set (see Section 2.4). For all methods, we report the RMSE on the individual SVBRDF maps, as well as the RMSE and LPIPS errors on re-renderings averaged over 20 random light and view configurations. We use the same 20 configurations to compare all methods on a given SVBRDF. We generate these configurations by sampling the light and view positions uniformly over a quad of the same size as the surface patch, parallel to and above the surface. This ensures that the images always contain a highlight.

Table 2.2 summarizes the results achieved by each method¹. When looking at individual maps, our method achieves the best result for diffuse albedo and normals, the second best

¹The numbers we report were computed by running all methods on our test set, which is composed of 61 synthetic materials provided by [54] and [26]. The difference between these numbers and the ones reported by Zhou and Kalantari [28] might be due to the fact that their test set (which is not available) only contains 52 of our 61 materials, and that the viewing and lighting conditions we used to render the dataset might differ from the ones used by [28] (which are unknown to us). Also, we observed that the synthetic inputs and roughness maps provided in [28] are visually different from ours, which suggests that they treated the roughness maps from [26] as linear while we treated them as gamma-corrected to agree with the ones from [54]. Nevertheless, the RMSE and LPIPS values reported in Table 1 of [28] remain suboptimal to ours on re-renderings.

	RMSE										LPIPS		Speed/sec
	Diffuse		Specular		Rough		Normal		Render		Render		
	mean	std	mean	std	mean	std	mean	std	mean	std	mean	std	
Method													
Des18	0.056	0.066	0.144	0.106	0.350	0.313	0.064	0.032	0.083	0.048	0.223	0.129	0.07
Guo21	0.094	0.115	0.119	0.128	0.185	0.158	0.071	0.039	0.095	0.054	0.214	0.090	NA
Zhou21	0.086	0.039	0.089	0.077	0.193	0.196	0.067	0.034	0.112	0.039	0.150	0.073	0.02
Gao19	0.070	0.040	0.119	0.087	0.296	0.279	0.073	0.035	0.084	0.035	0.139	0.076	42.70
Guo20	0.064	0.042	0.101	0.090	0.325	0.275	0.077	0.041	0.072	0.034	0.167	0.078	261.50
Zhou22	0.081	0.086	0.142	0.115	0.209	0.170	0.066	0.034	0.094	0.049	0.186	0.102	4.30
Ours	0.051	0.035	0.101	0.096	0.199	0.230	0.061	0.033	0.057	0.032	0.107	0.070	0.20

Table 2.2: Quantitative comparison on synthetic SVBRDFs. Des18, Guo21 and Zhou21 are fast feed-forward methods, while Gao19, Guo20 and Zhou22 are slower due to test-time optimization. Our approach achieves state-of-the-art quality while being an order of magnitude faster than the fastest optimization method. All timings were measured on an NVIDIA GeForce GTX 1080Ti GPU.

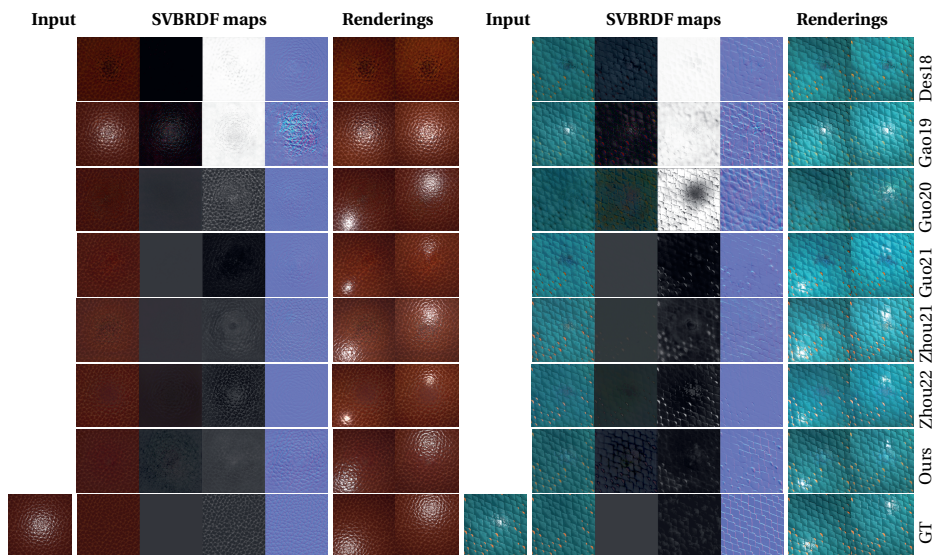


Figure 2.4: Visual comparison against other methods on synthetic images. The order of the SVBRDF maps is Diffuse, Specular, Roughness, and Normal maps. Note how our method recovers more faithful normal maps, as well as roughness and specular information away from the highlight. For visualization purposes, all images except Roughness and Normal maps are shown in gamma space.

result for specular albedo (outperformed by [7]), and the third best result for roughness (outperformed by [24] and [7]). Importantly, our method achieves the best results on re-renderings, both in terms of RMSE and LPIPS. Note also that our method is an order of magnitude slower than feedforward approaches [6, 7, 24], but an order faster than fine-tuning [28] and two to three orders faster than latent-space optimization [25, 26].

Fig. 2.4 provides a visual comparison on two representative SVBRDFs. Overall, our approach based on learned gradient descent recovers finer details in the normal maps, including away from the highlight, and better reproduces the colors and contrast of the input.

2.5.2. COMPARISON ON REAL IMAGES

We further examined a test set of 109 real scenes gathered by [28], composed of 33 scenes by [26] and 76 by [28]. Each scene has been captured under 9 calibrated view/light conditions, allowing us to use the central condition as input and the 8 other images as ground truth to compare re-renderings of the predicted SVBRDFs. To compute the test-time gradient for our method, we set the light intensity to be the same as during training and we assume that the camera, as well as the co-located light, are oriented perpendicularly the surface, even if this only approximately holds in practice.

Since the data and metrics are the same as the ones used by [28] for their evaluation,

we report their numbers in Table 2.3, along with our results, all of which were obtained by providing a single image as input to the different methods. Our method achieves the best results in terms of both RMSE and LPIPS, demonstrating its ability to generalize to real images despite being trained on synthetic data.

	Guo20		Zhou22	
Method	RMSE	LPIPS	RMSE	LPIPS
Des18	0.140	0.391	0.102	0.316
Gao19	0.158	0.361	0.110	0.290
Guo20	0.153	0.316	0.113	0.256
Guo21	0.161	0.391	0.103	0.303
Zhou21	0.154	0.314	0.132	0.266
Zhou22	0.133	0.286	0.093	0.216
Ours	0.122	0.276	0.084	0.211

Table 2.3: Quantitative comparison on real images, where one image serves as input and 8 other images are compared against re-renderings of the predicted SVBRDF.

Fig. 2.5 provides a visual comparison to the most recent method by [28] on six images including wood, ceramic, stone, canvas, and plaster. Our method is especially good at recovering details in the normal map, and at propagating roughness information away from the highlight. Comparisons on more images can be found in the supplementary materials.

We provide as supplemental materials a comparison with others on 93 flash photographs from [6], [24], [28], as well as images we captured ourselves with a hand-held consumer-level camera. For a fair comparison, all optimization-based methods were executed with their default light and view parameters as input. The initialization for [25] was obtained by running [6]. Fig. 2.6 illustrates some of these results. We show a re-rendering of the SVBRDF under the same lighting conditions as the input, as well as a re-rendering under novel lighting. Compared to others, our approach better reproduces the input (details in the normal map, color and contrast, extent of the highlight) and generalizes well to novel light with little residual of the highlight in the individual maps.

2.6. LIMITATIONS, EXTENSIONS AND FUTURE WORK

While we observed that learned gradient descent helps inpainting saturated pixels, Fig. 2.5 (top row), the quality of the prediction degrades for large highlights, where a lot of information is lacking (Fig. 2.7, top). Similarly, while test-time optimization helps the method generalize beyond its training set, it is challenged by input images that are too far from the expected capture conditions. The bottom part of Fig. 2.7 illustrates such as case, where the input image is captured under a light source that is far from the expected collocated flash, yielding worse results than when collocated lighting is used. An exciting direction to address these limitations is to extend our optimization framework beyond single-image capture. Specifically, Eq. 2.7 can be easily extended to compute $\mathcal{L}_{\text{reconstruct}}$ over multiple input images $\{I\}$. As a first step in this direction, we adapted our method to take 5 images

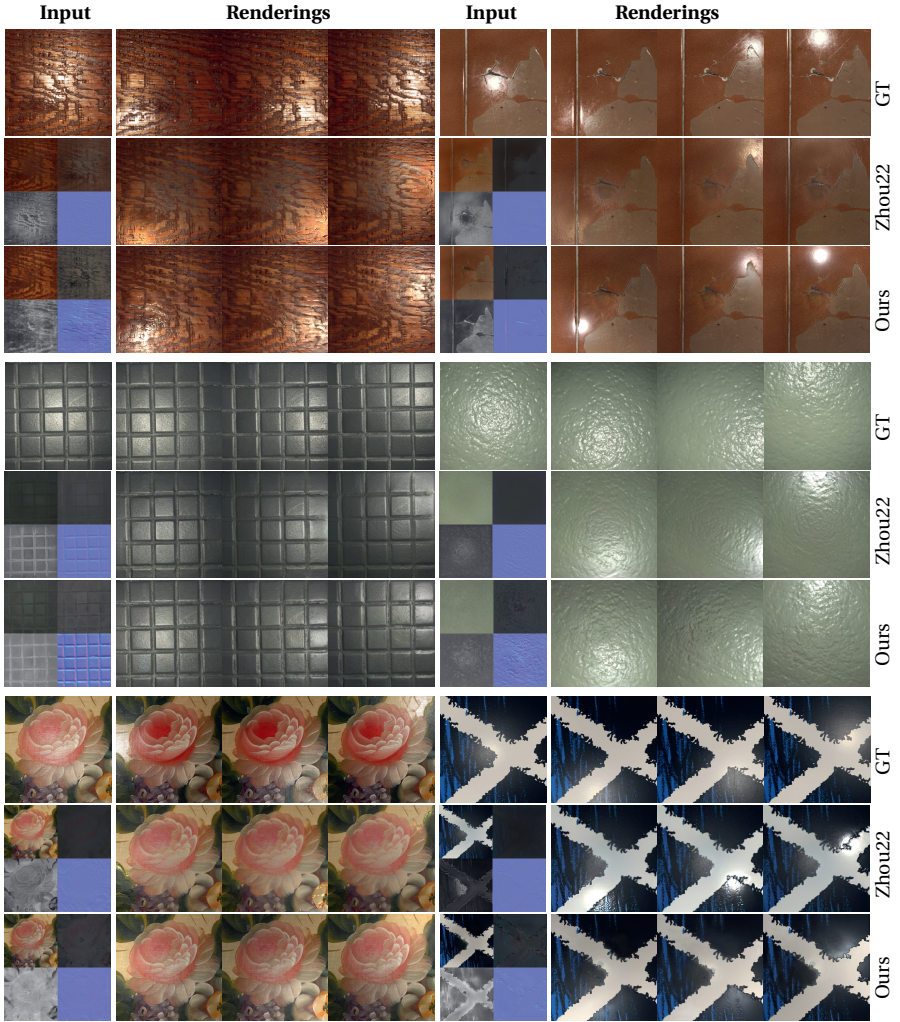


Figure 2.5: Visual comparison with [28] on real images with ground truth relighting. Note the fine geometric details in the normal maps and the propagation of spatially-varying roughness, which result in better reproduction of the ground truth appearance under novel lighting.

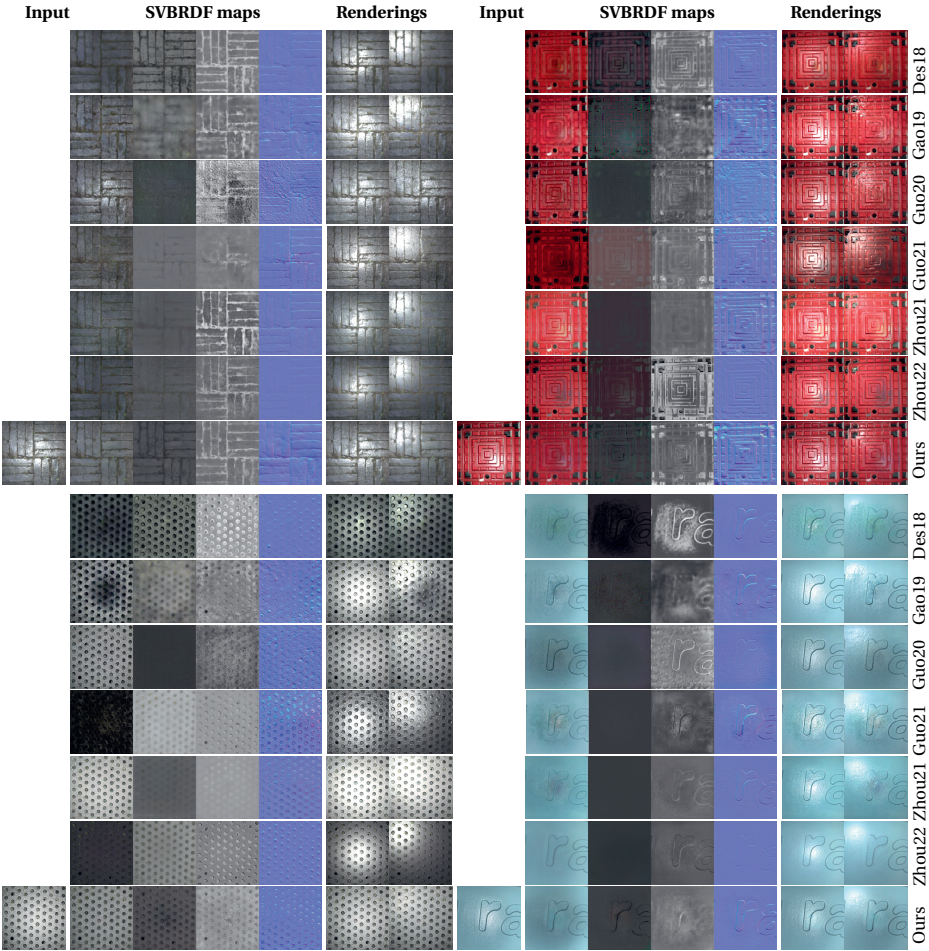


Figure 2.6: Comparison with other methods on four real images. Our SVBRDFs reproduce well the input images when re-rendered under the same lighting conditions, and produce plausible novel relighting thanks to detailed normal maps and propagation of diffuse albedo and roughness within and away from the high-light respectively.

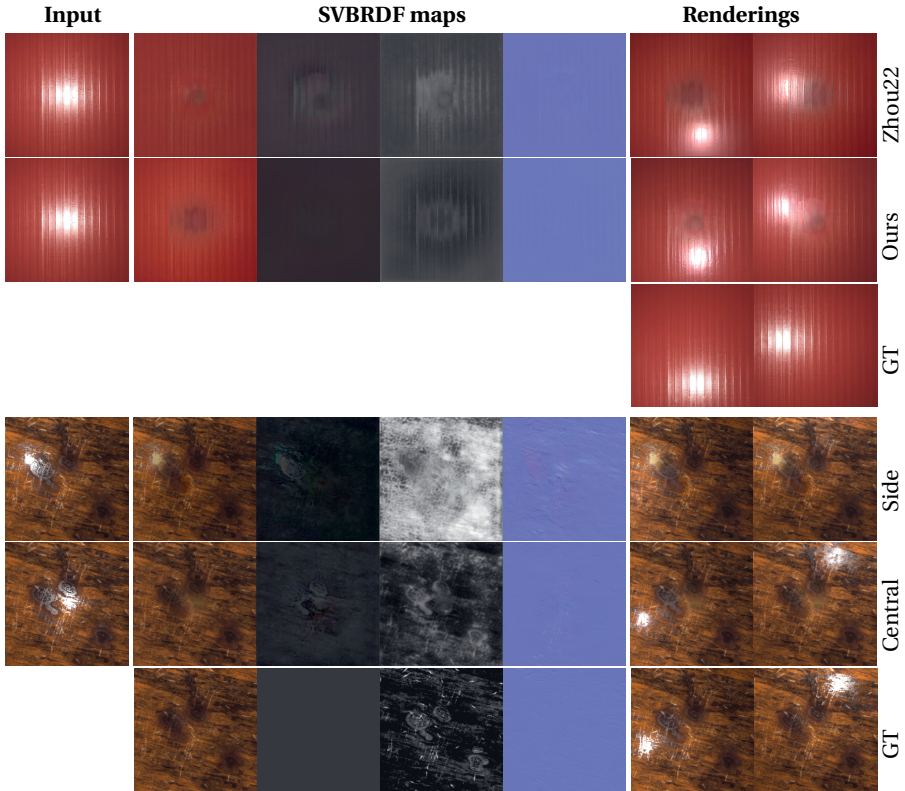


Figure 2.7: Limitations. Top: Our method struggles to inpaint saturated pixels over large highlights; a limitation shared by existing single-image methods. Bottom: Our method assumes a collocated flash light, thus, prediction quality degrades when the material is captured under a side light (bottom).

as input, taken under varying lighting and viewing conditions. Implementing this extension only requires modifying the initialisation network g_ω and the refinement network h_θ to process 5 images and 5 sets of gradient maps, respectively. While this extension increases the number of input channels of the network from 2×9 to $(N + 1) \times 9$ for N inputs, we kept the subsequent dimensions fixed (64, 64 and 9 channels). We trained this extended architecture with the same synthetic data as in Section 2.3.3, except that we rendered each SVBRDF under 5 configurations of light and view positions, which we selected at random among 9 pre-defined configurations. We used the same test set of SVBRDFs as in Section 2.4 to compare this extension (Ours-multi) to our single-image model (Ours-single) and to the state-of-the-art multi-image optimization MaterialGAN [26] (Guo20-multi).

Fig. 2.8 shows that our multi-image model outperforms [26] as well as our single-image model. In particular, having access to multiple images with different highlights helps

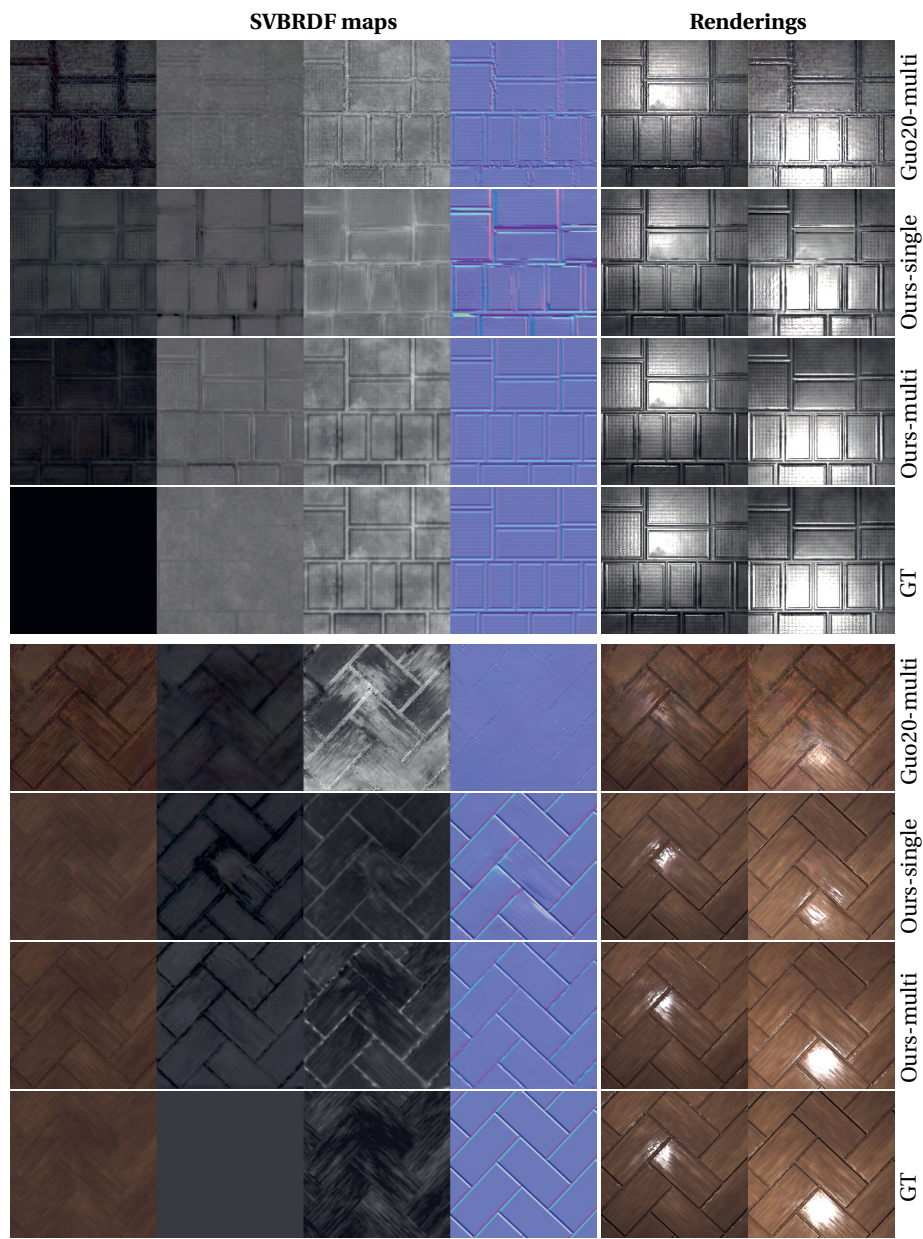


Figure 2.8: Visual comparison between MaterialGAN [26], our single-image model and our multi-image model on synthetic images.

recover material maps free of highlight residuals. Table 2.4 quantifies this improvement in terms of RMSE and LPIPS. Note that our multi-image model is only twice slower than the single-image model, while it is $600\times$ faster than the latent-space optimization of MaterialGAN.

While these preliminary results are promising, handling real-world multi-image data would require training our method with more diverse light and view configurations. Moreover, robustness to approximate light and view calibration might be achieved by treating the per-image light and view parameters (\mathbf{L}, \mathbf{V}) as additional unknowns to be optimized along with the material maps \mathbf{R} .

Method	RMSE		LPIPS		Speed/sec
	mean	std	mean	std	
Guo20-multi	0.068	0.030	0.148	0.071	261.50
Ours-single	0.057	0.032	0.107	0.070	0.20
Ours-multi	0.042	0.029	0.074	0.082	0.40

Table 2.4: Quantitative comparison of our multi-image extension (Ours-multi) against MaterialGAN [26] and our single-image model (Ours-single) on synthetic images.

2.7. CONCLUSION

Gradient descent is at the core of many inverse rendering algorithms, yet typically requires many steps and complementary regularization terms to converge to high-quality minima. We showed how *learned gradient descent* is well adapted to appearance capture, where the inherent structure of the problem can be leveraged by a neural network to perform gradient descent in a few high-quality steps. Intuitively, our recurrent neural network learns a prior about material appearance, while the forward rendering model gives a likelihood of reproducing the input. Feeding the network with the gradient of this rendering model effectively enables our method to solve for a maximum a posteriori estimate of the inverse problem of single-image SVBRDF capture. We also showed that the same formulation can be easily extended to a multi-image capture scenario. We strongly believe that a similar approach could benefit related inverse problems for which strong priors can be learned, such as facial and body capture, where feed-forward networks [56, 57] could be augmented with test-time optimization.

2.8. ACKNOWLEDGEMENTS

This work was partially funded by the NWO VIDI grant NextView, and was partially done while Adrien Bousseau was hosted by TU Delft for a year, supported by the Inria sabbatical exchange program. We would like to thank Valentin Deschaintre from Adobe Research, Jie Guo and Shuichang Lai from Nanjing University, and Yu Guo from University of California, Irvine for helping reproduce their results.

3

PERCEPTUALLY CONSISTENT INTERPOLATION USING MATERIALGAN

Interpolation in the latent space of pre-trained generative image models often results in perceptually nonlinear changes. This paper addresses this challenge in the context of material synthesis, specifically focusing on SVBRDF interpolation. We propose a method to achieve perceptually uniform interpolation by mapping linear changes in the latent space to perceptually consistent steps, ensuring that the SSIM remains close to constant between consecutive rendered images in the interpolation sequence. To this end, we introduce an optimization framework that adjusts interpolation weights such that a sequence is obtained with a uniform SSIM. Our approach is validated through experiments on a variety of material datasets and sequences. The results demonstrate effectiveness and robustness in producing interpolation sequences with consistent perceptual quality between given material image pairs.

3.1. INTRODUCTION

Generative models have made significant strides in the synthesis and manipulation of real images, enabling highly sophisticated and controlled editing capabilities. Recent developments in Generative Adversarial Networks (GANs) [17, 21, 58] and their variants [19, 59] can ensure detailed and realistic results. Guo et al. [26] developed MaterialGAN, a GAN-based model tailored to reconstruct spatially varying bidirectional reflectance distribution functions (SVBRDFs) from a small set of material appearance measurements through the use of a latent space. It generates high-quality material maps and performs well in interpolation.

However, like other GAN-based models, linear interpolation in the latent space of these models often leads to nonlinear perceptual changes, resulting in visual inconsistencies and disproportionate changes. This nonlinearity disrupts the smoothness and consistency desired when interpolating. Not only is this important for animations, also when imagining an interface for material design, interpolation sliders would otherwise not show a consistent behavior. These examples highlight the need for methods that can ensure uniform perceptual transitions. Moreover, in case of material synthesis, it is actually the rendered appearance, which is of relevance. Ensuring perceptually-uniform changes is an ongoing challenge [55, 60].

The concept of perceptual path length regularization introduced in StyleGAN2 [19] aims to smooth the latent-space traversal by penalizing large perceptual changes. However, this approach does not specifically achieve uniform perceptual similarity between consecutive interpolations. Lindow et al. [61] explored perceptually-driven techniques for interpolation, but their methods are not directly applicable to the latent space of generative models.

Our work is based on the pre-trained model MaterialGAN, and we propose a method to achieve uniform perceptual interpolation steps of material appearance between two sets of given material maps. We optimize the interpolation weights in the latent space to reach a constant perceptual similarity in the interpolation sequence. We define perceptual similarity through an image metric. Yet, we do not focus on developing a new image metric. Instead, we utilize the Structural Similarity Index (SSIM) [62] to demonstrate the effectiveness of our technique.

3.2. METHODOLOGY

3.2.1. PROBLEM FORMULATION

Utilizing the advanced features of MaterialGAN, we can obtain high-quality SVBRDF material maps R and their corresponding latent vectors z from a small number of mobile flash photographs for a specific material surface. Here, latent vectors z refer to latent vectors w^+ in space W^+ and noise vectors in space N of the MaterialGAN [26].

Given two sets of material maps, R_A and R_B , our goal is to interpolate between them to generate a sequence of $n + 1$ sets of material maps, such that the SSIM between every two consecutive images rendered under the same lighting and viewing condition remains constant. This objective ensures perceptually-uniform transitions across the sequence, providing smooth and consistent visual output.

3.2.2. LATENT SPACE INTERPOLATION

Assuming we have the latent codes z_A and z_B corresponding to the material maps R_A and R_B , respectively, the linear interpolation in the latent space can be expressed as:

$$z_i = (1 - \alpha_i)z_A + \alpha_i z_B \quad (3.1)$$

where α_i is the interpolation weight for the i -th intermediate material map, with $\alpha_0 = 0$ and $\alpha_n = 1$. The output interpolation sequence begins with the material maps of R_A and concludes with those of R_B .

To ensure perceptually-uniform steps as measured by the SSIM of the rendered images, we propose an optimization framework that adjusts the interpolation weights α_i .

3.2.3. REGULARIZATION ON INTERPOLATION WEIGHTS

To ensure the optimized weights α_i remain within the range $[0, 1]$ and are monotonically increasing, we exclude the adjustments of $\alpha_0 = 0$ and $\alpha_n = 1$ and introduce a penalty term \mathcal{P} in the loss function, which helps enforce the desired properties of the interpolation weights and is defined as:

$$\begin{aligned} \mathcal{P} := \lambda \left(\sum_{i=0}^n \text{ReLU}(-\alpha_i) + \sum_{i=0}^n \text{ReLU}(\alpha_i - 1) \right. \\ \left. + \sum_{i=1}^n \text{ReLU}(\alpha_{i-1} - \alpha_i) \right) \end{aligned} \quad (3.2)$$

where λ is a regularization parameter, and ReLU is the Rectified Linear Unit function that ensures non-negative penalties for violations of the constraints.

3.2.4. OPTIMIZATION FRAMEWORK

To achieve perceptually-uniform steps, we define an objective function that for consecutive images I_i and I_{i+1} computes the SSIM value, denoted as $\text{SSIM}(I_i, I_{i+1})$. The image I_i is rendered with material maps $f(R_i)$ generated by the generative model $G(z_i)$. Our goal is to keep this SSIM value constant across all consecutive image pairs. Formally, we minimize the variance of the SSIM values σ_{SSIM}^2 across all consecutive image pairs:

$$\sigma_{\text{SSIM}}^2 = \frac{1}{n} \sum_{i=0}^{n-1} (\text{SSIM}(I_i, I_{i+1}) - \mu_{\text{SSIM}})^2, \quad (3.3)$$

where μ_{SSIM} is the mean of the SSIM values, defined as:

$$\mu_{\text{SSIM}} = \frac{1}{n} \sum_{i=0}^{n-1} \text{SSIM}(I_i, I_{i+1}) \quad (3.4)$$

Including the penalty term, the overall objective function \mathcal{L} becomes:

$$\mathcal{L} = \sigma_{\text{SSIM}}^2 + \mathcal{P} \quad (3.5)$$

We employ gradient descent to optimize the interpolation weights α_i . The steps are as follows:

- i. Initialize α_i for $i = 0, \dots, n$ with linearly spaced values between 0 and 1.
- ii. Compute the latent codes z_i for each α_i .
- iii. Generate the intermediate images $f(G(z_i))$ using the rendering model under a (or a set of) given lighting and viewing condition.
- iv. Calculate the SSIM values between consecutive images.
- v. Adjust α_i to minimize the variance in SSIM values and the penalty term using gradient descent.

3.2.5. GRADIENT DESCENT OPTIMIZATION

The gradient descent optimization adjusts the weights α_i iteratively. At each iteration t , the weights are updated as follows:

$$\alpha_i^{(t+1)} = \alpha_i^{(t)} - \eta \left. \frac{\partial \mathcal{L}}{\partial \alpha} \right|_{\alpha=\alpha_i^{(t)}}, \quad (3.6)$$

where η is the learning rate and \mathcal{L} is the objective function defined earlier. The partial derivative $\frac{\partial \mathcal{L}}{\partial \alpha}$ represents the derivative of the loss function with respect to the interpolation weights.

3.3. RESULTS

We implemented our method using the pretrained MaterialGAN model [26]. The SSIM calculations were performed using a differentiable SSIM implementation available on GitHub [63]. The optimization process was carried out using the Adam optimizer, which adapts the learning rate η for each weight, improving convergence. We set the regularization parameter of the penalty term to $\lambda = 500$. On average, our optimization process takes about 2 minutes to complete 100 iterations with an interpolation number of eight on an NVIDIA GeForce GTX 1080Ti GPU. While the results often converge after approximately 50 iterations, we use 100 iterations in all cases for consistency and simplicity.

Comparison We used a variety of material maps to test the robustness and effectiveness of our method, which can be found in the supplemental material. Here, we compare our SSIM-optimized interpolation with direct linear interpolation in the latent space W^+ and in the noise space N . We closely analyzed the numerical quality and visual consistency of the images rendered from interpolated material maps produced by both methods across a range of interpolation numbers. We experimented with interpolation numbers of 2^i for $i \in [0, 6]$ and observed that the SSIM between consecutive rendered images in each of the interpolation sequences remains identical with our optimized method.

Fig. 3.1 presents two examples with different numbers of interpolations: one, two, and four. It is evident that our SSIM-optimized interpolation method significantly improves the smoothness of transitions between consecutive material appearances compared to the direct linear interpolation used as the initialization. Direct linear interpolation often results in perceptually non-uniform changes, causing noticeable inconsistencies

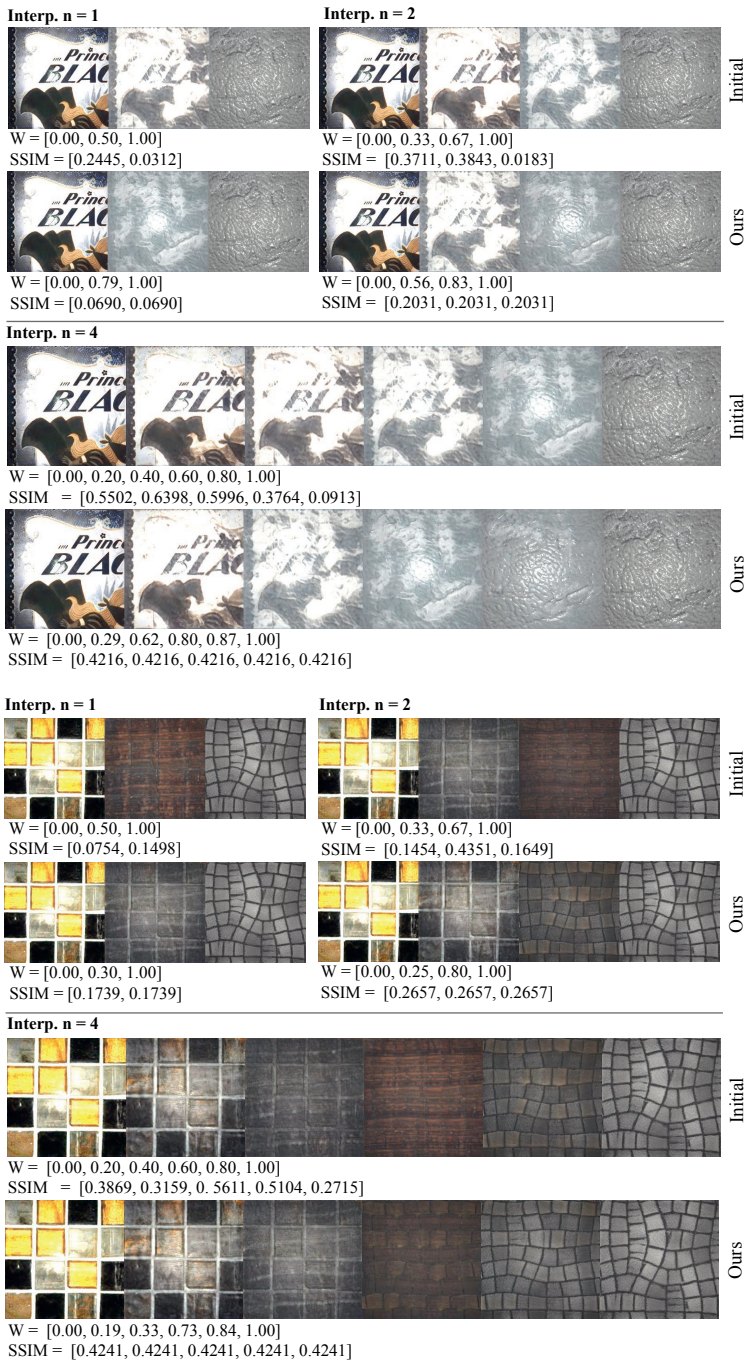


Figure 3.1: Numerical and visual comparisons of the interpolation sequences of rendered images with different interpolation numbers. For each example, we interpolate between two given sets of material images with interpolation numbers of one, two, and four.

in the material appearance. In contrast, our SSIM-optimized interpolation maintains a constant SSIM value between consecutive images, regardless of the interpolation number, ensuring perceptual uniformity and coherence. This improvement is particularly evident in sequences with a higher number of steps, where the direct interpolation method struggles to maintain visual consistency.

Quantitative evaluations also support our observations, with our SSIM-optimized interpolation achieving lower SSIM variance on average across all interpolated sequences. This indicates that our method not only produces visually appealing results but also quantitatively ensures SSIM uniformity and, thus, one can expect perceptual uniformity.

Discussion As indicated, our optimization could use a set of viewing and lighting conditions, yet, we currently optimize the SSIM of consecutive images rendered under the lighting and viewing condition for which MaterialGAN was trained. Yet, the uniform interpolation already generalizes relatively well to novel lighting and viewing conditions in the absence of strong highlights.

Fig. 3.2 shows the bottom material example from Fig. 3.1 re-rendered under two novel lighting and viewing conditions. The interpolation images re-rendered under the first novel condition maintain a similar SSIM value to the one used for optimization. However, for the re-rendering under the second novel condition, which produces strong highlight regions, the variance of the SSIM increases slightly. To achieve a uniform SSIM interpolation sequence in these special cases, the optimization could trivially integrate additional conditions, as shown in Fig. 3.3.

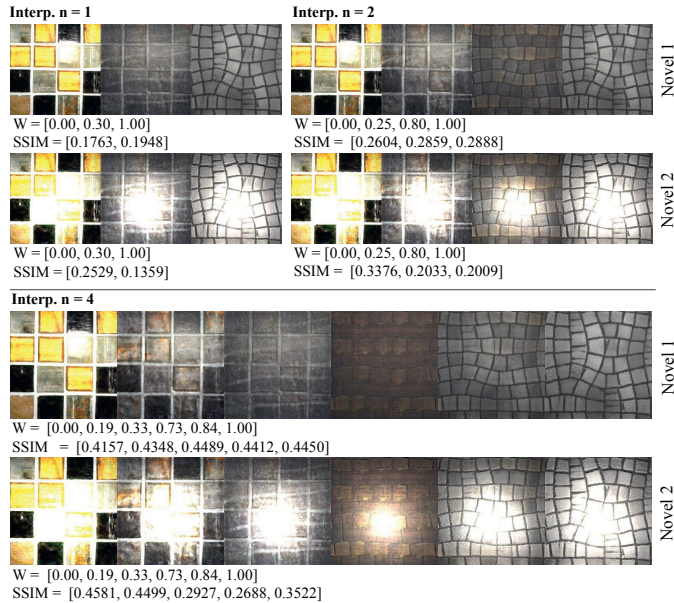


Figure 3.2: Interpolation sequences of images re-rendered under two novel lighting and viewing conditions with different interpolation numbers of one, two, four.

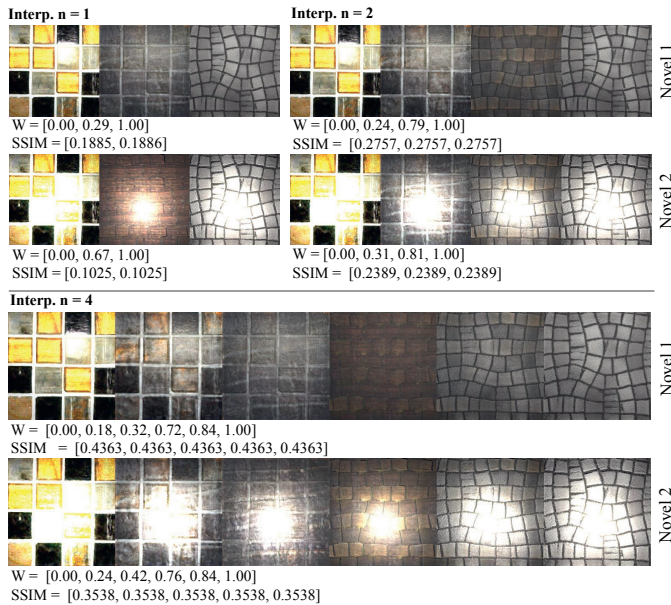


Figure 3.3: Optimization can be performed again for special lighting and viewing cases to achieve uniform interpolation.

3.4. CONCLUSION

We presented an optimization method for achieving a perceptually-uniform interpolation sequence in the latent space of generative models for SVBRDF material maps. By optimizing interpolation weights to maintain a constant SSIM between consecutive images rendered under given lighting and viewing conditions, our approach ensures smoother and more consistent visual transitions compared to direct linear interpolation. The convincing results make it a superior choice for applications requiring smooth material transitions. Future work will explore extending this method to other perceptual metrics and applications, such as video generation or image morphing, to enhance consistency in interpolated sequences.

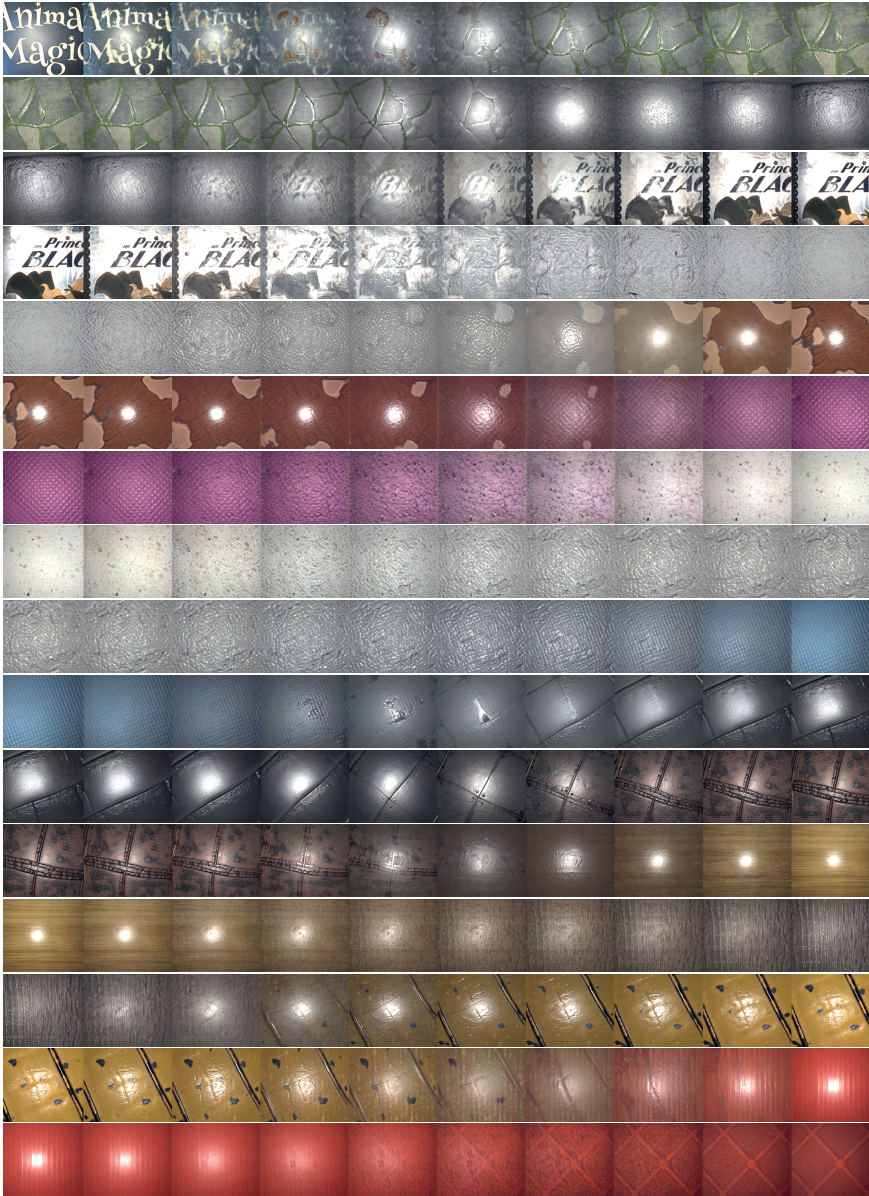


Figure 3.4: Uniform perceptual interpolation sequence across a variety of material maps with an interpolation number of ten, demonstrating the robustness and effectiveness of our method. This comparison highlights how our approach consistently maintains perceptual uniformity and smooth transitions between different material maps, regardless of the variations in their properties.

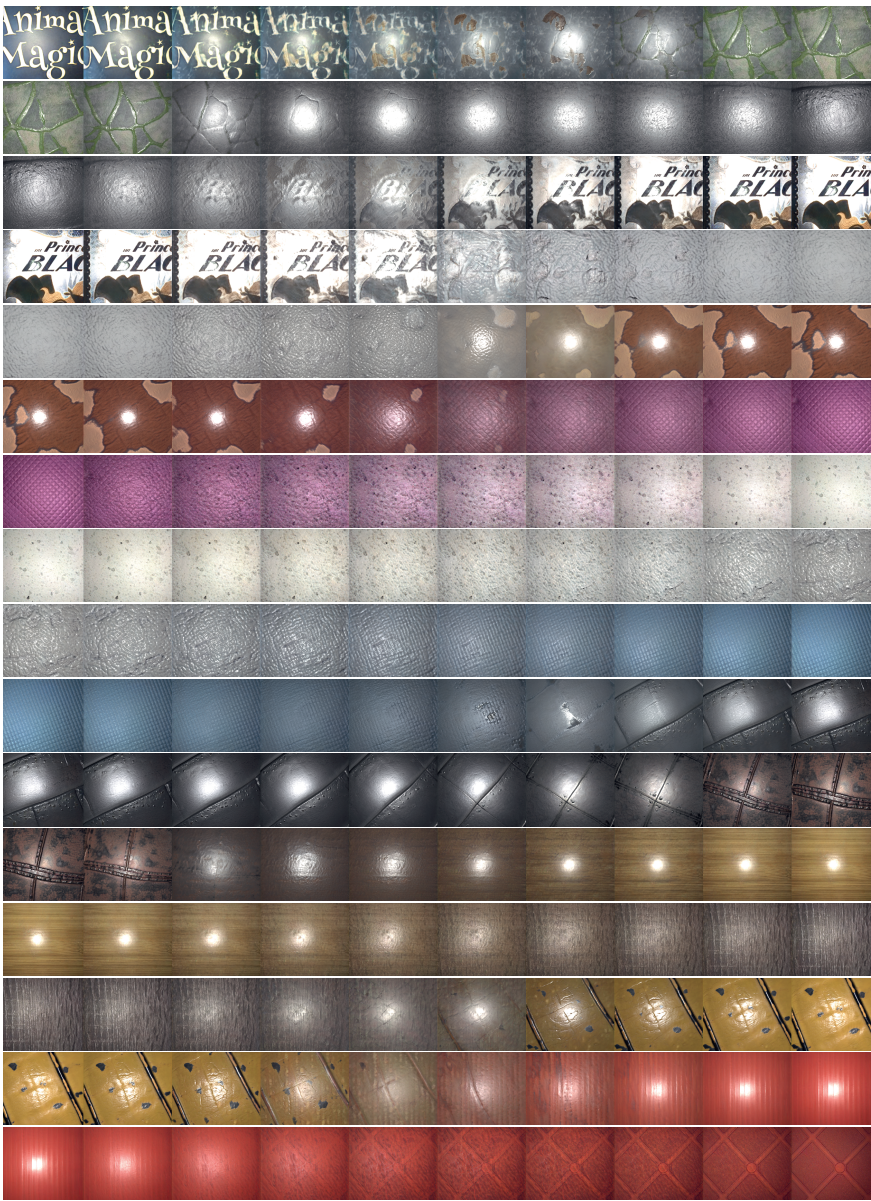


Figure 3.5: This figure presents an additional interpolation sequence, similar to Fig. 3.4, but achieved through direct interpolation within the latent space of Material-GAN. We can observe noticeable visual similarity jumps between consecutive frames in the sequence.

4

TEXTURE BROWSER: FEATURE-BASED TEXTURE EXPLORATION

Texture is a key characteristic in the definition of the physical appearance of an object and a crucial element in the creation process of 3D artists. However, retrieving a texture that matches an intended look from an image collection is difficult. Contrary to most photo collections, for which object recognition has proven quite useful, syntactic descriptions of texture characteristics is not straightforward, and even creating appropriate metadata is a very difficult task. In this paper, we propose a system to help explore large unlabeled collections of texture images. The key insight is that spatially grouping textures sharing similar features can simplify navigation. Our system uses a pre-trained convolutional neural network to extract high-level semantic image features, which are then mapped to a 2-dimensional location using an adaptation of t-SNE, a dimensionality-reduction technique. We describe an interface to visualize and explore the resulting distribution and provide a series of enhanced navigation tools, our prioritized t-SNE, scalable clustering, and multi-resolution embedding, to further facilitate exploration and retrieval tasks. Finally, we also present the results of a user evaluation that demonstrates the effectiveness of our solution.

4.1. INTRODUCTION

Texture is a fundamental visual characteristic that reflects the surface detail of an object, usually stored in an image. It plays an important role in object recognition in computer vision [64], rendering in computer graphics [65], and artistic or industrial design [66]. Most importantly, texture is recognized easily by an observer [67]. Nowadays, huge texture image collections exist, both natural and computer-generated but retrieving an ideal texture can be very difficult.

Traditional texture browsing methods tend to utilize metadata, such as keywords, captions, or descriptions, to manually cluster samples into groups. In this context, a vocabulary of 47 texture terms was created to describe a large collection of natural textures [68, 69]. This vocabulary tries to relate commonly-used texture words in English to the visual properties of textures, describing a wide variety of texture patterns. Such metadata-driven retrieval systems have several drawbacks. First, it requires considerable effort and time to manually annotate samples to build a database. Second, it is extremely difficult for a layman to describe the content of different types of textures in words. Finally, differences in language systems and bias on an individual's interpretation make it hard to reach a universal standard for properly and precisely describing a texture.

Alternatively, a texture can be procedurally generated, for example by modeling the process to form a texture from texel samples [70], or labels [71]. This process requires expertise and it is difficult to ensure a good exploration. Recently, a semi-procedural approach [72] has been presented, working with an exemplar and a label map to then create new variations. The chosen input is crucial to the process and a label map is not always available. Even in dedicated solutions for material generation, such as Substance [73], browsing for input textures remains a common task, which we address.

In this paper, we propose an efficient visual navigation tool that can organize an unlabeled texture collection in a semantically meaningful way. This enables users to efficiently navigate to a target texture. Potentially, this can also ease the task of labeling large collections of textures with semantic terms if wanted.

In our framework, we utilize the activations of the penultimate dense layer of a pre-trained image recognition network, VGG16, to represent the input images [9].

This high-dimensional activation vector represents multi-scale, and thus semantically-meaningful, features of the input and has been shown to link to texture information [74]. To visualize the distribution of textures based on these feature vectors, we use t-SNE to embed them into two dimensions. For easier navigation, we propose a modification of the original t-SNE algorithm, which we call prioritized t-SNE, which can modify the resulting 2D embedding to facilitate exploration of a chosen image neighborhood.

We also propose several navigation methods including flip zooming, scalable mean-shift clustering, user interactive selection, and an image-based retrieval algorithm to accelerate texture retrieval, as shown in a user evaluation. As texture structure is usually independent of its color palette, recoloring [75] is a very convenient way to achieve diverse and plausible alternatives of the selected target and is integrated as well for the completeness of our system.

Compared to traditional state-of-the-art image retrieval systems, our framework targets texture images and offers a simple, user-friendly, and efficient interface for retrieval, navigation, and re-coloring. More importantly, it does not require any manual pre-processing

or labeling. This is a key point, as labeling abstract textures is a hard problem, requiring the use of specific language that is not easily interpreted by end-users. Our system organizes and displays textures in a way that greatly expedites retrieval and exploration tasks.

The main contribution of our work is an intuitive retrieval and exploration system for large unlabeled collections of texture images that spatially groups textures sharing similar features. Our system integrates several targeted tools, and while it builds upon some known techniques (often to leverage familiarity, e.g., selection tools and 2D interaction), it also introduces novel solutions to control the embedding.

The rest of the article is organized as follows. We first review the related work (Section 4.2), before elaborating on our approach (Section 4.3), including the embedding and navigation solutions, as well as the proposed interface (Section 4.4). We then evaluate our work and compare to existing methods (Section 4.5) before we conclude (Section 4.6).

4.2. RELATED WORK

An abundance of work focuses on image retrieval but most is not geared specifically towards textures. Metadata-based image retrieval systems have been successfully applied in most web-based image search engines [76]. Nevertheless, these methods can still produce a lot of unwanted search results.

Color-based retrieval methods compute color histograms to guide the search task [77, 78] but are not focused on the actual content of the texture. Nevertheless, in our context, the structure of the images plays a much larger role and color can even be adapted in a post-process. Content-based image retrieval (CBIR) [79] relies on combinations of colors, textures, local geometry, or any other information that represents and can be extracted from images. In this situation, it is common to rely on example images when searching, which are not always available and these systems do not lend themselves well to a fine-grained search. Different metrics can be used to compare color and texture features [80] and many measures have been proposed in the past [62, 81–87]. Kokare et al. [88] explore the impact of different similarity measures in distance-based automated retrieval tasks.

Text-based queries are very common but not easily applicable to texture search. This also holds true when involving a hybrid image retrieval system that relies on keywords and images [89] or an interactive browsing solution [90], due to the large amount of manual effort in creating the text or image descriptions, and also due to the difference in each individual's interpretation. We compare to such an approach in our user study [68]. Sketch-based image retrieval methods utilize sketching to query target images [91]. While very powerful when focusing on important lines, e.g., when navigating sketch collections, not all properties of textures can be well captured. It is nonetheless a useful element in case that pronounced features are available and is also employed in our system.

With the tremendous success of convolutional neural networks (CNNs) on the ImageNet data set, image classification has received great attention [92], [93], [9]. These systems learn complex features that outperform hand-crafted ones. Hand-crafted features (e.g., SIFT [94] and Gaussian mixture models [95]) were not used since available pre-trained network features have proven more versatile in recent years [96, 97]. Their use for the task of object search within an image collection has been explored with promising results

[98–100], but not oriented towards texture retrieval. Danon et al. [101] proposed an unsupervised learning method towards a metric of similarity, leveraging the fact that the similarity of two patches can be learned from the prevalence of their spatial proximity in natural images, which does not hold for textures with spatially homogeneous structures.

Good examples in image retrieval with deep networks exist [96, 102–105] but require specific inputs, i.e., query images or keywords, and they do not target navigation/exploration. When a database of labeled images is available, networks can be trained for other applications as well, such as recognition [106], segmentation [107], or procedural texture generation [108]. Solutions to better address perceptual texture properties exist [109, 110] as well but these systems are based on similarity matrices generated by capturing subjective human judgments. This is time-consuming, costly, and would have to be applied for each newly inserted item. This limits their generalization to large collections or different databases. Our solution is inspired by these approaches, as we leverage the powerful semantic feature extraction capabilities of deep neural networks in combination with modern visual data analysis.

Visual exploration of embedded datasets using t-SNE is used extensively in the medical field, for example for mass cytometry data exploration [111]. Moreover, exploration methods based on hierarchical stochastic neighbor embedding [112] have been proposed [113, 114]. Nevertheless, in these cases, the focus is the exploration of relations between data points, which does not translate well to texture database exploration and retrieval tasks. The levels of the HSNE hierarchy are a subset of the dataset, and thus using such representation would hide information from the user. Our prioritized t-SNE approach follows a similar goal, has a simpler definition that does not require any preprocessing, continuously adjusts the relevance of each embedded point, and does not remove any images from view.

In the context of interactive exploration and browsing of image collection, previous work [90, 109, 110] proposed several visualization concepts, such as cylinder displays. However, these approaches are mostly intended for gaining an overview. Coarse semantic information-based methods are proposed to enhance the visualization and exploration. Yang et al. [115] and Mizuno et al. [116] propose to use multidimensional scaling (MDS) [117] when computing the similarity between images for visualization, but either keywords are involved in the annotation and search process [115] or the resulting embedding is less efficient than t-SNE [116]. MDS was further applied with weights to emphasize items of interest [118], while in Projection Explorer for Images [119, 120] (PEx-Image) it was used for image embedding. Worring et al. [121, 122] use pivot tables to visualize and explore the multimedia images over user-supplied metadata. To utilize the higher-level semantic features of the image, Xie et al. [123] propose to train an image captioning model based on existing semantic keywords. Previous work dealing with metadata agnostic CBIR systems includes the work of Rodden et al. [124], who propose a grid visualization of images based on an MDS arrangement with color histograms of image sections as features. Tian et al. [125] propose a series of tools to create a virtual reality system for CBIR, based on a weighted MDS distribution over manually crafted color and texture features, as well as available text metadata. Gomi et al. [126] introduce a hierarchical CBIR system, with a hierarchical arrangement of images based on aggregated color values of image regions, and also available keywords associated with each image. Finally, Schaefer et al. [127]

present an overview of different image collection browsing methods, mostly designed for the case of photography collections, highlighting their own sphere and honeycomb arrangement methods based on an MDS embedding over median color information. As absolute color is not a suitable feature for texture browsing, these methods are not well suited for our application context.

4.3. OUR APPROACH

Our work aims at facilitating navigation to a desired texture within a large database without any metadata. To this end, we position the textures in a 2D layout, following an embedding of their high-dimensional feature vectors, and propose interaction mechanisms to explore the dataset. The workflow is illustrated in Fig. 4.1. To support navigation, we provide solutions to influence the embedding at multiple scales. Features can be emphasized by allowing them to span a larger portion of space or irrelevant regions can be shrunk in real time. Additionally, a user can make use of clustering and image-based search (using existing images or sketches) to narrow down the search for texture structures. Finally, the colors of the target can be adapted to meet the wishes of the user. In the following, we describe the details of our solution.

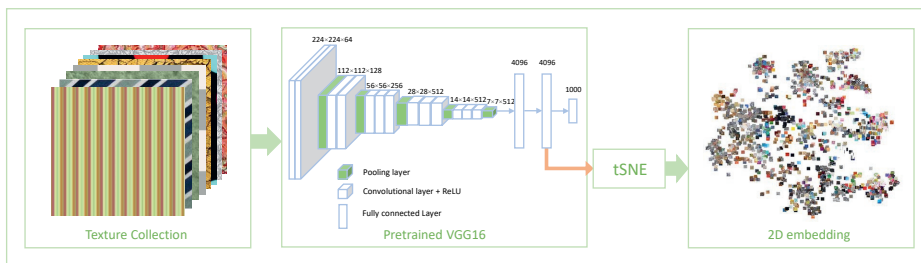


Figure 4.1: Workflow of our manifold embedding: each image in a texture collection is run through the VGG16 network to obtain a feature vector extracted from the penultimate layer of the network. These features are then embedded in 2D space using the t-SNE algorithm.

4.3.1. EMBEDDING

As discussed in Section 4.2, CNNs have proven to be a very powerful tool for extracting features from images. Typically, the latent vector that results from the penultimate layer of a network encodes relevant features that are important for classification. While it might sound attractive to build a specialized CNN for an image collection, it is a very expensive and time-consuming task, and suitable labels for training are typically not available.

Motivated by previous work in the area [9, 64, 74], we use (pre-trained) VGG16, which is a deep convolutional image classification network that was trained with ImageNet [128], a large image database. Evidence has been presented that deep convolutional image-classification networks trained on ImageNet rely on identifying texture rather than shape [64] and are therefore well suited for our tasks. The specialization of its descriptors for

texture recognition is also leveraged in style-transfer approaches [129]. The first layers of such networks encode filters for simple patterns, and later layers recognize progressively more complex textures [74]. Nevertheless, the last layers are strongly linked to the actual categories recognized by the network and are therefore less useful for our application. Therefore, we selected the descriptors of the penultimate fully connected layer given their generality and efficacy. As an alternative, we also tested the use of layers from AlexNet [92] but found them to be less effective in differentiating texture. We believe this is because its shallower architecture and large receptive fields force early layers to specialize more strongly on the recognition categories. In contrast, VGG16 is deeper and has smaller receptive fields, which recognize small abstract patterns at early layers which combine in later layers to recognize more complex patterns [74].

We scale all input images to a resolution of 256×256 to provide as an input to the network. The second fully connected layer, which is the optimal layer for retrieval [130], outputs a latent vector of 4096 dimensions, which we associate with each image.

The high dimensionality of the feature vector makes it impossible to directly visualize an organization of the images in such space. For visualization and interaction purposes on a 2D screen, a two-dimensional positioning is most suitable. We use t-SNE [15], a dimensionality-reduction method, to embed the latent vectors. Fig. 4.2, shows examples of the resulting embedding for the Describable Textures Dataset (DTD) [68] and the UIUC Texture Database [131]. t-SNE has the advantage of being non-linear and preserves neighborhood relations, which is particularly desirable for exploration purposes. Within a collection, some textures may manifest as transformations of others, such as rotation, scaling, or coloring. While it would be possible, we do not enforce transform invariance, and thus they are not necessarily mapped to similar locations by the t-SNE algorithm. This is by choice, since these aspects can play into a semantic meaning (e.g., tiger stripes are typically vertical).

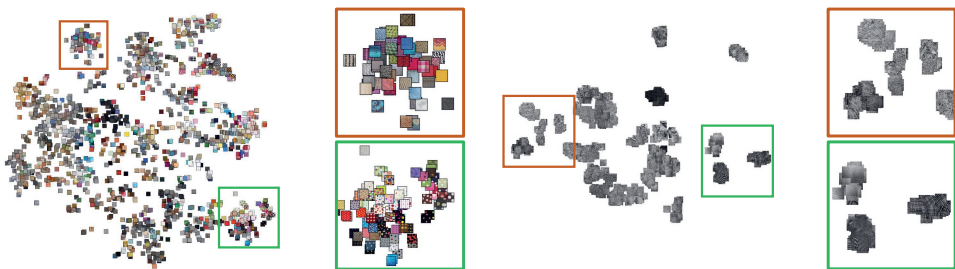


Figure 4.2: The 2D embedding distribution of the DTD collection (left) and the UIUC collection (right). The colored sections zoom in some clusters in the distribution.

Although non-convex and non-deterministic, the embedding of t-SNE is efficient, fairly stable, and has proven useful in many applications. The tunable perplexity parameter was set to 30 heuristically for all our examples, based on the recommended range (5-50). Moreover, the t-SNE algorithm is run for a maximum of 2000 iterations. Both the computational and the memory complexity of t-SNE are $O(n^2)$ but optimized implementations exist [132, 133].

4.3.2. PRIORITIZED T-SNE

The original t-SNE formulation attempts to create an embedding that maintains similar neighborhoods as in the high-dimensional space. It does so by minimizing the Kullback-Leibler divergence between the joint probabilities of the distances of the low-dimensional embedding and the high-dimensional vectors [15]. Instead, we would like to offer the possibility for a user to emphasize certain groups of textures (or features) that should then make use of more space in the embedding in a continuous manner to avoid generating artificial clusters. Similarly, textures that are of less relevance should use less space. To achieve this, we propose the prioritized t-SNE embedding. We assume that each of potentially multiple user-selected samples (textures) S has a weight $W \in [0, 10]$. The embedding region around this sample will be enlarged if $W \geq 1$ and shrunk if $0 \leq W < 1$. For a single selected sample S , we define weights w_i for all other samples as:

$$w_i = 1 - (1 - W)e^{-d_i * \max\left(W, \frac{1}{W}\right)} \quad (4.1)$$

where d_i is the distance between sample i and the selected sample S . When several samples are selected, the weights produced for each of the samples (Eq. 4.1) are multiplied. Once the weights are determined, we compute the joint probabilities q_{ij} of map point y_j and map point y_i in the low-dimensional space by

$$q_{ij} = \frac{(1 + w_i * w_j * \|y_i - y_j\|^2)^{-1}}{\sum_{k \neq l} (1 + \|y_k - y_l\|)^{-1}}. \quad (4.2)$$

The embedding then uses standard t-SNE with these modified joint probabilities. To accelerate convergence, we initialize q_{ij} with the originally computed t-SNE counterpart. Fig. 4.3 shows an example using two selected samples.

4.3.3. MULTI-SCALE REPLICATION

Some textures manifest different visual features at different scales, as shown in Fig. 4.4. The t-SNE algorithm clusters such images based on an overall dominating feature, while a user might have been interested in a feature at a different scale. For example, imagine a piece of cloth with a very fine structure that forms a larger-scale pattern. On the one hand, it could be grouped with cloth textures. On the other hand, its large-scale pattern might be more suitable to be represented by other textures. Especially large-scale (i.e., low-frequency) features can have an important impact, since a user will initially see an overview of the collection, where the images are small, and mostly large-scale features are visible. In this sense, we would like an image to be embedded adjacent to images sharing its small-scale visual features, but also adjacent to images sharing its large-scale features. To resolve this conflict, we create additional embedding positions for each image that correspond to image versions where small-scale features are removed, and we replicate the images at those positions when necessary. Specifically, for each image we generate two blurred versions via convolution with a Gaussian kernel, approximating human feature perception, with a standard deviation of 5 and 9 pixels ($\sim 2\%$ and $\sim 3.5\%$ of the input resolution). These kernel sizes were empirically chosen, since smaller kernels had little impact, and larger ones quickly resulted in converging feature vectors for all



Figure 4.3: Original embedding (left) and modified embedding using prioritized t-SNE (right). The texture circled in red was given a higher priority with $W = 8$ in Eq. 4.1, resulting in larger distances between images for all similar images, which are also enlarged to improve visibility. Conversely, the image circled in blue was given a low priority with $W = 0.2$ in Eq. 4.1, contracting similar images together.

images. Using several scales enables a successive removal of small-scale features (see Fig. 4.4).

After obtaining three sets of latent vectors (original texture collection and the two blurred versions), we embed these with t-SNE in parallel.

In this case, we initially get an embedding with three times the image samples of the original collection. Nevertheless, the 2D embedding of the different versions for most images ($\sim 95\%$) remains very similar. This means that the original latent vector for these images is dominated by the large-scale image features. To avoid unnecessary clutter, we only replicate images when their distance in the 2D embeddings exceeds a given threshold distance (25% of the embedding diagonal), as the texture then clearly exhibits distinct multi-scale features. If not, only the original texture is kept, as it already well reflects its large-scale features on its own.

4.3.4. CLUSTERING

While the presented embedding can successfully group similar textures, it fails to give a good overview due to too much information being displayed at once, especially with a large database. In consequence, we propose a clustering mechanism to ease navigation and to allow the user to identify categories in the embedding to then focus on clusters reflecting the desired properties, e.g., stripe patterns. To produce clusters, we apply Mean-Shift clustering [134] on the embedded feature space. This clustering algorithm has the advantage that it is guided by a single parameter, the scale at which the elements are grouped. Further, as the mean-shift can be computed relatively quickly by discretizing the

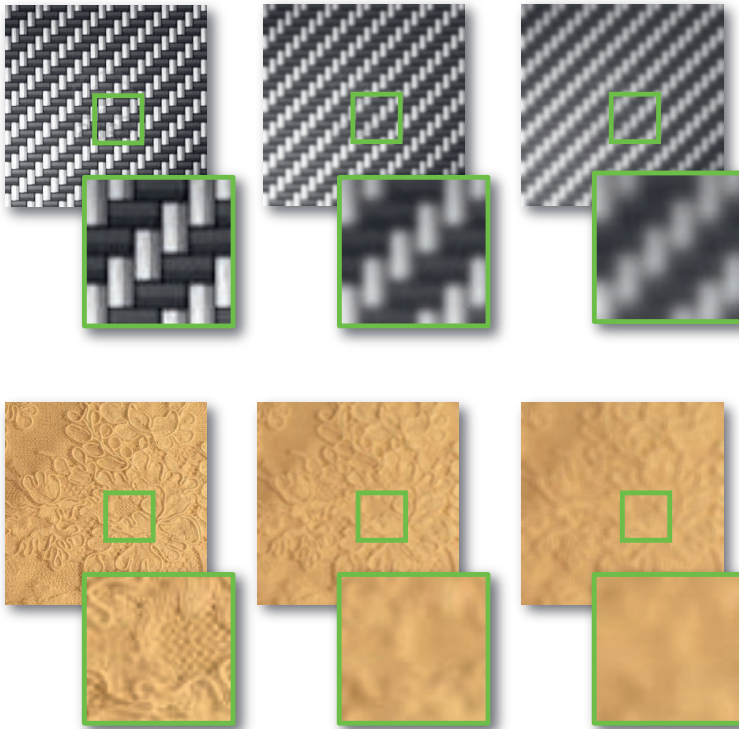


Figure 4.4: Features at different scales (original and 5, 9 pixel blur). Low scale detail is successively removed.

space [135, 136], the scale can be chosen interactively. Given clusters of similar textures, we can enhance them with a representative image (*landmark*); the image nearest to the centroid of each cluster. Fig. 4.5 shows three examples of clustering visualization using landmarks for 6, 10 and 20 clusters. The center landmark image is surrounded by eight more images, which are representatives for the farthest texture in the corresponding direction (horizontal, vertical, or diagonal) of the embedding that still belongs to the cluster, as seen from the centroid. Hereby, the observer gets an idea of the texture appearance at the boundaries of the cluster.

4.4. INTERFACE

The interface of our retrieval system is shown in Fig. 4.6. It consists of our two major components: the semantic embedding window and the priority texture selection tool. Further, we see an image-based search tool with an optional sketching board, and a recoloring tool. The latter is not a novel technical contribution, but is present for completeness. For

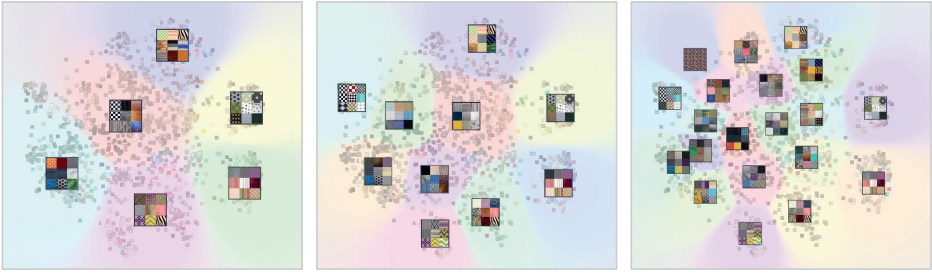


Figure 4.5: Overview using mean shift on the 2D embedding distribution resulting in 6 (left), 10 (middle) and 20 (right) clusters. Landmarks summarize the cluster information by showing representative cluster images, and background color indicates cluster extent.

4

our tests, we use the textures in the DTD collection. It is a database with 1518 images, grouped via 47 words (terms/categories) inspired from human perception.

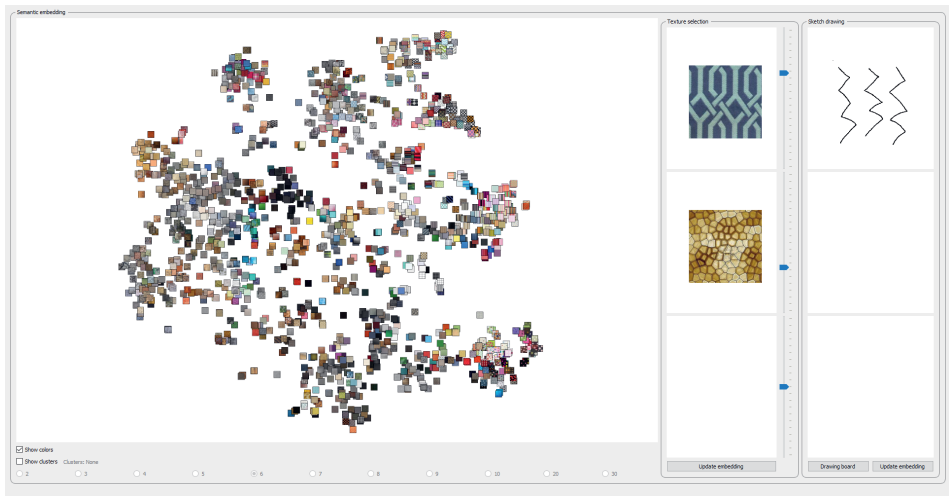


Figure 4.6: Image-retrieval Interface. The images in their semantically embedded positions (left). Controls for weighing selected images for our prioritized t-SNE (middle) and a sketch interface (right).

The semantic embedding window (Fig. 4.6 and Fig. 4.7) presents the collection of images to the user, positioned according to the computed embedding. This provides a holistic view of the image set and shows the global semantic transition among textures of different styles. The user can freely navigate through the overall display and change its zoom levels with the mouse. The texture that is pointed at with the mouse cursor is displayed in a large tooltip for a better detail visualization. In this window, the user can additionally enable the display of clusters to help obtain an overview of the features in

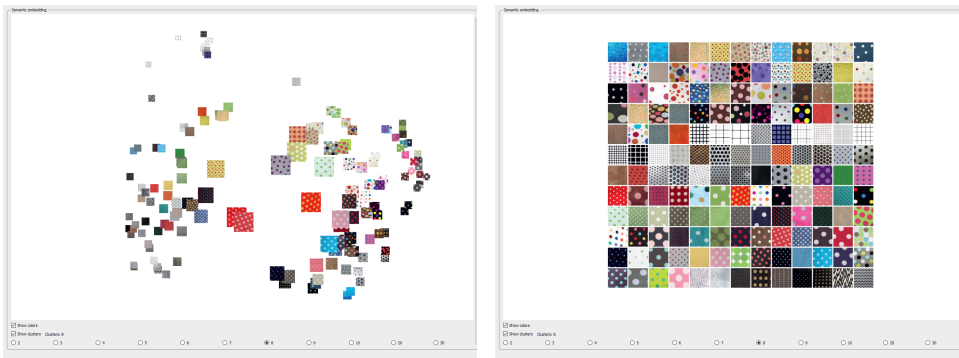


Figure 4.7: A selection can be shown as interactive flip-zoom interface (left) or a tiled interface (right).

4

different parts of the embedding (Fig. 4.7, left). The user can select the number of clusters and their landmarks and representative images of the clusters are shown (Section 4.3.4). Furthermore, the user can select a cluster or a rectangular region to restrict the view to only the chosen subset of images. For this restricted view, the user can then opt for a flip-zooming visualization style, or a tiled visualization, to prevent image overlap; this is shown on the right side of Fig. 4.7. Lastly, as we support recoloring, images can be displayed in grayscale to avoid user bias based on color. Nevertheless, loss of contrast during this conversion can hide important texture details, which is why both options exist. Note that a detail-preserving decolorization scheme (e.g., [137, 138]) could be applied as future work.

The priority texture selection tool allows a user to select one or more images, and assign a priority to them, which is then used to update the embedding using our prioritized t-SNE (Section 4.3.2). This updated embedding will show images related to high-priority selections at a larger scale and images related to low-priority ones at a smaller scale, as shown in Fig. 4.3.

The image-based search tool is added for completeness of the system and allows a user to search based on a user-provided image, provided via a file or as a sketch. The simple sketch tool supports controllable pen color and width. We rely on VGG16 to extract the corresponding latent vector of the user image, and find the closest texture image in feature space. Fig. 4.8 shows an example of user sketches and the corresponding result in the DTD collection. Once a texture has been identified in this way, it is possible for the user to highlight it in the semantic embedding window as an additional landmark (Section 4.3.4) or to change its weight to influence the embedding.

The recoloring tool (Fig. 4.9) allows a user, as a final step, to change the color of a selected texture with the subsequent option to save it to a new file for use outside of our application.

The recoloring algorithm is optimized based on [75] to simplify user interaction. We cluster the colors in the original image using a k-means algorithm, where we let the user select the number of clusters. The color of each cluster center is shown to the user, who can use a color picker to change them. Once a new cluster color is selected, we transfer

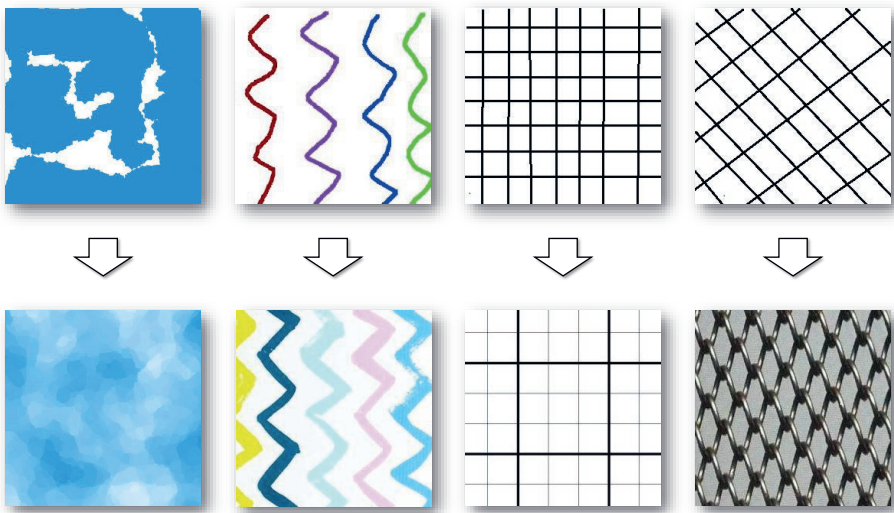


Figure 4.8: Examples of drawn sketches (top) used for retrieving a similar texture in the database (bottom).

the same color offset to all cluster pixels.

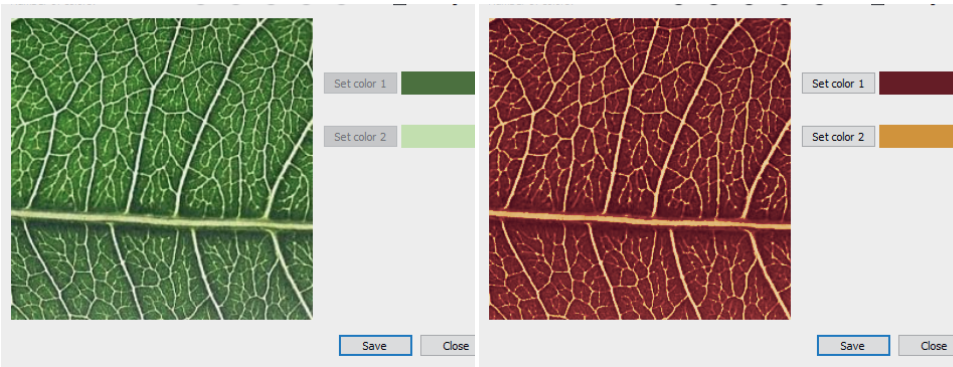


Figure 4.9: The image recoloring tool showing the original image (left) and a different choice of coloring using red tones (right).

4.5. EVALUATION

We conducted a user evaluation to validate the effectiveness and advantages of our method. We wish to perform two evaluations: comparison with alternative methods, and features evaluation. In the first part of our study, we compared against displaying images on a grid (using thumbnails), which is the standard file-system solution. It is the

visualization that users typically have the most experience in using. It also requires no metadata (which also holds as is the case for our system). Additionally, we compare to text-based solutions [68]. We also evaluate the four features compared to only using our basic *overview* mode, *multi-scale replication*, *clustering*, *prioritized-tSNE*, and *image-based search*. The first two features aim at general use cases and evaluation of them requires no special settings of the target. Therefore, they were integrated into the first part of the study, acting as additional comparisons. For the latter two features, they aim to improve the retrieval in cases where the number of images is very large, and thus were evaluated separately in the second part of the study.

In the evaluation, a Windows 10 system with an Intel i7-8700 CPU, 16GB RAM, and an NVIDIA GeForce GTX 1080Ti GPU with 11GB of VRAM was used. The GUI was displayed at a resolution of 1920×1200 . The system was implemented in C++ with machine-learning components running in Python.

In total, 16 users participated in the evaluation. They had no prior experience with our interface but were Computer-Science students. Additional details regarding the evaluation are given in the supplementary material.

4.5.1. COMPARISON WITH ALTERNATIVE METHODS

Grid view For the first task, the users were asked to retrieve four given textures from the same collection (DTD, containing 1518 texture images), using two different systems. One was a grid view, which consisted of a standard Windows 10 file explorer dialog with large thumbnails, where images were randomly ordered. The second was our interface, where we asked the users to search the textures starting from the overview panel and were free to use all the features our tool provides. Fig. 4.10 shows the images for this task.

As an introduction to our system, the users were shown a ~1 minute video that illustrated how our interface works (available as supplementary material). The users were not given time to familiarize themselves with the tool. Instead, their first search task was their first interaction with our tool.

Results (Fig. 4.11) show that the average time to retrieve the target textures with the grid view (427 seconds) is substantially reduced by using our interface (133 seconds); a more than three-fold improvement even when applied by an inexperienced user.

It is also noteworthy that all participants completed the task faster with our system than with the grid view. Ten users performed faster on each individual search query using our system. The rest performed faster on three out of four search queries using our system but even when just looking at the participants of this second group, the average time for completing the task using the grid view (496 seconds) was still significantly slower than using our system (159 seconds). Furthermore, one of the users failed to retrieve the final target texture via the grid view after searching for more than 17 minutes (only the time until they desisted was accounted in our averages), while this user succeeded via our method in less than three minutes.

With statistical hypothesis testing, retrieval time with our interface is shorter than with the grid view a p-value lower than 0.00001, indicating high statistical significance, and shorter by 4 minutes with a p-value of 0.0024, also statistically significant.



Figure 4.10: Retrieval-task Textures for method comparison. One texture was replicated due to our multi-scale method.

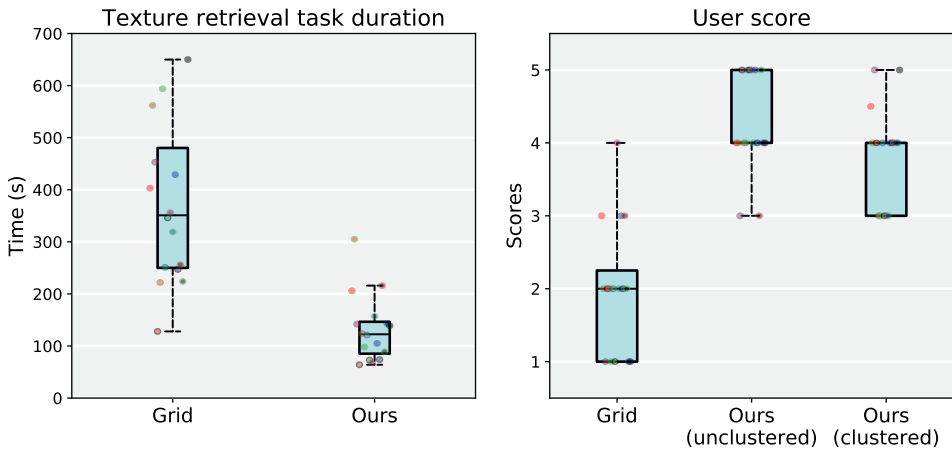


Figure 4.11: Retrieval time in seconds (left) and user score (right) for the standard grid view and our texture retrieval system with and without the clustering overlay.

Multi-scale Within this task, we also evaluated the usefulness of our multi-scale replication (Section 4.3.3). One of the textures that we chose as a target for the search (leftmost Fig. 4.10) was replicated via our scheme because of its multi-scale appearance. In 12 of the 16 cases, the users found the texture via the embedding position corresponding to one of the low-scale (blurred) versions, averaging 27 seconds. The rest obtained the target in its original scale embedding location, averaging 38 seconds. This coincides with our hypothesis that different users first notice features of different scales, and our multi-scale replication takes advantage of this phenomenon to facilitate retrieval in either case.

Clustering We asked the participants to repeat the retrieval while enforcing the use of the cluster view of the embedding, to assess the usefulness of the feature. In this case, we randomly rotated the embedding, to prevent users from relying on location knowledge from the previous retrievals. Users were then asked to assess their interaction satisfaction on a Likert scale of 5 for the grid view interface, and our interface with and without using the cluster view. Users rated our interface on average with a 4.25, and the grid view

satisfaction with an average of 2.0. (Fig. 4.11, right). In general, user preference for the *overview* exploration was mixed, as from the 16 participants, 9 preferred the non-clustered view, 4 preferred the clustered view, and 3 gave them the same score.

Fig. 4.12 gives further insight into the times required to complete the retrieval task with and without requiring the cluster view.

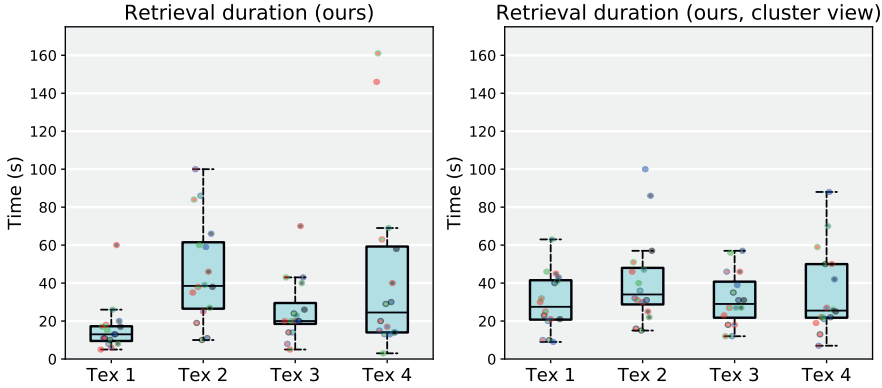


Figure 4.12: Individual retrieval time in seconds for the four textures of the initial retrieval (left) and when asked to repeat the task using the cluster view (right).

In that figure, it can be noted that Textures 2 and 4 required slightly longer search times. For Texture 2, this can be attributed to the fact that it does not exhibit immediately recognizable large-scale features. Indeed, our embedding system groups the texture together with lace-patterned textures, which can only be easily identified when zooming-in. In the case of Texture 4, we believe the dark colors and low contrast of the image may make it less identifiable from a zoomed-out view.

Those users that preferred the non-clustered view commented that the presented clusters did not always meet their expectations, since they grouped textures in an unexpected way. Our clustering implementation is based on the low dimensional 2D position of the textures, but could be improved by taking the high-level feature vector into account as future work to improve the quality of the resulting cluster display.

Text-based retrieval To test keyword use for texture selection, we provided users two out of the 47 labels from DTD: *porous* and *interlaced*. We asked users to retrieve one texture representative of each word, with no time limit. Fig. 4.13 shows the retrieved textures, with a red highlight for those that have the search keyword as a description label in the DTD database. For the first keyword, *porous*, only half of the participants found a texture that carried the corresponding label in DTD, but all textures exhibit porous features. For the second keyword, *interlaced*, only one of the retrieved textures was labeled as such in the database. Nevertheless, as seen in Fig. 4.13, all textures feature interlaced patterns. This highlights the difficulty of texture retrieval via labels, as user interpretation of a keyword can vary greatly. In contrast, a texture retrieval system navigated in a semantic way, such

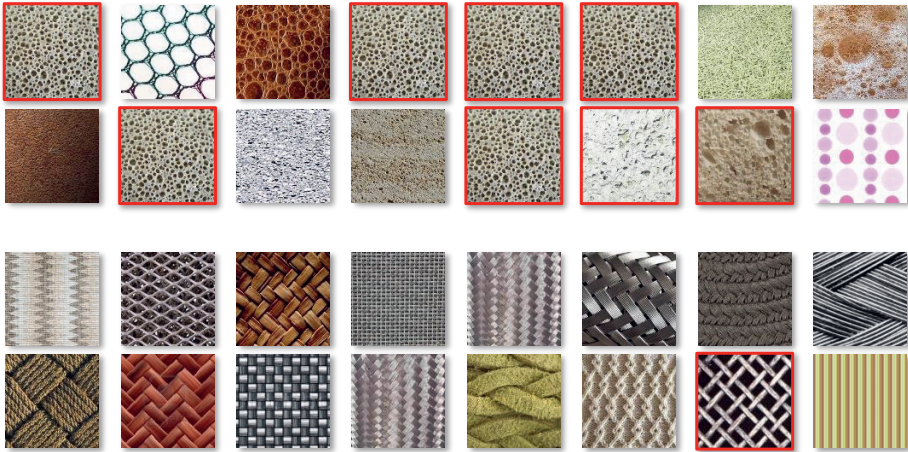


Figure 4.13: Images retrieved by users using the concept *porous* (top) and *interlaced* (bottom). Only the images marked in red have the corresponding label in the DTD database, showcasing the difference in label interpretation among different users.

as our presented system, provides a more intuitive and general way of approaching such tasks.

4.5.2. PRIORITIZED T-SNE AND IMAGE-BASED SEARCH TOOL EVALUATION

In this part of the study, we shift our focus to evaluating the importance of the *prioritized-tSNE* and *image-based search* functionality. Since these two features aim at aiding retrieval in large and crowded databases, the texture database used here contains 5824 texture images (see Fig. 4.14).

The 16 users were randomly divided into two groups, each with 8 users. For group A, users were asked complete a retrieval task with only the overview panel of our interface, and without using the *prioritized-tSNE*, *image-based search*, *multi-scale replication*, or *clustering* functionality. For group B, users were required to start their search with a specified feature, either *prioritized-tSNE* or *image-based search*, for different retrieval targets. After zooming into an area by using one of these features, they continued with that same navigation as group A until finding the target image. The textures for these retrieval tasks are shown in Fig. 4.15, where the two leftmost ones were used for the *prioritized-tSNE* evaluation and the two rightmost ones for the *image-based search* evaluation. In the latter case, the participants were given a small set of images they could use to start their search. The recorded retrieval time for each texture is shown in Fig. 4.16.

Prioritized-tSNE In the *prioritized-tSNE* test, the average time of group A (overview mode only) was 354 seconds, whereas group B (*prioritized-tSNE*) took in average 178

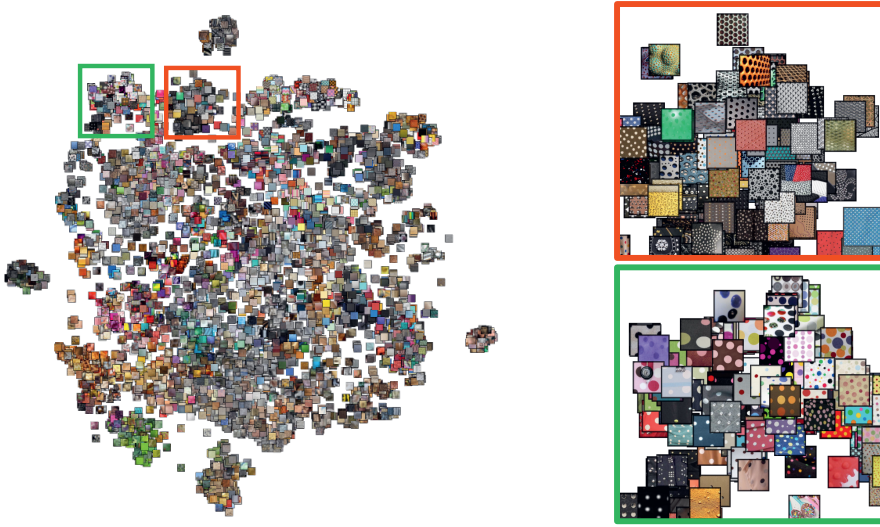


Figure 4.14: Embedding of the database of the feature-evaluation study (5824 textures, insets show grouping of similar images).



Figure 4.15: Retrieval-task textures for feature comparison.

seconds, roughly half the time. The results shown in Fig. 4.16, tasks 1 and 2, indicate that when using our *prioritized-tSNE* the retrieval times were substantially faster for all users except user #13. This user reported being misled by a similar texture (an image of cracked glass), and spent some time exploring the wrong region.

From the feedback in the questionnaire, some users from group A suggested that a tool that can spread the overlapped crowd out would be helpful, which is one of the goals of our *prioritized-tSNE* feature.

Image-based search In the *image-based search* test, group A (*overview mode only*) averaged 650 seconds, and group B (*image search*) averaged 147 seconds, a roughly 4x improvement. In group A, two users (#2 and #4) gave up retrieval after 5 minutes and 3 minutes, respectively. According to their feedback, user #2 switched the retrieval among several possible regions but was finally not able to locate the target. User #4 gave up due to overlapping textures and made a similar suggestion as reported before, that a tool to

spread the crowded regions would be useful. In group B, almost all the users succeeded in finishing the task using less time than those in group A, except user #10 for whom the retrieval was stopped at 5 minutes and the reason was similar to that of user #4. This can be probably solved if the users had access to the *prioritized t-SNE* tool. Overall, thanks to the fast identification of the target region when starting with the *image-based search* tool, the retrieval time is substantially shorter.

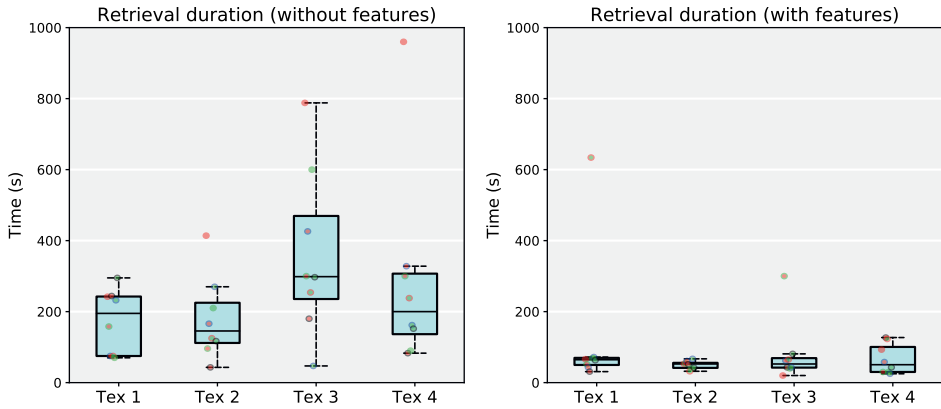


Figure 4.16: Individual retrieval time in seconds for the four textures using only the *overview* mode (left) and with additionally the specified features (right). Texture 1 and texture 2 are the retrieval targets for *prioritized-tSNE* evaluation, texture 3 and texture 4 are the retrieval targets for the *image-based search* evaluation.

4.6. CONCLUSION

In this paper, we presented a system that facilitates the task of exploration and retrieval in large unlabeled texture collections. We organize images based on semantic features and additionally provide several tools to improve the exploration of image groups that share similar features. One of these tools, our proposed prioritized t-SNE algorithm, can enhance the visualization of areas of interest for a user and might even find applications beyond the scenario of this work. The results of our user study show that our proposed system is a considerable upgrade to traditional filesystem and grid-based interfaces when exploring texture images.

4.7. ACKNOWLEDGMENTS

The authors would like to thank the University of Oxford for providing the texture database (DTD) used in this paper. This work is partly supported by VIDI NextView, funded by NWO Vernieuwingsimpuls.

5

MUSICON: GLYPH-BASED DESIGN FOR MUSIC VISUALIZATION AND RETRIEVAL

This paper introduces a novel glyph-based design for music representation that leverages deep latent features to improve user-directed search for music discovery. We propose a system that combines a pre-trained neural network model for high-level music feature extraction with dimensionality-reduction methods for effective visual mapping of the intrinsic characteristics that help distinguishing a song. We provide a search-by-icon user interface (UI) that integrates glyph based on the neural features in combination with other novel navigation methods to achieve intuitive search and exploration. A detailed user study validates our approach, demonstrating its efficacy in enabling swift song clustering, identification, and retrieval. Our findings reveal that our visual representation not only speeds up the music searching process but also fosters increased user interaction with digital music libraries, representing a valuable contribution to the domain of music exploration and retrieval.

5.1. INTRODUCTION

As music streaming platforms evolve, their role has expanded from merely providing access to vast music collections to becoming pivotal in music discovery [139, 140]. Users typically engage with new music through two main avenues: algorithmic recommendations and user-guided search and exploration. The latter, in particular, plays a critical role in diversifying users' music experiences, a factor increasingly recognized for its correlation with long-term user engagement [141]. Despite advancements in algorithms for navigating music collections, the integration of innovative visual cues into user interfaces remains limited. Conventional icons, such as album art, provide minimal insight into a song's characteristics, forcing users to rely solely on auditory exploration to determine their preference. This highlights a gap in user experience, as visual representations could potentially expedite the discovery process by indicating auditory similarities, even though music is primarily an aural medium [140, 142].

Platforms like Spotify provide their users the function to personalize the playlist, yet these innovations fall short in conveying the musical essence of unseen tracks from unfamiliar artists. The inherent challenge lies in the inability to avoid a time-consuming listening process. Our work posits that using visualization can provide information that can be efficiently parsed to compare characteristics, such as mode, tempo, or mood. Hereby, we can notably enhance user-guided search and exploration. Our solution reduces the time users spend finding music that aligns with their taste or current mood, especially when navigating with an open or exploratory mindset [140, 143]. This hypothesis is supported by evidence suggesting that visual identifiers can expedite navigation in user interfaces [144]. Yet, our goal is not necessarily to derive a global visual encoding of music, but rather a visualization method to enable comparisons to facilitate exploration. Users do not have to learn the meaning of individual representations in order to effectively use them, and even in rather homogeneous song collections, our visualization can provide a clear visual differentiation.

Our contributions are twofold. First, we propose an approach to extract latent characteristics of music utilizing state-of-the-art deep learning models. Second, we introduce a new visualization solution that employs custom-designed icons that embed these features in visual cues, facilitating music exploration, including interaction methods to compare, search, and categorize. It improves music visualization by combining advanced representation learning with user-centered glyph design principles.

The article is organized as follows: Section 5.2 reviews related work. Section 5.3 presents our approach, covering feature extraction, dimensionality reduction, and glyph generation. Section 5.4 integrates the icon into our UI prototype. Section 5.5 evaluates our work through a user study, followed by a conclusion in Section 5.6.

5.2. RELATED WORK

This section explores foundational work and recent advancements in music features, latent representation learning, and glyph-based music visualization, framing the context for our contributions to music discovery through visual representation.

Understanding music features spans from low-level signal descriptors to high-level semantic attributes. Traditional music information retrieval (MIR) approaches focused on

'hand-crafted' features, emphasizing explicit knowledge-based feature engineering [145]. The advent of deep learning shifted the attention towards automatic feature extraction, demonstrating strong performance in capturing complex musical characteristics without extensive domain knowledge [146]. Notable benchmarks such as the Million Song Dataset (MSD) and Spotify Web API highlight their utility in research, bridging content and context-based music information [147, 148]. However, concerns regarding the reproducibility, explainability and open research when using proprietary data (like features from Spotify) have motivated us to focus on alternative, open-source features for music representation.

Latent variable models have revolutionized the representation of music by learning abstract features that encapsulate the inherent characteristics of musical pieces. These models, especially Convolutional Neural Networks (CNNs), Variational Autoencoders (VAEs), and Transformers, have facilitated a broad range of MIR tasks, including genre classification, music recommendation, and emotion recognition [149–151]. The transition towards end-to-end learning models marks a significant shift from traditional feature engineering, enabling more nuanced and comprehensive understanding of music data. Our work leverages these advancements to derive latent representations that serve as a foundation for music visualization.

Visualization plays a pivotal role in music discovery, enabling users to navigate and explore music collections intuitively. Early efforts by Kolhoff et al. [152] introduced music icons, utilizing parameterized glyphs to represent music features visually. Subsequent research has expanded on this concept, exploring various visual mappings and interactive interfaces for music exploration [144, 153]. Yet, challenges remain in designing visual representations that effectively convey the complex nature of music features while supporting user-friendly exploration. We build upon this foundation and propose a novel glyph-based framework, using deep learning-derived features, coupled to specialized visualization and interaction techniques to enhance music discovery.

5.3. OUR APPROACH

Our research is positioned at the intersection of MIR and visualization, where we propose a novel strategy to enhance user-guided music discovery in a large database through visual representation. To achieve this goal, we introduce an innovative glyph design by leveraging advanced latent features from the MIR model to provide immediate, intuitive insights into the music's characteristics. We also propose interaction mechanisms to not only facilitate a more efficient and engaging music discovery experience but also address the needs of users exploring music collections, looking for new sounds that match their reference ideas. Fig. 5.1 shows an overview of our solution.

In this section, we will first discuss the feature extraction. Then, we will map this high-dimensional representation to a lower dimensional space to reduce the degrees of freedom of the information on a star-glyph representation, whose design choices are explained in the following. In the next section, we will then present our interface that builds upon this song representation.

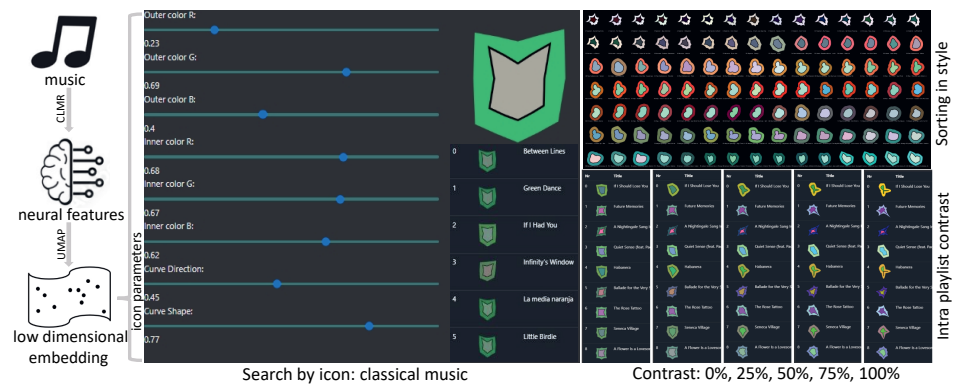


Figure 5.1: Workflow and interaction scheme of our Musicon system: the music data runs through a pre-trained Contrastive Learning of Musical Representations (CLMR) model towards high dimensional neural features, which are further embedded into eight dimensions as glyph parameters for characterising the music icon. The figure shows a customized icon for classical music and a list of retrieved songs. Applications include sorting music files by style, identifying similar songs, and navigation within a playlist. An overview-first, details-on-demand approach is used by enhancing icon contrast, useful for examining large song collections and distinguishing similar songs.

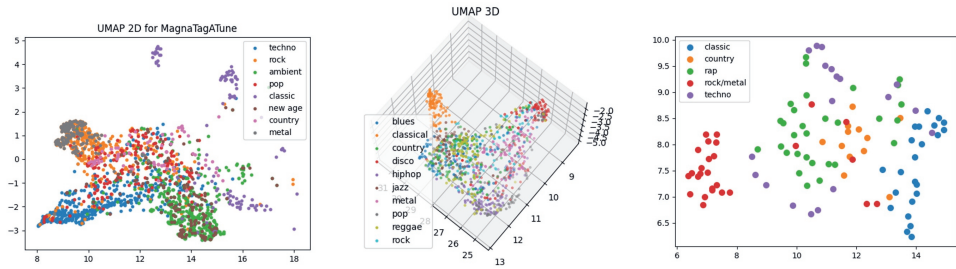


Figure 5.2: UMAP embeddings of features extracted for genre classification on: MagnaTagATune (left), GTZAN (middle), and custom dataset (right).

5.3.1. FEATURE EXTRACTION

We reviewed state-of-the-art research in music representation learning, focusing on papers that provide full code and trained weights due to the challenges of training. We reviewed 23 papers, of which ten presented pre-trained models suitable for downstream tasks. However, recommendation models were either multi-modal [154, 155] or lacked code and weights [156, 157]. Out of all options, we did identify the CLMR [158] model as particularly suitable for our task, as we also show via an analysis below. The CLMR model is an adaptation of the very effective SimCLR model [159], which was developed for contrastive learning of visual representations. Contrastive learning is an unsupervised representation learning technique with the objective to maintain similarities and

dissimilarities between data points in the representation space. CLMR shows excellent performance, is well documented, and lightweight to run. The network has learned a representation of 512 dimensions over an input sample of about 2.6 seconds. The model was trained for the downstream task of classification on the MagnaTagATune dataset [160]. We use the model and weights as provided by the authors. Representations over longer segments are averaged.

We verify the effectiveness of the CLMR representations by examining feature clusters to genre classification on multiple datasets and comparing feature embeddings to the Spotify features.

Evaluation of CLMR model To ensure the suitability of the model, we assessed its features for genre classification across three datasets: MagnaTagATune, GTZAN [161], and a small custom dataset. The custom data is composed of several albums that are considered iconic for various genres, details are given in Table 5.1. Each dataset provided genre labels. By extracting features and employing the UMAP algorithm [16] for 2D spatial embedding, color-coding by genre revealed that genre-based clusters closely matched the feature-based clusters. Fig. 5.2 plots the 2D embeddings of these three datasets, which suggests the captured features reflect high-level conceptual similarities across genres.

5

Genre	Artist	Album name
classic	Bach	A Musical Genius
classic	Vivaldi	The Four Seasons
rap	Eminem	The Eminem Show
rap	Nas	Illmatic
rock/metal	Nirvana	Nevermind
rock/metal	Slipknot	Iowa
techno	Paul Kalkbrenner	Berlin Calling
techno	Vitalic	Rave Age
country	Waylon Jennings	Dreaming My Dreams
country	Willie Nelson	Red Headed Stranger

Table 5.1: Composition of the custom test dataset with two iconic albums for each genre.

Comparison to Spotify features To explore how the selected representations compared with Spotify’s own feature metrics, we created a dataset of 10K data points which are selected from the over 1.2 million entries in the Spotify Dataset reported by Figueroa et al. [162]. For each data point, we obtained a 30-second MP3 sample through the Spotify API.

From these data points, we used the CLMR model to extract their neural features whose dimensionality was further reduced to 2D for visualization using the UMAP algorithm. Colors were assigned based on Spotify feature values such as acousticness, energy, valence, loudness, danceability and instrumentalness, to facilitate intuitive interpretation of the distribution. The corresponding feature results are shown in Fig. 5.3. The results show a clear correlation between the CLMR-derived features and Spotify’s features. This

demonstrates that the selected model effectively captures musical qualities that align with industry-recognized attributes.

5.3.2. FEATURE DIMENSIONALITY REDUCTION

The CLMR model provides a representative neural vector of 512 elements. We perform a comparative evaluation of various dimensionality-reduction methods to find a suitable approach to reduce the number of features. Several works [163–165] demonstrate that employing fewer but more representative dimensions than the original high dimensional vector in star glyphs greatly improves not only the computational efficiency but also their effectiveness across a spectrum of tasks. Aiming to retain the comprehensive nature of the data, we reduce the dimensionality to eight, which might seem arbitrary but is well motivated when opting for a star-shaped glyph, which will be discussed in Section 5.3.3.

We focus on identifying a method to effectively preserve data clustering and explored five algorithms for their ability to maintain data point similarities with its original high-dimensional representation: PCA [166], t-SNE [15] and UMAP, and two other more recent algorithms developed based on t-SNE and UMAP, respectively: TriMap [167] and PaCMAP [168].

To evaluate the preserved similarity of the reduced 8-dimensional space from the original 512-dimensional space, we calculated cosine similarity matrices for both spaces. We present the detailed statistics of the cosine similarities with the five assessed methods in Table 5.2.

Among the algorithms, t-SNE performed below our expectations, possibly the projection to an eight-dimensional space was off its optimal usage scenario. UMAP and PCA showed substantially better performance than TriMap and PaCMAP. Due to its non-linearity UMAP, outperforms PCA in maintaining data point similarities, while having acceptable computation cost, which made it our choice. Optimizing UMAP's hyperparameters (nearest neighbors=15, minimum distance=0.2 for normalized embeddings) further improved the results.

Method	Mean	Median	Std	Min	Max	Time/s
PCA	0.952	0.975	0.065	0.013	0.991	3
t-SNE	0.179	0.185	0.084	-0.161	0.373	4523
UMAP	0.962	0.963	0.009	0.889	0.983	49
PaCMAP	0.129	0.163	0.127	-0.257	0.335	28
TriMap	0.127	0.173	0.138	-0.273	0.352	38

Table 5.2: Statistics of the kept similarity for feature vectors reduced from 512 to eight dimensions for five different algorithms.

5.3.3. GLYPH DESIGN

Glyphs, often composed of various geometric elements and visual channels, are capable of encoding multiple data dimensions simultaneously [169]. This characteristic makes them particularly suitable for high-dimensional data visualization [170] and tabular data

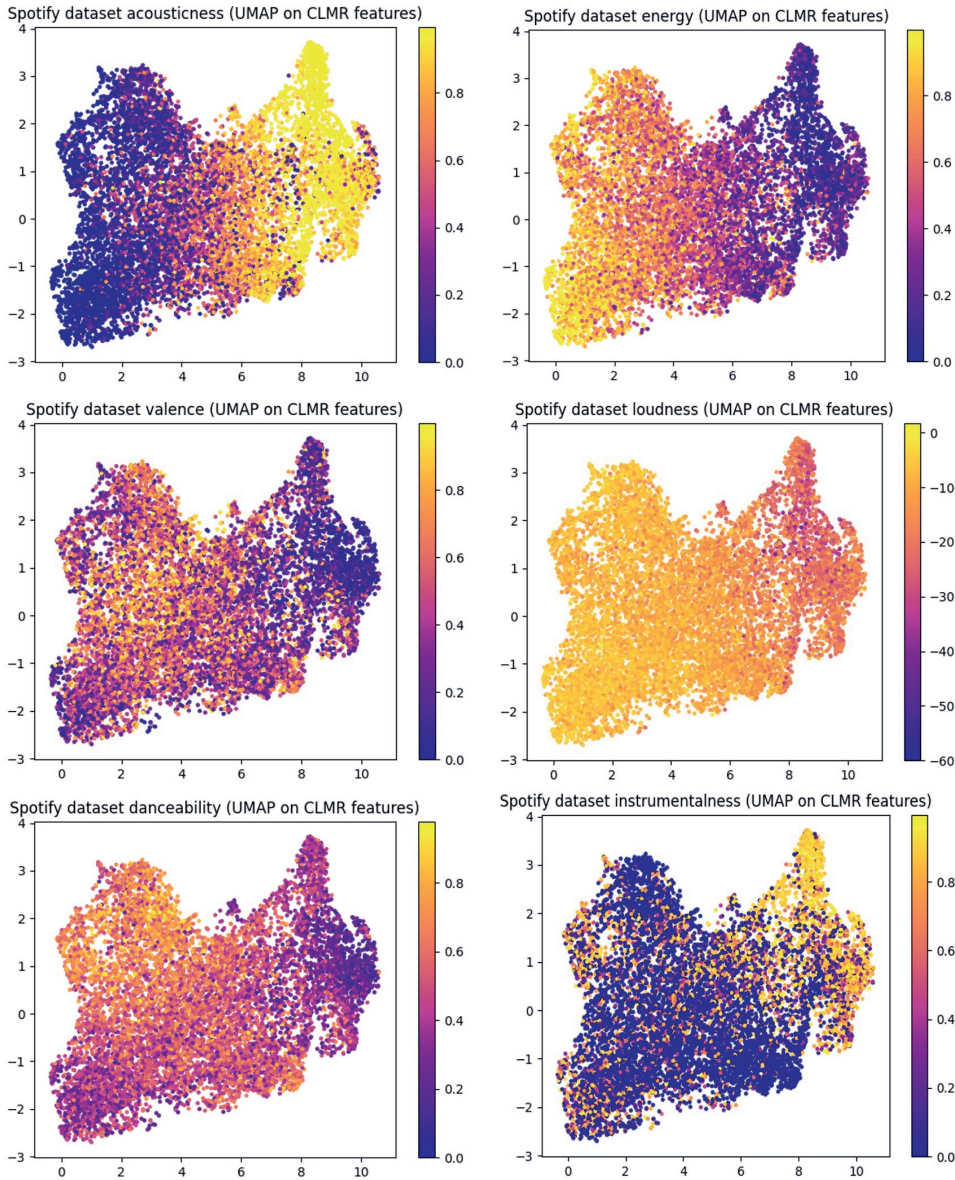


Figure 5.3: Examples of 2D UMAP embeddings of extracted CLMR-derived neural features colored with different Spotify features: acousticness (left in the 1st row), energy (right in the 1st row), valence (left in the 2nd row), loudness (right in the 2nd row), danceability (left in the 3rd row), and instrumentalness (right in the 3rd row).

representation [171]. Despite their utility, the design space for glyphs remains vast and largely unexplored [172]. Owing to its simplicity, versatility, and effectiveness in encoding multivariate observations and facilitating visual data comparison, we rely on the star glyph [163, 173, 174]. To enhance expressiveness and ease of comparisons, we opted for a contour plot instead of a whisker plot, as suggested by [175]. Hereby, shape is incorporated as a significant feature. Our design considered various elements, dimension ordering, categorization via colors, and shape emphasis by curvature, where some redundant encoding further eases shape distinction. Here, we detail our design decisions.

Dimension ordering The arrangement of variables on the axes of star glyphs significantly influences their shapes. The same data-point can be displayed as very different shapes when it is mapped in different orders according to Klippel et al. [176]. Finding the best order for variables is complex and depends on the analysis goal, but they observed that showing major differences on the main axes helps users identify shapes more quickly. Consequently, we sort the dimensions according to variance of the data.

Fig. 5.4 shows the effect on our final glyphs before and after sorting the dimensions for seven heavy metal songs. They look very alike before and quite distinct after reordering.

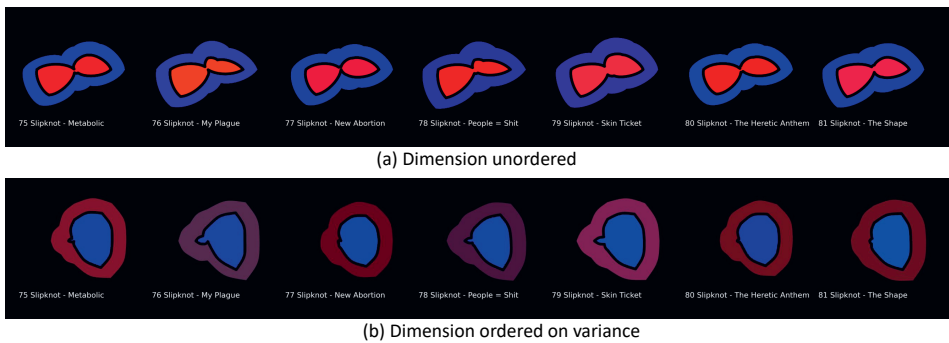


Figure 5.4: Comparison of unsorted (upper) and sorted (bottom) the axes of our glyph on variance. It is very hard to detect differences among the icons without axis sorted. In the bottom row, after sorting the axis, we can detect small differences between the icons.

Dual colour As color can be a good indicator for distinguishing categorical data [169], we use color prominently for the most distinctive dimensions, hereby making the implicit classification explicit. We adopted the automatic color mapping of Kolhoff et al. [152], which assigns six features to two RGB colors. Originally, we considered the first six dimensions, but for reasons explained below ('Curvature'), we chose the first three dimensions and dimensions five through seven to be mapped to RGB components. This is shown in Fig. 5.6 (left). We designed a distinct inner and outer glyph shape to receive these two colors, by superposing the glyph at different scales. The scale is chosen to ensure that both areas are balanced. The color mapping results can be seen in Fig. 5.6 (right).

Curvature As curvature is considered a pre-attentive visual stimulus [172], we leverage it to expand the variety of shapes and increase the expressiveness of our proposed glyph. While Klippel et al. [176] noted that distinctive shape features can speed up classification, they also cautioned that strong changes of shape may lead to a wrong impression of dissimilarity. For this reason, we chose to map dimensions four and eight, which were not yet redundantly encoded by color.

Specifically, the 4th dimension dictates the curvature's direction and strength. The 8th dimension, with the least variance, determines the positioning of control points relative to the line segment endpoints. An illustration of the influence of these two parameters on a curve can be seen in Fig. 5.5 (left).

A limitation of this mapping is that when the 4th dimension is close to zero, the impact of the 8th dimension is less noticeable. Given that the eighth dimension has the lowest variance, we consider this an acceptable shortcoming. An illustration of this singularity can be seen in Fig. 5.5 (right), where the effect of the curvature settings can be viewed on an 8D star shape.

To ensure clarity and prevent overly complex shapes, we employed an intersection detection method to adjust curvature strength and avoid intersecting lines, ensuring a coherent and interpretable glyph design. We confined curvature direction and strength to a range of $[-0.3, 0.3]$ and control point distances to $[0.3, 0.6]$. This configuration range avoids shapes that appear to have more line segments than intended.

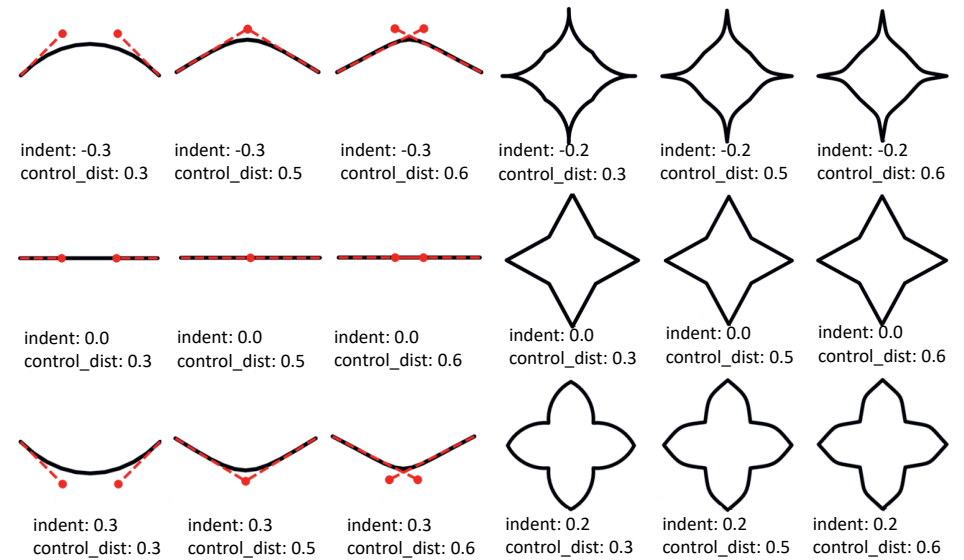


Figure 5.5: Influence of curve parameters: The x-axis varies the control distance, while the y-axis adjusts the direction and strength. The left image demonstrates the impact of parameter adjustments on curve construction, and the right image illustrates how curve parameters affect an 8-dimensional star shape.

Redundant Encoding As indicated, our glyph design contains redundant coding [177] by design. As the ideal mapping of data to star glyphs remains an open problem, encoding the important variables multiple times can significantly enhance glyph expressiveness and distinctiveness, bolstering visual search effectiveness. In our case, it also makes the design suitable for individuals with color vision deficiency.

Our scheme employs a dual encoding for each variable. It is always represented as a component of the glyph's shape, but also as a glyph's color or curvature. To judge the effectiveness of the resulting glyphs, we illustrate several representations in Fig. 5.6 (left) and show how each parameter influences the glyph in Fig. 5.6 (right). The design is carefully evaluated in Section 5.5.

5.4. INTERFACE

We introduce a new search interface that incorporates our glyph definition and interaction features to support the user. It has been built as a web application, accessible at <http://musicons.io/>. Involving a test database of 10K song snippets that were randomly selected from the Spotify Dataset provided by Figueroa [162], which contains more than 1.2M song samples. The principles of our glyph-based design are inherently flexible and can be adapted to mobile platforms. Nevertheless, we first chose a desktop interface for our Musicon system to better control its evaluation. Typically, the use of a desktop system ensured a larger display area and precise input capabilities for a detailed glyph-based visualization and interaction. It also facilitated the user study, as measuring user engagement and satisfaction was eased, as we could assume that people are familiar with the hardware. Designing touch-friendly controls and effectively managing visual complexity on a small screen of a mobile device remains promising future work.

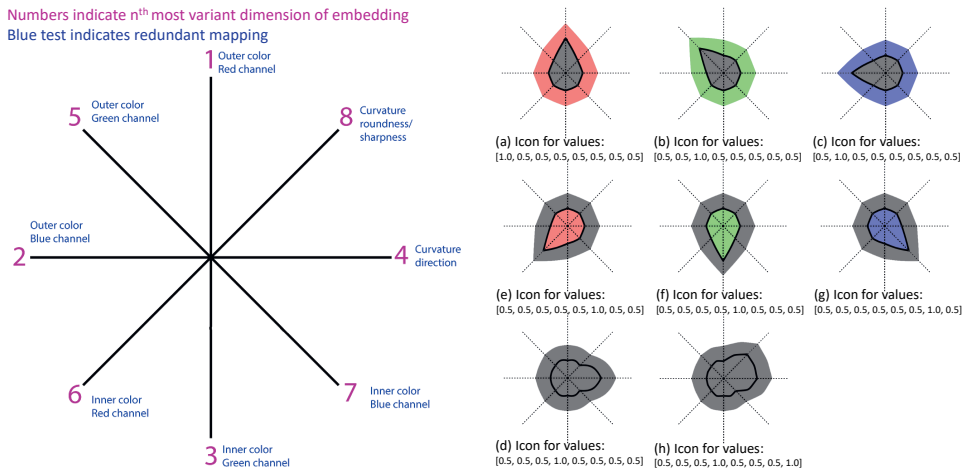


Figure 5.6: The proposed variable mapping order along the axis of the star glyph (left) and the influence of each parameter with redundant encoding (Initially all 0.5, then we vary one parameter at a time (right)).

5.4.1. CUSTOMIZED ICON

For an initial expression, we randomly selected 48 icons from the results. As can be seen in Fig. 5.7, the icons reveal the intended expressiveness and display a wide variety of shapes and colors.



Figure 5.7: Customized icons: 48 randomly selected icons from the results.

5.4.2. SEARCH-BY-ICON

The search-by-icon tool allows users to start with the icon of a preferred song and then explore other songs in the proximity, akin to a reverse search mechanism. This concept draws inspiration from Knees et al. [178], who explored audio search through the visualization of sound mental images.

The interface for search-by-icon can be seen in Fig. 5.8. It features eight adjustable sliders, corresponding to the glyph dimensions. The representation is updated in real time, as is the list of the most similar songs, which enables users to explore the music space interactively.

We use the cosine similarity to the user-generated icon and a comparison between the input and 10K vectors of songs in the database only takes milliseconds. It is done whenever sliders are adjusted. The ten most similar songs are then presented to the user.

5.4.3. PLAYLIST SORTING

Kolhoff et al. [152] introduced two sorting methods, 1D and 2D, by applying PCA on icon parameters. We explored various approaches within our dataset and found that UMAP embeddings surpassed PCA in performance.

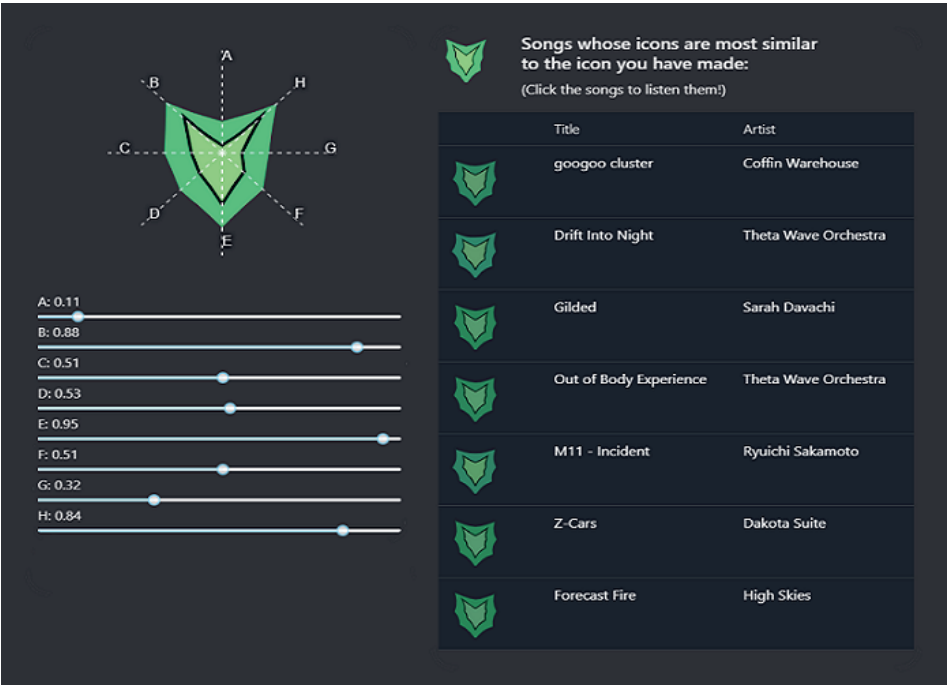


Figure 5.8: The interface for search-by-icon. The icon parameters are indicated with a dotted line and the icon is updated in real-time when the parameters are changed with the sliders (left). While the users drags the slider, the most similar songs are immediately updated and allow to be listened(right).

Specifically, we initiated our process by converting the dataset into an 8-dimensional (8D) UMAP embedding, subsequently transforming the resulting data into a 1-dimensional (1D) UMAP embedding. Although a 2-dimensional (2D) layout was considered, it introduced distortions when attempting to achieve the compactness of a 1D layout. Furthermore, the 1D layout better aligns with the format of song lists familiar to users of streaming applications, thus leading us towards a 1D embedding. Utilizing the 8D embedding initially maintains greater coherence with the space utilized for our glyphs, as demonstrated in Fig. 5.9, and reduces computational workload relative to the original 512 dimensions. While sorting within diverse playlists produces variable sequences, our method consistently positions similar icons in close proximity. This arrangement enhances user navigation and experience, outperforming standard PCA arrangements.

5.4.4. ENHANCING ICON CONTRAST

A global embedding will enhance inter-genre distinctions, while reducing intra-class variations. To amplify local contrast of icons when exploring, for example, homogeneous playlists of similar songs, we employed a min-max scaling to re-normalize the icon features' range of the selected subset to [0, 1]. Furthermore, users are offered the option

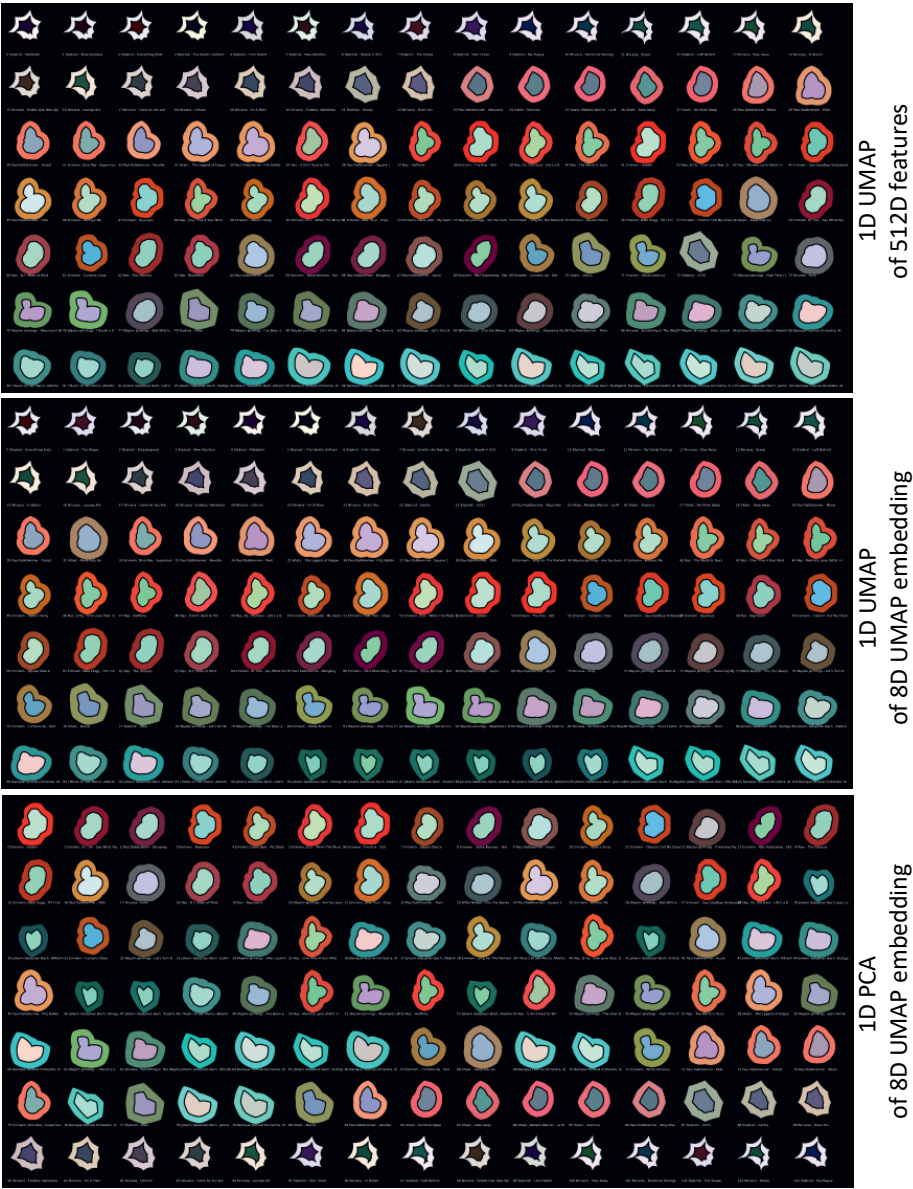


Figure 5.9: Comparative visualization of three sorting methods. From upper to bottom: icons sorted on 1D UMAP embedding of original 512D features, icons sorted on 1D UMAP embedding of the 8D UMAP embedding of the 512D features, and icons sorted on 1D PCA of the 8D UMAP embedding of the 512D features.

to tune the percentage of contrast linearly between the original and fully normalized embeddings. This maintains a link to the global representation and allows for applying a gradual contrast. Fig. 5.10 illustrates the effectiveness of this feature in a subset of jazz music, by gradually increasing the local contrast from 0% to 100%. As an example, the song zero and the song four in this list exhibit very high similarity in audio data - both share instruments, mood, and a prominent solo with the same instrument. At overview scale, the icons appear almost identical, as expected. By tuning up the contrast, the resolution of the icon parameters is locally amplified and the difference becomes more apparent.

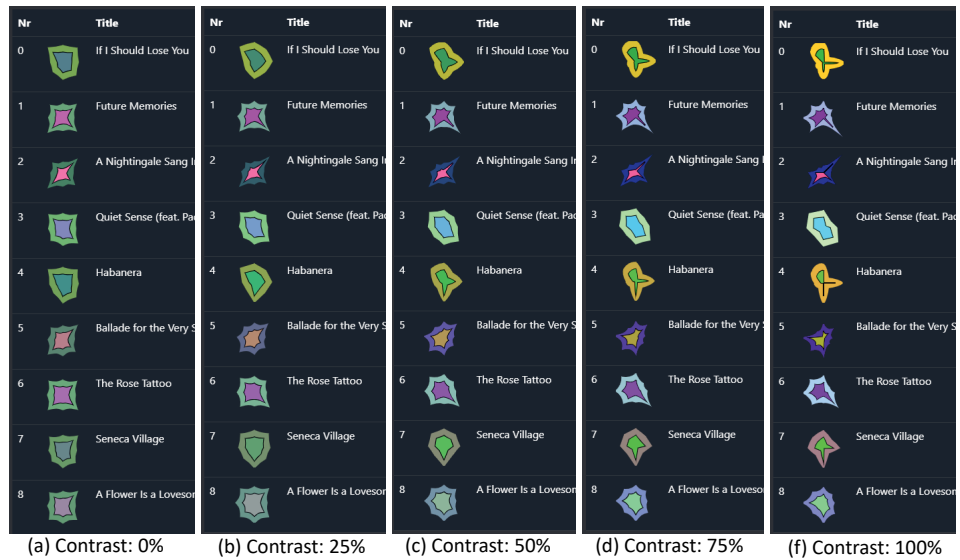


Figure 5.10: Local-contrast icons for a playlist of jazz music with contracts of 0%, 25%, 50%, 75%, and 100%.

5.5. EVALUATION

Given the subjective nature of visualizing and perceiving music, our hypothesis and experimental design are evaluated through an exploratory user study. We aim to conduct two types of evaluations: a comparison with alternative methods and an assessment of the proposed features.

We wanted to compare our work with Kolhoff et al. [152], who introduced content-based music icons. Unfortunately, after contacting the authors, we learned that the music resources and implementation are no longer available. Hence, we can only informally compare the two solutions. Our work benefits from deep features, a more detailed mapping of information on the glyph shape, as well as an improved robustness and inclusivity for color vision deficiency (CVD) by redundant encoding. Furthermore, we introduce a reverse search, which broadens the utility of music icons.

We target the evaluation of our proposed features: from the effectiveness of the icon to a larger system-evaluation. We used the 10K dataset due to its availability, although the system can be expected to handle larger song databases with minor optimizations in the implementation.

We targeted a participant demographic of ‘non-expert but generally computer-literate’ adults [179]. To reduce response bias, participation was anonymous. Using an a-priori sample size calculator with an expected medium effect size ($d = 0.5$), we determined that a minimum of 27 participants would achieve a statistical power of 0.8 and a significance level of 0.05, assuming analysis via paired samples t-test for certain tasks. We garnered 38 responses in the evaluation, with a roughly equal number of men and women and an age distribution ranging from 20-29 to over 70 years. To conduct the study, we developed a web interface that provided each participant with task instructions, managed the flow and timing, randomized the order of tasks, and collected the data. The study was conducted remotely to minimize the response bias caused by our presence. Additional details regarding the evaluation are given in the supplementary material.

5.5.1. CUSTOMIZED ICON

Visual clustering. This is a classic ‘free-grouping’ or ‘free-sorting’ task, widely used in the field of psychology [180]. This test assesses the effectiveness of the icons in representing feature similarity and the extent of user consensus on this aspect. Participants formed clusters from 60 icons based solely on visual cues, without song titles or additional metadata information. They were allowed to use any number of clusters and set aside non-fitting icons.

To ensure that there is a diversity in the selection yet still the possibility to make clusters, we sampled 10 data points from six of the clusters created by applying a k-means clustering algorithm on the original 512 dimensional embedding ($k = 10$). Each participant worked with the same set of icons but their presentation was in a random order.

To assess user consensus on clustering, we computed a co-occurrence matrix of participant-generated clusters and a cosine similarity matrix of the feature vectors, both shown in Fig. 5.11. Initial observations suggest a strong user agreement on the clusters, with the co-occurrence matrix displaying notable resemblance to the similarity matrix. To quantitatively evaluate this relationship, we calculated the pairwise Pearson correlation coefficient, resulting in a value of 0.6. This indicates a moderate linear correlation, suggesting a reasonable level of agreement among users in their clustering decisions.

Outlier Detection. This test, extending from the visual clustering test, utilizes participant-generated clusters to evaluate the glyphs’ effectiveness in representing music similarity and facilitating outlier detection. For each participant, we randomly selected four songs from one cluster and one song from a different cluster, presenting these five songs in a random order. Participants were then asked to identify the song that sounded distinct from the others. This process was repeated three times for each participant.

The descriptive statistics of the results is shown in Table 5.3. Given that random guessing would yield an expected recognition rate of 0.2, our observed mean recognition rate of 0.7451 represents a considerable enhancement. With a p-value lower than 0.00001 and an effect size of 2.375 (Cohen’s d), this improvement is statistically substantial.

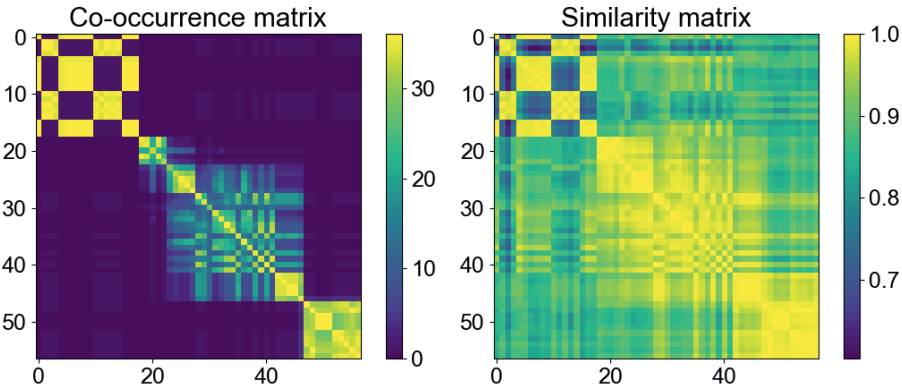


Figure 5.11: Co-occurrence matrix of the clusters made by participants (left) and a cosine similarity matrix of the feature vectors (right).

Mean	Median	Mode	Std	Variance
0.745	0.667	0.667	0.230	0.053

Table 5.3: Descriptive statistics of the recognition rates obtained for outlier detection.

Generalization, Contrast and CVD Robustness. This test is set up as a matching-to-sample task in the same manner as [163]. It aimed to assess three aspects: the alignment of the ‘most similar’ icon with the data point of highest cosine similarity, indicating the icon’s effectiveness in representing high-dimensional data; the impact of a contrast-enhanced icon version on task completion time and accuracy in identifying the most similar icon; and the icon design’s robustness against CVD, evaluated through task time and accuracy using a CVD simulation.

Participants were shown nine similar icons, including one target icon, and asked to identify the icon that is most similar to the target. This test was conducted across three rendering modes: the default design, a contrast-enhanced version with 100% contrast, and a color-blind mode simulating deuteranomaly the most common type of CVD. Each participant completed the task nine times, three times per rendering mode. The descriptive statistics of recognition rates obtained for these three modes is shown in Table 5.4.

Mode	Mean	Median	Mode	Std	Variance
Default	0.706	0.750	1.000	0.277	0.077
Contrast	0.785	1.000	1.000	0.271	0.074
CVD	0.741	0.667	0.667	0.231	0.053

Table 5.4: Descriptive statistics of recognition rates obtained for matching-to-sample with the ‘default’, ‘contrast’, and CVD mode of the icon.

With a random selection, the expected recognition rate is merely 0.125. Our study, however, demonstrates a marked enhancement with mean recognition rates more than 0.706 for all three modes. This substantial increase suggests that participants are better at identifying the icon that most accurately corresponds to a position within an 8-dimensional space. To validate these findings, we employed as ‘default’ mode a one-sample, one-tailed t-test, which indicated an essentially zero p-value and a pronounced effect size of 2.100 (Cohen’s d). These results robustly confirm the effectiveness of our approach.

We observed that ‘contrast’ mode outperforms ‘default’ mode with a higher mean recognition rate and a faster selection process, while, for the CVD mode, we discovered no obvious differences between the ‘default’ and CVD modes. A one-way ANOVA test conducted across all three rendering modes yielded a p-value of 0.450, indicating that rendering mode has no noticeable impact on performance in the matching-to-sample task. This outcome suggests that each icon rendering mode performs comparably well, affirming the robustness of our icon to CVD. The effectiveness of our redundant encoding strategy in enhancing recognition and matching accuracy is thus supported by these results.

5.5.2. SEARCH BY ICON

The test aimed to assess the efficacy of our ‘search-by-icon’ method for users. Participants were shown a target song with its custom icon and the search-by-icon interface shown in Fig. 5.8. They were tasked with using the interface to imitate the target icon and then retrieve the three songs most similar to the target one. Following the test, participants completed the System Usability Scale [179] to evaluate their experience with the interface.

Imitated icon and retrieved songs Cosine similarities between user-generated and target icon vectors, presented in Fig. 5.12 (left), with a high mean (0.989) and median (0.993), indicates that with vectors exhibiting a cosine similarity above 0.975 to the target, most users accurately replicated icons. Furthermore, the average cosine similarities between the target icon vector and the top three selected songs, detailed in Fig. 5.12 (right), reinforce the precision of these imitations, highlighting the effectiveness of participant selections in aligning closely with the target icons.

System Usability Scale (SUS) The SUS comprises of ten statements, each evaluated using the Likert Scale, which ranges from one for ‘strongly disagree’ to five for ‘strongly agree’. The details of this dataset can be found in the supplementary material.

Based on the feedback, we computed the SUS scores, as illustrated in Fig. 5.13(left), with the corresponding performance interpretations presented in Fig. 5.13 (right). We counted 13 ‘bad’ results, 7 ‘mediocre’, 11 ‘good’ and 6 ‘excellent’. We recognize that flattening the user experience into such a score is a gross simplification. Nonetheless, we observe that a majority, specifically 25 out of 38 participants, demonstrates a willingness to embrace our model.

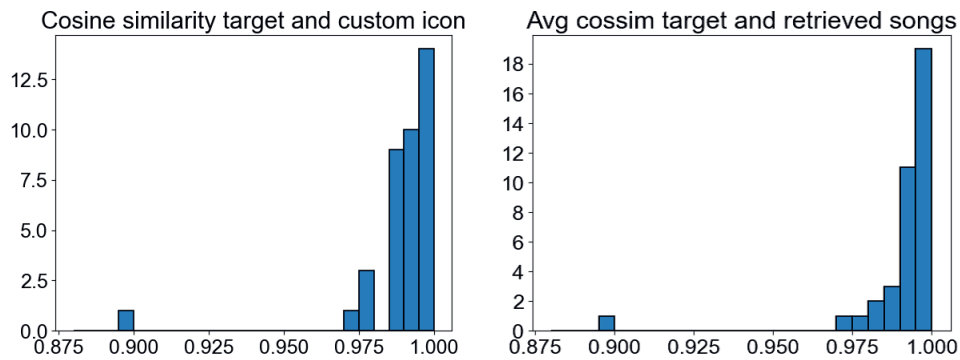


Figure 5.12: Cosine similarities between user-generated and target icon vectors (left), illustrating how closely users can imitate an icon. The average cosine similarities between the target icon vector and the top three selected songs (right), evaluating how effectively users can retrieve similar music using this tool.

5

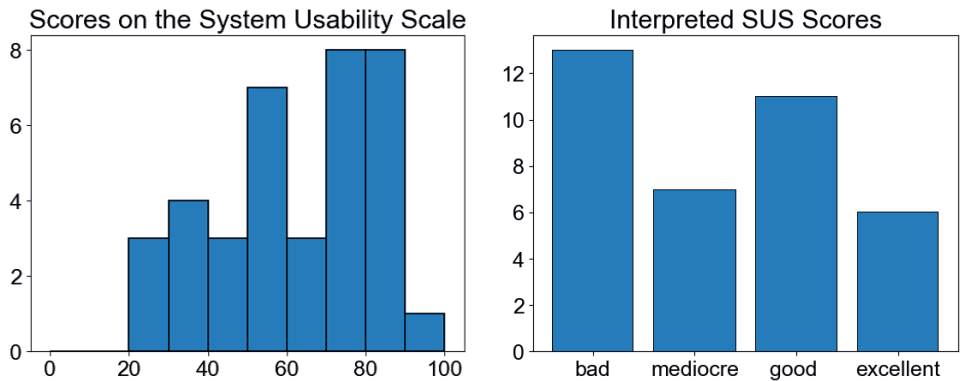


Figure 5.13: SUS Scores (left) and its corresponding interpretation (right). A score above 80.3 is interpreted as ‘excellent’, scores between 68 and 80.3 as ‘good’, scores between 50 and 68 ‘mediocre’, and anything below 50 ‘bad’.

5.5.3. SEARCH BY PLAYLIST

This test aimed to assess the icon’s effectiveness and sorting properties within a playlist context, comparing it against album art, which is typically used in streaming services. Participants were asked to select their top three songs from playlists featuring both album art and our custom design, with each format presented twice. We assessed the similarity between the top three selections and the target vector, time-on-task, plays per task, and additional insights from open-ended questions.

Retrieved songs Average cosine similarities between the target icon vector and the top three selected songs are presented in Fig. 5.14, with descriptive statistics in Table 5.5. Both

methods cover similar ranges of cosine similarities, but our method facilitates slightly higher similarity retrieval (one-tailed paired-samples t-test: $p = 0.03$, Cohen's d : 0.469), aligning with the icon's intended similarity representation.

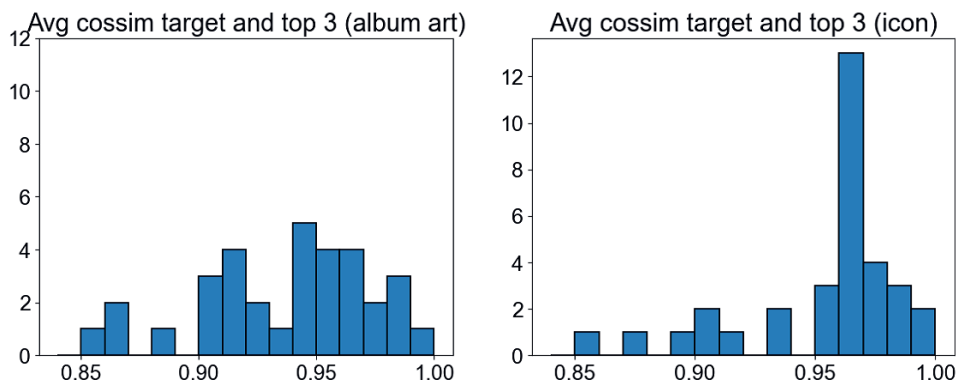


Figure 5.14: Average cosine similarities between the target icon vector and the top three selected songs, comparing between album art (left) and our custom icon (right).

Icon	Mean	Median	Std	Var	Min	Max
Album	0.955	0.967	0.033	0.001	0.858	0.991
Custom	0.938	0.945	0.036	0.001	0.851	0.993

Table 5.5: Descriptive statistics of the data as displayed in Fig. 5.14.

Time-on-task The time-on-task per participant for both album art and custom icon methods are detailed in Fig. 5.15, with descriptive statistics provided in Table 5.6. A notable reduction in average completion time, exceeding one minute, was observed. A left-tailed paired t-test confirmed these findings with $p = 0.00201$ and an effect size of 0.473 (Cohen's d).

Icon	Mean	Median	Std	Min	Max
Album	6:25	4:49	3:45	1:08	14:12
Custom	4:44	3:54	3:11	0:39	12:14

Table 5.6: Descriptive statistics of the data as displayed in Fig. 5.15, formatted as mm:ss.

Songs played per task Fig. 5.16 and Table 5.7 display the number of songs played per task per participant for both album art and the custom icon, showing similar ranges but a notably lower mean and median for the custom icon. A paired t-test confirms this difference, with $p = 0.00001$ and an effect size of 0.931 (Cohen's d).

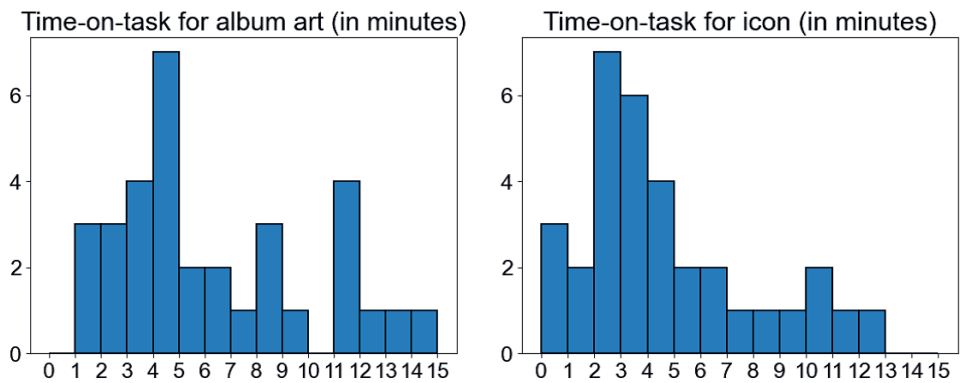


Figure 5.15: Completion times of time-on-task with album art (left) and our custom icon (right).

5

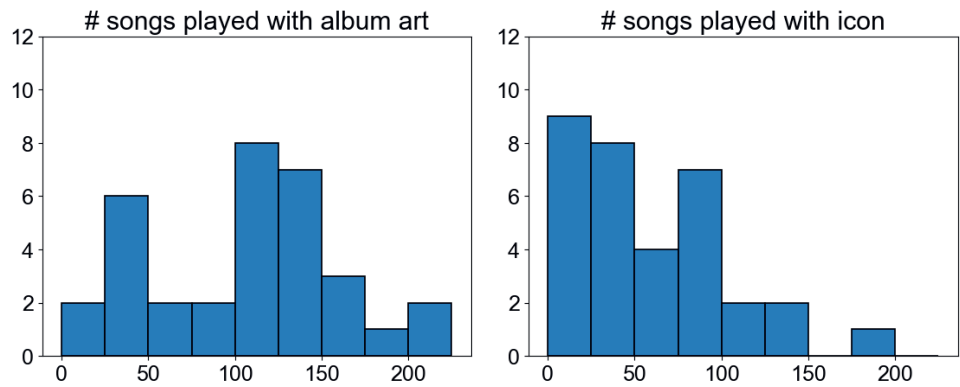


Figure 5.16: The number of songs played per task per participant for both album art (left) and the custom icon (right).

Icon	Mean	Median	Std	Var	Min	Max
Album	103.9	113.5	55.9	3121.1	10	222
Custom	57.7	40.0	42.4	1796.8	7	192

Table 5.7: Descriptive statistics of the data as displayed in Fig. 5.16.

Open Questions Upon study completion, participants responded to three open-ended questions regarding their playlist task experience. The detailed questions and corresponding analysis can be found in the supplementary material.

Our tool’s effectiveness was confirmed through strong quantitative results, notably speeding up task completion by over a minute compared to album art presentations and reducing the number of songs participants needed to listen to by almost 50%. When using our icon, selections tended to have similar or slightly higher cosine similarity to

the target song, suggesting the icons' visual cues enhanced both the speed and quality of decision-making.

While many participants valued our icon for its capacity to indicate similarity, some expressed a preference for album art due to its contextual and cultural insights. Acknowledging album art's value in certain situations, we argue that our icon meaningfully enhances user experience by addressing the variability of songs within an album. Further, our evaluation focused on the effectiveness of our contributions. In a practical system, we would envision the use of our icon in conjunction with potential metadata (including album art) if available.

An exciting direction for future work would be exploration for semantic latent representations and descriptive dimension mappings. Unfortunately, this is not straightforward, as it is unlikely that all aspects could be well captured in this way, e.g., what dimension would mean "adding a piano" or "adding strings"? Nevertheless, some meta information could be used to augment our solution. In a practical system, we would certainly argue for keeping genre information available for the user in selections and search. We refrained from doing so, to show the effectiveness of the automatically derived features and our visualization solution, which is already able to capture a considerable amount of information.

5.6. CONCLUSION

In this paper, we introduced a novel icon-based visualization approach for enhancing music discovery in streaming services by mapping latent characteristics of music to a novel glyph design. Our evaluation demonstrated that our method meaningfully improves user engagement and efficiency in music exploration, highlighting the potential of incorporating such visual cues into streaming platforms. The positive outcomes suggest a promising direction for further research in music visualization and user-interface design to refine and personalize the music-discovery process. Future development in this intersection of music information retrieval and visual interaction could potentially transform user experiences in digital music environments.

6

CONCLUSION

This thesis investigates the integration of advanced deep learning techniques within the field of computer graphics, with a particular emphasis on digital reconstruction, generation, and exploration. The primary focus is on addressing the workflow involved in the reconstruction of material properties, including the upstream processes of data exploration and generation.

Traditionally, the processes from finding the desired image until reconstructing materials have been heavily dependent on manual effort. Obtaining material properties typically requires expensive professional lighting equipment, as well as meticulous acquisition and calibration procedures. Even with the aid of summarization systems, a significant amount of time is required to search through large unlabeled databases for the desired image, a challenge that parallels in the search for specific music files. It can become even more challenging when the exact data needed is not readily available within the database.

The rapid advancements in AI offer promising solutions to these challenges, by achieving complex data tasks without the need for explicit feature engineering. Instead, systems exploit the structure of the underlying high-dimensional data manifold. Such methods are highly effective at uncovering and leveraging latent, yet semantically-rich features of data, and are particularly effective in automating digital tasks such as segmentation, detection, and generation. Yet, pure automation might not be the answer to all user needs.

People have different tastes, goals and might even change direction during the generation of their own content - an artist typically makes many attempts before converging. It is therefore important to keep the human in the loop. We want to be able to explore data sets in the directions we want, but need support in finding our way. Similarly, we want to generate the illustration we want, but will guide the creation with (simplified) examples we like. The key to enabling such a sense-making of the space of possibilities are features that describe the content with less information, to make it navigable, describable and possible to create with a limited input. In this work, we showed that it is possible to address this challenge.

We began by addressing the challenge of encoding hard-to-describe texture appearances, which are inherently difficult to describe and manipulate using traditional methods. Through the use of learned deep neural features, we structured the data space in a way that captures the intricate nuances of material appearances. In the single-image SVBRDF

capture, we demonstrated that with a learned gradient descent approach we can reconstruct detailed material properties from a single, simplified input, effectively bridging the gap between sparse data and richly detailed outputs. When dealing with large texture databases, we further integrated a suite of interactive tools based on the data distribution in the space of learned neural features for efficient exploration and retrieval.

Building on this foundation, we explored interpolation methods that allow us to generate content by example. By optimizing the interpolation paths in the latent space of generative models, we were able to produce visually smooth and perceptually uniform transitions between material images. This capability is particularly significant because it enables the generation of new, detailed material maps from just a few example inputs, highlighting the power of deep learning to generate high-quality results even when starting with minimal data.

The concept of generating detailed results from simplified inputs was further extended to the domain of music discovery. Here, we showed that a simple input, such as a glyph representing the latent characteristics of a song, could be used to deliver a detailed and engaging music discovery experience. This approach not only enhances user interaction but also demonstrates how deep neural features can be mapped to intuitive visual cues, making complex data more accessible and meaningful to users.

Throughout this thesis, we show that deep neural features enable us to extract and utilize the underlying structure of complex data, allowing for detailed and meaningful outputs even from simplified inputs. While our efforts build upon previous research, we present new advancements that overcome previous limitations, which have been detailed in the chapters.

Beyond the processes researched in this dissertation, there are many possibilities to explore future directions. Texture has been an important element for the appearance of material. We have so far created multiple methods to assist humans to explore texture databases. Even though existing databases are huge, they are still a limited representation of the world. Facilitating the generation of natural material appearance missing from such datasets would significantly contribute to the diversity of textures.

The texture generation by examples nicely provides humans with convenience and flexibility in their exploration. Similarly, music generation and editing could benefit from features.

Although neural features are powerful in assisting humans with digital tasks, they might still find their limitation in tasks, when accuracy and precision are prioritised. For material properties (i.e. the SVBRDF) synthesis, current AI-based estimation from a single image largely simplifies the traditional capturing procedure, yet its reconstruction accuracy falls short in comparison. While it is possible to expand the diversity of the training database to potentially improve performance, we do not see successful attempts on it yet, possibly due to the high but still limited potential of data-based fitting methods. An analytical physical model could nicely complement estimating subtle details that are hard to accurately capture with learned neural features. Therefore, it seems an interesting direction to incorporate more realistic physical models in the property estimation process of the deep networks, so that the capture of the sample image can be more flexible in lighting and viewing setups.

Overall, in this dissertation, which focused on reconstruction, generation and explo-

ration of digital content, we show that with the assistance of deep neural-features, intuitive and user-friendly solutions can be produced that support creativity and exploration with minimal manual input. Although it is not possible to prove that this concept is general, we showed that it does apply to many contexts, from image collections to textures, material properties, and even music. With this insight, we hope to have shown a glimpse of the many possibilities that we believe are yet to come to support the human with all its limitations in navigating this seemingly unlimited space of possibilities.

BIBLIOGRAPHY

- [1] I. Goodfellow, Y. Bengio, and A. Courville. *Deep learning*. MIT press, 2016.
- [2] M. Aittala, T. Weyrich, and J. Lehtinen. “Practical SVBRDF capture in the frequency domain”. In: *ACM Transactions on Graphics (TOG)* 32.4 (2013), pp. 110–1.
- [3] H. Wang, W. Zhang, and A. Dong. “Measurement and modeling of Bidirectional Reflectance Distribution Function (BRDF) on material surface”. In: *Measurement* 46.9 (2013), pp. 3654–3661.
- [4] X. Li, Y. Dong, P. Peers, and X. Tong. “Modeling surface appearance from a single photograph using self-augmented convolutional neural networks”. In: *ACM Transactions on Graphics (TOG)* 36.4 (2017), pp. 1–11.
- [5] Z. Li, K. Sunkavalli, and M. Chandraker. “Materials for masses: SVBRDF acquisition with a single mobile phone image”. In: *Proceedings of the European conference on computer vision (ECCV)*. 2018, pp. 72–87.
- [6] V. Deschaintre, M. Aittala, F. Durand, G. Drettakis, and A. Bousseau. “Single-image svbrdf capture with a rendering-aware deep network”. In: *ACM Transactions on Graphics (TOG)* 37.4 (2018), pp. 1–15.
- [7] X. Zhou and N. K. Kalantari. “Adversarial Single-Image SVBRDF Estimation with Hybrid Training”. In: *Computer Graphics Forum* 40.2 (2021), pp. 315–325.
- [8] X. Luo, L. Scandolo, A. Bousseau, and E. Eisemann. “Single-Image SVBRDF Estimation with Learned Gradient Descent”. In: *Computer Graphics Forum*. Vol. 43. 2. Wiley Online Library. 2024, e15018.
- [9] K. Simonyan and A. Zisserman. “Very deep convolutional networks for large-scale image recognition”. In: *arXiv preprint arXiv:1409.1556* (2014).
- [10] Viso.ai. *VGG - Very Deep Convolutional Networks*. <https://viso.ai/deep-learning/vgg-very-deep-convolutional-networks/>. Accessed: 2024-06-10.
- [11] A. Zhang, Z. C. Lipton, M. Li, and A. J. Smola. *Dive into deep learning*. Cambridge University Press, 2023.
- [12] V. Jade. *Texture vs. Shape: The Bias in CNNs*. <https://towardsdatascience.com/texture-vs-shape-the-bias-in-cnns-5ee423edf8db>. Accessed: 2021-08-26.
- [13] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson. “How transferable are features in deep neural networks?” In: *Advances in neural information processing systems* 27 (2014).

- [14] X. Luo, L. Scandolo, and E. Eisemann. “Texture Browser: Feature-based Texture Exploration”. In: *Computer Graphics Forum*. Vol. 40. 3. Wiley Online Library. 2021, pp. 99–109.
- [15] L. Van der Maaten and G. Hinton. “Visualizing data using t-SNE.” In: *Journal of machine learning research* 9.11 (2008).
- [16] L. McInnes, J. Healy, and J. Melville. “Umap: Uniform manifold approximation and projection for dimension reduction”. In: *arXiv preprint arXiv:1802.03426* (2018).
- [17] A. Radford, L. Metz, and S. Chintala. “Unsupervised representation learning with deep convolutional generative adversarial networks”. In: *arXiv preprint arXiv:1511.06434* (2015).
- [18] T. Karras, T. Aila, S. Laine, and J. Lehtinen. “Progressive growing of gans for improved quality, stability, and variation”. In: *arXiv preprint arXiv:1710.10196* (2017).
- [19] T. Karras, S. Laine, M. Aittala, J. Hellsten, J. Lehtinen, and T. Aila. “Analyzing and improving the image quality of stylegan”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, pp. 8110–8119.
- [20] T. Chakraborty, U. R. KS, S. M. Naik, M. Panja, and M. Bayapureddy. “Ten years of generative adversarial nets (GANs): a survey of the state-of-the-art”. In: *Machine Learning: Science and Technology* 5.1 (2024), p. 011001.
- [21] T. Karras, S. Laine, and T. Aila. “A style-based generator architecture for generative adversarial networks”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, pp. 4401–4410.
- [22] W. Ye, X. Li, Y. Dong, P. Peers, and X. Tong. “Single image surface appearance modeling with self-augmented cnns and inexact supervision”. In: *Computer Graphics Forum* 37.7 (2018), pp. 201–211.
- [23] G. Vecchio, S. Palazzo, and C. Spampinato. “SurfaceNet: Adversarial SVBRDF Estimation from a Single Image”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 12840–12848.
- [24] J. Guo, S. Lai, C. Tao, Y. Cai, L. Wang, Y. Guo, and L.-Q. Yan. “Highlight-aware two-stream network for single-image SVBRDF acquisition”. In: *ACM Transactions on Graphics (TOG)* 40.4 (2021), pp. 1–14.
- [25] D. Gao, X. Li, Y. Dong, P. Peers, K. Xu, and X. Tong. “Deep inverse rendering for high-resolution SVBRDF estimation from an arbitrary number of images”. In: *ACM Transactions on Graphics (TOG)* 38.4 (2019), pp. 134–1.
- [26] Y. Guo, C. Smith, M. Hašan, K. Sunkavalli, and S. Zhao. “MaterialGAN: Reflectance Capture Using a Generative SVBRDF Model”. In: *ACM Transactions on Graphics (TOG)* 39.6 (2020).
- [27] M. Fischer and T. Ritschel. “Metappearance: Meta-learning for visual appearance reproduction”. In: *ACM Transactions on Graphics (TOG)* 41.6 (2022), pp. 1–13.
- [28] X. Zhou and N. K. Kalantari. “Look-Ahead Training with Learned Reflectance Loss for Single-Image SVBRDF Estimation”. In: *ACM Transactions on Graphics (TOG)* 41.6 (2022), pp. 1–12.

- [29] M. Andrychowicz, M. Denil, S. Gomez, M. W. Hoffman, D. Pfau, T. Schaul, B. Shillingford, and N. De Freitas. “Learning to learn by gradient descent by gradient descent”. In: *Advances in neural information processing systems* 29 (2016).
- [30] P. Putzky and M. Welling. “Recurrent inference machines for solving inverse problems”. In: *arXiv preprint arXiv:1706.04008* (2017).
- [31] T. Weyrich, J. Lawrence, H. P. Lensch, S. Rusinkiewicz, T. Zickler, *et al.* “Principles of appearance acquisition and representation”. In: *Foundations and Trends® in Computer Graphics and Vision* 4.2 (2009), pp. 75–191.
- [32] D. Guarnera, G. C. Guarnera, A. Ghosh, C. Denk, and M. Glencross. “BRDF representation and acquisition”. In: *Computer Graphics Forum* 35.2 (2016), pp. 625–650.
- [33] Y. Dong. “Deep appearance modeling: A survey”. In: *Visual Informatics* 3.2 (2019), pp. 59–68.
- [34] S. R. Marschner, S. H. Westin, E. P. Lafortune, K. E. Torrance, and D. P. Greenberg. “Image-based BRDF measurement including human skin”. In: *Rendering Techniques’ 99: Proceedings of the Eurographics Workshop in Granada, Spain, June 21–23, 1999*. Springer. 1999, pp. 131–144.
- [35] W. Matusik. “A data-driven reflectance model”. PhD thesis. Massachusetts Institute of Technology, 2003.
- [36] H. P. Lensch, J. Kautz, M. Goesele, W. Heidrich, and H.-P. Seidel. “Image-based reconstruction of spatial appearance and geometric detail”. In: *ACM Transactions on Graphics (TOG)* 22.2 (2003), pp. 234–257.
- [37] M. Weinmann, J. Gall, and R. Klein. “Material classification based on training data synthesized using a BTF database”. In: *Computer Vision—ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6–12, 2014, Proceedings, Part III* 13. Springer. 2014, pp. 156–171.
- [38] K. Kang, Z. Chen, J. Wang, K. Zhou, and H. Wu. “Efficient reflectance capture using an autoencoder.” In: *ACM Transactions on Graphics (TOG)* 37.4 (2018), pp. 127–1.
- [39] L.-P. Asselin, D. Laurendeau, and J.-F. Lalonde. “Deep SVBRDF estimation on real materials”. In: *2020 International Conference on 3D Vision (3DV)*. IEEE. 2020, pp. 1157–1166.
- [40] Y. Dong, J. Wang, X. Tong, J. Snyder, Y. Lan, M. Ben-Ezra, and B. Guo. “Manifold bootstrapping for SVBRDF capture”. In: *ACM Transactions on Graphics (TOG)* 29.4 (2010), pp. 1–10.
- [41] P. Ren, J. Wang, J. Snyder, X. Tong, and B. Guo. “Pocket reflectometry”. In: *ACM Transactions on Graphics (TOG)* 30.4 (2011), pp. 1–10.
- [42] Z. Hui, K. Sunkavalli, J.-Y. Lee, S. Hadap, J. Wang, and A. C. Sankaranarayanan. “Reflectance capture using univariate sampling of brdfs”. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2017, pp. 5362–5370.
- [43] Z. Zhou, G. Chen, Y. Dong, D. Wipf, Y. Yu, J. Snyder, and X. Tong. “Sparse-as-possible SVBRDF acquisition”. In: *ACM Transactions on Graphics (TOG)* 35.6 (2016), pp. 1–12.

- [44] M. Aittala, T. Weyrich, J. Lehtinen, *et al.* “Two-shot SVBRDF capture for stationary materials.” In: *ACM Transactions on Graphics (TOG)* 34.4 (2015), pp. 110–1.
- [45] C.-P. Wang, N. Snavely, and S. Marschner. “Estimating dual-scale properties of glossy surfaces from step-edge lighting”. In: *Proceedings of the 2011 SIGGRAPH Asia Conference*. 2011, pp. 1–12.
- [46] Y. Zhao, B. Wang, Y. Xu, Z. Zeng, L. Wang, and N. Holzschuch. “Joint SVBRDF Recovery and Synthesis From a Single Image using an Unsupervised Generative Adversarial Network.” In: *EGSR (DL)*. 2020, pp. 53–66.
- [47] V. Deschaintre, G. Drettakis, and A. Bousseau. “Guided fine-tuning for large-scale material transfer”. In: *Computer Graphics Forum*. Vol. 39. 4. Wiley Online Library. 2020, pp. 91–105.
- [48] J. Flynn, M. Broxton, P. Debevec, M. DuVall, G. Fyffe, R. Overbeck, N. Snavely, and R. Tucker. “Deepview: View synthesis with learned gradient descent”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 2367–2376.
- [49] K. Lønning, P. Putzky, J.-J. Sonke, L. Reneman, M. W. Caan, and M. Welling. “Recurrent inference machines for reconstructing heterogeneous MRI data”. In: *Medical image analysis* 53 (2019), pp. 64–78.
- [50] P. Putzky and M. Welling. “Invert to learn to invert”. In: *Advances in neural information processing systems* 32 (2019).
- [51] K. Cho. “Learning phrase representations using RNN encoder-decoder for statistical machine translation”. In: *arXiv preprint arXiv:1406.1078* (2014).
- [52] R. L. Cook and K. E. Torrance. “A reflectance model for computer graphics”. In: *ACM Transactions on Graphics (TOG)* 1.1 (1982), pp. 7–24.
- [53] B. Walter, S. R. Marschner, H. Li, and K. E. Torrance. “Microfacet models for refraction through rough surfaces”. In: *Proceedings of the 18th Eurographics conference on Rendering Techniques*. 2007, pp. 195–206.
- [54] V. Deschaintre, M. Aittala, F. Durand, G. Drettakis, and A. Bousseau. “Flexible svbrdf capture with a multi-image deep network”. In: *Computer Graphics Forum* 38.4 (2019), pp. 1–13.
- [55] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang. “The unreasonable effectiveness of deep features as a perceptual metric”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 586–595.
- [56] K. Yoshihiro. “Relighting humans: occlusion-aware inverse rendering for full-body human images”. In: *ACM Transactions on Graphics (TOG)* 37 (2018), pp. 270–1.
- [57] S. Saito, T. Simon, J. Saragih, and H. Joo. “PifuHD: Multi-level pixel-aligned implicit function for high-resolution 3d human digitization”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, pp. 84–93.
- [58] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. “Generative adversarial nets”. In: *Advances in neural information processing systems* 27 (2014).

- [59] A. Brock, J. Donahue, and K. Simonyan. “Large scale GAN training for high fidelity natural image synthesis”. In: *arXiv preprint arXiv:1809.11096* (2018).
- [60] T. White. “Sampling generative networks”. In: *arXiv preprint arXiv:1609.04468* (2016).
- [61] N. Lindow, D. Baum, and H.-C. Hege. “Perceptually linear parameter variations”. In: *Computer Graphics Forum*. Vol. 31. 2pt3. Wiley Online Library. 2012, pp. 535–544.
- [62] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. “Image quality assessment: from error visibility to structural similarity”. In: *IEEE transactions on image processing* 13.4 (2004), pp. 600–612.
- [63] F. Wang. *pytorch-msssim*. <https://github.com/VainF/pytorch-msssim>. Accessed: 2024-04-30.
- [64] R. Geirhos, P. Rubisch, C. Michaelis, M. Bethge, F. A. Wichmann, and W. Brendel. “ImageNet-trained CNNs are biased towards texture; increasing shape bias improves accuracy and robustness”. In: *arXiv preprint arXiv:1811.12231* (2018).
- [65] P. S. Heckbert. “Survey of texture mapping”. In: *IEEE computer graphics and applications* 6.11 (1986), pp. 56–67.
- [66] E. Lupton and J. C. Phillips. *Graphic design: the new basics (revised and expanded)*. Chronicle Books, 2015.
- [67] J. J. Gibson. *The perception of the visual world*. Houghton Mifflin, 1950.
- [68] M. Cimpoi, S. Maji, I. Kokkinos, S. Mohamed, and A. Vedaldi. “Describing textures in the wild”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2014, pp. 3606–3613.
- [69] N. Bhushan, A. R. Rao, and G. L. Lohse. “The texture lexicon: Understanding the categorization of visual texture terms and their relationship to texture images”. In: *Cognitive Science* 21.2 (1997), pp. 219–246.
- [70] L.-Y. Wei and M. Levoy. “Fast texture synthesis using tree-structured vector quantization”. In: *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*. 2000, pp. 479–488.
- [71] J. Dong, L. Wang, J. Liu, Y. Gao, L. Qi, and X. Sun. “A procedural texture generation framework based on semantic descriptions”. In: *Knowledge-Based Systems* 163 (2019), pp. 898–906.
- [72] P. Guehl, R. Allegre, J.-M. Dischler, B. Benes, and E. Galin. “Semi-Procedural Textures Using Point Process Texture Basis Functions”. In: *Computer Graphics Forum*. Vol. 39. 4. Wiley Online Library. 2020, pp. 159–171.
- [73] Adobe. *Substance Painter*. <https://www.substance3d.com/>. Accessed: 2020-04-30.
- [74] C. Olah, A. Mordvintsev, and L. Schubert. “Feature visualization”. In: *Distill* 2.11 (2017), e7.
- [75] H. Chang, O. Fried, Y. Liu, S. DiVerdi, and A. Finkelstein. “Palette-based photo recoloring.” In: *ACM Transactions on Graphics (TOG)* 34.4 (2015), pp. 139–1.

- [76] K. H. Tan. "Text-based image retrieval using image captioning". In: *Journal of Visual Communication and Image Representation* (2019).
- [77] K. Seetharaman, S. Selvaraj, *et al.* "Statistical tests of hypothesis based color image retrieval". In: *Journal of Data Analysis and Information Processing* 4.02 (2016), p. 90.
- [78] C. Carson, S. Belongie, H. Greenspan, and J. Malik. "Region-based image querying". In: *1997 Proceedings IEEE Workshop on Content-Based Access of Image and Video Libraries*. IEEE. 1997, pp. 42–49.
- [79] L. Piras and G. Giacinto. "Information fusion in content based image retrieval: A comprehensive overview". In: *Information Fusion* 37 (2017), pp. 50–60.
- [80] J. Yue, Z. Li, L. Liu, and Z. Fu. "Content-based image retrieval using color and texture fused features". In: *Mathematical and Computer Modelling* 54.3-4 (2011), pp. 1121–1127.
- [81] B. Julesz. "Visual pattern discrimination". In: *IRE transactions on Information Theory* 8.2 (1962), pp. 84–92.
- [82] J. I. Yellott. "Implications of triple correlation uniqueness for texture statistics and the Julesz conjecture". In: *JOSA A* 10.5 (1993), pp. 777–793.
- [83] B. S. Manjunath and W.-Y. Ma. "Texture features for browsing and retrieval of image data". In: *IEEE Transactions on pattern analysis and machine intelligence* 18.8 (1996), pp. 837–842.
- [84] J. Portilla and E. P. Simoncelli. "A parametric texture model based on joint statistics of complex wavelet coefficients". In: *International journal of computer vision* 40.1 (2000), pp. 49–70.
- [85] X. Dong and M. J. Chantler. "Texture similarity estimation using contours". In: *25th British Machine Vision Conference 2014*. BMVA Press. 2014, pp. 1–11.
- [86] J. Zujovic, T. N. Pappas, and D. L. Neuhoff. "Structural texture similarity metrics for image analysis and retrieval". In: *IEEE Transactions on Image Processing* 22.7 (2013), pp. 2545–2558.
- [87] X. Zhao, M. G. Reyes, T. N. Pappas, and D. L. Neuhoff. "Structural texture similarity metrics for retrieval applications". In: *2008 15th IEEE International Conference on Image Processing*. IEEE. 2008, pp. 1196–1199.
- [88] M. Kokare, B. Chatterji, and P. Biswas. "Comparison of similarity metrics for texture image retrieval". In: *TENCON 2003. Conference on convergent technologies for Asia-Pacific region*. Vol. 2. IEEE. 2003, pp. 571–575.
- [89] B. Dinakaran, J. Annapurna, and C. A. Kumar. "Interactive image retrieval using text and image content". In: *Cybern Inf Tech* 10 (2010), pp. 20–30.
- [90] M. Porta. "Browsing large collections of images through unconventional visualization techniques". In: *Proceedings of the working conference on Advanced visual interfaces*. 2006, pp. 440–444.

- [91] L. Liu, F. Shen, Y. Shen, X. Liu, and L. Shao. “Deep sketch hashing: Fast free-hand sketch-based image retrieval”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 2862–2871.
- [92] A. Krizhevsky, I. Sutskever, and G. E. Hinton. “Imagenet classification with deep convolutional neural networks”. In: *Advances in neural information processing systems*. 2012, pp. 1097–1105.
- [93] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. “Going deeper with convolutions”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 1–9.
- [94] D. G. Lowe. “Distinctive image features from scale-invariant keypoints”. In: *International journal of computer vision* 60.2 (2004), pp. 91–110.
- [95] H. Permuter, J. Francos, and I. H. Jermyn. “Gaussian mixture models of texture and colour for image database retrieval”. In: *2003 IEEE International Conference on Acoustics, Speech, and Signal Processing, 2003. Proceedings. (ICASSP'03)*. Vol. 3. IEEE. 2003, pp. III–569.
- [96] L. Zheng, Y. Yang, and Q. Tian. “SIFT meets CNN: A decade survey of instance retrieval”. In: *IEEE transactions on pattern analysis and machine intelligence* 40.5 (2017), pp. 1224–1244.
- [97] L. Liu, J. Chen, P. Fieguth, G. Zhao, R. Chellappa, and M. Pietikäinen. “From BoW to CNN: Two decades of texture representation for texture classification”. In: *International Journal of Computer Vision* 127.1 (2019), pp. 74–109.
- [98] A. Babenko, A. Slesarev, A. Chigorin, and V. Lempitsky. “Neural codes for image retrieval”. In: *European conference on computer vision*. Springer. 2014, pp. 584–599.
- [99] L. Xie, R. Hong, B. Zhang, and Q. Tian. “Image classification and retrieval are one”. In: *Proceedings of the 5th ACM on International Conference on Multimedia Retrieval*. 2015, pp. 3–10.
- [100] N. Garcia and G. Vogiatzis. “Learning non-metric visual similarity for image retrieval”. In: *Image and Vision Computing* 82 (2019), pp. 18–25.
- [101] D. Danon, H. Averbuch-Elor, O. Fried, and D. Cohen-Or. “Unsupervised natural image patch learning”. In: *Computational Visual Media* 5.3 (2019), pp. 229–237.
- [102] O. Mohamed, O. Mohammed, A. Brahim, *et al.* “Content-based image retrieval using convolutional neural networks”. In: *First International Conference on Real Time Intelligent Systems*. Springer. 2017, pp. 463–476.
- [103] A. Gordo, J. Almazan, J. Revaud, and D. Larlus. “End-to-end learning of deep visual representations for image retrieval”. In: *International Journal of Computer Vision* 124.2 (2017), pp. 237–254.
- [104] J. Yue-Hei Ng, F. Yang, and L. S. Davis. “Exploiting local features from deep networks for image retrieval”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*. 2015, pp. 53–61.

- [105] H. Noh, A. Araujo, J. Sim, T. Weyand, and B. Han. “Large-scale image retrieval with attentive deep local features”. In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 3456–3465.
- [106] J. Xue, H. Zhang, and K. Dana. “Deep texture manifold for ground terrain recognition”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 558–567.
- [107] O. Fried, S. Avidan, and D. Cohen-Or. “Patch2vec: Globally consistent image patch representation”. In: *Computer Graphics Forum*. Vol. 36. 7. Wiley Online Library. 2017, pp. 183–194.
- [108] Y. Hu, J. Dorsey, and H. Rushmeier. “A novel framework for inverse procedural texture modeling”. In: *ACM Transactions on Graphics (TOG)* 38.6 (2019), pp. 1–14.
- [109] F. Halley *et al.* “Perceptually relevant browsing environments for large texture databases”. PhD thesis. Heriot-Watt University, 2012.
- [110] S. Padilla, F. Halley, D. A. Robb, and M. J. Chantler. “Intuitive Large Image Database Browsing using Perceptual Similarity Enriched by Crowds”. In: *International Conference on Computer Analysis of Images and Patterns*. Springer. 2013, pp. 169–176.
- [111] N. Li, V. van Unen, T. Höllt, A. Thompson, J. van Bergen, N. Pezzotti, E. Eisemann, A. Vilanova, S. M. Chuva de Sousa Lopes, B. P. Lelieveldt, *et al.* “Mass cytometry reveals innate lymphoid cell differentiation pathways in the human fetal intestine”. In: *Journal of Experimental Medicine* 215.5 (2018), pp. 1383–1396.
- [112] N. Pezzotti, T. Höllt, B. Lelieveldt, E. Eisemann, and A. Vilanova. “Hierarchical stochastic neighbor embedding”. In: *Computer Graphics Forum*. Vol. 35. 3. Wiley Online Library. 2016, pp. 21–30.
- [113] T. Höllt, N. Pezzotti, V. van Unen, F. Koning, B. P. Lelieveldt, and A. Vilanova. “CyteGuide: Visual guidance for hierarchical single-cell analysis”. In: *IEEE Transactions on Visualization and Computer Graphics* 24.1 (2017), pp. 739–748.
- [114] T. Höllt, A. Vilanova, N. Pezzotti, B. P. Lelieveldt, and H. Hauser. “Focus+ context exploration of hierarchical embeddings”. In: *Computer Graphics Forum*. Vol. 38. 3. Wiley Online Library. 2019, pp. 569–579.
- [115] J. Yang, J. Fan, D. Hubball, Y. Gao, H. Luo, W. Ribarsky, and M. Ward. “Semantic image browser: Bridging information visualization with automated intelligent image analysis”. In: *2006 IEEE Symposium On Visual Analytics Science And Technology*. IEEE. 2006, pp. 191–198.
- [116] K. Mizuno, H. Y. Wu, and S. Takahashi. “Manipulating bilevel feature space for category-aware image exploration”. In: *2014 IEEE Pacific Visualization Symposium*. IEEE. 2014, pp. 217–224.
- [117] M. A. Cox and T. F. Cox. “Multidimensional scaling”. In: *Handbook of data visualization*. Springer, 2008, pp. 315–347.
- [118] M. Dowling, J. Wenskovitch, J. Fry, L. House, and C. North. “SIRIUS: Dual, Symmetric, Interactive Dimension Reductions”. In: *IEEE transactions on visualization and computer graphics* 25.1 (2018), pp. 172–182.

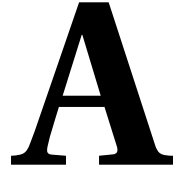
- [119] F. V. Paulovich, M. C. F. Oliveira, and R. Minghim. "The projection explorer: A flexible tool for projection-based multidimensional visualization". In: *XX Brazilian Symposium on Computer Graphics and Image Processing (SIBGRAPI 2007)*. IEEE. 2007, pp. 27–36.
- [120] D. M. Eler, M. Y. Nakazaki, F. V. Paulovich, D. P. Santos, G. F. Andery, M. C. F. Oliveira, J. B. Neto, and R. Minghim. "Visual analysis of image collections". In: *The Visual Computer* 25.10 (2009), pp. 923–937.
- [121] M. Worring and D. C. Koelma. "Insight in image collections by multimedia pivot tables". In: *Proceedings of the 5th ACM on International Conference on Multimedia Retrieval*. 2015, pp. 291–298.
- [122] M. Worring, D. Koelma, and J. Zahálka. "Multimedia pivot tables for multimedia analytics on image collections". In: *IEEE Transactions on Multimedia* 18.11 (2016), pp. 2217–2227.
- [123] X. Xie, X. Cai, J. Zhou, N. Cao, and Y. Wu. "A semantic-based method for visualizing large image collections". In: *IEEE transactions on visualization and computer graphics* 25.7 (2018), pp. 2362–2377.
- [124] K. Rodden, W. Basalaj, D. Sinclair, and K. Wood. "Evaluating a visualisation of image similarity as a tool for image browsing". In: *Proceedings 1999 IEEE Symposium on Information Visualization (InfoVis' 99)*. IEEE. 1999, pp. 36–43.
- [125] G. Y. Tian and D. Taylor. "Colour image retrieval using virtual reality". In: *2000 IEEE Conference on Information Visualization. An International Conference on Computer Visualization and Graphics*. IEEE. 2000, pp. 221–225.
- [126] A. Gomi, R. Miyazaki, T. Itoh, and J. Li. "CAT: A hierarchical image browser using a rectangle packing technique". In: *2008 12th International Conference Information Visualisation*. IEEE. 2008, pp. 82–87.
- [127] G. Schaefer. "Approaches for interactive browsing of large image datasets". In: *2016 IEEE 10th International Conference on Application of Information and Communication Technologies (AICT)*. IEEE. 2016, pp. 1–4.
- [128] L. Fei-Fei. "Imagenet: crowdsourcing, benchmarking & other cool things". In: *CMU VASC Seminar*. Vol. 16. 2010, pp. 18–25.
- [129] L. A. Gatys, A. S. Ecker, and M. Bethge. "Image style transfer using convolutional neural networks". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 2414–2423.
- [130] Z. Yang, J. Yue, Z. Li, and L. Zhu. "Vegetable image retrieval with fine-tuning VGG model and image hash". In: *IFAC-PapersOnLine* 51.17 (2018), pp. 280–285.
- [131] N. Kazak. "Performance analysis of spiral neighbourhood topology based local binary patterns in texture recognition". In: *International Journal of Applied Mathematics Electronics and Computers* Special Issue-1 (2016), pp. 338–341.
- [132] D. M. Chan, R. Rao, F. Huang, and J. F. Canny. "t-SNE-CUDA: GPU-Accelerated t-SNE and its Applications to Modern Data". In: *2018 30th International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD)*. IEEE. 2018, pp. 330–338.

- [133] N. Pezzotti, J. Thijssen, A. Mordvintsev, T. Höllt, B. Van Lew, B. P. Lelieveldt, E. Eisemann, and A. Vilanova. “GPGPU linear complexity t-SNE optimization”. In: *IEEE transactions on visualization and computer graphics* 26.1 (2019), pp. 1172–1181.
- [134] K. Fukunaga and L. Hostetler. “The estimation of the gradient of a density function, with applications in pattern recognition”. In: *IEEE Transactions on information theory* 21.1 (1975), pp. 32–40.
- [135] H. Bezerra, E. Eisemann, X. Décoret, and J. Thollot. “3d dynamic grouping for guided stylization”. In: *Proceedings of the 6th international symposium on Non-photorealistic animation and rendering*. 2008, pp. 89–95.
- [136] T. Höllt, N. Pezzotti, V. van Unen, F. Koning, E. Eisemann, B. Lelieveldt, and A. Vilanova. “Cytosplore: interactive immune cell phenotyping for large single-cell datasets”. In: *Computer Graphics Forum*. Vol. 35. 3. Wiley Online Library. 2016, pp. 171–180.
- [137] C. O. Ancuti, C. Ancuti, and P. Bekaert. “Enhancing by saliency-guided decolorization”. In: *CVPR 2011*. IEEE. 2011, pp. 257–264.
- [138] C. Ancuti and C. O. Ancuti. “Laplacian-guided image decolorization”. In: *2016 IEEE International Conference on Image Processing (ICIP)*. IEEE. 2016, pp. 4107–4111.
- [139] A. T. Chodos. “What does music mean to Spotify? An essay on musical significance in the era of digital curation”. In: *INSAM Journal of Contemporary Music, Art and Technology* 2 (2019), pp. 36–64.
- [140] C. Hosey, L. Vujović, B. St. Thomas, J. Garcia-Gathright, and J. Thom. “Just give me what I want: How people use and evaluate music search”. In: *Proceedings of the 2019 chi conference on human factors in computing systems*. 2019, pp. 1–12.
- [141] A. Anderson, L. Maystre, I. Anderson, R. Mehrotra, and M. Lalmas. “Algorithmic effects on the diversity of consumption on spotify”. In: *Proceedings of the web conference 2020*. 2020, pp. 2155–2165.
- [142] M. Schedl, H. Zamani, C.-W. Chen, Y. Deldjoo, and M. Elahi. “Current challenges and visions in music recommender systems research”. In: *International Journal of Multimedia Information Retrieval* 7 (2018), pp. 95–116.
- [143] J. M. Wolfe. “Visual search”. In: *Current biology* 20.8 (2010), R346–R349.
- [144] J. P. Lewis, R. Rosenholtz, N. Fong, and U. Neumann. “VisualIDs: automatic distinctive icons for desktop interfaces”. In: *ACM Transactions on Graphics (TOG)* 23.3 (2004), pp. 416–423.
- [145] M. Schedl, E. Gómez, J. Urbano, *et al.* “Music information retrieval: Recent developments and applications”. In: *Foundations and Trends® in Information Retrieval* 8.2-3 (2014), pp. 127–261.
- [146] M. Müller, R. Bittner, J. Nam, M. Krause, and Y. Özer. “Deep learning and knowledge integration for music audio analysis (Dagstuhl Seminar 22082)”. In: *Dagstuhl Reports* (2022).

- [147] T. Bertin-Mahieux, D. P. Ellis, B. Whitman, and P. Lamere. *The Million Song Dataset*. <http://millionsongdataset.com/>. Accessed: 2020-06-13.
- [148] P. Skidén. *API improvements and Update*. <https://developer.spotify.com/community/news/2016/03/29/api-improvements-update/>. Accessed: 2020-06-15.
- [149] P. Hamel and D. Eck. “Learning features from music audio with deep belief networks.” In: *ISMIR*. Vol. 10. Citeseer. 2010, pp. 339–344.
- [150] D. P. Kingma and M. Welling. “Auto-encoding variational bayes”. In: *arXiv preprint arXiv:1312.6114* (2013).
- [151] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. “Attention is all you need”. In: *Advances in neural information processing systems* 30 (2017).
- [152] P. Kolhoff, J. Preuß, and J. Loviscach. “Content-based icons for music files”. In: *Computers & Graphics* 32.5 (2008), pp. 550–560.
- [153] V. Setlur, C. Albrecht-Buehler, A. A. Gooch, S. Rossoff, and B. Gooch. “Semantics: Visual metaphors as file icons”. In: *Computer Graphics Forum*. Vol. 24. 3. Blackwell Publishing, Inc Oxford, UK and Boston, USA. 2005, pp. 647–656.
- [154] S. O. Martín. “Knowledge Extraction and Representation Learning for Music Recommendation and Classification”. PhD thesis. Ph. D. thesis, Universitat Pompeu Fabra.[Cited on page 139.], 2017.
- [155] K. Chen, B. Liang, X. Ma, and M. Gu. “Learning audio embeddings with user listening data for content-based music recommendation”. In: *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2021, pp. 3015–3019.
- [156] A. Saravanou, F. Tomasi, R. Mehrotra, and M. Lalmas. “Multi-Task Learning of Graph-based Inductive Representations of Music Content.” In: *ISMIR*. 2021, pp. 602–609.
- [157] A. Van den Oord, S. Dieleman, and B. Schrauwen. “Deep content-based music recommendation”. In: *Advances in neural information processing systems* 26 (2013).
- [158] J. Spijkervet and J. A. Burgoyne. “Contrastive learning of musical representations”. In: *arXiv preprint arXiv:2103.09410* (2021).
- [159] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton. “A simple framework for contrastive learning of visual representations”. In: *International conference on machine learning*. PMLR. 2020, pp. 1597–1607.
- [160] E. Law, K. West, M. I. Mandel, M. Bay, and J. S. Downie. “Evaluation of algorithms using games: The case of music tagging”. In: *ISMIR*. Citeseer. 2009, pp. 387–392.
- [161] G. Tzanetakis and P. Cook. “Musical genre classification of audio signals”. In: *IEEE Transactions on speech and audio processing* 10.5 (2002), pp. 293–302.
- [162] R. Figueroa. *Spotify 1.2M+ Songs*. <https://www.kaggle.com/datasets/rodolfofigueroa/spotify-12m-songs/data>. Accessed: 2023-12-15.

- [163] J. Fuchs, P. Isenberg, A. Bezerianos, F. Fischer, and E. Bertini. “The influence of contour on similarity perception of star glyphs”. In: *IEEE transactions on visualization and computer graphics* 20.12 (2014), pp. 2251–2260.
- [164] B. Dy, N. Ibrahim, A. Poorthuis, and S. Joyce. “Improving Visualization Design for Effective Multi-Objective Decision Making”. In: *IEEE Transactions on Visualization and Computer Graphics* 28.10 (2021), pp. 3405–3416.
- [165] Y. Hou, H. Zhu, H.-N. Liang, and L. Yu. “A study of the effect of star glyph parameters on value estimation and comparison”. In: *Journal of Visualization* (2022), pp. 1–15.
- [166] K. Pearson. “LIII. On lines and planes of closest fit to systems of points in space”. In: *The London, Edinburgh, and Dublin philosophical magazine and journal of science* 2.11 (1901), pp. 559–572.
- [167] E. Amid and M. K. Warmuth. “TriMap: Large-scale dimensionality reduction using triplets”. In: *arXiv preprint arXiv:1910.00204* (2019).
- [168] Y. Wang, H. Huang, C. Rudin, and Y. Shaposhnik. “Understanding how dimension reduction tools work: an empirical approach to deciphering t-SNE, UMAP, TriMAP, and PaCMAP for data visualization”. In: *The Journal of Machine Learning Research* 22.1 (2021), pp. 9129–9201.
- [169] T. Munzner. *Visualization analysis and design*. CRC press, 2014.
- [170] D. Kammer, M. Keck, T. Gründer, A. Maasch, T. Thom, M. Kleinstaub, and R. Groh. “Glyphboard: Visual exploration of high-dimensional data combining glyphs with dimensionality reduction”. In: *IEEE transactions on visualization and computer graphics* 26.4 (2020), pp. 1661–1671.
- [171] M. Brehmer, R. Kosara, and C. Hull. “Generative design inspiration for glyphs with diatoms”. In: *IEEE Transactions on Visualization and Computer Graphics* 28.1 (2021), pp. 389–399.
- [172] R. Borgo, J. Kehr, D. H. Chung, E. Maguire, R. S. Laramée, H. Hauser, M. Ward, and M. Chen. “Glyph-based Visualization: Foundations, Design Guidelines, Techniques and Applications.” In: *Eurographics (state of the art reports)*. 2013, pp. 39–63.
- [173] M. Friendly. “Statistical graphics for multivariate data”. In: *SAS SUGI* 16 (1991), pp. 1157–1162.
- [174] M. Keck and L. Engeln. “Sparkle glyphs: A glyph design for the analysis of temporal multivariate audio features”. In: *Proceedings of the 2022 International Conference on Advanced Visual Interfaces*. 2022, pp. 1–3.
- [175] S. E. Palmer. *Vision science: Photons to phenomenology*. MIT press, 1999.
- [176] A. Klippel, F. Hardisty, and C. Weaver. “Star plots: How shape characteristics influence classification tasks”. In: *Cartography and Geographic Information Science* 36.2 (2009), pp. 149–163.
- [177] J. Fuchs. “Glyph design for temporal and multi-dimensional data: Design considerations and evaluation”. In: *Information Visualization* (2015).

- [178] P. Knees and K. Andersen. “Searching for audio by sketching mental images of sound: A brave new idea for audio retrieval in creative music production”. In: *Proceedings of the 2016 ACM on International Conference on Multimedia Retrieval*. 2016, pp. 95–102.
- [179] D. W. Cunningham and C. Wallraven. *Experimental design: From user studies to psychophysics*. CRC Press, 2011.
- [180] S. J. Blanchard and I. Banerji. “Evidence-based recommendations for designing free-sorting experiments”. In: *Behavior research methods* 48 (2016), pp. 1318–1336.



SUPPLEMENTARY MATERIAL OF CHAPTER 4

A.1. INTRODUCTION

We conducted a user evaluation to validate the effectiveness and advantages of our method. We wish to perform two evaluations: comparison with alternative methods, and features evaluation. In a first part of our study, we compared against displaying images on a grid (using thumbnails), which is the standard file-system solution. It is the visualization that users typically have the most experience in using. It also requires no metadata (which also holds as is the case for our system). Additionally, we compare to text-based solutions. We also evaluate the four features compared to only using our basic *overview* mode, *multi-scale replication*, *clustering*, *prioritized-tSNE*, and *image-based search*. The first two features aim at general use cases and evaluation of them requires no special settings of the target. Therefore, they were integrated in the first part of the study, acting as additional comparisons. For the latter two features, they aim to improving the retrieval in cases where the number of images is very large, and thus were evaluated separately in the second part of the study.

A.2. COMPARISON WITH ALTERNATIVE METHODS

A.2.1. TASKS

1. Given a reference texture, users are asked to find it in two ways from the same database: standard grid view, and our texture browser system. The four textures for retrieval are shown in Fig. A.1. With our system, users will retrieve the images twice, once starting from the overview without clustering enabled, and later we enforced the cluster view. In this task, we record the times taken for the task for the different procedures. Note that the positions of images in our system changed for each participant, and for each task, since the embedding was rotated by a random angle.
2. Given an abstract descriptive word from the DTD database, users are asked to search for an appropriate texture, which matches this description. In this task, we

record the label that the retrieved texture had in the DTD database and compare it to the goal word.

A.2.2. PROCEDURE

1. Provide access to the texture browser software, as well as access to a folder with randomly indexed images for each of the retrieval tasks.
2. Provide a tutorial video of our TextureBrowser tool to familiarize the participants with the GUI and all functionalities of the tool.
3. Let participants carry out the tasks described above, record time, the used functions for each task, and their feedback. Let participants fill in a questionnaire.



Figure A.1: Images used for the retrieval task of methods comparison.

A.2.3. RESULTS AND DISCUSSION

In total, 16 users participated in our user evaluation. It was their first exposure to our interface and were asked to finish the two tasks listed in the evaluation method above. For each task, retrieval time was recorded and an evaluation questionnaire was filled.

For Task 1, the retrieval times were recorded for the different systems. Table A.1 and Table A.2 show the detailed timing results, as well as the user satisfaction scores reported by the users in Table A.3. The metrics for satisfaction of user interaction is a 5 level Likert scale, where higher means better (specifically, 1: very unsatisfied; 2: unsatisfied; 3: moderate; 4: satisfied; 5: very satisfied).

According to Table A.2, the time to retrieve the target textures via the grid interface is substantially reduced via the usage of our tool, either with non-clustered mode or with cluster view mode, with at least a factor of 2, except for user #13 where the reduction is mild. One case worth noticing is that user #15 gave up the retrieval of the last target texture via the grid view after a total search time that exceeded 23 minutes, while succeeding with our solution in less than three minutes (with no cluster view).

On average, the users rated our method higher than the grid view on interaction score. Among all users, only user #9 scored the grid view higher than our system, and only for the cluster view mode, despite having a shorter retrieval time in our system with the cluster view. Overall, the users preferred their interaction without the cluster view (avg. score 4.25) slightly over the cluster view (avg. score 3.84).

User	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
	Grid															
Texture 1	0:10	0:41	0:58	0:23	0:44	0:48	0:34	0:28	0:35	1:30	0:27	0:48	1:16	0:46	0:18	4:02
Texture 2	5:07	2:06	1:01	0:43	0:08	5:40	1:10	1:55	2:21	0:51	4:44	0:29	1:00	1:05	3:23	2:15
Texture 3	0:47	1:17	1:23	0:17	1:04	2:00	0:27	1:37	2:02	1:00	0:56	0:30	0:50	0:27	1:48	1:33
Texture 4	0:39	5:18	4:11	0:48	1:46	1:26	2:00	1:46	0:58	1:58	1:02	2:20	1:10	1:26	17:40	3:00
	Ours															
Texture 1	0:05	0:17	0:11	0:11	0:18	0:15	0:08	0:10	0:06	0:26	0:13	0:13	1:00	0:08	0:20	0:17
Texture 2	0:35	1:24	1:40	0:19	0:38	1:00	1:26	0:10	0:25	0:39	0:59	0:11	0:46	0:27	1:06	0:38
Texture 3	0:20	0:43	0:08	0:14	0:05	0:20	0:14	0:24	0:20	0:23	0:20	0:20	1:10	0:40	0:43	0:26
Texture 4	2:26	2:41	0:15	0:20	1:03	0:03	0:13	0:29	0:17	1:09	0:13	0:30	0:40	0:14	0:14	0:58
	Ours (cluster view)															
Texture 1	0:30	0:32	0:10	0:23	0:25	0:46	0:20	0:10	0:21	1:03	0:09	0:40	0:45	0:41	0:43	0:21
Texture 2	0:46	0:51	0:32	0:16	0:31	0:40	0:36	0:15	0:30	0:47	1:40	0:31	0:25	0:22	1:26	0:57
Texture 3	0:23	0:12	0:46	0:18	0:27	0:56	0:12	0:35	0:18	0:27	0:39	0:31	0:46	0:27	0:57	0:31
Texture 4	0:19	0:59	0:07	0:13	0:22	0:22	0:21	0:50	0:27	1:10	1:22	0:22	0:50	0:26	0:42	0:25

Table A.1: Task 1 retrieval times for individual textures

User	Grid	Ours	Ours (clustering)
1	06:43	03:26	01:58
2	09:22	05:05	02:34
3	07:33	02:22	01:35
4	02:08	01:04	01:10
5	03:42	02:04	01:45
6	09:54	01:38	02:44
7	04:11	02:01	01:29
8	05:46	01:13	01:50
9	05:56	01:08	01:36
10	05:19	02:37	03:27
11	07:09	01:45	03:50
12	04:07	01:14	02:04
13	04:16	03:36	02:46
14	03:44	01:29	01:56
15	$\geq 23:09$	02:23	03:48
16	10:50	02:19	02:14
Average	07:07	02:13	02:18

Table A.2: Total user timings for task 1

User	Grid	Ours	Ours (clustering)
1	3	4	4.5
2	2	4	4
3	1	3	5
4	2	5	4
5	2	4	3
6	1	4	3
7	2	5	4
8	1	5	3
9	4	5	3
10	2	4	3
11	3	5	4
12	2	4	4
13	3	3	4
14	2	5	4
15	1	4	4
16	1	4	5
Average	2	4.25	3.84

Table A.3: User interaction scores




























User	DTD Word		DTD Word	
		Keyword: Porous		Keyword: Interlaced
1		Porous		Zigzagged
2		Honeycombed		Grid
3		Scaly		Woven
4		Porous		Crosshatched
5		Porous		Braided
6		Porous		Braided
7		Fibrous		Braided
8		Bubbly		Braided
9		Pitted		Woven
10		Porous		Woven
11		Flecked		Woven
12		Pitted		Braided
13		Porous		Braided
14		Porous		Knitted
15		Porous		Interlaced
16		Dotted		Banded

Table A.4: Images selected by users in task 2

From open feedback the users provided on the questionnaire, the main reason of preference for the unclustered exploration is that it feels more intuitive for them. This is reasonable since the cluster view requires more involved control (i.e., defining an appropriate number of clusters). Furthermore, some users noted that the resulting clusters did not group textures according to their expectation, and choosing the wrong cluster to explore would not allow them to find the target texture. Nevertheless, some users

noted that the cluster view is better suited for retrieval of textures with very well defined features, such that the corresponding cluster is well defined and easier to find, especially when the overall number of textures is large. With a more sophisticated clustering method, this situation can be alleviated. Finally, some users also mentioned that they believe that training would improve their performance.

MULTI-SCALE

We also evaluated the effectiveness of our multiscale replication. The first target texture among the given four was present twice in the embedding, via the embed position of the feature vector for the original version and one of the blurred versions. Out of 16 participants, 12 obtained the target based on the position corresponding to the blurred version, in an average 27 seconds, while the rest obtained the target in its original embedding position, in an average 38 seconds. This validates our hypothesis that different users first search for textures in groups according to features of different scales, and our multiscale replication allows them to find the texture in either case. On average, users were inclined to retrieve this image via the features of its blurred version, and the retrieval time in that case is shorter. We believe that this is because the large-scale features are easier to identify starting from an unzoomed view, as is the case in our tool.

TEXT-BASED RETRIEVAL

In Task 2, each user retrieved two textures via our interface according to their understanding of the two descriptive words they were given. These words are taken from the Describable Textures Dataset (DTD), which contains 1519 images, and groups them according to 47 words (terms/categories) inspired from human perception. The retrieved textures for each participant are shown in Table A.4. The words provided to the users were *porous* and *interlaced*.

From the results shown in Table A.4, we can see that the interpretation of a specific adjective differs from person to person. For the first word, *porous*, half of the users obtained a texture whose describing word from DTD is identical to the given word. For the other half, the retrieved images have a common feature of small holes, which could roughly match the expectation of the word *porous*.

For the second word, *interlaced*, only one user (#15) retrieved a texture that is labeled with that word in DTD. Most of the users retrieved textures that can be described with the words *woven* and *braided*. These words can clearly describe texture features that are very similar, and we believe most non-expert users of such a word-based classification system would struggle to differentiate among them, as shown by our results.

A.3. PRIORITIZED T-SNE AND IMAGE-BASED SEARCH TOOL EVALUATION

The prioritized-tSNE and image-based search features were evaluated in a second part of the study. We divided the participants into two groups (group A and group B) to evaluate each feature independently, group A without using any feature and group B using the

specified features. Since these two features aim at solving retrieval in a large and crowded database, the texture database used here contains 5824 textures.

A.3.1. TASKS

1. **Prioritized-tSNE:** Given a reference texture, users are asked to find the identical texture via our interface. For group A, users can browse around freely with only the overview panel of our interface, but without using any of the advanced features (prioritized-tSNE, image-based search, or clustering). For group B, users first try to select a texture similar to the target texture from the overview panel as an input to the prioritized-tSNE tool, and assign it with a large weight to spread out the crowd where the target texture may belong. After that, they are suggested to browse around the highlighted vicinity for further retrieval. The textures for retrieval are shown as the two in the left in Fig. A.2.
2. **Image-based search:** Given a reference texture, users are asked to find the identical texture using our interface. For group A, users can again browse around freely with only the overview panel of our interface, but without using any of the advanced features. For group B, users first use our image-based tool to locate a similar texture in the database, after which they can freely browse until they locate the reference image. We provide two textures similar to the reference texture as as input to the image-based retrieval, and users are also allowed to draw sketches as well. The reference textures for retrieval are shown as the two in the right in Fig. A.2.



Figure A.2: Images used for the retrieval tasks of features evaluation.

A.3.2. PROCEDURE

1. Provide access to our texture browser software.
2. Provide a tutorial video for Group A on how to use the overview mode of the interface, including zooming in and out, translating, selecting images, and tiling in a grid. Additionally, create a tutorial video for Group B that covers using the overview mode, the prioritized-tSNE, and the image-based search tools.
3. Let participants carry out the tasks described above, record time, and their feedback (optional). Let participants fill in a questionnaire.

A.3.3. RESULTS AND DISCUSSION

In total 16 users were randomly divided into two groups, each with 8 users. The recorded retrieval time for each texture is shown in Table A.5 and Table A.6.

PRIORITIZED T-SNE

According to Table A.5, in the prioritized-tSNE evaluation, the average time of group A (using only the overview mode) is more than two times that of group B (using the prioritized-tSNE tool). This suggests that by using our prioritized-tSNE tool, the retrieval can be substantially faster. There is an outlier, user #13, who was misled by a similar texture, an image of cracked glass, and spent a long time in the wrong region according to the feedback. This suggests that when using our prioritized-tSNE tool, it is important for the user to have an impression of the overall distribution first, such that an appropriate image can be selected as input.

From the free feedback in the questionnaire, some users from the group A suggested that a tool that can spread the overlapped crowd out would be helpful. This also reflects the usefulness of the prioritized-tSNE tool.

Group	User	Texture 1	Texture 2	Total
A	1	04:02	06:54	10:56
A	2	02:38	01:36	04:14
A	3	01:15	02:46	04:01
A	4	04:04	00:43	04:47
A	5	01:15	02:05	03:20
A	6	01:10	03:30	04:40
A	7	03:52	04:30	08:22
A	8	04:56	01:57	06:53
A	Average	02:54	03:00	05:54
B	9	01:06	00:53	01:59
B	10	00:52	00:54	01:46
B	11	00:43	01:01	01:44
B	12	00:31	00:52	01:23
B	13	10:34	00:32	11:06
B	14	01:09	00:40	01:49
B	15	01:12	01:07	02:19
B	16	01:03	00:42	01:45
B	Average	02:08	00:50	02:58

Table A.5: Retrieval times for individual textures in feature “prioritized-tSNE” evaluation

IMAGE-BASED SEARCH

In the evaluation of the image-based search tool, as shown in Table A.6, the average time to finish the task of group A (using only the overview mode) is at least 4 times that of group B (using the image-based search tool). In group A, two users (#2 and #4) gave up

the task after 5 minutes and 3 minutes, respectively. From their feedback, we learned that user #2 switched the retrieval among several possible regions but was still not able to locate the target. User #4 gave up retrieval due to overlapping textures and suggested that a tool to spread the crowded region out would be useful. In group B, almost all the users succeeded in finishing the task using less time than those in group A, except for user #10 for whom the retrieval was stopped at 5 minutes and the reason was similar to that of user #4. This can probably be solved by the use of our prioritized-tSNE tool. Overall, thanks to the fast identification of the target region in the large database when using the tested tool, the retrieval time was substantially shorter.

Group	User	Texture 3	Texture 4	Total
A	1	13:08	16:00	29:08
A	2	≥05:00	≥05:00	≥10:00
A	3	07:06	05:28	12:34
A	4	≥03:00	01:23	≥04:23
A	5	04:14	03:58	08:12
A	6	10:00	01:30	11:30
A	7	00:47	02:42	03:29
A	8	04:57	02:32	07:29
A	Average	06:01	04:49	10:50
B	9	00:20	01:33	01:53
B	10	≥05:00	00:30	≥05:30
B	11	01:02	00:58	02:00
B	12	00:43	02:07	02:50
B	13	01:05	02:03	03:08
B	14	00:40	00:30	01:10
B	15	00:43	00:25	01:08
B	16	01:21	00:43	02:04
B	Average	01:21	01:06	02:27

Table A.6: Retrieval times for individual textures in feature “sketching” evaluation

B

SUPPLEMENTARY MATERIAL OF CHAPTER 5

B.1. INTRODUCTION

Given the subjective nature of visualizing and perceiving music, our hypothesis and experimental design are evaluated through user tests. The major focus of the evaluation lies in our proposed features: from the effectiveness of the icon to a larger system-evaluation.

The interface was implemented in a web app to facilitate remote user testing. It allows for real-time search in a database of 10K songs.

We targeted a participant demographic of “non-expert but generally computer-literate” adults and emphasized diversity in gender and age across ranges from 20-29 to over 70 years. To reduce response bias, participation was anonymous. Using an a-priori sample size calculator with an expected medium effect size ($d = 0.5$), we determined that a minimum of 27 participants would achieve a statistical power of 0.8 and a significance level of 0.05, assuming analysis via paired samples t-test for certain tasks. Consequently, we garnered 38 responses in the evaluation. The distributions for their respective age, gender and experience with the Spotify streaming service can be seen in Fig. B.1.

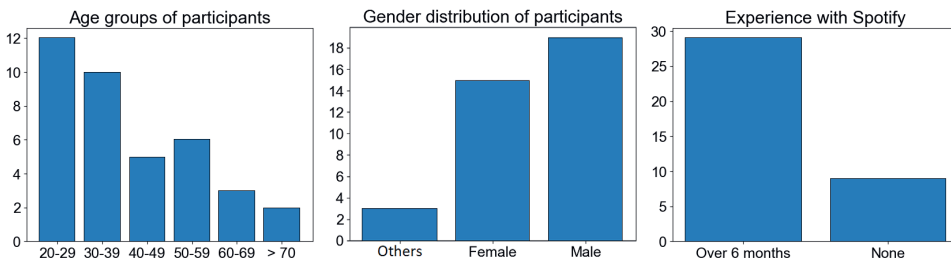


Figure B.1: Demographic information on the participants of the user study.

The user study consists of five tests and each of them is based on the following tasks:

1. Test one: Visual clustering (free-grouping task).

2. Test two: Outlier detection (five-alternative forced-choice task).
3. Test three: Generalisation, Contrast and CVD Robustness (matching-to-sample task).
4. Test four: Search-by-icon (real-world task).
5. Test five: Search-in-playlist (real-world task)

B.2. TEST ONE: VISUAL CLUSTERING

B.2.1. TASK

The goal of this test was to evaluate how well the icons capture the similarity of their features and how much users agree on this. Participants were asked to visually form clusters from a set of 60 icons, without knowing song titles or other information. Participants could use any number of clusters and were allowed to leave a set of spare icons that did not fit to anything else. An example screenshot for this test is shown in Fig. B.2. To ensure that there is a diversity in the selection yet still the possibility to make clusters, we sampled 10 data points from six of the clusters grouped by applying the k-means clustering algorithm ($k = 10$). Each participant worked with the same set of icons but their presentation was in a random order.



Figure B.2: An example screenshot for test one.

B.2.2. RESULTS AND DISCUSSION

To see how users agree on the clustering, we calculated a co-occurrence matrix of the clusters made by participants and a cosine similarity matrix of the feature vectors. The resulting matrices can be seen in Fig. B.3.

Our analysis indicates a consensus among users regarding the clusters. Initial examination reveals a striking resemblance between the co-occurrence matrix and the similarity

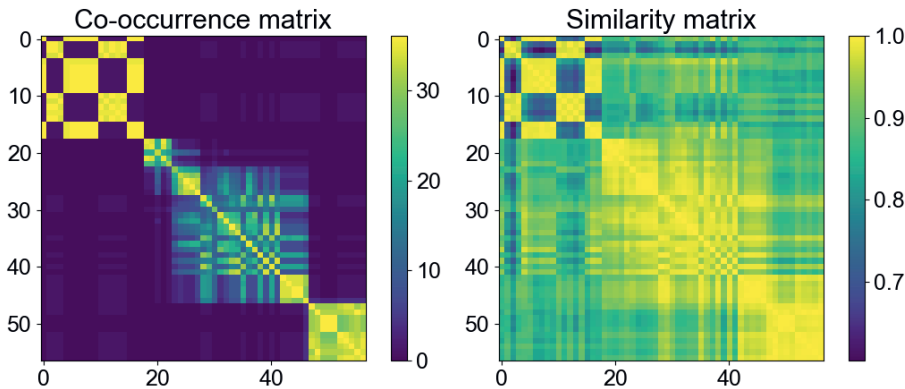


Figure B.3: Co-occurrence matrix of the clusters made by participants (left) and a cosine similarity matrix of the feature vectors (right).

matrix. To quantify this relationship, we computed the pairwise Pearson correlation coefficient, which resulted in a value of 0.6. This value denotes a ‘moderate’ linear correlation, suggesting a significant degree of agreement among users in their clustering decisions.

B.3. TEST TWO: OUTLIER DETECTION

B.3.1. TASK

This test builds upon test one, by using the cluster data the participants provided themselves. It is in essence a five-alternative forced-choice task and is to see how well the icons represent similar music and if the clustering allows users to spot outlier songs easily.

For each participant, we randomly selected four songs from a single cluster, along with one song from a different cluster, and then presented these five songs in a randomized sequence. Participants were then asked to identify the song that sounded distinct from the rest. An example screenshot for this test is shown in Fig. B.4. We repeated this three times for each participant.

B.3.2. RESULTS AND DISCUSSION

The outlier recognition rates achieved by the participants can be seen in Figure B.5 and the descriptive statistics of the results can be found in Table B.1.

The expected recognition rate when of random guessing would be 0.2. It seems rather likely that our results with a mean recognition rate of 0.745 is a considerable improvement. We observe that $p = 0.000000000000002$ (one sample one-tailed t-test), finding an effect size of 2.375 (Cohen’s d).

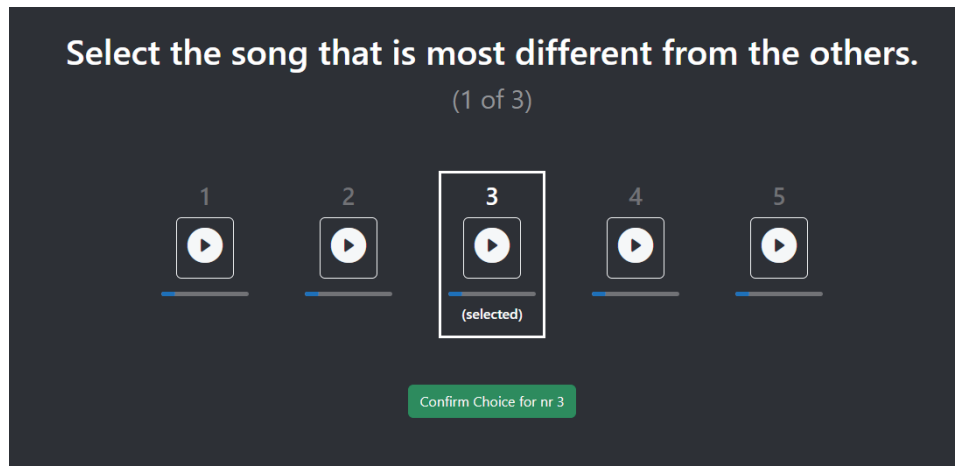


Figure B.4: An example screenshot for test two.

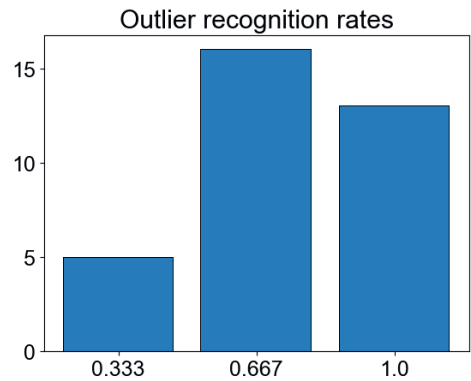


Figure B.5: Recognition rates obtained for outlier detection in test two.

Mean	Median	Mode	Std	Variance
0.745	0.667	0.667	0.230	0.053

Table B.1: Descriptive statistics of the recognition rates obtained for outlier detection.

B.4. TEST THREE: GENERALISATION, CONTRAST AND CVD ROBUSTNESS

B.4.1. TASK

The goal of this test was threefold:

1. Evaluate if the most similar icon aligns with the data point having the highest cosine similarity. In other words: if the icon is generally meaningful for representing high-

dimensional data.

2. Evaluate if the contrast enhanced version of the icon improves performance in terms of time-on-task and accuracy for finding the most similar icon.
3. Evaluate the robustness of the icon design against colour blindness by testing the time-on-task and accuracy with a color vision deficiency-simulated version of the icon.

B

We presented the user with nine icons that are all rather similar. One of the icons was the target icon. We asked participants to select the icon most similar to the target icon. An example screenshot can be seen in Fig. B.6.

We performed this test for three different ‘rendering modes’:

- ‘default’, as the icon was designed and explained in Section 3.
- ‘contrast’, with 100% increase of contrast between the nine icons, as explained in Section 4.4.
- ‘CVD’, with CVD simulated on the colour rendering, more specifically deuteranomaly - the most common form of colour blindness.

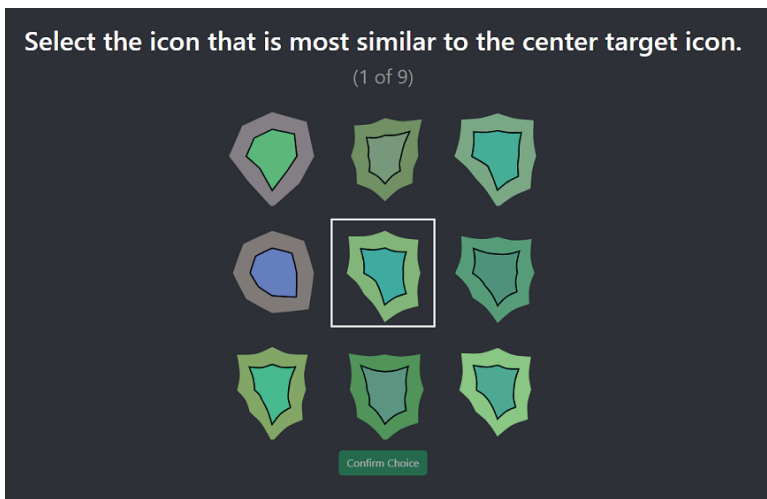


Figure B.6: An example screenshot for Test 3.

We repeated the task nine times for each participant: three times for each rendering mode.

B.4.2. RESULTS AND DISCUSSION

Overall, we find that the icon performs well in the matching-to-sample task. A one-way ANOVA test conducted across all three rendering modes yielded a p-value of 0.450, indicating that rendering mode has no significant impact on performance in the matching-to-sample task. This outcome suggests that each icon rendering mode performs comparably

well, affirming the robustness of our icon to color blindness. The effectiveness of our redundant encoding strategy in enhancing recognition and matching accuracy is thus supported by these results.

B

COMPARISON BETWEEN ‘DEFAULT’ AND ‘CONTRAST’ ICONS.

Fig. B.7 and Table B.2 present a comparison of recognition rates between the high contrast and ‘default’ rendering of the icon. Similarly, Fig. B.8 and Table B.3 display a comparison of time-on-task between the high contrast and ‘default’ rendering of the icon.

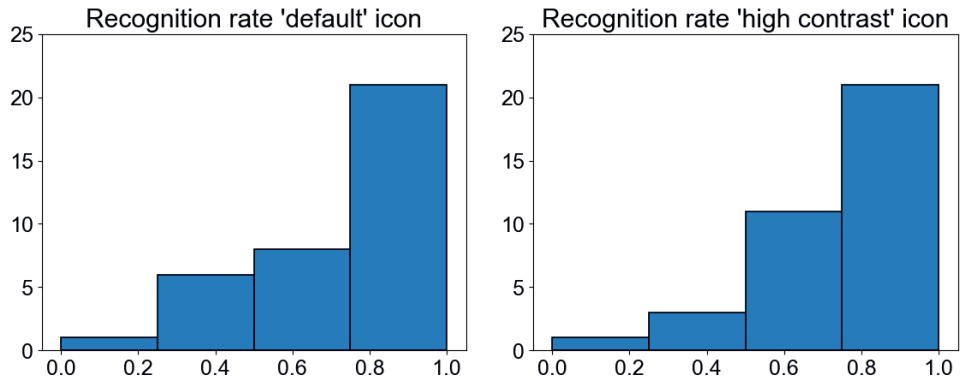


Figure B.7: Comparison of recognition rates obtained for matching-to-sample with the ‘default’ and ‘contrast’ version of the icon.

Mode	Mean	Median	Mode	Std	Variance
Default	0.706	0.750	1.000	0.277	0.077
Contrast	0.785	1.000	1.000	0.271	0.074

Table B.2: Descriptive statistics of the data as displayed in Fig. B.7.

We had expected the high contrast version to yield higher recognition rates than the default version. Inspecting the plots and the statistical descriptions, this seems to be the case: From Table B.2, we notice that for high-contrast rendering, the mean recognition rate is higher than for the ‘default’ version of icon and indeed we see the distribution in Fig. B.7 shift a bit to the right. However, for a statistical analysis, we find a medium-sized effect size of 0.283 (Cohen’s d), meaning that there is no sufficient statistical significance ($p = 0.103$ for a one-tailed paired-samples t-test).

In terms of time-on-task, we had expected the high contrast icon to allow for faster selection than the default icon and that seems to be indeed the case. As with the recognition rates, we find a medium-sized effect of 0.202 (Cohen’s d) but fail to establish strong significance: $p=0.084$ (one-tailed paired-samples t-test).

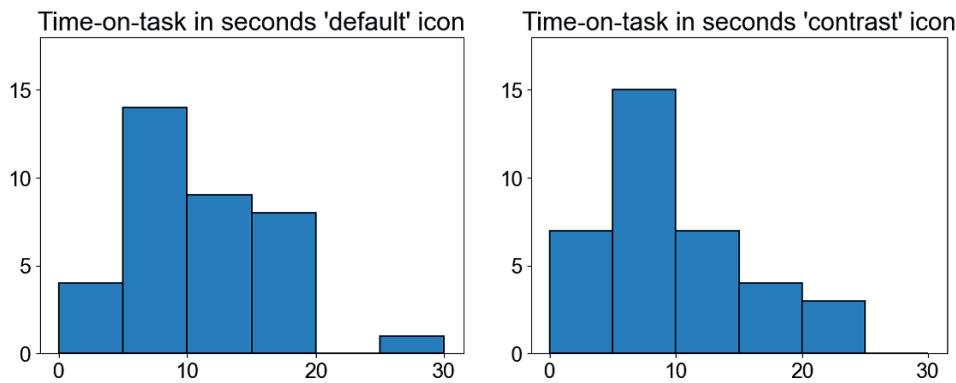


Figure B.8: Comparison of the time-on-task for matching-to-sample with the ‘default’ and ‘contrast’ version of the icon.

Mode	Mean	Median	Mode	Std
Default	10.862	9.742	5.446	29.658
Contrast	9.754	7.812	5.385	28.999

Table B.3: Descriptive statistics of the data as displayed in Fig. B.8.

CVD ROBUSTNESS

We are interested in a comparison of a CVD simulated mode icon with the ‘default’ icon, to see if the redundant encoding in our icon indeed makes the icon more robust to CVD. Therefore we compare the recognition rates we found in the sample matching for the ‘default’ and CVD rendering modes. In Fig. B.9, we can compare the distribution of the recognition rates participants achieved for the high contrast version of the icon with the default rendering of the icon, Table B.4 provides the descriptive statistics for the data.

Up front, we hypothesised that the CVD version would under-perform slightly in comparison with the default version of the icon. There seems to be a change in the distribution, where the median value does shift from 0.750 to 0.667 (Table B.4. We find that the mean recognition is a bit higher but this might be statistical noise, as we cannot confirm any statistical significance between these distributions: a two-tailed paired-samples t-test yields a p value of 0.715.

Mode	Mean	Median	Mode	Std	Variance
Default	0.706	0.750	1.000	0.277	0.077
CVD	0.741	0.667	0.667	0.231	0.053

Table B.4: Descriptive statistics of the data as displayed in Fig. B.7.

B

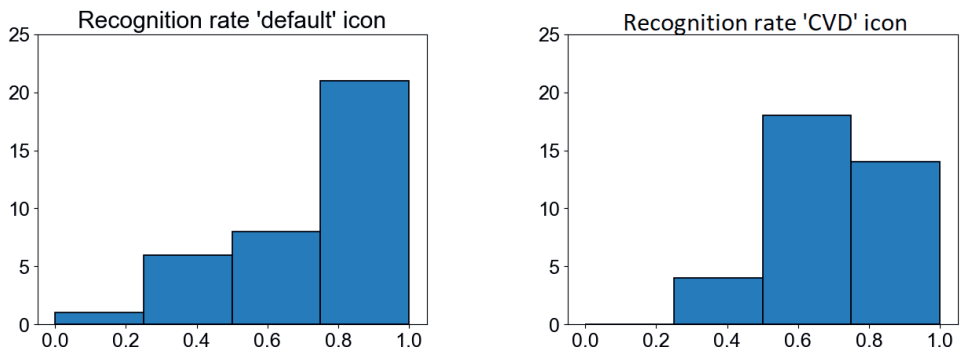


Figure B.9: Comparison of recognition rates obtained for matching-to-sample with the 'default' and 'CVD' version of the icon.

B.5. TEST FOUR: SEARCH-BY-ICON

B.5.1. TASK

The test aimed to assess the efficacy of our 'search-by-icon' method for users. Participants were shown a target song with its custom icon and the search-by-icon interface shown in Fig. B.10. They were tasked with using the interface to imitate the target icon and then retrieve the three songs most similar to the target one.

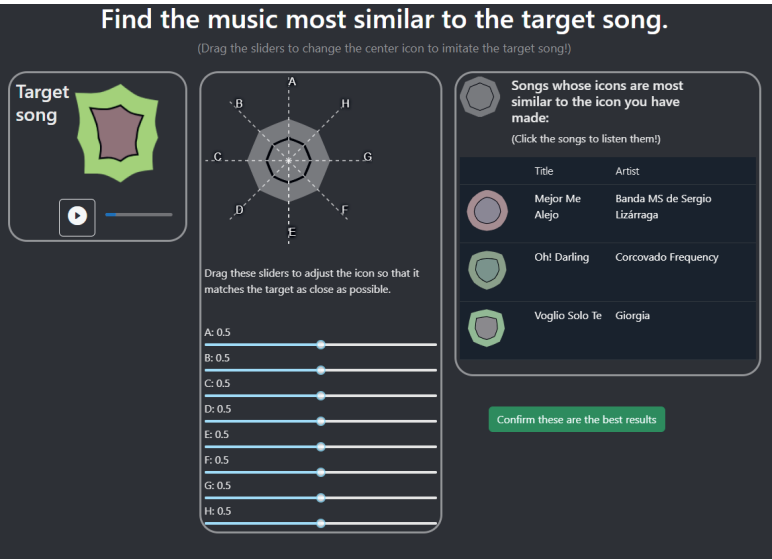


Figure B.10: An example screenshot for Test 4.

B.5.2. RESULTS AND DISCUSSION

We evaluate this novel method by evaluating the different sub-tasks we distinguished: how close the user can imitate an icon, how well the user can retrieve ‘similar’ music with this tool, and the willingness of users to adopt this tool with the System Usability Scale.

IMITATED ICON AND RETRIEVED SONGS

Cosine similarities between user-generated and target icon vectors, presented in Fig. B.11 (left), with a high mean (0.989) and median (0.993), indicates that with vectors exhibiting a cosine similarity above 0.975 to the target, most users accurately replicated icons. Furthermore, the average cosine similarities between the target icon vector and the top three selected songs, detailed in Fig. B.11 (right), reinforce the precision of these imitations, highlighting the effectiveness of participant selections in aligning closely with the target icons. The descriptive statistics of the results can be seen in Table B.5.

SYSTEM USABILITY SCALE (SUS)

The SUS consists of the ten questions:

- 1. I think I would like to use this product frequently.
- 2. I found it unnecessarily complicated.
- 3. I found the product easy to use.
- 4. I think I need technical support to use the product.
- 5. I found the different functions of the product well integrated with each other.
- 6. I felt there were too many contradictions in the product.
- 7. I can imagine that most people can quickly get to grips with the product.
- 8. I found the product cumbersome to use.
- 9. I felt confident while using the product.
- 10. I had to learn a lot about the product before I could use it properly.

Each of these statements was ranked with the Likert Scale anchored with one for ‘fully disagree’ and five for ‘fully agree’. The answers that were given in response to each of the questions in the SUS can be seen in Fig. B.12.

Task	Mean	Median	Std	Variance
Imitation	0.989	0.993	0.017	0.00003
Retrieval	0.991	0.995	0.017	0.00003

Table B.5: Descriptive statistics of the data from imitated icon task and retrieved songs task as displayed in Fig. B.11.

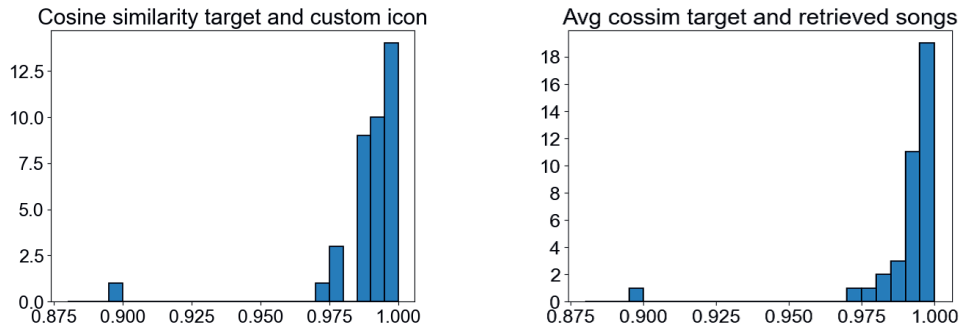


Figure B.11: Cosine similarities between user-generated and target icon vectors (left) and the average cosine similarities between the target icon vector and the top three selected songs (right).

Based on the feedback, we computed the SUS scores, as illustrated in Fig. B.13(left), with the corresponding performance interpretations presented in Fig. B.13 (right). We recognize that condensing the user experience into a singular score significantly simplifies the nuanced nature of their feedback. Nonetheless, we observe that a majority, specifically 25 out of 38 participants, demonstrates a willingness to embrace our model, despite the fact that it currently does not integrate the standard features derived from meta data, which we would not want to avoid in a final system but excluded to examine the effectiveness of our core contribution.

B.6. TEST FIVE: SEARCH-IN-PLAYLIST

B.6.1. TASK

This test aimed to assess the icon's effectiveness and sorting properties within a playlist context, comparing it against the prevalent use of album art in streaming services. Participants were asked to select their top three songs from playlists featuring both album art and our custom design, with each format presented twice. An example screenshot is shown in Fig. B.14. To prevent order effect, the target song was selected randomly from the selection of possible target songs, the playlist order was randomised for each participant, as was the order in which they were presented with custom icon and album art icons.

B.6.2. RESULTS AND DISCUSSION

We assessed the similarity between the top three selections and the target vector, time-on-task, plays per task, and additional insights from open-ended questions.

RETRIEVED SONGS

Average cosine similarities between the target icon vector and the top three selected songs are presented in Fig. B.15, with descriptive statistics in Table B.6. Both methods

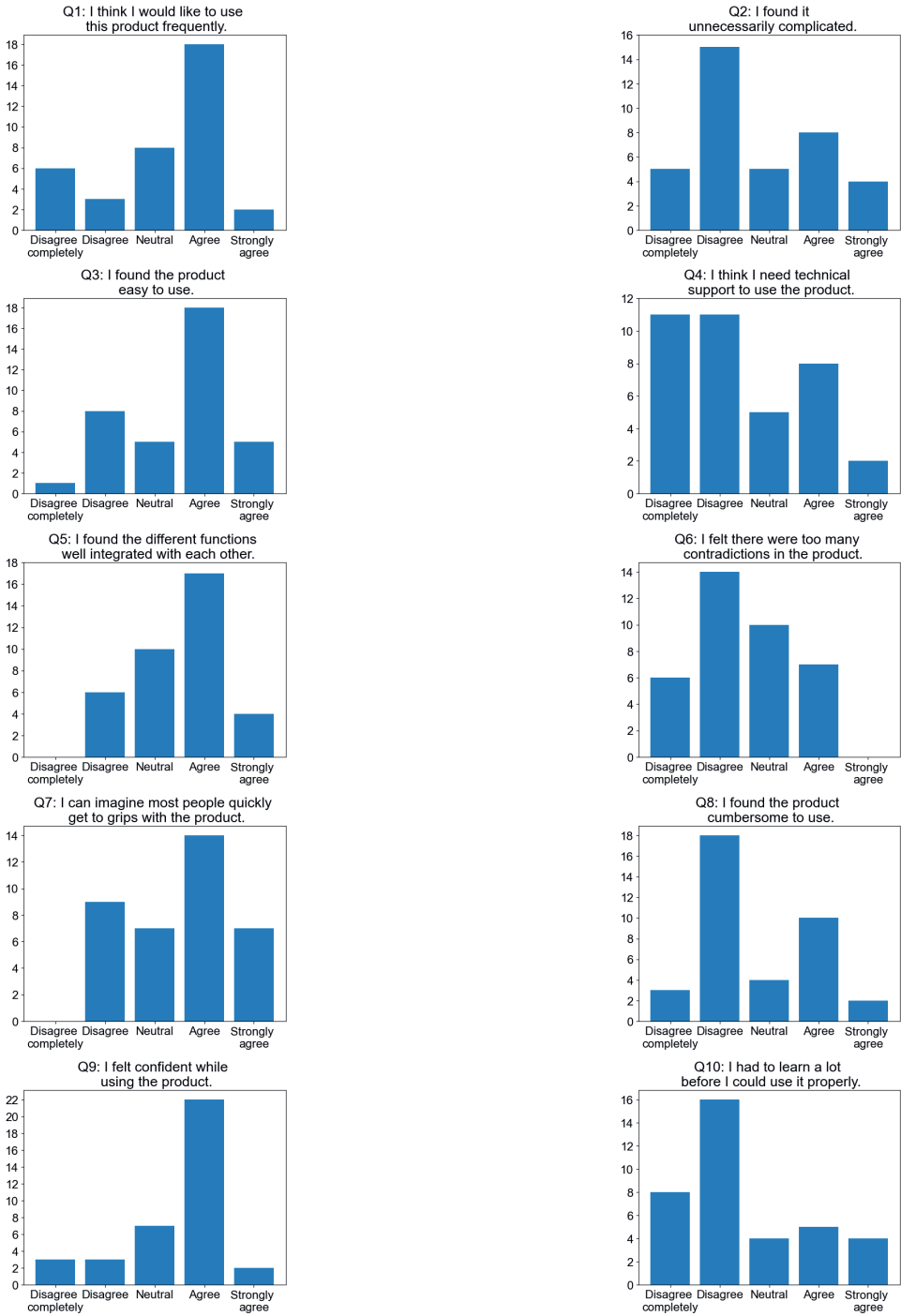


Figure B.12: Answers given to questions the SUS.

B

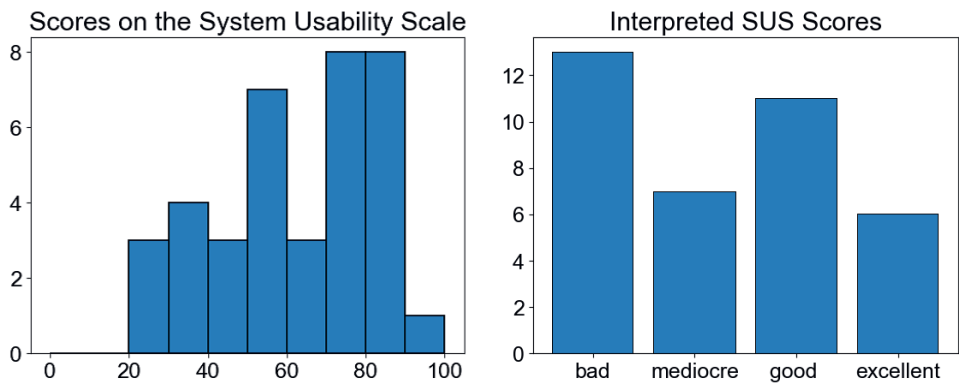


Figure B.13: SUS Scores (left) and its corresponding interpretation (right).

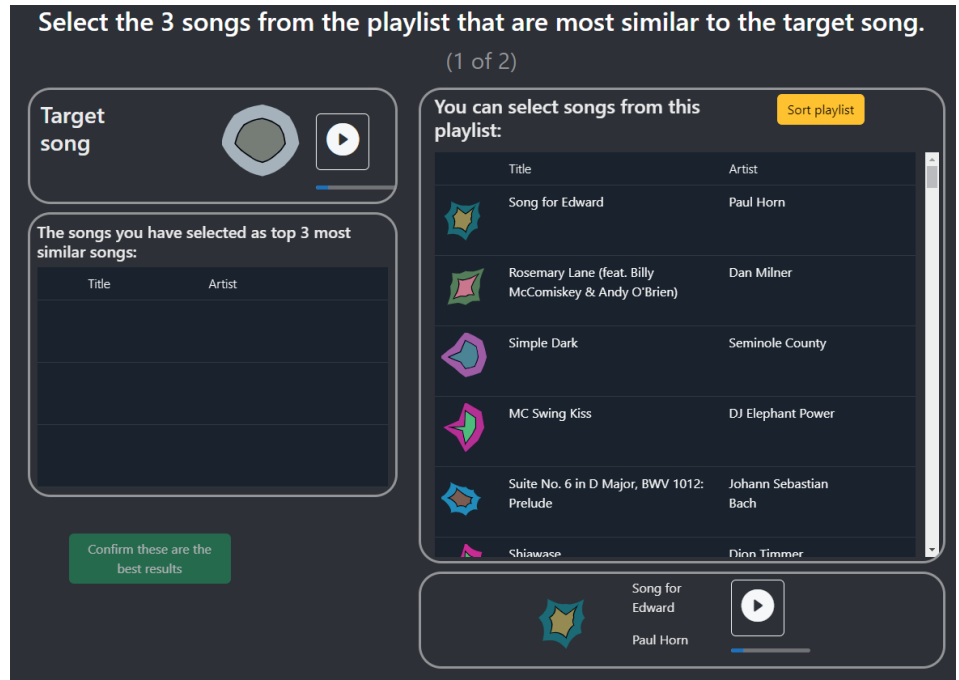


Figure B.14: An example screenshot for Test 5.

cover similar ranges of cosine similarities, but the icon method facilitates slightly higher similarity retrieval (one-tailed paired-samples t-test: $p = 0.03$, Cohen's $d: 0.469$), aligning with the icon's intended similarity representation.

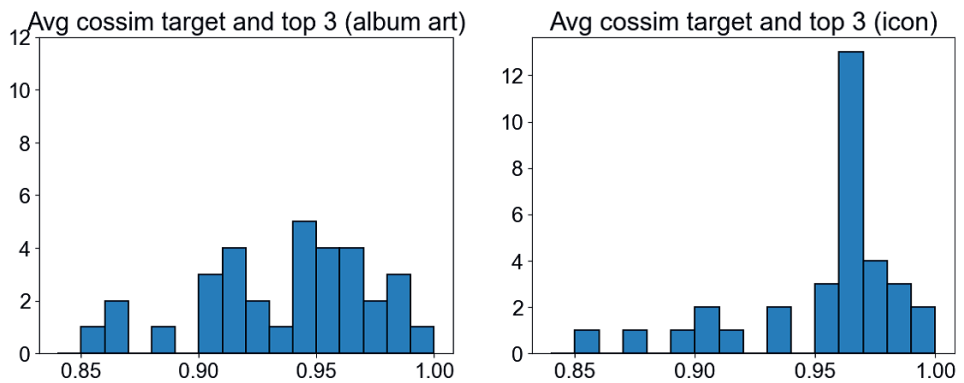


Figure B.15: Average cosine similarities between the target icon vector and the top three selected songs, comparing between album art (left) and our custom icon (right).

Icon	Mean	Median	Std	Var	Min	Max
Album	0.955	0.967	0.033	0.001	0.858	0.991
Custom	0.938	0.945	0.036	0.001	0.851	0.993

Table B.6: Descriptive statistics of the data as displayed in Fig. B.15.

TIME-ON-TASK

The time-on-task per participant for both album art and custom icon methods are detailed in Fig. B.16, with descriptive statistics provided in Table B.7. A significant reduction in average completion time, exceeding one minute, was observed. A left-tailed paired t-test confirmed these findings with $p = 0.00201$ and an effect size of 0.473 (Cohen's d).

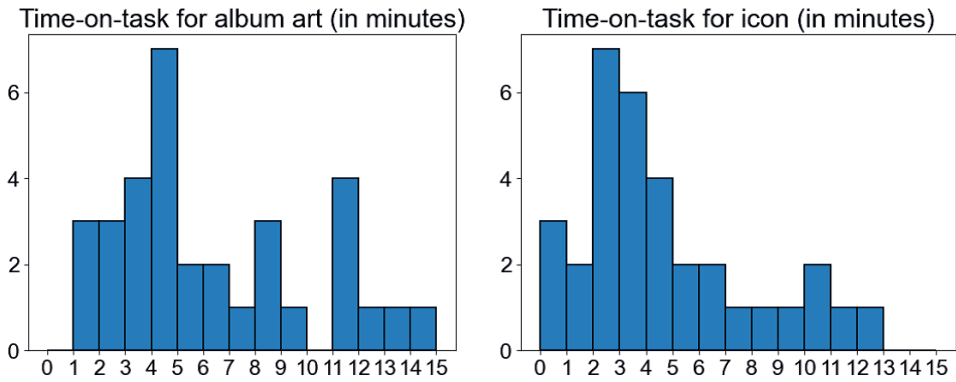


Figure B.16: Completion times of time-on-task with album art (left) and our custom icon (right).

Icon	Mean	Median	Std	Min	Max
Album	6:25	4:49	3:45	1:08	14:12
Custom	4:44	3:54	3:11	0:39	12:14

Table B.7: Descriptive statistics of the data as displayed in Fig. B.16, formatted as mm:ss.

SONGS PLAYED PER TASK

Fig. B.17 and Table B.8 display the number of songs played per task per participant for both album art and the custom icon, showing similar ranges but a notably lower mean and median for the custom icon. A paired t-test confirms this difference, with $p = 0.00001$ and an effect size of 0.931 (Cohen's d).

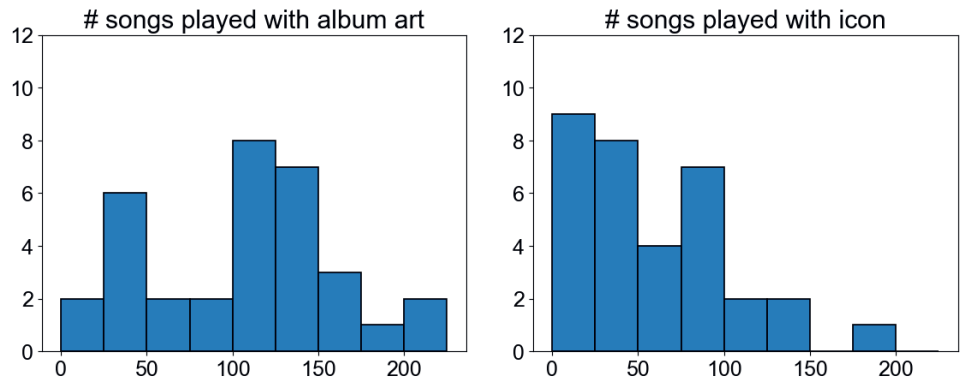


Figure B.17: The number of songs played per task per participant for both album art (left) and the custom icon (right).

Icon	Mean	Median	Std	Var	Min	Max
Album	103.9	113.5	55.9	3121.1	10	222
Custom	57.7	40.0	42.4	1796.8	7	192

Table B.8: Descriptive statistics of the data as displayed in Fig. B.17.

OPEN QUESTIONS

Upon study completion, participants responded to three open-ended questions regarding their playlist task experience. We summarise and highlight the responses here.

Q1: HOW DID YOU EXPERIENCE USING THE CUSTOM ICON DIFFER FROM THE REGULAR SETTING?

To this first, question 23 participants mentioned a positive experience. 14 mentioned explicitly that they experienced that it made their task of music selection easier. Three

participants mentioned that they considered the album art icon more fit for the task, of which two explicitly mentioned that they found the album art to give more information about the song than the custom icon.

Q2: WAS THERE ANYTHING SURPRISING OR UNEXPECTED?

To this second question, five participants mentioned that they were surprised how well the icon had supported their task. In contrast five participants mentioned they had encountered, what they considered outliers, which made them doubtful in how well the icon reflected the music content in some cases.

Q3: WHAT COULD BE DONE TO IMPROVE THE ICON?

To the third and final question we got some very concrete feedback from participants: four participants mentioned that they would like the icon to be more expressive, in terms of shape and colour. In particular, three mentioned the colours as a bit bland. In addition, 12 participants expressed a longing for a better understanding of the parameters of the icons: something more semantic or a more elaborate explanation, several of them mentioned a desire for genres mapped to axis or colours.

Overall, our tool's effectiveness was confirmed through strong quantitative results, notably speeding up task completion by over a minute compared to album art presentations and reducing the number of songs participants needed to listen to by almost 50%. When using our icon, selections tended to have similar or slightly higher cosine similarity to the target song, suggesting the icons' visual cues enhanced both the speed and quality of decision-making. While many participants valued our icon for its capacity to indicate similarity, three expressed a preference for album art due to its contextual and cultural insights. Acknowledging album art's value in certain situations, we argue that our icon significantly enhances user experience by addressing the variability of songs within an album.

ACKNOWLEDGEMENTS

Finally it comes to the end of my PhD research journey. I am deeply grateful to those who have supported, guided, and encouraged me throughout this challenging yet rewarding experience.

First and foremost, I would like to express my sincere gratitude to my supervisor, Prof. dr. E. Eisemann. Your unwavering support, insightful guidance, and constant encouragement have been invaluable and indispensable for me to be a better researcher and person. I always got so much inspiration from the stories you shared from your own experiences. Thanks for being a friendly and easy-going supervisor. Your expertise and support have not only shaped this thesis but also significantly contributed to my personal and professional growth. I am truly lucky to have had you as my supervisor.

I have always appreciated working with so many brilliant and kind collaborators, colleagues and friends in the department. Your supports have been invaluable in both my research and personal life. Adrien, your deep insights in the field impressed me. Your sharp comments on the manuscript and EG slide gave me important guidance. Thank you for all the help during the time we worked together. Leo, we collaborated in two chapters of my thesis. Thank you for always being available to offer insightful advice whenever I had questions, and thanks for helping me catch up with OpenGL programming. I learned a lot from you and truly enjoyed our collaborations. Nestor, thank you for supervising my master's thesis. Your valuable feedback made it into a better shape. Vera, your unique taste in art and the way you combine it with computer science is impressive. I am happy that you are working in a field you are passionate about and I admire it. Annemieke, we shared some unforgettable course experiences and many food events together. I really enjoyed our hallway snack talks. Thank you for inviting my family and me to your home and your wedding; I wish you and Peter a wonderful life together. Jerry, thank you for your useful tips and tricks, which helped me so much on my work and life in the Netherlands. Soumyadeep, your smiling face and cheerful morning greetings were very encouraging. Thank you for helping me reset my PC so many times. A big thank you to Mark for his invaluable assistance in proofreading the Dutch summary. I would also like to thank the nice and friendly people in the group: Ricardo, Anna, Rafael, Klaus, Thomas, Michael, Petr, Martin, Chang, Ali, Baran, Ruben, Mathijs, Guowei, Nicolas, Faizan, Amir, Lukas, Yang, Alex, Nasikun. It has been a pleasure getting to know and work with you all. I will never forget the wonderful memories we made together.

Special thanks to Ruud for all your swift supports in IT. I really appreciate that you helped me deliver the screen to my apartment. To the administrative staff, particularly Marloes and Lauretta, thank you for your invaluable help with the countless trivial tasks and for always being there to ensure everything ran smoothly.

Finally, I am deeply grateful to my family for their constant love and support. To my parents, your belief in me and your care have been my source of strength. To my brothers

and sister, your support and love have helped me go forward without worries. To my daughter, Xuran, you are the best thing in the world. Although it was hard at the beginning when you were born, we made it together. Thank you for your purest love. You bring so much happiness into my lives. Dear Chaoping, thanks for always being with me in so many years, your unwavering support, patience, and love have been my rock throughout this journey.

This thesis would not have been possible without the support and contributions of everyone of you. Thank you all from the bottom of my heart.

CURRICULUM VITÆ

Xuejiao LUO

Xuejiao Luo was born in Guangdong, China.

She joined the Computer Graphics and Visualization group as a PhD student under the supervision of Prof. Elmar Eisemann in August 2018. Her doctoral research focused on the Reconstruction, Generation, and Exploration of Digital Content with Deep Neural Features, which is presented in this dissertation. Prior to her move to the Netherlands, she completed her M.Sc. in Multimedia Networking at Télécom ParisTech, France, in February 2018, with a master thesis on Adding Motion Blur to Still Images. She also holds a B.Eng. in Electronic Science and Technology from South China University of Technology, China, which she earned in July 2012. Between 2012 and 2016, she worked at China Mobile as a machine learning scientist, developing AI-based models for fraud detection and prevention.

LIST OF PUBLICATIONS

1. **X. Luo**, N.Z. Salamon, E. Eisemann, “Adding Motion Blur to Still Images”. *Graphics Interface*, pp. 108-114, 2018.
2. **X. Luo**, N.Z. Salamon, E. Eisemann, “Controllable motion-blur effects in still images”. *IEEE Transactions on Visualization and Computer Graphics*, Vol. 26, No. 7, pp. 2362-2372, 2020.
3. **X. Luo**, L. Scandolo, E. Eisemann, “Texture Browser: Feature-based Texture Exploration”. *Computer Graphics Forum*, Vol. 40, No. 3, pp. 99-109, 2021.
4. **X. Luo**, L. Scandolo, A. Bousseau, E. Eisemann, “Single-Image SVBRDF Estimation with Learned Gradient Descent”. *Computer Graphics Forum*, Vol. 43, No. 2, p. e15018, 2024.
5. **X. Luo**, V. Hoveling, E. Eisemann, “Musicon: Glyph-Based Design for Music Visualization and Retrieval”. *Vision, Modeling, and Visualization*, 2024.
6. **X. Luo**, E. Eisemann, “Perceptually Consistent Interpolation using MaterialGAN”. *In preparation*, 2024.

