Visual Analytics for High-Dimensional Images via Dimensionality Reduction

Vieth, A.

DOI
10.4233/uuid:0fdde46e-23cc-4984-b03b-7710a03e2b46

Publication date
2026

Document Version
Final published version

Citation (APA)
Vieth, A. (2026). *Visual Analytics for High-Dimensional Images via Dimensionality Reduction*. [Dissertation (TU Delft), Delft University of Technology]. https://doi.org/10.4233/uuid:0fdde46e-23cc-4984-b03b-7710a03e2b46

VISUAL ANALYTICS

FOR HIGH-DIMENSIONAL IMAGES VIA DIMENSIONALITY REDUCTION

Alexander Vieth

Analyzing high-dimensional images is a complex task. Unlike regular color images, they are not straightforward to visualize. Their additional information content increases the complexity of interpretation, both in terms of computational processing and human comprehension. Visual analytics — the combination of visualization, interaction, and automated analysis methods — has proven useful to gain insights into such large, difficult-to-handle data.

In this thesis, we investigate non-linear dimensionality-reduction methods for the exploration of high-dimensional images. Specifically, we address the problem that current dimensionality-reduction methods are image-agnostic: they treat spatially resolved data without considering their spatial layout. We present algorithmic solutions that yield image-informed embeddings, and interactions techniques that connect images and embedding representations. Further, we present an open-source visual analytics software framework for rapid prototyping and extensible workflow.

Visual Analytics for High-Dimensional Images via Dimensionality Reduction

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
Monday, 12 January 2026 at 17:30 o'clock

by

Alexander VIETH

Master of Science in
Electrical Engineering, Information Technology and Computer Engineering,
RWTH Aachen University, Germany
born in Münster, Germany.

This dissertation has been approved by the promotors.

Composition of the doctoral committee:

Rector Magnificus,	chairperson
Prof. dr. A. Vilanova,	Delft & Eindhoven University of Technology, *promotor*
Prof. dr. ir. B.P.F. Lelieveldt,	Leiden University Medical Center & Delft University of Technology, *promotor*
Prof. dr. E. Eisemann,	Delft University of Technology, *promotor*
Asst. Prof. dr. T. Höllt,	Delft University of Technology, *copromotor*
Independent members:	
Prof. dr. M.J.T. Reinders,	Delft University of Technology
Assoc. Prof. dr. N. Gehlenborg,	Harvard Medical School, USA
Assoc. Prof. dr. F.V. Paulovich,	Eindhoven University of Technology
Assoc. Prof. dr. R. Raidou,	TU Wien, Austria

Keywords: Visual Analytics, High-Dimensional Images, Dimensionality Reduction, Software Framework
Printed by: Ridderprint
Cover by: Linda Hoogendam

An electronic version of this dissertation is available at https://repository.tudelft.nl/.
This document was typeset with the help of LaTeX using the kaobook class.

The act of writing
Brings frustration, insight, joy
Right in that order

– based on *academic haiku* by Sally Wyatt
from *Letters in Lockdown*

Summary

Analyzing high-dimensional images is a complex task. Unlike regular color images, they are not straightforward to visualize. Their additional information content increases the complexity of interpretation, both in terms of computational processing and human comprehension. Visual analytics — the combination of visualization, interaction, and automated analysis methods — has proven useful to gain insights into such large, difficult-to-handle data. For example, non-linear dimensionality reduction is a commonly employed technique in visual analytics for exposing interesting patterns through lower-dimensional representations of high-dimensional data.

In this thesis, we investigate non-linear dimensionality-reduction methods for the exploration of high-dimensional images. Specifically, we address the problem that current dimensionality-reduction methods are image-agnostic: they treat spatially resolved data without considering their spatial layout. We present algorithmic solutions that yield image-informed embeddings and interactions techniques that connect images and embedding representations. To a large extend, we utilize hierarchical approaches to handle the image data. We show how these techniques enable more insightful exploration of high-dimensional images.

Further, we present an open-source visual analytics software framework for rapid prototyping and extensible workflow development for high-dimensional data analysis. All algorithms and techniques described in this thesis are made available in or fully implemented as plugins for this framework.

Samenvatting

Het analyseren van hoogdimensionale beelden is een complexe taak. In tegenstelling tot gewone kleurafbeeldingen zijn ze moeilijker te visualiseren. Hun hogere informatiedichtheid verhoogt de complexiteit van de interpretatie, zowel wat betreft de computationele verwerking als het menselijk begripsvermogen. Visual analytics — de combinatie van visualisatie, interactie en geautomatiseerde analysemethoden — is nuttig gebleken om inzicht te verkrijgen in dergelijke grote, moeilijk te verwerken data. Niet-lineaire dimensiereductie is bijvoorbeeld een veelgebruikte techniek in visual analytics om interessante patronen bloot te leggen met behulp van lagerdimensionale representaties van hoogdimensionale gegevens.

In dit proefschrift onderzoeken we niet-lineaire dimensiereductiemethoden voor het verkennen van hoogdimensionale beelden. We richten ons met name op het probleem dat de huidige dimensiereductiemethoden beeldagnostisch zijn: ze behandelen ruimtelijk gestructureerde gegevens zonder rekening te houden met hun ruimtelijke structuur. We presenteren algoritmische oplossingen die beeldgestuurde inbeddingen opleveren en interactietechnieken die beelden en inbeddingsrepresentaties met elkaar verbinden. We maken grotendeels gebruik van hiërarchische benaderingen om de beeldgegevens te verwerken. We laten zien hoe deze technieken een meer inzichtelijke verkenning van hoogdimensionale beelden mogelijk maken.

Verder presenteren we een open-source softwareframework voor visual analytics, bedoeld voor snelle prototyping en uitbreidbare workflowontwikkeling bij de analyse van hoogdimensionale gegevens. Alle algoritmen en technieken die in dit proefschrift worden beschreven, zijn beschikbaar binnen of volledig geïmplementeerd als plugins voor dit framework.

Contents

List of Figures and Tables

Figures

Tables

Lists of Symbols and Abbreviations

Symbols

Definitions are provided at first use in the text, and listed here for reference. Unless otherwise specified, when a letter is reused, a **bold** style indicates vectors while regular style indicates scalars.

$\mathbf{a}$	High-dimensional attribute vector
$\mathbf{b}$	Low-dimensional projection vector
c	Specific attribute dimension/channel
C	Number of attribute dimensions
E	Number of embedding dimensions
$\mathscr{C}$	Context indices
d	Standard distance function (attribute space)
d_s	Spatial distance function
$\mathscr{D}_E$	Embedding view data indices
$\mathscr{D}_I$	Image view data indices
e_{ij}	An edge between vertices i and j
$\mathscr{E}$	Edges
f	High-dimensional image function
$\mathscr{F}$	Focus indices
$\mathscr{G}$	Graph
$\boldsymbol{h}$	Histogram feature (vector, bold font)
h	Histogram feature (scalar)
i	Data index (flattened)
$\mathbf{i}$	Image position (x and y coordinates)
I	Area of influence
$\widehat{I}$	Normalized area of influence
$\mathscr{I}$	Image Graph
$\mathscr{K}$	Selection indices on a level in image
$\mathscr{K}_f$	Selection focus indices in image
L_i^k	Landmark i on level k
$\mathscr{L}$	Hierarchy levels
m	Number of hierarchy levels
n	Total image/data size
$n_{\mathcal{N}}$	Total size of neighborhood
$\mathcal{N}^{C,k}$	Neighborhood in attribute space, set of k indices, abbreviated as $\mathcal{N}$
$\mathcal{N}^{S,\eta}$	Neighborhood in image space, set of η indices
$\mathbb{Q}$	Superpixel segmentation, ground truth
q	Component in ground truth segmentation
s	Number of superpixels
$\mathcal{s}$	Individual superpixel
$\mathcal{S}$	Superpixel segmentation, set of superpixels
T	Transition probability matrix
v_b	Visual budget
v_t	Visual target
v_i	A specific vertex
$\mathscr{V}$	Vertices
$\mathbf{w}$	Spatial neighborhood weights vector
w	Spatial neighborhood weights scalar
x	Image position x
X	Image domain x direction/set
y	Image position y
Y	Image domain y direction
$\mathscr{X}_f$	Selection focus indices in embedding
$\boldsymbol{Z}$	Feature matrix
$\boldsymbol{z}$	Feature vector
$\mathscr{F}_\eta$	Feature extraction function
$\gamma^{\uparrow}$	Threshold level upwards
$\gamma^{\downarrow}$	Threshold level downwards
η	Size of spatial neighborhood

λ	Length of a random walk	σ	Covariance
μ	Mean vector (bold font)	$\mathbf{\Sigma}$	Covariance matrix
μ	Mean scalar	ω	Number of random walks

Abbreviations

AE autoenconder. 7
API application programming interface. 4

CyCIF cyclic immunofluorescence. 29, 49, 50

DM diffusion maps. 7
DR dimensionality reduction. 2, 6–8, 15, 29, 39

EV explained variation. 51

GUI graphical user interface. 56

IMC imaging mass cytometry. 25, 26
IS isomap. 7

*k***NN** *k*-nearest neighbor. 8, 42

LoD level of detail. 39

MDS metric multidimensional scaling. 7, 8, 11, 13, 15

PCA principal component analysis. 2, 6
PCP parallel coordinates plot. 5

QF quadratic form. 19, 22, 23

ROI region of interest. 29, 39

SPLOM scatterplot matrix. 5

t-SNE t-distributed Stochastic Neighbor Embedding. 2, 7, 8, 15

UE undersegmentation error. 51
UMAP Uniform Manifold Approximation and Projection. 2, 7, 8, 15

VA Visual Analytics. 2, 55, 56, 59

1. Introduction

A molecular biologist assesses highly multiplexed images of tumor tissues to classify cancerous cells which potentially leads to advanced treatment plans. An agricultural planner inspects hyperspectral images of crops for large scale monitoring of irrigation and pesticide dissemination systems. An art conservator examines scans of old paintings to document their condition and possibly restore them to closer versions of their original state. An astrophysicist studies near-infrared images of the night sky, aiming to improve our understanding of the evolution of the universe.

Each of these examples are set in widely varying applications, yet in all of them a domain expert is working with a specific type of data: high-dimensional images. These types of images do not just contain three color channels (red, green, and blue) like a common photograph does; they generalize the typical three-dimensional color feature of each pixel to any number of numerical attributes — from the intensity of reflected light in multiple spectral ranges to the relative abundance of specific proteins. High-dimensional images typically have dozens or hundreds of such channels. This increased amount of information goes along with increased difficulties to handle the data — both computationally and for human understanding.

Domain experts regularly engage with data of unknown content. A novel acquisition device produces new types of images, an object or specimen of interest is recorded for the first time, or the image captures a yet unrecorded event. After acquiring new images, each above domain expert seeks to understand their content. The data analysis begins with exploration. Among the main goals during this stage are the identification of general characteristics, novel patterns and the generation of hypotheses for their underlying causes. Statistical methods are powerful tools for validating theoretical models and can also help uncover patterns, but often do not suffice in gaining insight into unknown data. Merely plotting data can reveal patterns in the data that many summary statistics fail to capture, e.g., a dinosaur emerging from a large collection of observations, as in Figure 1.1. But visualizing high-dimensional images effectively is not straight-forward at all.

"Exploratory data analysis is detective work – numerical detective work, counting detective work or graphical detective work."
John W. Tukey in *Exploratory Data Analysis*, page 1 [1]

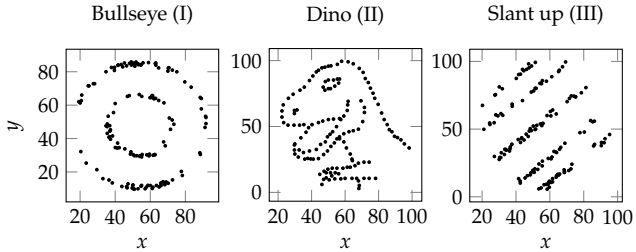
1.1. Visual Analytics

Processing large amounts of data effectively is challenging for both humans and computers. Especially when aiming to obtain actionable insights from novel data, neither purely algorithmic approaches nor human interpretation of static visualizations suffice:

> The credo of [...] ***visual analytics*** is therefore to use a combination of visualization, interaction, and automated analysis methods to explore huge amounts of data and to obtain solid insights.

From the foreword by Jarke van Wijk to *Interactive Visual Data Analysis* [2]

How does that apply to our situation? High-dimensional images qualify as large data twofold, both in terms of their number of data points, as images typically contain hundreds of thousands or millions of pixels, and with regard to the number of attribute dimensions attached to each pixel. With grayscale or RGB color images, all information can be mapped to the three RGB channels of modern displays, allowing for immediate visualization and human interpretation of all their content. In contrast, the many channels of high-dimensional images cannot be displayed simultaneously in the same manner. Indeed, we face multiple difficulties: Which (combination of) image channel(s) best reveals underlying structures? What are effective strategies for mapping high-dimensional pixel attributes onto the two-dimensional image space? And, in general, how can



Summary statistics

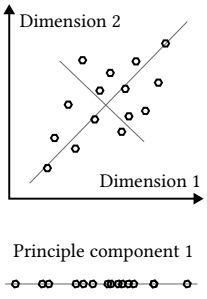
	μ_x	μ_y	σ_x	σ_y
I	54.27	47.83	16.77	26.94
II	54.27	47.83	16.77	26.94
III	54.27	47.83	16.77	26.94

μ: mean, σ: standard deviation

Figure 1.1. *Three datasets from the Datasaurus dozen collection [6], a recent take on Anscombe's Quartet [7]: even though each data set is characterized by the same summary statistics (up to several digits of precision), the differences in distribution show immediately when visualized.*

patterns in the high-dimensional attribute space be related to their spatial embedding in the image domain?

There is seldomly a single representation of complex data that shows all interesting aspects simultaneously. One major aim of Visual Analytics (VA) is to facilitate data exploration by coupling automated analysis with multiple interactive visual representations, steering users towards potentially interesting views on or parts of their data. For high-dimensional images this could manifest in several coordinated views [3] on the data. For example, one view focusing on the spatial layout, displaying a false color image of a user-selectable set of channels. Another view solely concerned with the data attributes, e.g. a parallel-coordinates plot [4] that automatically applies edge bundling [5] to highlight clusters. And finally, a plot of a derived data representation that distills specific data properties. In fact, lower-dimensional representations of high-dimensional data have proven remarkably successful at displaying interesting patterns.



1.2. Dimensionality Reduction

Embedding points from a high-dimensional space into a lower dimensional space aims to preserve intrinsic structure of the data while enabling interpretable visualization and reducing computational complexity, e.g., to facilitate clustering. Both linear and non-linear projections techniques have been well explored for these purposes [8].

While the mathematical foundation of these dimensionality reduction (DR) techniques vary, they generally assume that the data lies in a subspace of the original high-dimensional space. For example, principal component analysis (PCA) keeps the linear subspace with the largest variance. In recent years, non-linear, neighborhood-based techniques like t-distributed Stochastic Neighbor Embedding (t-SNE) and Uniform Manifold Approximation and Projection (UMAP) have become widely adapted methods for exploring and presenting data structure.

Applying dimensionality reduction methods to high-dimensional images remains challenging. Figure 1.2 illustrates the correspondence of embedded data points and image pixels. The large number of pixels, i.e., the data points, leads to dense and cluttered embeddings, and ignores the image layout. Views on the data via channel-wise image representations and static embeddings separates the exploration of image layout and attribute space. Fundamentally, most dimensionality reduction methods are not designed to take both the image layout and attribute space into account, rendering them unable to capture important image structure.

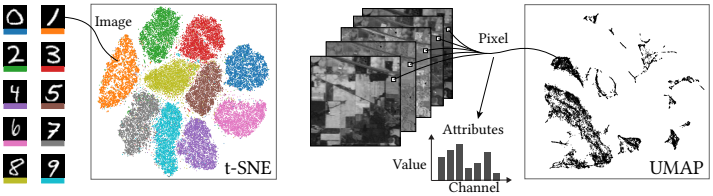


Figure 1.2. *Embeddings of (left) a collection of images, MNIST [9], and (right) a single high-dimensional images, the Indian Pines data [10]. The embedding points represent whole images and individual pixels respectively. In this thesis, we focus on the latter setting.*

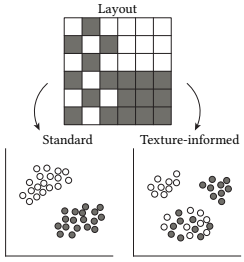
1.3. Contribution and Outline

Two decades ago, *Illuminating the Path* [11] set forth an ambitious research agenda for visual analytics that remains largely applicable today. Among their goals, several stand out as particularly pertinent to our discussion of high-dimensional images:

> Facilitate the understanding of massive and continually growing data; facilitate knowledge discovery through information synthesis, which is the integration of data based on their meaning rather than the original data type; support multiple levels of data and information abstraction; provide frameworks for analysis of spatial data.

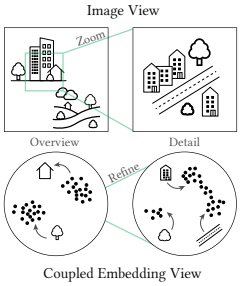
Excerpt from the recommended visual paradigms that support analytical reasoning in *Illuminating the Path*, page 8 [11]

This thesis presents three methods and a novel visual analytics framework which contribute to making progress toward these goals with a particular focus on exploring high-dimensional images with dimensionality reduction techniques.
We present algorithms that yield image-informed embeddings and interaction techniques connecting images with their embedding representations. Each thesis chapter addresses different aspects of the relationship between the attribute space and pixel layout of high-dimensional images: texture integration in DR, interaction paradigms, scalability in the spatial domain, and analysis software architecture. Specifically, this thesis presents a method to incorporate spatial neighborhood information into dimensionality reduction techniques (Chapter 4), a method to couple the exploration of high-dimensional images and hierarchical embeddings (Chapter 5), a method to compute manifold-preserving superpixel hierarchies of high-dimensional images (Chapter 6), and finally, a flexible and extensible visual analytics software framework for high-dimensional data (Chapter 7).

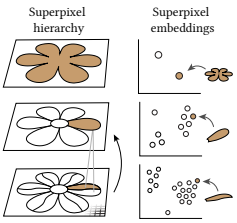


Chapter 4 Visual exploration of high-dimensional data is commonly facilitated by dimensionality reduction. However, common dimensionality reduction methods do not include spatial information present in images, such as local texture features, into the construction of low-dimensional embeddings. Consequently, exploration of such data is typically split into a step focusing on the attribute space followed by a step focusing on spatial information, or vice versa. In this chapter, we present a method for incorporating spatial neighborhood information into distance-based dimensionality reduction methods, such as t-SNE. We achieve this by modifying the distance measure between high-dimensional attribute vectors associated with each pixel such that it takes the pixel's spatial neighborhood into account. Based on a classification of different methods for comparing image patches, we explore a number of different approaches. We compare these approaches from a theoretical and experimental point of view. Finally, we illustrate the value of the proposed methods by qualitative and quantitative evaluation on synthetic data and two real-world use cases.

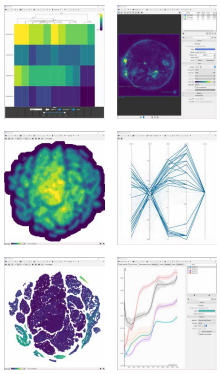
Chapter 5 The spatial and attribute information of high-dimensional images are typically explored separately, e.g., by using coordinated views of an image representation and a low-dimensional embedding of the high-dimensional attribute data. Facing ever-growing image data sets, hierarchical dimensionality



reduction techniques lend themselves to overcome scalability issues. However, current embedding methods do not provide suitable interactions to reflect image space exploration. Specifically, it is not possible to adjust the level of detail in the embedding hierarchy to reflect changing level of detail in image space stemming from navigation such as zooming and panning. In this chapter, we propose such a mapping from image navigation interactions to embedding space adjustments. We show how our mapping applies the "overview first, details-on-demand" characteristic inherent to image exploration in the high-dimensional attribute space. We compare our strategy with regular hierarchical embedding technique interactions and demonstrate the advantages of linking image and embedding interactions through a representative use case.



Chapter 6 The above linking strategy between image space interactions and the level-of-detail shown in hierarchical embedding views facilitates intuitive exploration. However, available data hierarchies for hierarchical dimensionality reduction methods ignore the spatial layout of pixels in the images. This impedes the exploration of regions of interest in the image space, since there is no congruence between a region of interest in image space and an associated embedding of the high-dimensional attributes. In this chapter, we present a superpixel hierarchy for high-dimensional images. In contrast to classical superpixel methods, our agglomerative superpixels construction takes the high-dimensional attribute manifold into account. Based on the same considerations, we show how to use this hierarchy for hierarchical dimensionality reduction, facilitating an "overview first, details-on-demand" exploration of high-dimensional images. Finally, we compare our image-guided hierarchy with classical hierarchical embedding-based image exploration in two use cases.



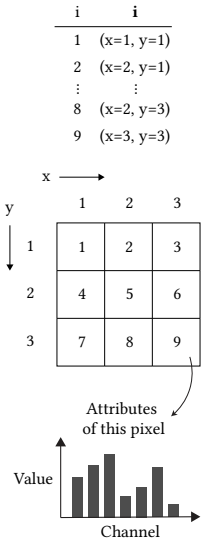
Chapter 7 Commonly, tailor-made visual analytics software is developed for a given exploration and analysis setting. This limits their applicability in other scenarios or fields. However, as diverse as these settings are, their characteristics and requirements for data analysis are conceptually similar. Many applications share abstract tasks and data types and are often constructed with similar building blocks. Developing such applications, even when based mostly on existing building blocks, requires significant engineering efforts. This chapter presents ManiVault, a flexible and extensible open-source visual analytics framework for analyzing high-dimensional data. The primary objective of ManiVault is to facilitate rapid prototyping of visual analytics workflows for visualization software developers and practitioners alike. ManiVault is built using a plugin-based architecture that offers easy extensibility. While our architecture deliberately keeps plugins self-contained, to guarantee maximum flexibility and re-usability, we have designed and implemented a messaging application programming interface (API) for tight integration and linking of modules to support common visual analytics design patterns. ManiVault provides several visualization and analytics plugins, and ManiVault's API makes the integration of new plugins easy for developers. ManiVault facilitates the distribution of visualization and analysis pipelines and results for practitioners through saving and reproducing complete application states. As such, ManiVault can be used as a communication tool among researchers to discuss workflows and results.

2. Background

This chapter introduces the key concepts underlying the methods presented later in this thesis. We will define high-dimensional image data and spatial neighborhoods more rigorously in Sections 2.1 and 2.3 respectively. Then, in Sections 2.2 and 2.4, we briefly review the field of dimensionality reduction methods and introduce t-SNE in more detail, as an representative example of non-linear projection methods.

2.1. High-dimensional Image Data

While, in general, dimensionality reduction methods can be applied on other high-dimensional datasets as well, we focus on high-dimensional, digital 2D images. Each pixel is an image element at a unique location with an associated attribute vector. We can formalize such a **high-dimensional image** as a discrete function $f : X \times Y \rightarrow \mathbb{R}^C$ from the spatial domain $X \times Y \subset \mathbb{N}^2$ to the attribute range $\mathbb{R}^C$. X and Y are the sets of pixel coordinates that span the image domain along its two dimensions while C is the number of attributes, or image channels. The i-th pixel is indexed with the tuple $\mathbf{i} = (x, y)_i$ where $1 \leq i \leq n$ and the number of pixels $n = |X \times Y|$. For easy of notation, we will refer to the x-coordinate of the i-th pixel as $x_\mathbf{i}$, and handle the y-coordinate likewise. Now, $f(\mathbf{i}) = \mathbf{a_i}$ yields the pixel's high-dimensional attribute vector at location $\mathbf{i}$ with $\mathbf{a_i} = [a_{\mathbf{i}1}, \ldots, a_{\mathbf{i}C}]$. In this work we focus on 2D images, but the above definitions are straightforward to extend to 3D images consisting of voxels.

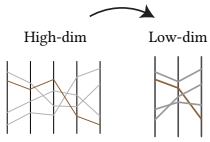


We can also interpret such a high-dimensional image as a combination of several scalar images, each representing one channel of the high-dimensional image. Therefore, we refer to the c-th channel of the image with f_c, where $1 \leq c \leq C$, such that $f_c = [a_{\mathbf{1}c}, \ldots, a_{\mathbf{n}c}]$ denotes the values of the c-th channel for all pixels in the image.

In chapters 4 and 6, it will be beneficial to interpret images as four- or eight-connected graphs. The spatial image domain $X \times Y$, a rectangular grid, then is a graph $\mathcal{I} = (\mathcal{V}, \mathcal{E})$ where each vertex $v_\mathbf{i} \in \mathcal{V}$ is an image pixel with edges $e_{\mathbf{ij}} \in \mathcal{E}$ between neighboring pixels. The image remains a discrete function, we merely conceptualize it as a mapping $f : \mathcal{V} \rightarrow \mathbb{R}^C$ from the spatial domain of the image $\mathcal{I}$ to the attribute space $\mathbb{R}^C$. In a four-connected graph, all pixels sharing an edge are neighbors. Section 2.3 provides an generalized definition for square neighborhood, e.g., eight-connected graphs.

2.2. Dimensionality Reduction Methods

High-dimensional data can be difficult to visualize and analyze, particularly when datasets contain not only many dimensions (hundreds or thousands) but also numerous observations (millions). In the first instance, visualizing the entire breadth of the data, e.g., using tables-based visualizations like heatmaps [12], scatterplot matrixs (SPLOMs) [13] or parallel coordinates plots (PCPs) [4] faces significant scalability challenges due to cluttering, screen space constraints and increased cognitive load. In the second instance, analyzing data in high-dimensional spaces is burdensome due to high computational costs and potentially limited results, resulting from the curse of dimensionality. This *curse of dimensionality* is a lose umbrella term for various undesired characteristics of high-dimensional data which exacerbate their handling. As the dimensionality of the data space increases, the data tends to populate this space more and more sparsely, leading to counterintuitive behaviors such as norm concentration, where distances between data points become increasingly uniform [14]. In practice, many high-dimensional datasets have a lower intrinsic dimensionality than the number of dimensions of the data space itself. I.e., it is possible to



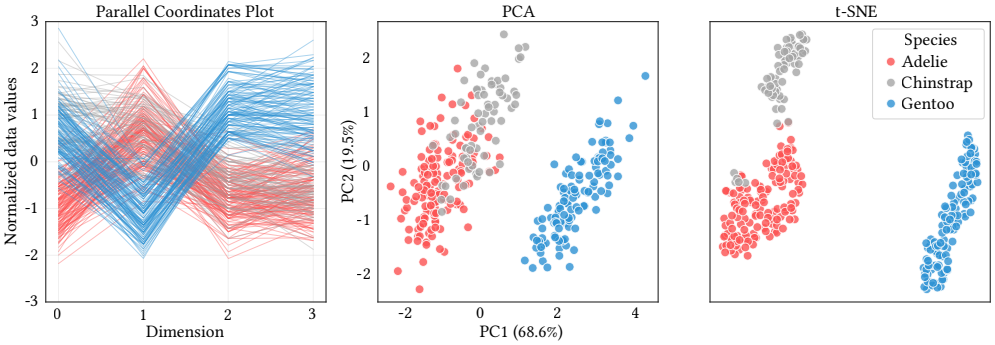


Figure 2.1. *The Palmer penguins data [15] describes four physical attributes of three penguin species, shown in a PCP (left). The dimension 1-4 correspond to bill length, bill depth, flipper length and body mass measurements. The first two principal components (center) explain* 88.1% *of the total variance of the data, yet the Adelie and Chinstrap species and are not well-distinguishable in the PCA plot. The t-SNE embedding (right) clearly reveals the three clusters in the data, (largely) corresponding to the three penguin species. The plot intentionally does not show axis descriptors or scale indication, as neither the axis orientation nor the absolute coordinates of the embedded points provide insight into the data.*

represent the data with minimal loss of information in a lower-dimensional space. The lower-dimensional representation — the embedding — can then be used for downstream analysis and visualization.

To obtain a lower-dimensional representation of the data, we can either focus on a subset of dimensions or project the high-dimensional data faithfully into fewer dimensions. Since selecting a subset of dimensions rarely suffices to preserve the essential structure of complex data, we are primarily interested in projection methods which are also capable of capturing structure that isn't aligned with the original feature axes. Such a dimensionality reduction (DR) method now maps a high-dimensional data point $\mathbf{a} \in \mathbb{R}^C$ to a low-dimensional embedding point $\mathbf{b} \in \mathbb{R}^E$ with typically $E \ll C$:

$$DR : \mathbb{R}^C \to \mathbb{R}^E; \quad DR(\mathbf{a}) = \mathbf{b} \tag{2.1}$$

The map DR aims to preserve the intrinsic structure of the data. In general, there are various ways of quantifying the quality of this structure preservation. Typically though, they compare the distances between the high-dimensional points and corresponding distances between the respective embedding points. However, any projection to lower dimensionality incurs some loss of information, i.e., the distribution of points in the embedding does not perfectly represent the original data. Different priorities regarding which aspects of data structure to preserve, and other trade-offs with respect to, e.g., computational speed and memory consumption, have motivated the development of numerous DR methods. There has been plenty of writing about these methods in both literature reviews [8, 16–18] and topical books [19–21]; here, we will focus on a selection of pertinent methods.

Projection traits:

1. Linearity
2. Input type (point vs. distance)
3. Neighborhood (local vs. global)
4. Ease of use
5. Computational complexity
6. Out-of-sample
7. Inverse transform
8. Determinism

See [17] for a more comprehensive description of these traits.

DR methods are commonly distinguished based on several traits. Among the most important of these characteristics are their linearity, input type and neighborhood scope. Whereas linear projections apply the same transformation across the entire data space, non-linear DR maps can behave differently in different data space regions. Next, some projections methods take high-dimensional points as their input, i.e. attribute vectors $\mathbf{a}$, others work with distances or dissimilarities between points. Any method working with distances can naturally also handle points by computing distances between them. Finally, a global DR map focuses on preserving all point-to-point distances whereas a local DR map only aims to preserve distance between points that are closely neighboring each other.

Principal component analysis (PCA) [22] is one of the most widely-used DR methods. PCA performs a linear, global transformation: it projects the data into a new coordinate system, such that the first new coordinate explains the greatest variance in the data and following coordinates explain the next-greatest

variance in descending order. The first two such principal coordinates are used for visualizing data as in Figure 2.1. Often, the first n principal coordinates that explain a large percentage (e.g. 90 – 95%) of the total variance of the data are used as a pre-processing step for denoising.

Metric multidimensional scaling (MDS) [23] takes a different approach by attempting to preserve pairwise distances between data points. Multidimensional scaling (MDS) is a nonlinear, global method that iteratively updates a low-dimensional embedding by minimizing a loss function which measures the discrepancy between the pairwise distances of all high-dimensional data points and the respective low-dimensional representations. Notably, when the loss function is defined as the sum of squared differences between pairwise distances, MDS is equivalent to PCA [8] and referred to as classical MDS or Torgerson scaling. Other specialization, like Sammon mapping [24] adapts classical scaling by weighting each difference by the inverse of the pairwise distance in the high-dimensional space, thereby retaining more detail in local data structure. Throughout this manuscript, we refer to metric multidimensional scaling (MDS) simply as MDS.

An Autoenconder (AE) [25] is a multi-layer neural network that is trained to reconstruct its input data. The network consists of an encoder that maps high-dimensional data points to a low-dimensional latent representation and a decoder that reconstructs the original data from this representation. Crucially, the middle hidden layer (the bottleneck) has far fewer dimensions than the input data, forcing the network to learn a compressed representation that captures the most salient features of the data.

Many DR methods aim to capture the structure of high-dimensional data by representing them as graphs, where each vertex corresponds to a data point and each edge describes the (Euclidean) distance between two data points. The underlying intuition behind this approach is the hypothesis that the high-dimensional data lies on a lower-dimensional manifold — a topological space where each local neighborhood resembles Euclidean space — embedded in the high-dimensional space. The graph structure then helps approximate the geometry of this underlying manifold. Isomap (IS) [26] exemplifies this idea by first constructing a nearest-neighbor graph of the data (see Section 2.3) and then computing geodesic distances between points on this graph (shortest path lengths along the edges). These geodesic distances better approximate distances along the underlying manifold than straight-line Euclidean distances. isomap (IS) then applies MDS using these geodesic distances to compute a low-dimensional embedding.

The Diffusion maps (DM) [27] framework also operates on graph representations but aims to capture the manifold geometry more robustly than geodesic distances by considering multiple paths between points. Specifically, diffusion maps (DM) defines diffusion distances between two points based on random walks on the graph: this distance reflects the probability of transitioning between the points in a Markov process on the nearest neighbor graph. A low-dimensional embedding is then obtained by solving an eigenvalue problem on the graph's transition matrix. This approach is more robust shortcuts in the graph compared to geodesic distances because it averages over many paths rather than relying on a single shortest path, but like isomap (IS) it focuses more on preserving distance between very dissimilar points instead of the often more interesting local data structure [8].

Over recent years, non-linear, local methods like t-distributed Stochastic Neighbor Embedding (t-SNE) [29, 30] and Uniform Manifold Approximation and Projection (UMAP) [28] have established themselves as the go-to DR methods for visualizing high-dimensional data. These DR methods also define point-to-point distance using nearest neighbor graph structures but prioritize preserving local neighborhood structure instead of all point-to-point distances. Both methods optimize their embeddings by minimizing a loss function that compares probability distributions over pairwise similarities, keeping similar points close together while allowing dissimilar points to spread apart. Appro-

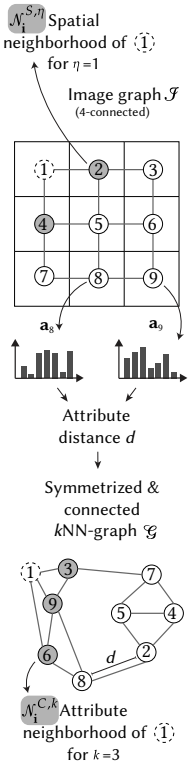
Method	Type	Input	Scope
PCA [22]	Lin	Attr	Glo
MDS [23]	NL	Dist	Glo
AE [25]	NL	Attr	Glo
IS [26]	NL	Dist	Glo
DM [27]	NL	Dist	Glo
UMAP [28]	NL	Dist	Loc
t-SNE [29, 30]			

Abbreviations:
Lin = Linear, NL = Nonlinear,
Attr = Attributes, Dist = Distances,
Glo = Global, Loc = Local.

priate initialization schemes (such as PCA-based initialization) can further help preserve global structure to a certain extent [31]. Below, Section 2.4 discusses t-SNE in more detail.
In contrast to linear, global embedding techniques like PCA, nonlinear, local methods generally yield better cluster separability but are harder to interpret: directions in the embedding space are not linear combinations of the original features, and the axes do not correspond to any meaningful data attributes. Consequently, while these embeddings effectively reveal clustering structure, they do not provide insight into which original variables drive the observed patterns. As such, the t-SNE embedding in Figure 2.1 intentionally does not label any axis. Further, current best practices suggest to use multiple DR methods, as each can highlight different aspects of the data, and incorporate meta data into their visualization, e.g., via re-coloring to help explain patterns [18].
In this dissertation we focus mainly on nonlinear, local DR methods and their hierarchical variants (see Section 3.3). However, the techniques proposed in chapter 4, 5 and 6 mainly focus on incorporating image information into DR methods by integrating image-layout and and high-dimensional attribute information into the definition of (dis)similarities between pixels. They do not further adjust the loss function or optimization procedure of the embedding methods used. Thusly, these proposed techniques may be extended to any distance based DR method.

Supplemental Material SA3 applies the ideas form chapter 4 to UMAP and MDS.

Local DR methods focus on preserving distance between a data point and its close neighbors. But in an image, a data point has neighbors in two spaces: the image layout, i.e. neighboring pixels, and the attribute space, i.e. similar attribute vectors. This duality invites a more thorough discussion of the term neighborhood.



2.3. Neighborhood Definition

The notion of neighborhood is twofold for high-dimensional image data: one can distinguish between neighbors in the spatial and the attribute domain. We refer to the *attribute neighborhood* $\mathcal{N}_{\mathbf{i}}^{C,k}$ of pixel $\mathbf{i}$ as the set of indices of the k pixels with the smallest distances d to the attributes of pixel $\mathbf{i}$:

$$\mathcal{N}_{\mathbf{i}}^{C,k} = \underset{\mathbf{j}=(x,y)_j,\ 1\le j\le n}{k\text{-arg min}} \ d(\mathbf{a_i}, \mathbf{a_j}). \tag{2.2}$$

k-arg min performs a selection of the k arguments in $\{\mathbf{j} = (x, y)_j : 1 \le j \le n\}$ that yield the k smallest distances.
Typically, the squared Euclidean distance is chosen as the distance measure $d(\mathbf{a_i}, \mathbf{a_j}) = \|\mathbf{a_i} - \mathbf{a_j}\|_2^2$, but other distances like cosine or Hamming distance are popular choices as well. It is useful to represent the k nearest neighbors of each pixel in a k-nearest neighbor (kNN) graph $\mathcal{G}$. This sparse graph is often symmetrized. At times (e.g., in chapter 6) it is also advantageous to connect disconnected components in this graph.
Next, we define the *spatial neighborhood* $\mathcal{N}_{\mathbf{i}}^{S,\eta}$. In chapter 4 we assume an 8-connected graph for which the spatial neighborhood is a square of $n_{\mathcal{N}} = (2 \cdot \eta + 1)^2$ pixels centered at $\mathbf{i}$ and with $\eta = 1$. More generally, the set of spatially-neighboring indices for a square neighborhood is given by:

$$\mathcal{N}_{\mathbf{i}}^{S,\eta} = \{(x_{\mathbf{i}} + r,\ y_{\mathbf{i}} + s) : -\eta \le r, s \le \eta\}. \tag{2.3}$$

For a 4-connected graph the spatial neighborhood is not square-shaped and we omit an algebraic definition here. Later, in chapter 6 we are mainly interested in directly neighboring pixels/superpixels and refer to the set of pixels/superpixels immediately connected by an edge as the spatial neighborhood.

We focus on two-dimensional, rectilinear spatial layouts for the sake of simplicity, but in principle our method is trivially extendable for data resolved in three spatial dimensions, i.e., multivariate volumetric data.

2.4. t-distributed Stochastic Neighbor Embedding (t-SNE)

Many non-linear, local, distance-based dimensionality reduction methods like LargeVis [32], UMAP [28] and t-distributed stochastic neighbor embedding (t-SNE) [29, 30] share a similar basic structure. First, based on a distance measure in the attribute space, they construct a neighborhood graph that captures local neighborhoods. Then, a low dimensional layout is produced, with the aim to represent these neighborhoods as faithfully as possible; this process is guided by optimizing a specific cost function. We will discuss the methods presented in this dissertation using the example of t-SNE. While the same concepts are applicable to all distance-based dimensionality reduction methods, we deem it easiest to follow one specific example.
To create a low dimensional embedding as described above, t-SNE uses a symmetric joint probability distribution P to describe similarities between high-dimensional points. Likewise, a joint probability distribution Q encodes the similarity of the corresponding low-dimensional points in the embedding space. Starting with a random initialization of Q, the embedding points are iteratively moved such that the distribution Q matches P well. This optimization process is guided by the Kullback-Leibler (KL) divergence that measures the divergence of P and Q as cost function $Cost(P, Q)$

$$Cost(P,Q) = KL(P,Q) = \sum_{i}^{n} \sum_{j,j\neq i}^{n} p_{\mathbf{ij}} \ln\left(\frac{p_{\mathbf{ij}}}{q_{\mathbf{ij}}}\right), \tag{2.4}$$

where the probability $p_{\mathbf{ij}}$ represents the similarity of two high-dimensional data points $\mathbf{a_i}$ and $\mathbf{a_j}$, and $q_{\mathbf{ij}}$ represent the similarity of the two corresponding low-dimensional data points in the embedding. $p_{\mathbf{ij}}$ is symmetric and computed as

$$p_{\mathbf{ij}} = \frac{p_{\mathbf{i|j}} + p_{\mathbf{j|i}}}{2n}, \tag{2.5}$$

where $p_{\mathbf{j|i}}$ can be interpreted as the probability that the point $\mathbf{a_j}$ is in the neighborhood of the point $\mathbf{a_i}$ in the attribute space. $p_{\mathbf{j|i}}$ is calculated using the distance measure $d(\mathbf{a}_i, \mathbf{a_j})$ between the high-dimensional points:

$$p_{\mathbf{j|i}} = \begin{cases} \frac{\exp\left(-d(\mathbf{a_i},\mathbf{a_j})/(2\sigma_{\mathbf{i}}^2)\right)}{\sum_{\mathbf{k}\in\mathcal{N}_{\mathbf{i}}^{C,k}} \exp\left(-d(\mathbf{a_i},\mathbf{a_k})/(2\sigma_{\mathbf{i}}^2)\right)} & \text{if } \mathbf{j} \in \mathcal{N}_{\mathbf{i}}^{C,k} \\ 0 & \text{otherwise.} \end{cases} \tag{2.6}$$

The number of nearest neighbors $|\mathcal{N}_{\mathbf{i}}^{C,k}| = 3\varphi$ can be steered with a user-defined perplexity φ. The bandwidth σ_i, in turn, is determined based on the given perplexity value such that

$$\varphi = 2^{-\sum_{\mathbf{j}\in\mathcal{N}_{\mathbf{i}}^{C,k}} p_{\mathbf{j|i}} \log_2 p_{\mathbf{j|i}}}. \tag{2.7}$$

3. Related Work

All work in this thesis treats high-dimensional images and thus shares a large portion of its related literature. Here, we gather relevant literature to Chapters 4 to 6 about the exploration of high-dimension images with dimensionality reduction methods. Additionally, each chapter will cover related work specific to the topics discussed therein individually. While high-dimension image exploration is also essential to Chapter 7, it shifts focus from methodology to software frameworks and therefore covers it related work separately.
While dimensionality reduction methods have been extensively studied across various domains, their effective application to high-dimensional image data specifically remains relatively under-explored, a gap that this thesis seeks to address.

3.1. Exploratory Analysis of High-Dimensional Images

Exploration of high-dimensional images is a two-fold issue: exploring the high-dimensional data attribute space and exploring the spatial layout.
There is a multitude of approaches for visual exploration of high-dimensional data [33–36]. A challenge when facing high-dimensional data that is additionally spatially resolved, is to effectively visualize spatial and attribute characteristics in an integrated fashion. MulteeSum [37] compares spatio-temporal gene expression data from fruit fly embryos by segmenting cells in the image and providing multiple attribute summaries per cell. Another approach is to characterize the data attributes in terms of specific features and represent them as glyphs at their respective regions in space [38]. For multivariate volume data, high-dimensional transfer functions have been employed in combination with standard volume rendering techniques [39].
Generally, regarding both high-dimensional and color or grayscale images, visualization and interaction systems for large images use image tiles of various resolutions taken from image pyramids [40–42]. Image exploration might be performed solely based on zoom and pan operations in the image pyramid as presented by Jeong et al. [43], or can be supplemented with additional information. Molin et al. [44], e.g., propagate low-level features to the current zoom level. These examples deal specifically with grayscale or color images, but high-dimensional images cannot be displayed or explored following the same approaches: they do not trivially extend to more than three image channels.

In Chapter 4 we use DR techniques for image exploration.

Chapter 5 couples DR techniques with image interaction techniques.

3.2. Dimensionality Reduction for High-Dimensional Images

Extensive reviews on dimensionality reduction and multidimensional projection techniques can be found [16–18]. Non-linear dimensionality reduction techniques such as t-SNE [29] and UMAP [28] have become popular techniques to visualize and explore high-dimensional data. However, these techniques have been applied to high-dimensional imaging data without considering image-specific characteristics like texture information. Abdelmoula et al. [45] use t-SNE in a segmentation pipeline for high-dimensional images. They embed pixels according to their high-dimensional attribute values to a three-dimensional space, followed by coloring the pixels by using the 3D embedding coordinates as coordinates in the L^*a^*b color space. The resulting color images are then used to aid the segmentation with the goal of identifying tissue segments with similar properties, according to the original attribute space. Recently, Evers et al. [46] followed a similar approach to identify regional correlations in spatio-temporal weather ensemble simulations with the main difference of using MDS instead of t-SNE.

Chapter 4 introduces the notion of spatial layout in DR methods.

Other approaches combine dimensionality reduction with segmented image data. Facetto [47] combines un- and semi-supervised learning to aid in the visual analysis of high-dimensional imaging data in the field of structural biology. After segmenting cells and aggregating their corresponding attributes to features, they use UMAP to display the cells according to their similarities. ImaCytE [48] is a visual analysis tool for similar data that focuses on the analysis of cell neighborhoods. Again, cells are segmented and the attributes of pixels within the cells aggregated. Cells are laid out according to their similarity in the attribute space using t-SNE and the resulting information is used to analyze spatial neighborhoods. All of these applications and tools make use of standard dimensionality reduction methods that do not incorporate spatial information and instead follow a two-step approach using the results of either dimensionality reduction or spatial analysis as input to the other. Instead, in Chapter 4, we directly include spatial information in the dimensionality reduction process to reduce the number of steps and potential points of failure in interactive analysis.

3.3. Hierarchical Dimensionality Reduction

Many image acquisition methods yield images containing many million pixels. Even though UMAP [28] and modern implementations of t-SNE [49] enable efficient embedding computation of data sets the size of single digit million data points, these DR techniques reach their limits when being applied to mega- or gigapixel images. The embeddings insufficiently display detailed data characteristics like intra-cluster dissimilarities, and thus small-scale structures will not be visible anymore, or larger structure is lost. Hierarchical DR techniques tackle this issue and come with a lower computational cost [16]. Recently, several methods that extend existing single-level techniques have emerged. For example, Glimmer [50] (i.e., a hierarchical version of classical multidimensional scaling) and HiPP [51] perform recursively subsampling and hierarchical clustering respectively to create data hierarchies. The more recent HSNE [52] (i.e., a hierarchical version of t-SNE [29]), and HUMAP [53] (i.e. a hierarchical version of UMAP [28]) both select landmarks from lower hierarchy levels as points in higher levels and use random walks to define similarities between them. Multiscale PHATE [54] (i.e., a hierarchical version of PHATE [55]) simulates a diffusion process using random walks to "smoothen" the data and coarse-grain the k-nearest-neighbor data graph repeatedly. These methods all share hierarchical aggregation to represent data points on various levels of abstraction and aim to facilitate exploration of high-dimensional data, but are based on different DR methods. In Chapter 5, we propose a direct coupling between the interactions with an image, i.e., zooming and panning, and representations of hierarchical dimensionality reduction methods.

Chapter 5 connects interactions with hierarchical DR methods and image interactions.

While it is possible to apply these methods to high-dimensional images, none of them takes the spatial information of the data into account when creating their hierarchies or defining similarities between landmarks or sample points. To combine both spatial layout and attribute information into the hierarchy construction we use a recurring motive from the above techniques: random walks. They are typically employed on nearest-neighbor graphs of the attribute data as a means to capture high-dimensional similarities. Diffusion Maps [56] describes a unified framework of random walk based dimensionality reduction methods and shows their equivalence to an eigenvector problem. In Chapter 6 we propose a superpixel algorithm in image space based on similarities in pixel attribute space.

Chapter 6 introduces image information into the hierarchy creation for hierarchical DR methods.

3.4. Multivariate Graph Visualization and Node Embeddings

Graph-based techniques are commonly applied to pattern recognition and computer vision problems on imaging data [57]. For that, images are interpreted as graphs: each pixel is interpreted as a node and neighboring pixels are connected by a link. Several techniques for graph drawing aim to incorporate *network structure* and *node attributes* [58] and are closely related to dimensionality reduction techniques as they also embed complex data into low-dimensional space while preserving certain structure.

GraphTPP [59] focuses on a visual combination of node attributes and connections in a 2D graph layout. First, principal component analysis (PCA) is applied to the data using only the attributes. Then links between nodes are overlaid on the resulting scatterplot. The user can then manually reposition points according to their interpretation, and compute a new linear projection that best fits the modified layout. GraphTSNE [60] aims to preserve graph connectivity and node attribute similarity. It does so by training a graph convolutional network on a modified t-SNE loss that combines the squared Euclidean distance between node attributes and the shortest-path distance between the nodes on the graph. Their design seeks to position two points close in the embedding either when their attributes are similar or they are connected by an edge. Similarly, MVN-Reduce [61] defines a distance measure between two nodes as the sum of a node's attribute distance and their weighted shortest path distance on the graph. The resulting distances are used as input to distance-based dimensionality reduction methods like MDS. The Heterogeneous Network Embedding (HNE) framework [62] aims to create embeddings that position data points with links closer and those without further away from each other. Therein, a neural network is trained with a loss function that builds on a similarity term between point attributes that is weighted depending on their respective node linkage.

Chapter 4 embeds pixels while considering their image-space neighbors.

Embedding nodes from a graph structure is a problem closely related to dimensionality reduction methods. Khosla et al. [63] compare several such node embedding methods, which compute a low-dimensional feature vector for each node in a graph; in contrast to the above discussed dimensionality reduction techniques, these methods are typically designed for the downstream tasks of multi-label classification and link prediction. Variations of random walks are often used to define node features, e.g. the frequency of node visits, here as well. In *node2vec* [64], for example, random walks are biased by adjusting the transition probability for walking backwards and walking to a node that is not connected to the previous node. *tsNET* [65] follows a different approach: they take geodesic distances between nodes in graphs as input for the similarities as defined in t-SNE and map them to 2D based on a modified cost function. Earlier, *Isomap* [26] introduced the idea of using geodesic distances into DR methods that build on multidimensional scaling. However, both Lee and Verleysen [66] as well as Lafon and Lee [56] discuss that using shortest-path-based similarities (i.e., using geodesic distances) can be susceptible to creating shortcuts that jeopardize the representation of the underlying data manifold whereas random-walk-based similarities seem to be more robust. In fact, random walks have been used to estimate geodesic distances in various settings [67, 68].

Chapter 6 uses random walk based similarities for computing superpixels.

Spatial Information in Dimensionality Reduction for High-Dimensional Images

4.

We have seen in our discussion on DR techniques for high-dimensional images that these methods are applied to image data, but do not incorporate image layout information, see Section 3.2. Specifically, spatial patterns (texture) are completely lost in resulting embeddings. In this chapter, we propose an extension to existing distance based DR techniques which informs them with texture information.

This chapter is based on the paper "Incorporating Texture Information into Dimensionality Reduction for High-Dimensional Images" published at the 15*th* IEEE Pacific Visualization Symposium (PacificVis 2022) [69].

4.1. Introduction

High-dimensional data is commonly acquired and analyzed in various application domains, from systems biology [70] to insurance fraud detection [71]. Typically, high-dimensional data are tabular data with many columns (or attributes), corresponding to the dimensionality per item, but there are no connections between items. Dimensionality reduction techniques like t-SNE [29] or UMAP [28] are well-established tools used for exploratory visual analysis of such high-dimensional data [72]. Advances in imaging techniques have introduced an increasing number of imaging data modalities producing high-dimensional images (every pixel represents a high-dimensional attribute-vector). Current state-of-the-art dimensionality reduction methods are commonly used for the exploratory analysis of such imaging data, for example in cultural heritage [73], biology [74], or geospatial applications [75]. However, they rely only on attribute data of pixels and do not take additional spatial information, such as texture, present in such imaging data into account. Thus, in the resulting low-dimensional embeddings, the pixels are only arranged according to their individual attributes (Figure 4.1b), but do not provide any insight into texture, neighborhoods or other spatial relations common in image analysis.

The spatial configuration is, however, commonly of interest when analyzing high-dimensional image data. For example, taking spatial neighborhood information into account, in addition to high-dimensional attributes, has led to new discoveries in single-cell biology [76]. Typical approaches to combine high-dimensional attributes and spatial information, however, rely on a two-stage process: first, high-dimensional attributes are aggregated, for example to classify pixels, then standard image analysis is performed on the aggregate images, see Section 3.2. Decoupling the high-dimensional and spatial analysis in such a way has several downsides: Most importantly, boundaries between clusters in an embedding are often not well-defined, and as such classification is ambiguous and has a level of arbitrariness. Issues with inaccurate classification might appear undetected and lead to wrong conclusions. Furthermore, if problems with the classification become apparent in the spatial analysis, one has to go back to the high-dimensional analysis and potentially loses all progress in the spatial analysis. Moreover, the necessary aggregation in the first step limits what is discoverable in the spatial analysis step. Therefore, we deem the integration of spatial information directly into the dimensionality reduction desirable for exploratory analysis.

We present an approach to integrate spatial information directly into the dimensionality reduction process with the goal to combine attribute and spatial information in a single embedding (Figure 4.1c). Specifically, we propose to adapt the similarity computation, used in distance-based dimensionality reduction, such as t-SNE, UMAP, or MDS, to incorporate different spatial neighborhood features. We exemplarily present different similarity computation methods for such neighborhood comparisons by extending an existing classification [77] to high-dimensional images.

The main contributions of this chapter are:

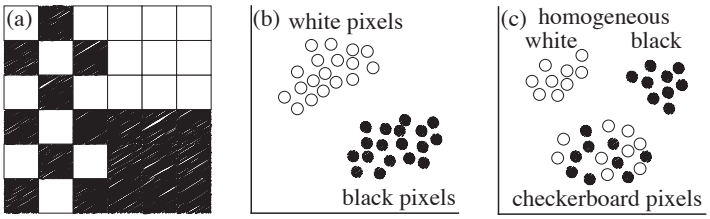


Figure 4.1. ***Texture-aware dimensionality reduction.*** *An image (a) with black and white pixels forms multiple textures. Standard distance-based dimensionality reduction produces one cluster of black and one cluster of white pixels (b), a texture-aware version should create clusters for the different textures (c).*

- incorporating texture information into distance-based dimensionality reduction for exploratory analysis of high-dimensional images through distance measures including image neighborhoods.
- the exemplary extension of t-SNE using different classes of neighborhood distance measures and their analysis.

4.2. Related Work

Relevant work on dimensionality reduction on images is discussed in Chapter 3.

In the following, We aim to report the work most relevant to this chapter, namely, texture and feature extraction in high-dimensional images, and dimensionality reduction in hyperspectral imaging.

Often, high-dimensional image data is visualized indirectly by first extracting features that capture interesting data characteristics and then displaying those [34, 35]. One such feature is texture. We follow Haidekker's definition of texture as "any systematic local variation of the image values" [78] since it emphasises that texture encodes spatially local relationships of pixel values. Texture feature extraction is a broad, well-established field [79] but the extension of single-channel texture features to multi-channel images is not trivial. Typically, single-channel texture features are extracted for each channel and concatenated to a feature vector. Multi-channel texture features such as color co-occurrence matrices proposed by Palm [80] are less common and engineering such features is an ongoing process in the image processing community [81].

Dimensionality reduction has been used to *create* texture features [82] and *explore* texture databases [83], but such approaches are generally out of the scope of this work.

In the analysis of hyperspectral images, dimensionality reduction is an important step for pixel classification. A common approach to include spatial information relies on computing the first couple of principal components of the high-dimensional data and then continuing with classic image processing methods that work on scalar or color data to extract spatial neighborhood information [84, 85]. For example, morphological image processing techniques are used to capture spatial structure in high-dimensional images [86]. Spatial-spectral local discriminant projection [87] takes a more direct way of combining spatial and spectral information into dimensionality reduction by incorporating a weighting factor into the neighborhood preserving embedding that represents the spectral similarity between spatially neighboring pixels. However, this and similar hyperspectral image analysis methods [88] rely on training with ground truth data, which is typically not available in exploratory data analysis. Recently, Halladin-Dąbrowska et al. [75] proposed a workflow using t-SNE for cleaning ground truth data. However, they do not include spatial neighborhood information in their embeddings.

A straightforward way to inform dimensionality reduction techniques of images about their spatial domain is to consider each data point's spatial location in the point similarity measure used during the embedding. Spherical SNE [89] devises a similarity function between data points in the style of bilateral filtering that weights attribute distance with pixel location distance. This approach, however,

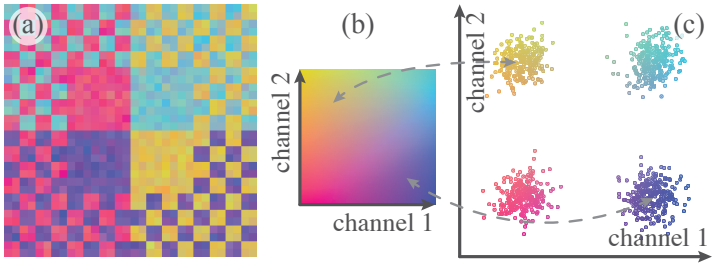


Figure 4.2. ***Synthetic test image dataset*** *with* $X \times Y = 32 \times 32$ *and* $C = 2$*. The distribution in attribute space (c) reveals four groups. We use a 2D colormap (b) to color pixels in image space (a) according to their attribute information.*

does not capture the similarity of the local structure around the compared points, which we aim for.
Applied to an image (interpreted as a 4- or 8-connected graph), all the approaches mentioned in Section 3.4 essentially combine the pixel location distance (geodesic distances on the graph) with the attribute distance, not dissimilar to what is described for the Spherical SNE [89]. In contrast, the goal of our proposed approach is to compare local texture structures rather than absolute distances. Two pixels are compared by taking into account the structure of the high-dimensional values in the spatial neighborhoods of the two pixels.

4.3. Texture-Aware Dimensionality Reduction

Figure 4.2 shows a synthetic toy-example of a 'high-dimensional' image with two attribute channels (i.e., $C = 2$). The spatial layout is displayed in Figure 4.2a with each pixel color coded according its two attribute values using a 2D colormap (Figure 4.2b). Figure 4.2c shows a scatterplot of all attribute values. Four groups are clearly distinguishable based on the attributes. In image space the four groups form eight visually distinct regions, four group-homogeneous and four consisting of checkered patches. We use this image to showcase the characteristics of our proposed approaches and compare it to a standard t-SNE embedding.
To incorporate spatial neighborhood information into low-dimensional embeddings, we propose a set of distance measures that take the spatial neighborhood $\mathcal{N}^{S,\eta}$ of pixels in high-dimensional images into account. The distance between the attributes of two attribute vectors $d(\mathbf{a_i}, \mathbf{a_j})$ is thus replaced by a new texture-informed distance $d_s(\mathbf{i}, \mathbf{j}, f, \eta)$ and the k nearest neighbors in Equation 2.6 will be based on this new measure as well. Since we aim to compare the spatial neighborhood of two pixels, it does not suffice to include their image coordinates or spatial distance; rather, it is necessary to involve each pixel's spatial local neighborhood. Essentially, we are comparing image patches instead of single pixel values. All other steps of the embedding process remain as they were.

4.3.1. Comparing image patches

In the following, we will present a number of texture-aware distance measures d_s. As the space of potential measures is vast, we will focus on a few exemplary measures, following the classification of distance measures for image patches introduced by Zitová and Flusser [77]. In particular, they distinguish image patch comparison into area-based methods (ABM) and feature-based methods (FBM).
ABM and FBM for image patch comparison differ in their approach to compute similarity scores. ABM directly work with the pixel's attribute values of the two image patches to compare. They are sometimes called intensity-based instead, which might reflect the immediate usage of the attribute values more aptly. In contrast, FBM follow a two-step approach of first computing features

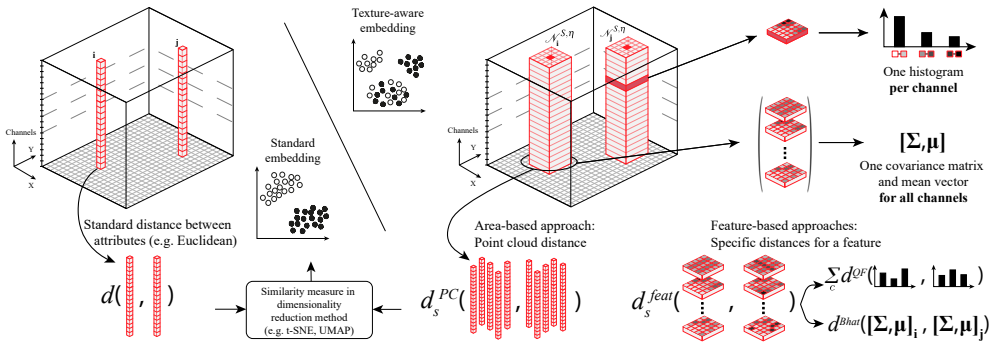


Figure 4.3. ***Incorporating image texture information into dimensionality reduction*** *by adapting the distance measure that defines pixel similarity. High-dimensional image data is depicted in a data cube. Standard t-SNE compares pixels based on their attribute vectors only, e.g., using Euclidean distance (left). We propose to also consider the spatial neighborhood* $\mathcal{N}_{\mathbf{i}}^{S,\eta}$ *of the pixels (right) and present different approaches. Feature-based methods (FBM) derive texture features per channel (e.g., local histogram) or across channels (e.g, covariance matrix) and compare those, while area-based method (ABM) (e.g., point cloud distance) compare sets of original attribute vectors directly.*

for each patch and then comparing those. ABM can be further categorized into correlation-like methods (for example point cloud distances), Fourier methods, mutual information methods, optimization methods [77]. We refer to Goshtasby [90] for an extensive overview and discussion of area-based similarity and dissimilarity measures. Likewise, there exists a rich body of literature discussing image patch descriptors and appropriate feature matching methods used in FBM [91, 92]. To note, most FBM extract various salient features per image with the goal of matching the whole images. Such methods do not necessarily produce one feature value per pixel which is desired for computing the pairwise distance between pixels as in our case. More extensive and general discussions of ABM and FBM can be found in [77, 93]. Here, we will present how to structurally extend image patch comparison to high-dimensional images, and will showcase t-SNE with exemplary methods for each category.

4.3.2. Application to high-dimensional images

Classically, both FBM and ABM are applied to grayscale or color images, but we work with high-dimensional images instead. Typical ABM involve direct comparison of individual points and thus can be applied to multi-channel data, by directly comparing the multi-dimensional points with any applicable metric. For FBM, a direct extension of a single-channel feature to multi-channel data does not always exist or a straightforward extension to multiple dimensions suffers from problems. Consider, for example, local histograms. They work well to summarize single-channel data, but as the number of dimensions of the histogram reflects the number of channels in the data, a local histogram for high-dimensional data would typically be sparse due to the curse of dimensionality [94]. Therefore, in addition to creating multi-dimensional features, we also consider computing traditional one-dimensional features per channel and then compute the distance between the resulting feature vectors.

In the following, we will present a point cloud distance, namely the Chamfer distance, to showcase an area-based method. We use channel-wise histograms as single-channel features and the more general covariance matrix of the neighborhood values as a multi-channel feature to provide examples for FBM. Figure 4.3 illustrates the concept behind the three approaches. An example of these methods on synthetic data will be discussed in more detail in Section 4.4.1.

4.3.3. Feature-based methods

A wide range of image features exist and an adequate choice depends on the application as well as the goal of the analysis [79]. It is out of the scope of this paper to cover all possibilities. We focus on spatial heterogeneity which has been successfully applied, for example, in biomedical tumor analysis [95] or geospatial data analysis [96].
We investigate two texture features that capture local heterogeneity in scalar images. As a single-channel feature example, we capture heterogeneity with local histograms per pixel and channel. Histograms have been successfully used for texture synthesis [97] and lend them-self well as texture features. But since local histograms do not adapt well to high-dimensional data, we use the covariance matrix $\mathbf{\Sigma}$ and channel-wise means μ, roughly generalizing the histogram measure of dispersion, as a multi-channel feature. Covariance information has as well been shown to be useful texture information for texture synthesis in generative adversarial networks [98]. For FBM, the neighborhood distance becomes $d_s(\mathbf{i}, \mathbf{j}, f, \eta) = d_s^{feat}(\mathfrak{F}_\eta(f, \mathbf{i}), \mathfrak{F}_\eta(f, \mathbf{j}))$ with $\mathfrak{F}_\eta$ being the chosen feature extraction operator that will depend on the use case and neighborhood size parameter η.
It is worth noting that the approach of computing the features separately per channel assumes independence between all channels f_c, which is typically not the case. This means that in some cases certain combinations of attribute values and texture features cannot be distinguished. The covariance matrix feature (Section 4.3.3), and point cloud distance-based, (Section 4.3.4), approaches do not have this limitation since they use the full attribute space to measure the distances.

Local histograms features Local histograms are a common way to characterize texture in scalar image processing. We compute one feature, i.e., the normalized local histogram, per pixel and channel (confer the right side of Figure 4.3). The histogram of attribute values of channel c in the spatial neighborhood $\mathcal{N}_\mathbf{i}^{S,\eta}$ is referred to as the vector $\boldsymbol{h}_{\mathbf{i}c} = [h_{\mathbf{i}c1}, \ldots, h_{\mathbf{i}cB}]$, where B is the total number of bins. All entries are normalized by the total amount of pixels in the neighborhood. As the histogram is represented as a vector, rather than a single scalar, this yields a feature matrix per pixel:

$$\mathfrak{F}_{hist}(f, \mathbf{i}) = [\boldsymbol{h}_{\mathbf{i}1}, \ldots, \boldsymbol{h}_{\mathbf{i}C}]. \tag{4.1}$$

This means, to compute the distance between two pixels, we now need to compare two vectors of histograms. We can interpret a histogram as an estimate of a probability density function. As such, we can choose one of the many distance functions defined between probability distributions. One such distance is the quadratic form (QF) distance [99], defined as

$$d^{QF}(\boldsymbol{h}_{\mathbf{i}c}, \boldsymbol{h}_{\mathbf{j}c}) = (\boldsymbol{h}_{\mathbf{i}c} - \boldsymbol{h}_{\mathbf{j}c})^\intercal \, \mathbf{A} \, (\boldsymbol{h}_{\mathbf{i}c} - \boldsymbol{h}_{\mathbf{j}c}) \tag{4.2}$$

$$= \sum_{b,k=1}^{B} a_{bk}(h_{\mathbf{i}cb} - h_{\mathbf{j}cb})(h_{\mathbf{i}ck} - h_{\mathbf{j}ck}). \tag{4.3}$$

Here, $\mathbf{A} = \{a_{bk}\}$ with $0 \leq a_{bk} \leq 1$ and $a_{bb} = 1$, enables attributing a weight between bin indices, e.g., to take distance into account. We use $a_{bk} = 1 - (\mid b - k \mid)/B$ as proposed by Equitz et al. [99].

With the distance per channel in place, we can define a distance d_s^{feat} for all channels as the sum of all channel-wise feature distances:

$$d_s^{hist}(\mathfrak{F}_{hist}(f, \mathbf{i}), \mathfrak{F}_{hist}(f, \mathbf{j})) = \sum_{c=1}^{C} d^{QF}(\boldsymbol{h}_{\mathbf{i}c}, \boldsymbol{h}_{\mathbf{j}c}). \tag{4.4}$$

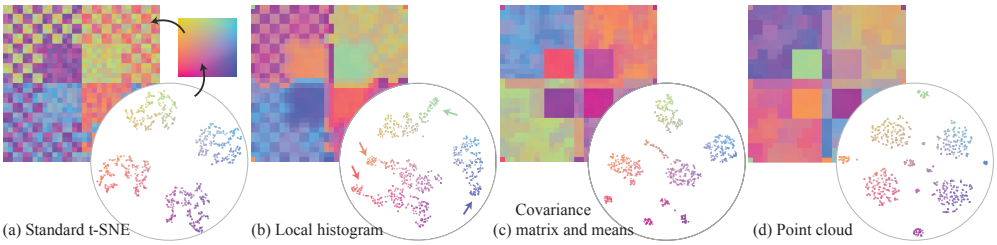


Figure 4.4. ***Comparison of different embeddings*** *of the synthetic image data shown in Figure 4.2 using standard t-SNE (a) and the three presented texture-aware approaches (b-d). We use the re-coloring approach introduced in Figure 4.2 to indicate embedding structure in image space. Here, we use the 2D embedding coordinates to index the colormap shown in (a). As a result, pixels that are close in the embedding space have similar colors in image space.*

Covariance matrix and means A more general dispersion feature for multi-channel data are covariance matrices. We will use these in combination with channel-wise mean values as an example for multi-channel features $\mathfrak{F}_{cov}(f, \mathbf{i}) = [\mathbf{\Sigma_i}, \mu_\mathbf{i}]$. Each entry in $\mathbf{\Sigma_i} = \{\sigma_{ab}\}$ represents the variance between the attribute values within the spatial neighborhood $\mathcal{N}_\mathbf{i}^{S,\eta}$ of the channels f_a and f_b; the vector $\mu_\mathbf{i}$ holds the mean values within the same neighborhood per channel. One measure that is suited for comparing our covariance matrix feature is the Bhattacharyya distance [100]:

$$d_s^{Bhat}(\mathfrak{F}_{cov}(f, \mathbf{i}), \mathfrak{F}_{cov}(f, \mathbf{j})) = \frac{1}{8}(\mu_\mathbf{i} - \mu_\mathbf{j})^T \mathbf{\Sigma}^{-1}(\mu_\mathbf{i} - \mu_\mathbf{j}) + \frac{1}{2} \ln\left(\frac{\det \mathbf{\Sigma}}{\sqrt{\det \mathbf{\Sigma_i} \det \mathbf{\Sigma_j}}}\right) \tag{4.5}$$

with $\mathbf{\Sigma} = \frac{\mathbf{\Sigma_i} + \mathbf{\Sigma_j}}{2}$ and $\det(\mathbf{\Sigma})$ denoting the determinant of a matrix $\mathbf{\Sigma}$.

4.3.4. Area-based methods

A straightforward example ABM is to interpret the attribute vectors of pixels in the spatial neighborhood $\mathcal{N}_\mathbf{i}^{S,\eta}$ as a high-dimensional point cloud, see Figure 4.3. Instead of comparing explicit texture features, we are now computing distances in the high-dimensional space defined by the data attributes directly. Point cloud distances have been used to compare single-channel images [101] and many naturally extend to higher dimensions since they are based around norms of differences between attribute vectors. To stay consistent with the previously-established notation, one can think of it as simply defining the feature as the data values in the spatial neighborhood without any transformation, $\mathfrak{F}_{PC}(f, \mathbf{i}) = \mathbf{Z_i} = \{\mathbf{a_j} : \mathbf{j} \in \mathcal{N}_\mathbf{i}^{S,\eta}\}$, leading to a point cloud distance $d^{PC}(\mathbf{Z_i}, \mathbf{Z_j})$.
Multiple distance measures between point clouds exist [38]. The choice, which one to use, depends on the application that needs to be addressed. A commonly used point cloud distance is the Chamfer (pseudo-) distance [102]. Conceptually, to calculate this point cloud distance d^{PC} between the spatial neighborhoods of two pixels $\mathbf{i}$ and $\mathbf{j}$, for each point in the spatial neighborhood $\mathcal{N}_\mathbf{i}^{S,\eta}$ we find the closest point in the other neighborhood $\mathcal{N}_\mathbf{j}^{S,\eta}$ with respect to a metric (e.g., squared Euclidean distance in our implementation) and average these closest point distances. This yields:

$$\begin{aligned} d_s^{PC}(\mathbf{Z_i}, \mathbf{Z_j}) = &\frac{1}{n_\mathcal{N}} \sum_{\mathbf{q} \in \mathcal{N}_\mathbf{i}^{S,\eta}} \min_{\mathbf{p} \in \mathcal{N}_\mathbf{j}^{S,\eta}} \|\mathbf{a_q} - \mathbf{a_p}\|_2^2 + \\ &\frac{1}{n_\mathcal{N}} \sum_{\mathbf{p} \in \mathcal{N}_\mathbf{j}^{S,\eta}} \min_{\mathbf{q} \in \mathcal{N}_\mathbf{i}^{S,\eta}} \|\mathbf{a_q} - \mathbf{a_p}\|_2^2 \end{aligned} \tag{4.6}$$

In comparison with other point cloud distances, like the closely related Hausdorff distance, the Chamfer distance is more robust against outliers in the neighborhoods. Unlike the max-min Hausdorff distance, here, we take the average of all point-wise minima instead of their maximum. Supplemental Material SA4 compares several Hausdorff family distances for the previously introduced toy-example image discussed in Section 4.4.

4.3.5. Computational complexity

The computational complexity for the presented approaches can be split into two parts: first the feature extraction and second the actual distance functions.
The histogram feature computation in our implementation scales linearly with the number of spatial neighbors. For a single pixel and channel, this yields the complexity for the **local histogram** feature extraction: $\mathcal{O}(n_\mathcal{N})$, scaling linearly, only with the number of pixels in the neighborhood $n_\mathcal{N} = (2 \cdot \eta + 1)^2$. The **covariance matrix** feature calculation is dominated by the computational complexity of a matrix multiplication between two matrices of size $n_\mathcal{N} \times C$, namely $\mathcal{O}(C n_\mathcal{N}^2)$.
The distance calculation is the more time-consuming step for the presented methods. For the **local histogram** feature approach the QF distance computation scales quadratically with the number of bins B, which dominates its complexity in $\mathcal{O}(C\, B^2)$. **Covariance matrices and mean** comparison with the Bhattacharyya distance is more expensive. Its computation involves matrix-vector multiplication and determinant calculation. Using LU decomposition for the latter yields $\mathcal{O}(C^3)$.
Finally, the **point cloud distance** requires the computation of all pairwise distances between the two neighborhoods. As a result, it's complexity scales quadratically with the neighborhood size $\mathcal{O}(C\, n_\mathcal{N}^2)$. Note, that neighborhood-based dimensionality reduction methods, like t-SNE and UMAP, use the point distances to construct k-nearest neighbor (kNN) graph, which requires the computation of all pairwise distances for the whole dataset. Naively, the kNN-graph graph construction would scale quadratically with the number of pixels rather than the neighborhood. However, most modern implementations of t-SNE, and UMAP avoid quadratic complexity by using approximated kNN algorithms [28, 103], as do we, see Section 4.5.
We have also experimentally verified this analysis, showing that the local histogram approach is the fastest and the Bhattacharyya distance the slowest. The full data can be found in the Supplemental Material SA6.

4.3.6. Spatial weighting

So far, we treated all pixels in the spatial neighborhood uniformly. In order to define specific patterns of interest within the neighborhood, for example by assigning pixels further away from the center a lower importance, we introduce a spatial weighting $\mathbf{w}$. Weights are consistent with respect to the center $\mathbf{i}$ of a neighborhood, which implies that a pixel position $\mathbf{p}$ receives the weight: $\mathbf{w}(\mathbf{i} - \mathbf{p})$.
For the histogram features, spatial weights with $\sum \mathbf{w} = 1$ can be included in the histogram construction by scaling pixel attributes by the weight. Weights can be introduced into the covariance matrix and mean computation as detailed in the Supplemental Material SA2, where we also cover the integration into the Chamfer point cloud distance. A two-dimensional Gaussian kernel is a natural weighting choice as it assigns smoothly decreasing importance to pixels further away from the neighborhood center.

4.4. Application on Synthetic Data

To illustrate some of the properties of the different approaches, presented in Section 4.3.3 and Section 4.3.4, we created a simple synthetic image data set,

shown in Figure 4.2. The image consists of 32×32 pixels, with two attribute channels, separating the pixels into four groups (Figure 4.2c). As seen in Figure 4.2a the spatial layout includes four homogeneous regions in the center and checkered patches around them, each constructed by alternating 2×2 pixel blocks of two different classes.

Figure 4.4 shows a standard t-SNE embedding of the synthetic data set as well as embeddings using the three described methods. All four embeddings were computed using a perplexity of 20 and $1,000$ gradient descent iterations. For the three texture-aware approaches we considered a uniformly weighted 3×3 neighborhood. To indicate structure derived from the t-SNE embedding in image-space without clustering, we use a simple re-coloring previously shown by Höllt et al. [104]. In short, 2D coordinates derived from the embedding are added to each pixel. The pixel is then assigned a color by using these coordinates as a lookup into a 2D colormap. As a result, pixels that are close in the embedding space, and thus are similar according to the used distance metric, will have a similar color in the image representation. We use t-SNE as an example throughout the chapter, similar embeddings using UMAP and MDS can be found in Supplemental Material SA3.

Standard t-SNE (Figure 4.4a) separates the pixels into four groups, one per class, with some small scale structure within each class, introduced by noise in the data. However, the embedding does not give any insight into the spatial layout of the four classes. In particular, the pixels of each class positioned in the checkerboard pattern cannot be distinguished from pixels of the same class in the central homogeneous regions.

Figure 4.4b, shows the embedding and re-coloring using the **Local histograms and QF distance**. We defined the number of histogram bins using the Rice rule: $B = \lceil 2\sqrt[3]{M} \rceil = 5$, with the neighborhood size $M = 3 \times 3 = 9$. The resulting embedding is somewhat less clear than the standard t-SNE one, consisting of only three major clusters, however, with more structure within those clusters. The four homogeneous areas in the center show up in separated areas in the embedding (arrows), indicated by their individual colors. These regions are loosely connected to larger regions in the embedding containing the pixels from the checkerboard regions. Notably, the individual two classes forming a checkerboard region do form separate regions within the larger clusters to some degree. However, pixels of the same class in two different checkerboard regions, for example, pixels of with small values in both channels (Figure 4.2), which are present in both checkerboards on the left image half, are separated, as indicated by the blue-ish and orange-ish colors.

The result of our approach using the **covariance matrix and means feature** can be seen in Figure 4.4c. We can clearly identify nine separate clusters in the embedding. The four homogeneous regions correspond to the four small clusters on the bottom of the embedding, while the four larger clusters represent the four checkerboard regions, as indicated by the recolored image. The ninth cluster corresponds to the boundaries between the checkered regions. Again, the homogeneous areas are separated from the checkered but with much sharper boundaries. Different from the previously described approaches, however, each checkered area is recognized as a single cluster in the embedding, meaning that the checkerboard pattern is not visible anymore in the re-colored image. In an exploratory visual analysis setting, this would facilitate the selection and further analysis of regions with specific spatial neighbourhood characteristics.

The **Point cloud distance** approach, here specifically using the Chamfer distance, yields a similarly straightforward partitioned embedding in Figure 4.4d. Again, all four homogeneous image patches are separately clustered as well as the checkered regions. The borders between the different regions now also created individual clusters in the middle of the embedding.

To quantitatively analyse the approaches, we compute the k-nearest neighbor hit, as described by Espadoto et al. [17]. The average neighbor hits for 63-nearest neighbors in embedding space are 77.9% (point cloud distance), 79.4% (Bhattacharyya distance) and 80.4% (QF distance) whereas the standard t-SNE

embedding yields 35.1%. While the point cloud and Bhattacharyya distance result in a higher neighbor hit for small neighbor numbers, their quality decreases slightly faster for larger numbers than the QF distances hit. See Supplemental Material SA1 and Figure A.2b in Supplemental Material SA3 for full details.

4.4.1. Discussion

All three presented texture-aware approaches are able to distinguish between several spatial arrangements of the high-dimensional image.
A drawback of local histogram features is the number of bins as an additional hyperparameter. Since there is no obvious choice for a good setting, the user has to fall back to heuristically setting this value. Further, in the scope of this work, we only discuss the QF distance for comparing histograms. Other distance measures are available and would likely produce different results. As a per-channel feature-based approach the local histograms implicitly assume channel independence. Thus, they cannot capture multidimensional texture patterns. Using multidimensional histograms instead of a 1D histogram per channel might be able to capture such patterns. However, such an approach would drastically increase computational complexity. The histogram size grows exponentially with the number of dimensions, quickly making storing histograms and computing distances infeasible. Further, such histograms are in danger of quickly becoming very sparse and as such would not provide a useful basis for comparison anymore.
The covariance matrix feature can capture multiple attribute dimensions without requiring channel independence with the same goal of comparing the distribution of values within the defined neighborhood. However, instead of comparing all individual values, they are represented in an approximate way based on the assumption that the values are Gaussian distributed. The point cloud distance does not make this assumption and compares all attribute vectors to each other. If we compare the covariance matrix feature with the point cloud distance results, the most prominent difference between the embeddings is how they treat the borders between cluster regions. When data has a bi-modal distribution in a channel, the Gaussian assumption in the covariance feature does not reveal this case, whereas the point cloud distance would.
An advantage of the FBM methods is that they produce features that can aid the interpretation of structure in the resulting embeddings. Visualizing those features in combination with the embeddings is an interesting avenue for future work.
The spatial weights as introduced in Section 4.3.6 and spatial neighborhood size η affect the approaches to different degrees. See the Supplemental Material SA5 for a brief overview of different neighborhood sizes and spatial weights. For example, the Chamfer point cloud distance produces very similarly clustered embeddings for several neighborhood sizes and is — with respect to the synthetic data set — not much affected by radially decreasing spatial weights for this example. Meanwhile, using the histogram feature, the checkered pixels are clustered differently when weights are applied.

4.5. Implementation

We implemented the described distance measures as an extension for the open-source t-SNE implementation in HDI [103, 105], where we use HNSW [106] with our custom distance functions to create the approximated k-nearest neighbor graph. The framework is implemented in C++ and OpenGL for GPU-based calculations; a Python wrapper is provided as well. Our library is available as open-source on GitHub [107].

Code available on GitHub in the repository biovault/Spidr

4.6. Use cases

Here, we illustrate the application of the presented approaches for visual data exploration using two use cases. The first use case (Section 4.6.1) describes the exploration of hyperspectral images, commonly used in geospatial analysis. For the second use case (Section 4.6.2), we applied our method to imaging mass cytometry data, a method that is recently gaining attention in systems biology. We use the Chamfer point cloud distance as an example for an area-based method and the covariance matrix feature exemplarily as a multi-channel feature-based method. The Bhattacharyya distance comes with a high computational cost for high large channel numbers, hence the usage of the point cloud distance for the hyperspectral image example and the covariance matrix feature for the systems biology use case.

4.6.1. Hyperspectral imaging

Similar to digital photography, hyperspectral imaging captures bands of the electromagnetic spectrum. Instead of only three channels for red, green, and blue in digital photography, hyperspectral imaging captures hundreds of narrow spectral bands at the same time, each corresponding to a channel in the output image. This spectral signature can be used to recognize materials or objects in the image. To fully exploit the information in hyperspectral images, spatial relations between the high-dimensional pixels need to be considered [85].
Here, we present a use case, based on the Indian Pines data set [10]. The data set is an established reference hyperspectral image obtained by airborne visible/infrared imaging spectrometry, covering 220 adjacent spectral bands, of which 200 are typically used (after discarding 20 water absorption bands that do not contain useful information). We consider a 145×145 pixel cutout of the data set, known as *Site 3*. This subset of the data is one of three *'intensive'* test sites and thus has been well documented. A ground truth (Figure 4.5e),

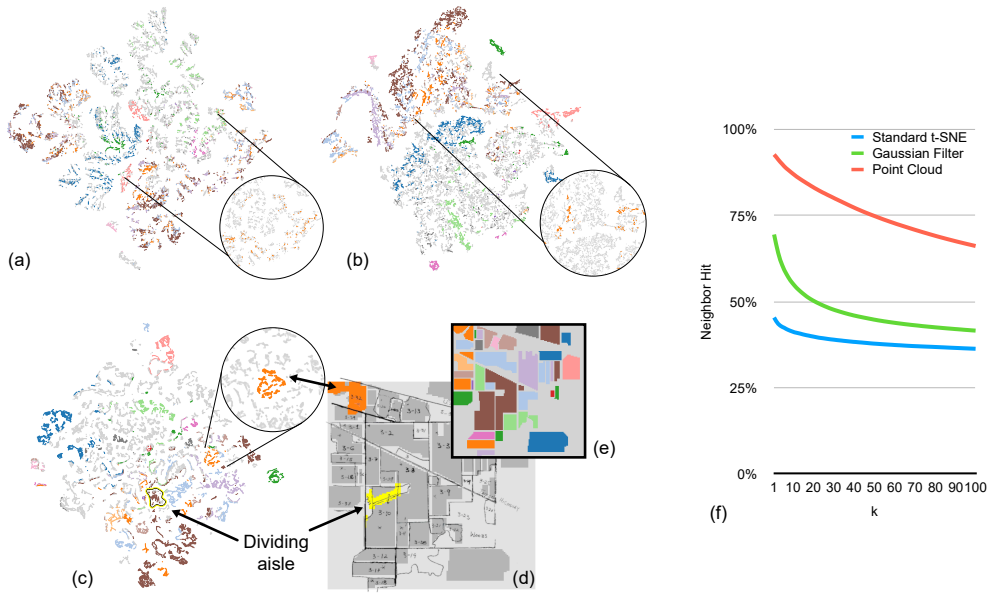


Figure 4.5. ***Indian Pines: Comparison of texture-aware and standard t-SNE embeddings.*** *Embeddings using standard t-SNE (a), standard t-SNE applied after bilateral filtering (b), and our point cloud based t-SNE (c). A manually annotated map and ground truth labels are shown in (d) and (e), respectively. Points in the embeddings are colored, according to ground truth labels for a qualitative comparison of embedding structure. Finally, we show the ration of k-nearest neighbors in embedding space with the same ground truth label for different k in (f). Note, that for an exploratory use-case no ground truth is available and we only show it here to illustrate the properties of the embeddings.*

providing labels indicating ground usage, such as fields, grassland, or houses for all pixels is available for the data set. We use this ground truth data to verify that the structure in our embeddings is meaningful; however, it must be noted that in explorative analysis such a ground truth is not available and information would need to be derived from the embedding structure. Each pixel in the data set maps to a 20×20 meter area on the ground. The 200 channels of this cutout cover wavelengths range from to 400-1300 nm in roughly 9-10 nm steps.
We computed embeddings using standard t-SNE (Figure 4.5a) and our texture-aware t-SNE using the Chamfer point cloud distance with a 5×5 neighborhood (Figure 4.5c). To provide a better baseline than standard t-SNE, we applied bilateral filtering to each channel of the original image data and derived a standard t-SNE embedding from the filtered image (Figure 4.5b). A bilateral filter applies edge-preserving smoothing to an image. Thus, in the resulting image every pixel is a combination of a small neighborhood, providing a straightforward way of incorporating some spatial neighborhood information. We computed all embeddings using a perplexity of 30 and $5,000$ gradient descent iterations. In Figure 4.5, we color-code the ground truth labels on each embedding to indicate how well the structure in the embedding corresponds to structure in the images.
The embedding based on the Chamfer point cloud distance shows more structure than the other two embeddings. Notably, the colored points, corresponding to the labels of the ground truth, form more clearly distinguished clusters (see, for example, the orange points in the insets of Figure 4.5). The other two embeddings show many clusters containing points belonging to multiple regions. Most notably is the weak separation of the background, unlabeled points (light gray) in Figs. 4.5a and 4.5b, compared to the coherent, strongly separated groups in the point cloud-based embedding (Figure 4.5c).
This visual impression is reinforced by a quantitative analysis using the neighborhood hit [17], the average ratio of k-nearest neighbors in embedding space with the same ground truth label. Figure 4.5f shows the neighborhood hit for the first 100 nearest neighbors. The point cloud distance approach yields a significantly higher hit for all k-nearest neighbors than both standard t-SNE and the bilateral filtering approach.
Figure 4.5d shows the original hand-drawn annotations overlaid on the individual fields taken from the ground-truth data. The arrow points at an aisle dividing two parts of a field that was given in the manual annotation but was lost in the ground truth. In our point cloud-based embedding, we could identify a cluster (arrow in Figure 4.5c) corresponding to this aisle and an unlabeled area next to it. The yellow area in Figure 4.5d indicates the pixels corresponding to that cluster, which illustrates the ability to distinguish structure, even beyond the ground truth, in the case of the embedding using the point cloud distance. Hereby, we illustrate the usefulness of combining spatial information with the full attribute space for exploration purposes. While there exist clusters in the other two embeddings that partially correspond to the aisle, they also contain pixels from areas in different regions (see Figure A.14a in Supplemental Material SA7 for an example).
In summary, our point-cloud embedding outperforms the other two with respect to the exploration of spatially continuous, meaningful regions. More examples for similar behaviour, for instance that a specific field is well captured in a single cluster of the point-cloud embedding but divided between multiple clusters in the other embeddings, are shown in Figure A.15 in Supplemental Material SA7.

4.6.2. Imaging mass cytometry

Imaging mass cytometry (IMC) [108] is a recent imaging modality used to study cellular biology. IMC simultaneously captures the expression of up to 50 different proteins in tissue by ablating tissue sections spot-by-spot. Combining the resulting measurements in a regular grid results in a high-dimensional

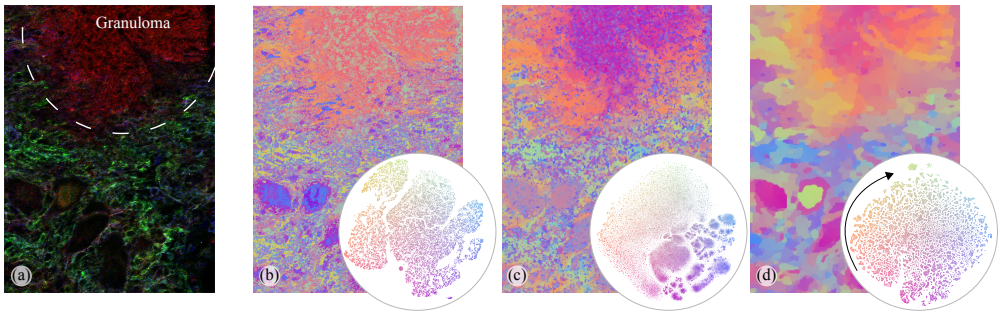


Figure 4.6. ***Imaging mass cytometry.*** *A false coloring image of a lung tissue sample gives an idea of the tissue structure in (a) with the granuloma enclosed by the dashed line. Embeddings derived using standard t-SNE (b), and texture-informed embeddings using the covariance matrix and Gaussian weighting with a* 3×3 *(c) and* 7×7*-sized (d) neighborhood are shown in combination with recolored images to indicate embedding structure in image space.*

image, where the pixel position corresponds to the position in the tissue and the channels to the different measured proteins. Visual analytics and exploratory analysis based on dimensionality reduction is used in practice for the analysis of IMC data, as for example presented by Somarakis et al. [48]. They present a multi-step approach where cells are segmented, followed by aggregating high-dimensional profiles per cell. These are then used to identify cell types using dimensionality reduction and the resulting classification is the basis for exploration of local neighborhoods of cells.

For this use case, we consider a tissue sample from a mammalian lung provided by collaborating researchers. The image measures 272×374 pixels, each pixel represents a 1 µm area, and we consider ten attribute channels, corresponding to ten different proteins describing immune cells and structural properties. Figure 4.6 shows an RGB re-coloring of the sample, mapping red, green, and blue channels to one of three structural proteins each, to give an impression of the tissue layout.

Figure 4.6b shows a standard t-SNE embedding and re-colored following the same re-coloring scheme as in Figure 4.4. Figs. 4.6c and 4.6d show texture-aware embeddings based on the covariance matrix feature and Bhattacharyya distance. We used two different neighborhood sizes, 3×3 in Figure 4.6c and 7×7 in Figure 4.6d to show structures on different scales in the image. For both, we make use of a Gaussian spatial weighting. All embeddings were computed with a perplexity of 30 and 5,000 gradient descent iterations to ensure convergence.

Our collaborators are interested in the composition of cell structures called granuloma, indicated in Figure 4.6, and their surrounding cells. A granuloma is an agglomeration of immune cells, typically to isolate irritants or foreign objects. Current analysis pipelines separate the analysis of the high-dimensional attribute data and spatial layout of the cell data [48]. Our collaborators stated that it would be useful to combine these two steps for early data exploration.

In the re-colored image in Figure 4.6b we can see that the granuloma as a whole is already differentiated from the rest of the image, indicated by a bright orange area with mint green, purple, and blue inclusions. The bright orange indicates a combination of proteins, characteristic for a specific set of immune cells (macrophages) which are expected in the center of the granuloma. It is known that the area around a granuloma is made up of layers consisting of said macrophages and different combinations of other immune cells. The structure of these cell layers and interaction of cells within and between adjacent layers is subject of current research. Hints of this changing composition can be seen in Figure 4.6b where the center largely consists of mint-green inclusions which slowly change to purple and blue inclusions towards the outside. A first hypothesis when analysing the given tissue was that these layers are similar all around the granuloma.

Comparing the small-scale texture-aware embedding and re-colored image in Figure 4.6c, we get a similar impression with a bright-purple colored area with

some inclusions defining the granuloma. Note, that the colors are not directly comparable due to the heuristic nature of t-SNE and the different structures of the embeddings. However, we already see some hints at larger scale structure. The central area of the granuloma (consisting of many blue (mint in Figure 4.6b) inclusions) and the outer layers are now separately colored in a deep pink and orange, respectively, indicating separation in the embedding. Individual cells can still roughly be identified, for example the blue and purple patches within and around the structure.
Finally, using a larger neighborhood as in Figure 4.6d clearly creates areas of similar color, corresponding to higher-level structures. This is expected as the neighbourhood is enlarged. Here, no individual cells are recognizable anymore. The granuloma as a whole is clearly recognizable by a pink to orange area, but in addition a clear layering structure is visible. The granuloma center is a relatively homogeneous dark pink area. Around the granuloma, we can see the layering of structures in different shades of orange to a greenish tone on the far outside, following the colormap applied to the embedding from bottom to top on the left side (arrow Figure 4.6d). Upon inspection of this texture-aware embedding our collaborators were very interested in these layers surrounding the granuloma center and how clearly they were identifiable in Figure 4.6d, hereby eliminating the need for a multi-step approach which was typical for their work flow. They also noted that the layer structure was more varied than they expected which they intend to study further and verify that this is indeed consistent across biological replicates.

4.7. Conclusion

In this chapter, we have presented a framework of texture-aware dimensionality reduction for visual exploration of high-dimensional images and illustrated its potential through examples based on t-SNE and three different texture-aware distance metrics. The generated embeddings combine attribute similarity with spatial context, and, thereby, support the exploration of high-dimensional images. Our method adapts the point similarity calculation of distance-based dimensionality reduction methods by taking the spatial nature of images into account. We presented two classes of approaches for comparing spatial pixel neighborhoods and extended them to high-dimensional images: Feature-based methods (FBM), extracting and comparing features of neighborhoods, and area-based methods (ABM), applying distance measures between the sets of attributes within the neighborhoods directly. We have shown strengths and weaknesses of the different approaches, illustrated them in a synthetic example and presented their applicability via two use cases.
The presented method opens several avenues for future work. We focused on images, i.e., structured, rectangular grids. Extensions to unstructured grids, common in geographic information systems, or graphs are thinkable. While we show several examples of different neighborhood sizes, we did not investigate this parameter in-depth. Using varying neighborhood sizes might reveal spatial structures that are only present at a specific scale. Another interesting avenue might be to investigate the potential of other feature extraction methods, like Markov random field texture models or neural network approaches to capture domain-specific texture characteristics when training data is available. Additionally, visualizing the extracted features alongside the embeddings is an interesting idea in itself.
We have shown that texture-aware dimensionality reduction methods can provide insights into high-dimensional images that cannot be captured with standard dimensionality reduction methods alone. Yet, algorithmic enhancements alone cannot fully support the demands of high-dimensional image exploration. In the next chapter, we will discuss how image interactions like zooming and panning can steer with level-of-detail changes in hierarchical embeddings.

Coupled Exploration of High-Dimensional Images and Hierarchical Embeddings

5.

In the previous chapter we explored an algorithmic modification to inform DR techniques about image information. In this chapter we shift our focus to interaction mechanism with high-dimensional images and hierarchical embedding representations.

This chapter is based on the paper "Interactions for Seamlessly Coupled Exploration of High-Dimensional Images and Hierarchical Embeddings" published at the 28*th* Symposium on Vision, Modeling, and Visualization (VMV 2023) [109].

5.1. Introduction

A common exploration setup of high-dimensional images consists of multiple coordinated views showing an image representation and a low-dimensional embedding of the attribute data side-by-side. The image space is explored mainly by navigation, i.e., panning and zooming interactions to focus on a region of interest (ROI), since large images typically exhibit a higher resolution than a screen is able to display physically. Therefore, navigation in image space is commonly supported with image pyramids: Each attribute channel is repeatedly downsampled to yield smaller images at multiple scales of detail. Exploration starts at a lower resolution, matching the viewport pixels closely, from where a user can then zoom into ROIs, which will automatically move down the pyramid into higher level-of-detail views.
Whereas scalar or three-dimensional data can be easily mapped to colors, high-dimensional attribute data cannot be directly shown in screen space without a mapping from the high-dimensional to a color space. This mapping is often achieved through a selection of attributes, clustering, or coloring of 2D/3D-projections of the attribute data. Typically, the attribute space exploration of high-dimensional images still cannot be performed well in image space alone but is augmented with views on the attribute data. Attribute vectors are often embedded with DR techniques like UMAP [28] or t-SNE [110] and subsequently explored in the resulting low-dimensional embedding spaces, e.g., in single-cell analysis with cyclic immunofluorescence (CyCIF) images [47], hyperspectral images of artworks [111], or remote sensing [112]. While image sizes in the order of a million pixels are common, data set sizes of over 100,000 points are considered very large for DR techniques like t-SNE [113]: the resulting embeddings usually cannot capture all desired detail and come with increased computational cost.
Hierarchical DR techniques, such as HiPP [51], HSNE [52] or HUMAP [53], have been developed to tackle issues that emerge from large amounts of data points. They decrease the embedding size by using landmarks to create a hierarchical data structure, in which each level represents the original data set at a different level of abstraction. Hierarchical DR techniques follow the "*overview first, zoom, filter, details-on-demand*" approach [114] for interactive data exploration. They start out presenting the user an overview embedding, which shows dominant data structures. From there, the user can request more refined embeddings by selecting clusters, which will show a subset of the data at a more detailed hierarchy level. This refinement interaction can be seen as analogous to zooming in image space, based on an image-pyramid, to achieve higher levels of detail.

Existing (hierarchical) DR methods largely target abstract high-dimensional data and thus lack interactions specific to exploration of high-dimensional images. Most importantly, there is no coupling between user interactions in image space and embedding space. E.g., zooming into a part of the image, i.e., requesting more detail for this part of the data, has no effect on the level-of-detail of the embedding view. This requires a set of interactions (i.e., selection and zoom) in the embedding space to achieve the desired detail. Ideally, navigation in image space comes with a desired adaptation of the view of the hierarchical embedding space. Figure 5.1 shows an abstract example of such coupled image

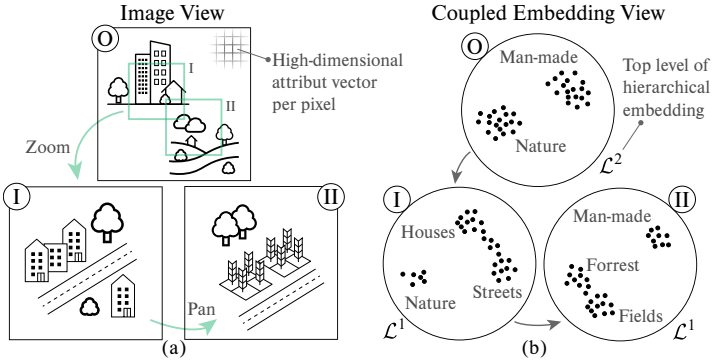


Figure 5.1. ***Coupling concept:*** *(a) Image interactions with a high-dimensional image showing mostly man-made object (top) and nature (bottom) specifically zooming (I) and panning (II) to focus on mode detailed views. (b) Coupled view of a hierarchical embedding corresponding to the overview (O), zoom (I) and pan (II) image views. Interacting with the image view triggers a corresponding change of detail level in the embedding, from hierarchy level 2 to 1.*

and embedding views: Zooming into an image region Figure 5.1a triggers an update of the embedding view Figure 5.1b, which is set to display a higher detail embedding level, e.g., the roads that were not visible previously. A coupling between the image scale space and hierarchical embedding space interactions would thus enable a fully image-aware high-dimensional data exploration and analysis.
The main contribution of this chapter is an interaction paradigm that couples interactions in image space to hierarchical-embedding actions, including

- a mapping from image navigation interactions to embedding space actions,
- an optimization strategy for level-of-detail adjustment based on ROIs in image space, and
- its implementation as an extension of HSNE, and an evaluation on a representative data set.

5.2. Related Work

Relevant work on visual analysis and exploration of high-dimensional data is discussed in Chapter 3.

In the following, we aim to report the work most relevant to this chapter, namely, interaction paradigms for the exploration of and interaction with high-dimensional images.
To adequately explore high-dimensional images, both the high-dimensional attribute space and the image layout have to be taken into account. Ellsworth et al. [115] discuss a holistic approach of showing multiple channels side by side using a wall of monitors. Toolboxes like PySpacell [116] provide various spatial statistics functions to analysis pre-segmented high-dimensional images. SquidPy [117] is a framework that brings together high-dimensional image viewers and image analysis tools.
State-of-the-art high-dimensional image analysis toolkits stress the importance of region-of-interest based exploration of large images. Scope2Screen [118] is a Focus+Context oriented application, which provides lens views on ROIs and lets the user define false RGB recolorings of the viewport, based on manually selected attribute channels. They mention the need for DR techniques and suitable visual representations of found features in image space in order to couple image and feature space more closely. Others, like histoCAT [119], ImaCytE [48], or Facetto [47] offer multiple coordinated views to analyze high-dimensional image data, including image viewers, parallel coordinate plots, and DR plots. Our interaction coupling between hierarchical embedding and high-dimensional image data allows for embedding the entire image data and recoloring of image ROIs based on the entire high-dimensional attribute space.
Interacting is essential for the exploration of dimensionality-reduced data. There exist various classification approaches, e.g., Liu et al. [36] who divide these interactions into computation-centric, exploratory, and model manipulating. Sacha et al. [120] described common user interactions with DR methods more

thoroughly. Past discussions of interactions with visualization techniques for high-dimensional data by Yang et al. [121] and Sifer et al. [122] focused on parallel coordinates and table-based approaches. Recently, Höllt et al. [104] proposed Focus+Context-based interaction techniques specifically for the exploration of hierarchical embeddings. Marcílio-Jr. et al. [123] similarly specified an interaction technique for single-level embeddings. These prior works, however, focus on interactions solely with embeddings. In this chapter, we discuss how a hierarchical embedding should react to user interaction with an image representation of the data.
Elmqvist and Fekete [124] propose a generalized model for interactions with visualizations of hierarchically aggregated data. Their model, though, assumes a single view of the data, whereas we tackle the problem of interacting with two separate views: a spatial data layout (image view) and an embedding hierarchy (embedding view), coupled to the image view. We aim to specify suitable interactions with the image view and corresponding actions of the embedding view.

5.3. Tasks and Requirements

The main purpose of coupling image-space interactions to embedding-space actions is to enable a user to navigate in image space (**T1**), while simultaneously exploring the attribute space of the currently visible image region (**T2**). Two-dimensional embeddings are a useful modality for exploring similarities in the attribute space. Albeit not a direct interaction with the image space, the user should still be able to coarsen and refine the level of detail in the embedding directly (**T3**), as it is already possible in traditional approaches.
In summary, the user should be able to

- **T1** navigate (zoom or pan) in image space,
- **T2** explore the attribute space of an image ROI, and
- **T3** request more or less detail for a ROI in attribute space.

A typical image exploration starts with the entire image in view (*overview*), followed by zoom and pan operations to different ROIs for *detail* inspection. The embedding space exploration should mirror this *"overview first, details-on-demand"* characteristic of the image navigation (**R1**) with the intention of providing analogous reactions in the attribute-space depiction to a single image-space interaction. This entails that, as the user focuses on a spatial region of interest, the embedding should be limited to a set of points which represents the ROI (**R2**, **R4**). This contrasts other conceivable approaches that might follow Focus+Context paradigms and would represent image areas outside a ROI as well. Instead, **R1** ensures a maximal appropriate detail level for a given ROI and reduces computational costs by restricting both view and computation to a subset of all data point. Additionally, to minimize cognitive load on the user, the embedding should preserve coherence between updates when changing ROIs (**R3**). In order to allow for an interactive data exploring, the embedding has to update fast, that is, any additional computational effort on top of the embedding procedure should be minimal (**R4**). Further, to enable linking of image and attribute spaces, e.g., through highlighting of representative embedding points and their represented pixels, a data mapping between arbitrary selections in either space has to be supported (**R5**).
Thus, to successfully accomplish the user tasks, the image-to-embedding coupling should:

- **R1** follow the *"overview first, details-on-demand"* approach,
- **R2** represent the ROI,
- **R3** provide stable transitions between embeddings,
- **R4** update at interactive exploration speeds, and
- **R5** link selections between image and embedding space.

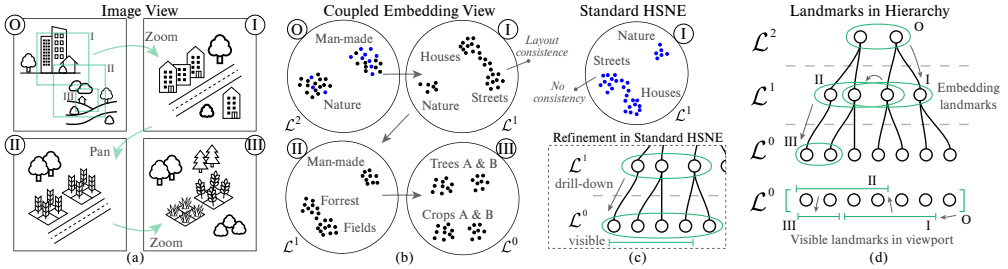


Figure 5.2. ***Interaction overview:*** *Image interactions and embedding hierarchy reactions: (a) exemplary image space depicting a region of mostly man-made object (top) and mostly nature (bottom) with three viewports (I zoom from overview, II pan, III pan and zoom). (b) depicts the embeddings as shown after each interaction. The data points that are currently in the viewport and landmarks that are shown in the corresponding embedding views are shown in (d). In contrast to standard HSNE interactions, where a refinement of the landmarks that represent ROI I, marked with blue ●, leads to inconsistent cluster placement, our method keeps a consistent layout, (b, I) and (c, top). The scattered placement of those top-level landmarks renders manual selection practically impossible. Changing the level of detail for a given ROI (e.g., drilling down) can lead to embeddings containing invisible landmarks as seen in (c, bottom).*

5.4. Coupling Image Navigation and Embedding Space

An intuitive way of coupling image-space navigation and high-dimensional attribute-space exploration is to create a new embedding for each new ROI in image space. However, this approach does not fulfill all our embedding-view requirements and user tasks. First and foremost, neighborhood-based DR techniques would have to recompute neighborhood graphs over and over, which severely limits interactivity, thus breaking **R4**. Gigapixel or larger images are not uncommon and already the first top-level embedding, which encompasses all data points, can be infeasible to compute in reasonable time. With such large images, even ROIs that cover only a small part of the image space can contain hundreds of thousands or even millions of data points. Further, the large number of points leads to indistinguishable similarity structures within clusters, which might obscure interesting data characteristics. Finally, this approach only enables a single level of detail per ROI since it is not possible to refine or coarsen a standard embedding, thus breaking **T3**.

Hierarchical DR techniques overcome these issues. They are better suited for embedding large data sets and do not require neighborhood re-computations for data subsets (**R4**). Their hierarchical structure also allows a user to coarsen or refine the level of detail shown for the attributes of an image ROI (**T3**). An additional benefit to exploring image patches using hierarchical embeddings is that the resulting data representations are informed by the entire data manifold and not just a subset of the data.

5.4.1. Interactions: Zoom and Pan, Drill-Down and Roll-Up

Large images are most commonly navigated by zooming and panning operations. These operations change the size (zooming) and position (panning) of a viewport over the image and determine the ROI shown to the user. We generally follow the mathematical notations for hierarchical embeddings and interactions with hierarchical data structures as used in the literature [104], and fully laid out in Supplemental Material SB1. This notation does not support the concept of linking two spaces, which motivates the introduction of two new symbols: We denote the set of all data points within an image viewport as $\mathcal{D}_I$ and the set of all corresponding landmarks shown in the embedding view used to represent this viewport as $\mathcal{D}_E$.

Zooming and panning focus exclusively on the spatial data layout. Both operations modify $\mathcal{D}_I$ by adding visible or removing invisible data points. In our coupled image and embedding setting, we want zooming and panning in the image view to update the embedding (**T1**, **T2**). Thus, we need a mapping of the image interactions to possible actions in the embedding. Whenever $\mathcal{D}_I$ changes, we have to recompute $\mathcal{D}_E$ so that every landmark $L_j^k \in \mathcal{D}_E$ on embedding

hierarchy level k represents at least one data point $L_i^0 \in \mathcal{D}_I$ and all data points in $\mathcal{D}_I$ are represented by a landmark in $\mathcal{D}_E$ (**R2**). Given a single level of detail in the image viewport we define all landmarks in the embedding to be from a single hierarchy level as well. The selection of the hierarchy level will be discussed in Section 5.4.2.
Figure 5.2a showcases zoom and pan actions in an abstracted high-dimensional image and indicates which set of data landmarks $\mathcal{L}^0$ are currently in $\mathcal{D}_I$ (Figure 5.2d). Each viewport change triggers an update of $\mathcal{D}_E$ (Figure 5.2b). The updated embeddings aim to be consistent with their predecessors, i.e., clusters that represent similar data points remain in nearby embedding positions. Standard hierarchical embedding refinement do not feature such coherence (Figure 5.2c).
Starting an exploration, the viewport encompasses the entire image and thereby all $\mathcal{L}^0$ landmarks. The corresponding top-level embedding contains all top-level landmarks. Zooming-in only ever removes elements from $\mathcal{D}_I$ and, in line with this, the updated embedding will either only contain a subset of the previous landmarks from the same abstraction level or landmarks from a finer abstraction level that are represented by the previous landmarks. Panning and zooming-out, however, may add previously unseen data points into the viewport and thereby might require the inclusion of previously unrepresented landmarks into the embedding. The pan, labeled II in Figure 5.2a, shows such an update of $\mathcal{D}_E$.
The set of data points in the viewport $\mathcal{D}_I$ corresponds to a selection $\mathcal{K}^0$ of landmarks from the data level: $\mathcal{K}^0 = \mathcal{D}_I \subseteq \mathcal{L}^0$. To find a set of landmarks $\mathcal{D}_E$ on a level k that represents $\mathcal{K}^0$, we need a general mapping of landmark selections between levels from $\mathcal{K}^k$ to $\mathcal{K}^{k+1}$.

5.4.2. Landmark Mapping

The importance of landmark mapping is twofold: defining which embedding landmarks $\{L_j^k\} \in \mathcal{D}_E$ best represent $\mathcal{D}_I$ (**R2**), as well as linking selections between image and embedding views (**R5**).
Standard approaches use top-down mappings since users define selections in embedding space: starting at the top-level, refinement actions should represent all landmarks contained in the selection; but in our image-driven scenario this yields many landmarks outside the image ROI, see Figure 5.2c (bottom). We use a **bottom-up mapping** approach to map $\mathcal{K}^0$ to $\mathcal{K}^k := \mathcal{D}_E$, which represent the image ROI (**R2**). For rolling-up one level we define a set $\mathcal{K}^{k+1}$ that represents a set $\mathcal{K}^k$ as the union of all parents of the landmarks in $\mathcal{K}^k$. To avoid traversing the hierarchy when rolling-up several levels, we can cache the representative landmark on each level for every data point when computing the embedding hierarchy in the first place. We use the same approach when drilling-down into the hierarchy, based on a given viewport $\mathcal{D}_I$. Instead of computing all children $\mathcal{K}^{k-1}$ that are represented by $\mathcal{K}^k$ we use the bottom-up mapping to find the minimal set $\mathcal{K}'^{k-1} \subseteq \mathcal{K}^{k-1}$, that contains only the landmarks needed to represent $\mathcal{D}_I$. This means any set $\mathcal{K}^k$, representative for the data points $\mathcal{K}^0$ in the viewport, can immediately be computed as the union of the representative landmarks on level k.
For linking selections between a subset of the image viewport to the embedding we follow the same bottom-up approach, the only difference being that instead of rolling-up the entire viewport we start with the selected subset $\mathcal{K}^0 \subseteq \mathcal{D}_I$. Vice versa, for linking selections from the embedding to the image, we traverse the hierarchy downwards for all selected embedding landmarks $\mathcal{K}^k \subseteq \mathcal{D}_E$ to find the corresponding data point selection $\mathcal{K}^0 \subseteq \mathcal{L}^0$.
At this point, we need to define how to couple the zoom factor in the image, i.e., the ROI's fraction of the full image space, to the selection of the hierarchy level k in the embedding, according to the underlying goals and requirements. We define such a heuristic with the aim to keep the number of landmarks in the embedding space within a pre-defined budget, similar to the visual entity

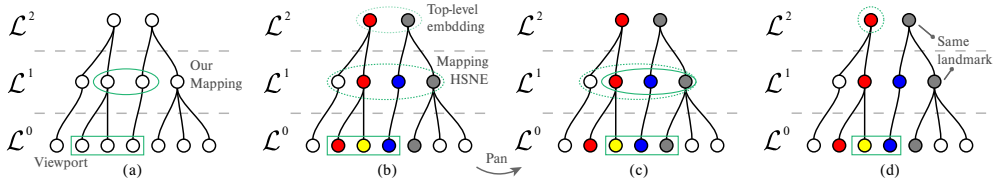


Figure 5.3. ***Mapping comparisons:****. (a) shows the $\mathcal{D}_E$ in resulting from our mapping. (b) depicts the HSNE landmarks in resulting from the top-down approach. Colors indicates data level indices. (d) shows how a pan action updates landmarks in the coupled bottom-up approach whereas the top-down approach might not display any change at all and (c) depicts HSNE's bottom-up, landmarks-in-viewport-only mapping.*

budget introduced by Elmqvist et al. [124]. Different from their approach, we propose to find the level with the number of landmarks closest to a target number v_t, instead of applying a hard maximum to accommodate hierarchies with large differences in the number of data points between neighboring levels. In our framework, the budget is user defined. According to our requirements, the budget should be chosen small enough to allow for interactive computation of the embedding (**R4**) but could, e.g., also be defined as to not overload the visual capacity of the linked embedding view. Given a budget v_t, we determine the hierarchy level k by calculating the set $\mathcal{H}^k$ that represents $\mathcal{H}^0$ for all hierarchy levels. We pick the level k for which $|\mathcal{H}^k|$ is closest to v_t.
We also enable refining or coarsening the level of detail of the representative landmarks $\mathcal{H}^k$ in the embedding view for a given viewport selection $\mathcal{D}_I$ (**T3**). Knowing the current hierarchy level k and $\mathcal{D}_I$, we can immediately compute $\mathcal{H}^{k-1}$ or $\mathcal{H}^{k+1}$ with our bottom-up mapping. In contrast, simply rolling-up or drilling-down the current landmarks in the embedding hierarchy view without considering the viewport does not yield a satisfying result. As exemplified by Figure 5.2c (bottom), e.g., requesting more detail for an embedding by drilling-down all landmarks can yield landmarks outside the viewport (breaking **R2**).

5.4.3. Implementation using HSNE

Code available on GitHub in the repository ManiVaultStudio/ImageEmbeddingCoupling

Our landmark and interaction mapping is suitable for any tree-based hierarchical embedding method. As proof of concept, we implement our interactions with HSNE [52] in the visual analytics framework ManiVault [125]. To do so, we had to adjust HSNE in some aspects. Most importantly, HSNE defines representation between levels probabilistically. HSNE defines an *area of influence* $I_{L_j^{k+1}}(L_i^k)$ that indicates the probability that a landmark $L_j^{k-1} \in \mathcal{L}^{k-1}$ is well represented by $L_i^k \in \mathcal{L}^k$. The resulting data structure is not a proper tree since each landmark in $\mathcal{L}^k$ can be represented by multiple landmarks in $\mathcal{L}^{k+1}$. In order to convert this structure into a tree, we compute for every data point the landmark in each scale with the maximum influence exercised on the respective data point. Notably, HSNE landmarks are not aggregates but each landmark L_i^k corresponds directly to an actual data point L_j^0.
In contrast to our method, regular HSNE employs a **top-down mapping** for selection linking. It only applies this mapping from embedding to the data level (image) and does not use a reverse mapping from image to embedding view. Particularly, when linking from a lower to a higher hierarchy level, selections are linked based on the landmarks that are contained in both levels only, but not their parents. That means, rolling-up some $\mathcal{H}^0$ in the image will result in $\mathcal{H}^k = \left\{L_i^k \mid L_i^k \in \mathcal{H}^0 \text{ and } L_i^k \in \mathcal{L}^k\right\}$. Figure 5.3 shows a simplified HSNE hierarchy with three levels, where each landmark is connected to its most representative landmark in the next abstract level. Figure 5.3a indicates embedding landmarks resulting from our mapping for the three data points in

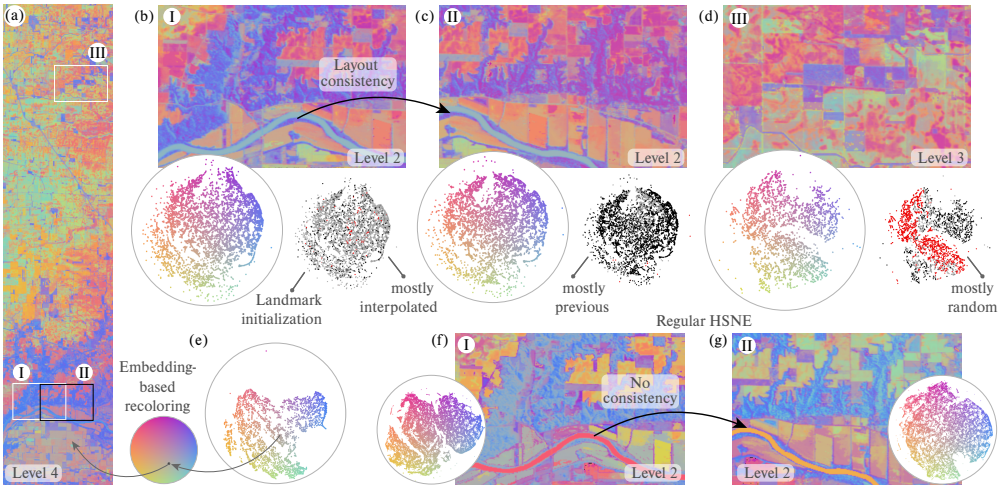


Figure 5.4. ***Indian Pines:*** *(a) Recolored image based on top-level embedding as shown in (e), with three ROI viewports as obtained after a zoom (I) and pan (II, III). The top row (b, c, d) shows corresponding coupled embeddings as well as the current re-colored viewports. When changing the viewport to a region which is represented by a similar set of landmarks, e.g., spatially neighboring regions, to embedding layout stays consistent. The initialization mode of each landmark is indicated in a second scatterplot as based on previous ●, interpolated ●, and random ● positions. Standard HSNE refinements of landmark sets that are representative of a current viewport, in (f) and (g), do not enable consistent embedding exploration and contain more landmarks for the same level of detail.*

view. In contrast, as regular HSNE is top-down-oriented, it coarsens all top-level landmarks (or those representing the viewport), resulting in an embedding that contains all level 1 landmarks, of which some do not represent the viewport at all (Figure 5.3b). Using a visual budget target of 2, HSNE would also not refine the top-level at all in this example. Selecting the data points ● and ● in the image viewport (Figure 5.3d), would only highlight ● in $\mathscr{L}^1$ and none in $\mathscr{L}^2$, since neither point is a landmark in that level.

In the standard HSNE exploration this mapping is sensible. Typically, interactions with an HSNE embedding, like refining a selection, build on the assumption that the selection is continuous in the embedding space [104]. The likelihood that $\mathscr{K}^0 \cap \mathscr{L}^k$ contains all relevant landmarks is thus high. But when interacting with the image space, that is, selecting a spatially connected region in the image, the linked embedding landmarks usually do not correspond to similarly confined regions in the hierarchy or the embedding layout. Rather, representative landmarks will be scattered throughout the embedding, since neighboring image pixels might depict data points with vastly dissimilar attribute vectors (Figs. 5.2b and 5.2c).

5.4.4. Initialization of Embedding Updates

When drilling-down or rolling-up in a hierarchical embedding a new embedding needs to be created. Initialization is a crucial step when calculating neighborhood-based embeddings. In the default implementations of HSNE these new embeddings are initialized randomly. In order to preserve the analysis coherence, we want to re-use landmark positions from the current embedding for the initialization of the updated embedding (**R3**). Similar to [104], we initialize all landmarks in $\mathscr{D}_E$ that were preserved during drill-down or roll-up with their previous positions from before the interaction Landmarks that were added during a drill-down are initialized with the position of their respective parents. When moving horizontally in the hierarchy, newly added landmarks are initialized based on their neighborhood in the hierarchy. Therefore, we query the existing neighborhood graph of the scale and interpolate the position using the three closest neighbors that were present in the corresponding embedding

before the interaction. In cases, where $\mathcal{D}_E$ changes strongly, it might not always be possible to find a neighbor for a given added landmark. In this case, the new landmark is initialized at a random position.
As a result, upon panning to a region that remains similar with respect to the attributes of the shown data points, the embedding is not expected to change strongly; our initialization ensures that there is consistency between embedding updates (**R3**). Nevertheless, when panning to a region containing points with rather different attribute vectors, there will be little overlap between the previous Current $\mathcal{D}_E$ and the embedding will essentially be initialized randomly.
Initializing t-SNE embeddings (or derivatives that follow the same optimization procedure) with small values is important for their convergence [113]. During the optimization process, the embedding's extends grow due to the repulsing forces that are responsible for creating space between dissimilar points. To re-enable proper convergence behaviour, we utilize the embedding extents at a given moment of the first optimization and re-scale all points into this frame (shrink the embedding) during re-initialization of the embedding for each update.

5.5. Exemplary Use Case: Hyperspectral Image Exploration

Here, we show the application of our approach to a representative hyperspectral data set. We indicate the performance in comparison to exploration of the same data using standard HSNE and focus on embedding transitions and visual stability.
The Indian Pines Test Site 3 [10] data sets contain a $614 \times 2,678 \approx 1.6$M pixel large image with 200 dimensions attached to each pixel that depict the light spectrum reflected by the objects in view, specifically fields (e.g., corn and soy), forests, roads, rivers and houses (more information on the data set in Supplemental Material SB3). A typical workflow during the exploration of such data using DR techniques starts with an *overview* of the image layout and attribute data, in our scenario given by the top-level HSNE embedding.
In the following, we use image re-colorings to indicate embedding structure — we map the 2D embedding positions to color using a 2D colormap by Bernard et al. [126], as shown in previous work [69]. This color-coding allows connecting the embedding and image space while minimizing imposing additional structure that comes with choosing a specific clustering algorithm. We computed a 5-level HSNE hierarchy by picking the 25% most important landmarks (with respect to their area of influence) at every level to go to the next abstraction level [104], leading to a top-level embedding with 4,205 landmarks (Figs. 5.4a and 5.4e). This embedding is laid out over 2000 iterations. For better reproducibility and to improve global structure we initialize the top-level embedding with the first two PCA components of the top-level landmark attribute data [113]. Each following embedding is laid out over 500 iterations with otherwise default parameters as introduced by Pezzotti et al. [52]. Fewer iterations are sufficient due to the initial embedding structure given by our re-initialization scheme; additional iterations can be run on demand. The visual budget target is set to 10,000 landmarks to provide an appropriate balance between detail and performance, aiming for total update times of less than one second between changing the viewport and finished embedding. The embedding view is constantly updated during the gradient descent iterations starting about 300 ms after the user interaction, providing visual feedback and thus visual coherence. For detailed timings, we refer to Supplemental Material SB2. The highlighted ROIs mostly depict three spectrally distinct top-level clusters (Figure B.3) with blue and violet hues corresponding to woods, and reddish and yellow-green hues corresponding to different types of fields.
Starting with the top-level embedding and image in full view, we will first focus on an area in the lower part of the image (**T1**) which is predominantly

covered by pixels from two regions in the top-level embedding, woods and various fields (**T2**). We *zoom-in* to ROI Ⓘ to focus on a subset of 57,424 pixels (Figure 5.4b). Driven by the image-space zoom, we automatically drill down the HSNE hierarchy in a single interaction step to level 2 with a representative landmark count of 9,738, closest to our visual budget target of 10,000 (**R1**, **R2**). The entire embedding update took a little under a second of which our hierarchy traversal only contributed 50 ms (**R4**). Compared to the overview recoloring Figure 5.4b, we see that the recolored patch in the new embedding shows a more detailed structure, within the forest areas and also between (sub)types of fields, due to the more detailed hierarchy level. Notably, since we initialize most of the landmarks in the updated embedding, based on the previous landmark positions (indicated in Figure 5.4b lower right), their general global positions and thus corresponding colors are preserved (**R3**). E.g., the forest region is still represented by landmarks in blue and violet hues, preserving a user's mental frame. In contrast, following standard HSNE interactions we start with a selection of top-level landmarks that are representative for the ROI (obtained using our bottom-up mapping, since standard HSNE does not provide such correspondence) and drill-down twice to obtain the same level of detail. The resulting embedding contains 50,116 landmarks — almost as many as pixel in the viewport. This embedding represents larger image regions outside the current ROI than our embedding and fails at preserving coherent landmark positions between embedding updates. Our approach requires fewer interaction to yield a more detailed and ROI-specific embedding.

As a next exploration step, we are interested if the observed patterns can also be found in close vicinity of the current ROI. In Figure 5.4c we show the result of a *pan* to the partially overlapping ROI Ⓘ. Using our coupled interaction, we stay on the same abstraction level 2, now with 9,533 representative landmarks. Since the data points covered by the current ROI overlap substantially with the previous one, many representative landmarks remain the same during the embedding update. Landmark clusters that are representing the same image regions also remain in similar embedding regions, preserving coherence between the updates. In contrast, standard HSNE interactions would require drill-down recomputations, leading to inconsistencies between embedding updates (Figure 5.4g).

To see how the method behaves when moving to an unrelated region, we *pan* to ROI Ⓘ (Figure 5.4d). The pan triggers with a roll-up in the embedding hierarchy to level 3 with 3,979 representative landmarks (level 2 had 16,404 landmarks for this viewport). Covering a very different set of data points, the new ROI is represented by many landmarks that have not been in the previous embedding, as visible by the abundance of randomly initialized landmarks. Since the number of representative landmarks in level 2 and 3 are almost equally distant to the visual target, a user might decide to request more detail by manually *drilling down* the entire ROI (**T3**). Figure 5 in Supplement S2 shows the result of this operation. We can observe that the refined level-3 embedding follows the layout of the level-2 embedding, indicating that the level 3 embedding was already capturing most of the variation.

5.6. Limitations

Our method currently restricts all embedding landmarks $\mathcal{D}_E$ to be from the same hierarchy level k. For scenarios in which a ROI contains regions of varying homogeneity, an embedding view that reflects this with multiple levels of detail could be helpful. To address this, and further following the terminology of Elmqvist [124], our set of image interactions might be extended with a new *local-aggregation* interaction. For this Focus+Context inspired action the user would define a focus selection $\mathcal{K}_f^0 \subset \mathcal{K}^0$, a subset of the current viewport. We would then need to find sets of focus $\mathcal{F} = \mathcal{K}_f^{k-1}$ and context landmarks $\mathcal{C} = \mathcal{K}^k$ such that the embedding $\mathcal{D}_E = \mathcal{C} \cup \mathcal{F}$ represents the current ROI well. However,

even though Höllt et al. [104] already proposed such a Focus+Context framework for hierarchical embeddings, it is not straightforward to extend to our setting. Contiguous selections in the embedding space ensure that the represented data points on lower levels are well-connected. As discussed in Section 5.4.3, these selections can be assumed when interacting with the embedding rather than the image. Selections of spatially connected regions in image space typically do not correspond to similarly confined regions in the embedding hierarchy since data points with vastly different neighbours might be spatially close. Rather, linked landmarks might be scattered throughout the embedding. Landmarks with few HSNE transition matrix connections experience lower attracting forces during the embedding optimization and are pushed to the periphery by dominating repulsive forces.
Further, our re-initialization scheme in Section 5.4.4 does not guarantee consistency between embeddings when moving back and forth between two ROI without any overlap in their corresponding $\mathcal{D}_E$. We consider approaching these limitations future work.

5.7. Conclusion

In this chapter we have presented an interaction strategy for the coupled exploration of high-dimensional images with hierarchical embeddings. Our method couples image navigation interactions (zooming and panning) with an embedding space that represents the attribute data. We showed how an *"overview-first, details-on-demand"* approach is well-suited for driving embedding detail based on user interactions with the image space.
Future lines of research may focus on extending the embedding view from a single to multiple levels of detail by showing landmarks from several hierarchy levels Another possible direction would be exploring ways of providing a user with guidance on where to zoom and pan to.
We showed the potential benefits of coupling image navigation with simultaneously updating embeddings for exploration analysis workflows. However, so far the construction of the embedding hierarchy has been agnostic of the image itself. In the next chapter we will explore superpixels as a way of including information about the image layout into data hierarchies for hierarchical embeddings.

Manifold-Preserving Superpixel Hierarchies

6.

This chapter is based on the paper "Manifold-Preserving Superpixel Hierarchies for Exploration of High-Dimensional Images" [under review].

In the previous chapters, we discussed the coupling image interactions and of hierarchical DR and incorporated spatial arrangement of image data into DR techniques. In this chapter we further examine how hierarchical data structures may support the exploration of high-dimension images. Specifically, we build on an image hierarchy compatible with existing DR techniques that maintains fidelity to both the spatial layout of the images and the high-dimensional attribute space.

6.1. Introduction

Hierarchical DR techniques [51–53] tackle the scaling problem of single-level embedding methods. They reduce the number of embedded points by creating hierarchical data structures. Upper levels in these hierarchies represent the original data at increased rates of abstractions. Now, instead of projecting the original data, hierarchical DR techniques embed (subsets of) the abstracted points on a given hierarchy level. With each added abstraction level the DR techniques embed fewer points, thus reducing scaling issues. However, abstracted points often represent a wide spread of pixels, i.e., they do not necessarily correspond to connected spatial regions. Figure 6.1 illustrates this shortcoming of data-space-based hierarchical DR techniques applied to images: Without a structural link between data hierarchy and image, exploring ROI in the image space using the embedding becomes complicated. Points in the embedding correspond to widely spread pixels and pixels might be represented by several embedding points at the same time.

Previous approaches tackle the lacking awareness of spatial arrangement in embeddings of images by incorporating spatial neighborhood information into the similarity measure which is used for DR, see chapter 5. Even so, if one were to use a spatially-informed distance measure in a standard hierarchical DR techniques, the problem of non-spatially-continuous abstractions would remain. Other approaches link the image and data space exploration by coupling the level of detail (LoD) in a hierarchical embedding based on the LoD presented in image space, guided by zooming and panning [109]. Still, each image-coupled LoD embedding tends to represent large regions outside the ROI. To fully enable a coupled exploration of image and attribute space, we need a hierarchical data representation that couples these spaces both in its creation and respects their link during interaction.

Our approach for such an image hierarchy comprises superpixels, i.e., non-uniform, continuous pixel groups which segment the image. In this context, a superpixel hierarchy is a sequence of superpixel segmentations with increasingly large and consequently fewer superpixels. Segments in higher hierarchy levels result from merging lower-level segments. When dealing with classic color images, superpixel segmentation is typically based on a similarity measure between perceptual features like distances in perceptually uniform color spaces. We propose to merge segments using similarity measures based on the high-dimensional manifold structure of the attribute data that is associated with each pixel. Thereby, and in contrast to the data hierarchies used in other hierarchical DR techniques, our superpixel hierarchy is informed not only by the high-dimensional attributes but also the spatial component of image data.

In this chapter, we connect the exploration of image and attribute space of high-dimensional images by incorporating spatial information into a hierarchical embedding. To accomplish this, we propose:

- a superpixel hierarchy for high-dimensional images, based on
- a manifold-aware similarity measure between superpixels, and
- integrating this similarity in dimensionality reduction methods.

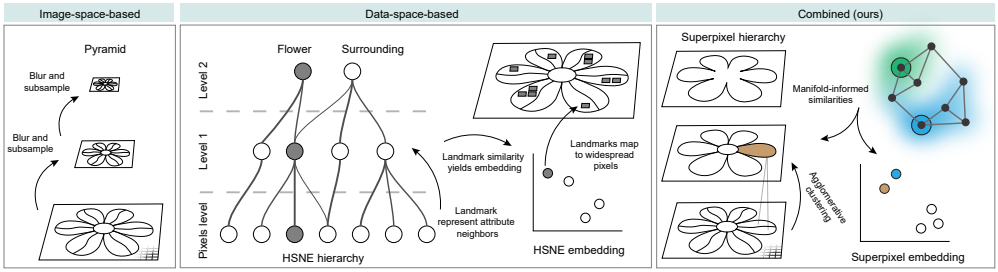


Figure 6.1. ***Image Hierarchies:*** *Classical image-space-based hierarchies, like image pyramids, progressively blur and subsample the image (left). Data-space-based hierarchies, as used in hierarchical DR methods (center), ignore the image space entirely and only aim to preserve manifold structure of the attribute space. A pixel, as highlighted in the hierarchy, might actually be represented by multiple landmarks in abstraction levels. And, in turn, a landmark can represent scattered pixels. Combined, this complicates an image-based exploration of the high-dimensional space. Our superpixel hierarchy (right) combines both image layout and attribute space manifold structure.*

6.2. Related Work

Relevant work on hierarchical dimensionality reduction is discussed in Chapter 3.

In the following, we aim to report the work most relevant to this chapter, namely, superpixels and node embedding methods.

6.2.1. Superpixels

There exist a wide range of superpixel methods and their full discussion is out of scope of this chapter; instead, we refer to more extensive reviews [127–129]. Typically, superpixel methods work with three-dimensional color images, though here we will focus on methods most pertinent to our work, which are those specific to high-dimensional images.

Hyperspectral imaging (HSI) is the main high-dimensional image domain which uses superpixel methods, usually aimed at downstream tasks like pixel classification, endmember detection and hyperspectral unmixing. Using the Felzenszwalb and Huttenlocher's [130] image segmentation method Thompson et al. [131] describe a superpixel segmentation methods for hyperspectral images with the goal of endmember detection. They perform agglomerative clustering to define superpixels and merge clusters based on spectral distances of pixels within and between superpixels. While this approach does create a superpixel hierarchy, it does not fully preserve the manifold structure of the underlying spectral data. Also, while one technically could use their inter-superpixel distance as a basis for an embedding of a hierarchy level, it does not capture the similarity of all pixels contained in the respective superpixels well, since it only incorporates distances of pixels along superpixel boundaries. Similarly adapting an originally color-focused method, Xu et al. [132] follow the k-means based SLIC [133] and substitute the color distance for a spectral distance. Barbato et al. [134] also adapt SLIC, augmenting it with both hyperspectral distance and an additional spectral clustering preprocessing step.

Another popular superpixel method in HSI are entropy rate superpixel [135]. They employ an entropy-based objective function for superpixel segmentation that combines the entropy rate of a random walk on a graph with a balancing term to promote compact, homogeneous, and similarly sized clusters. The method constructs a graph in image space by iteratively adding edges between pixels that maximize random walk entropy within the new superpixel. Several authors extend this method to HSI: whereas Tang et al. [136] advocate using spectral distances to set up the random walk transition probabilities, others use the first or first three principal components to create a false-color image and apply color distances [137]. The entropy rate superpixel method also creates a superpixel hierarchy, but does not explicitly define distances or similarities between superpixels, which we aim to do in order to embed each hierarchy level.

Grady [138] describes another method that uses random walks in image space to define merge criteria for superpixels. Given specific seed pixels, a pixel is assigned to a superpixel based on which seed pixel is visited most often by random walks started on the pixel itself. Follow-up papers introduce additional shape constraints and allow self-loops in their walks, as well as data-based seed selection techniques [139, 140]. However, these methods require a fixed number of seed points, and thereby superpixels, and do not define similarities between superpixels either.
Agglomerative methods all implicitly create superpixel hierarchies but usually chose a single hierarchy level as the resulting image segmentation. In contrast, Wei et al. [141] explicitly compute all levels of a superpixel hierarchy based on the Borůvka's algorithm for finding a hierarchy of minimum spanning trees applied to images. Yan et al. [142] extend this method for asymmetrical distance measures between pixels. We base our superpixel hierarchy on the adaption of the Borůvka's algorithm presented by Wei et al. [141]. However, they ignore the manifold structure of high-dimensional input data. To inform the superpixel distance and superpixel merging with the data manifold structure we also employ random walks, but use them in the pixel attribute space instead of the image space as all methods above.

6.2.2. Node embedding methods

Embedding nodes (vertices) from a graph structure is a problem related to dimensionality reduction methods, which are often used for visualizing and exploring data. Khosla et al. [63] compare several such node embedding methods, which compute a low-dimensional feature vector for each vertex in a graph; in contrast to the above dimensionality reduction techniques, these methods are typically designed for the downstream tasks of multi-label classification and link prediction. Variations of random walks are often used to define vertex features, e.g. the frequency of vertex visits, here as well. In *node2vec* [64], for example, random walks are biased by adjusting the transition probability for walking backwards and walking to a vertex that is not connected to the previous vertex. *tsNET* [65] follows a different approach: they take geodesic distances between vertices in graphs as input for the similarities as defined in t-SNE and map them to 2D based on a modified cost function. Earlier, *Isomap* [26] introduced the idea of using geodesic distances into DR methods that build on multidimensional scaling. In fact, random walks have been used to estimate geodesic distances in various settings [67, 68]. However, both Lee and Verleysen [66] as well as Lafon and Lee [56] discuss that using shortest-path-based similarities (i.e. using geodesic distances) can be susceptible to creating shortcuts that jeopardize the

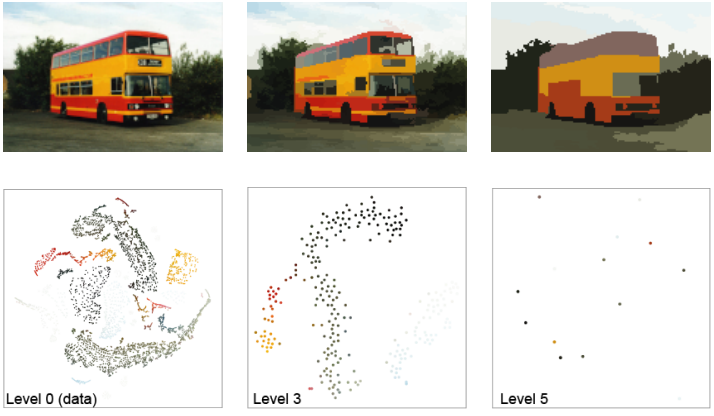


Figure 6.2. *RGB image (top left) and 2 levels of abstraction (3 and 5). Superpixels are recolored with the average color of all pixels they contain. Numbers of components: 15300 (150x102 pixels), 271, 14.*

representation of the underlying data manifold whereas random-walk-based similarities seem to be more robust.
Embeddings are certainly not the only way of visualizing and exploring high-dimensional image data in a hierarchical manner. For example, Jalba et al. [143] use watershed based supervoxels for exploration of diffusion tensor images (DTI). They set up a linked view of the segmented DTI and the watershed tree that represents the multidimensional data hierarchically. Their method is similar to ours in that it also creates a hierarchical superpixel representation but uses domain specific superpixel merging criteria that involve computing means in the data space, which is both difficult to generalize for non-DTI data and ignores the data manifold structure.

6.3. Superpixel Hierarchy

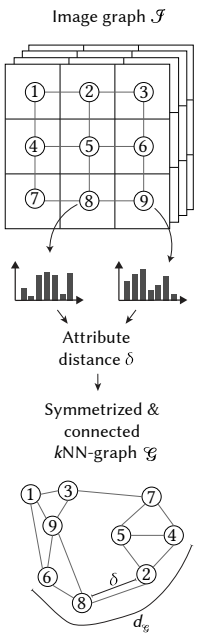
See chapter 2 for our image indexing notation.

Superpixels segment the image domain irregularly. We define a superpixel segmentation $\mathcal{S}$ as a partition of $\mathcal{I}$ with $s = |\mathcal{S}|$ disjoint components. Each superpixel $\mathcal{s}_p \in \mathcal{S}$ is associated with a connected subgraph of $\mathcal{I}$. A **superpixel hierarchy** of L levels is an ordered set of superpixel segmentations $\{\mathcal{S}^0, \ldots, \mathcal{S}^L\}$ with $\mathcal{S}^0$ containing all vertices from $\mathcal{I}$ as individual components and the property that each superpixel of $\mathcal{S}^{l+1}$ is obtained by merging one or more superpixels from $\mathcal{S}^l$. We read $\mathcal{s}_p^l$ as the p-th superpixel on level l. Each superpixel in this hierarchy can be seen as a set of superpixels from a lower level, down to image pixels in the lowest level. The number of image pixels contained in a superpixel is notated as $|\mathcal{s}_p^l|$.

Merging might not always be possible, see the merging criterion in Section 6.5.

Such a superpixel hierarchy can be constructed by merging components in the image graph $\mathcal{I}$ following Borůvka's algorithm for finding a minimum spanning tree (MST) [141]. Starting with each pixel as a component/tree, spatially neighboring trees are iteratively merged based on a distance criterion. We use the Borůvka method similarly but where Borůvka's algorithm connects trees, we merge superpixels.

6.4. Preliminary Considerations



A superpixel hierarchy is a hierarchy both in image space and pixel attribute space: spatially adjacent pixels are merged based on similarities of their attributes or features, e.g., color intensity or texture respectively. Rather than color images, we are interested in high-dimensional images. High-dimensional data often exhibits an underlying manifold structure which is of high interest for data exploration. Using Euclidean distance or measures defined for color images would ignore this manifold structure. Therefore, geodesic distances, approximations thereof, or local measures are preferred. We will construct the superpixel hierarchy with a manifold-aware similarity measure. To explore the high-dimensional image both in image and pixel attribute space consistently, we later use the same measure to embed each superpixel on a given hierarchy level.
For capturing local manifold structure in high-dimensional data a k-nearest neighbor (kNN) graph in $\mathbb{R}^C$ is commonly used. We can compute a kNN-graph based on distances between the high-dimensional attributes, e.g., using squared Euclidean distances $\delta(\mathbf{i}, \mathbf{j}) = \|\mathbf{a_i} - \mathbf{a_j}\|_2^2$. A kNN-graph is not guaranteed to be made up of a single connected component, but we want to enforce the graph to be connected, since it allows us to define geodesic distances between all vertex pairs. Later in Section 6.5, we require connectedness as it ensures the existence and uniqueness of a stationary distribution of random walks. We convert the directed kNN-graph into a symmetrized and connected kNN-graph $\mathcal{G}$ following the scheme laid out in Supplemental Material SC2. We index vertices in this attribute-based graph $\mathcal{G}$ with $\mathbf{i}$, like in the distinct image graph $\mathcal{I}$, since the $\mathbf{i}$-th vertex (associated with $\mathbf{a_i}$) in $\mathcal{G}$ and $v_\mathbf{i}$ in $\mathcal{I}$ refer to the same data point, albeit with respect to the different spaces. The set of vertices that are

directly connected to a vertex $\mathbf{i}$ via one edge was previously referred to as its attribute neighborhood $\mathcal{N}_{\mathbf{i}}^{C,k}$ but for notational simplicity we will use $\mathcal{N}(\mathbf{i})$ in this chapter.
Once the symmetrized and connected kNN-graph $\mathcal{G}$ has been build, a straightforward manifold-preserving distance $d_{\mathcal{G}}(\mathbf{i}, \mathbf{j})$ between two data points is their geodesic distance, i.e., the accumulated distance along the shortest-paths between their vertices in $\mathcal{G}$. Since on the data level every component $\mathcal{s}_p^0 \in \mathcal{S}^0$ is a single pixel and corresponds directly to a single data point we can immediately apply the geodesic distance to compare them. On higher levels though, we need to broaden the definition of this measure since there we are associating sets of pixels/data points. One successfully applied measure used for comparing sets of points is the Hausdorff distance:

$$d_H(\mathcal{s}_p, \mathcal{s}_q) = \max\left\{\max_{\mathbf{i}\in\mathcal{s}_p}\left(\min_{\mathbf{j}\in\mathcal{s}_q} d(\mathbf{i},\mathbf{j})\right),\ \max_{\mathbf{j}\in\mathcal{s}_q}\left(\min_{\mathbf{i}\in\mathcal{s}_p} d(\mathbf{j},\mathbf{i})\right)\right\} \tag{6.1}$$

with $d(\mathbf{i}, \mathbf{j}) = d_{\mathcal{G}}(\mathbf{i}, \mathbf{j})$. That is, we capture the worst-case mismatch between the sets by finding the maximum distance from any pixel in one superpixel to any pixel in the other. When both superpixel just contain a single pixel, d_H simplifies to the geodesic distance $d_{\mathcal{G}}$ between them.
While this approach can yield reasonable results with respect to both the superpixel hierarchy and respective embeddings (see Figure 6.2), its intrinsically expensive computation renders it undesirable in practice. For high abstraction levels it would involve computing all geodesic distance combinations between sets of thousands of pixels, which becomes computationally taxing fast. One way to address this could be taking a number, e.g., 100, sample pixels from each superpixel and compute the Hausdorff distance between these representative subsets. However, geodesic distance based approaches like this may be jeopardizing manifold representation by over-valuing shortcuts in the graph which are not necessarily essential for the graph structure [56, 66]. Instead, we propose a more robust similarity measure based on features obtained from random walks on the data graph.

6.5. Method

Previous work on creating manifold-preserving hierarchies of high-dimensional data [52, 53] used random walks to compute features for each data point, and employed a distance between those features as a proxy for exact geodesic distances to define manifold-preserving similarities between landmarks, i.e., vertices of the graph that represents a neighborhood on a more detailed level of the hierarchy. We follow a similar approach, but do not use landmarks for data-abstraction. Instead, we want to define a similarity between superpixels that can be used both as a merging criterion during the superpixel hierarchy creation with Borůvka's algorithm as discussed above, and as the basis for the embedding at each hierarchy level.

6.5.1. Constructing the image hierarchy

To compute embeddings of superpixels on each hierarchy levels we need to define transition matrices T^l on each level l. We propose to define the data level transitions T^0 based on random walks in attribute space and compute T^l for abstraction levels $l \geq 0$ by merging rows and columns of lower level transition matrices.
As in the geodesic approach before, we begin on the data level with an undirected and connected graph $\mathcal{G}$. For each vertex in $\mathcal{G}$, we start ω walks with λ steps governed by the edge weights of $\mathcal{G}$. On the data level $l = 0$, we use a Gaussian

See Section 2.4 for breakdown of t-SNE.

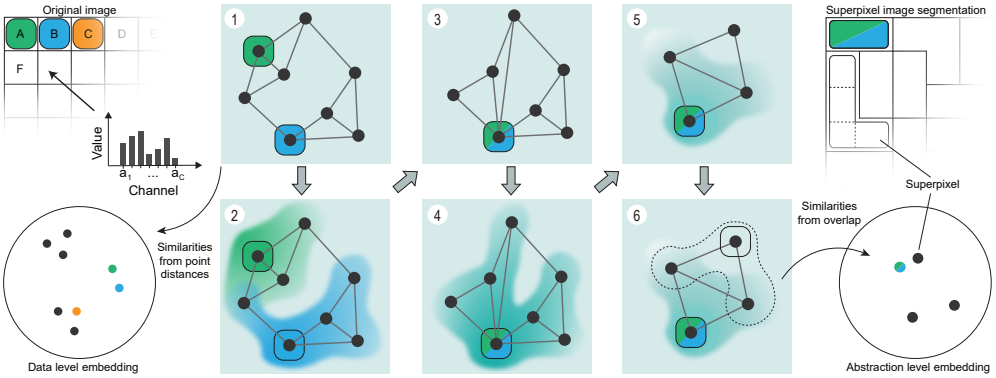


Figure 6.3. ***Method overview:*** *Each image pixel is associated with a high-dimensional attribute vector (top left). (1) We compute a nearest neighbor graph $\mathcal{G}$, whose vertices correspond to pixels and edges are based on attribute similarities. The data level embedding is computed from this neighbor graph, like in, e.g., t-SNE. (2) We compute a feature vector per vertex, describing the local graph structure, using random walks. (3) For the next abstraction level, vertices in the attribute graph $\mathcal{G}$ are merged with the spatially (in the image layout, i.e., $\mathcal{F}$) neighboring vertex of largest feature similarity. (4) The new vertex retains all outgoing connections of the merged vertices. All vertex features are added and re-normalized. (5) This is repeated for each vertex. (6) Similarities between merged vertices are used for creating embeddings where each point corresponds to one superpixel (right).*

kernel to transform the edge weight (data distances) into transition probabilities for the walks:

$$p^0_{\mathbf{j}|\mathbf{i}} = \frac{\exp\left(-\delta(\mathbf{i},\mathbf{j})/\sigma_\mathbf{i}\right)}{\sum_k \exp\left(-\delta(\mathbf{i},\mathbf{k})/\sigma_\mathbf{i}\right)} \quad \text{with } \mathbf{j},\mathbf{k} \in \mathcal{N}(\mathbf{i}) \tag{6.2}$$

$\sigma_\mathbf{i}$ is chosen such that the conditional probability $P_\mathbf{i}$ equals the perplexity $u = |\mathcal{N}(\mathbf{i})|/3$ as in t-SNE [110]. Small data distances are assigned a high transition probability and large distances a small one. Notably, a random walk is confined to the edges of the data graph $\mathcal{G}$. The self-step probability $p^0_{\mathbf{i}|\mathbf{i}}$ is zero, since we do not include a vertex into its own neighborhood. A walk may return to its starting point. We adjust the perplexity (and thereby the number of nearest neighbors) on a level following Kobak et al. [113] to $u = \max(10,\ \min(n/100,\ 100))$.

The random walks populate a sparse matrix T. A random walk starting at vertex $\mathbf{i}$ in the attribute graph $\mathcal{G}$ populates the matrix row $T(\mathbf{i},-)$, an unnormalized feature for the vertex. At each step we increase the feature $T(\mathbf{i},\mathbf{j})$ by a weight, where $\mathbf{j}$ is the vertex the current step landed on. The weight decreases exponentially with the number of already taken steps starting with a weight of 1. Finally, we normalize each row $T(\mathbf{i},-)$ such that all entries sum to 1 yielding a valid transition matrix. Figure 6.3 illustrates the coverage of the random walks, i.e., the vertex feature, for two vertices in green and blue.

The transition probability vector $T(\mathbf{i},-)$ is a descriptor (feature) of the high-dimensional neighborhood of vertex $\mathbf{i}$ in the attribute graph $\mathcal{G}$. Now, to quantify the similarity between two vertices, and later superpixels, we measure the similarities of their features, which are probability distributions after normalization. Such a measure is given by the Bhattacharyya coefficient BC which measures the overlap of two distributions, in our case:

$$BC^l(\mathcal{s}^l_r, \mathcal{s}^l_s) = \sum_{\mathcal{s}^l_t \in \mathcal{S}^l} \sqrt{T^l(\mathcal{s}^l_r, \mathcal{s}^l_t)\, T^l(\mathcal{s}^l_s, \mathcal{s}^l_t)} \tag{6.3}$$

The Bhattacharyya coefficient always lies within $[0,1]$ where 0 indicates no overlap/similarity and 1 identical distributions. Note that on the data level $l = 0$, a superpixel $\mathcal{s}^0_i$ corresponds to a pixel at $\mathbf{i} = (x,y)_i$, yielding $BC^0(\mathbf{i},\mathbf{j}) = \sum_{\mathbf{k}\in\mathcal{S}^0} \sqrt{T^0(\mathbf{i},\mathbf{k})\, T^0(\mathbf{j},\mathbf{k})}$.

Iterating over all superpixels, the Borůvka superpixel algorithm merges each superpixel with their spatially neighboring superpixel in $\mathcal{I}$ with the smallest distance in $\mathcal{G}$. Pixels are either 4- or 8-connected in $\mathcal{I}$, but on higher abstraction levels superpixel neighbor-relation in $\mathcal{I}$ can be less regular. We can use the Bhattacharyya distance $d_{Bhat} = -\ln BC$ as a distance measure between the superpixel features. However, instead, we can skip the double inversion (negation and ln of a number smaller than 1) and use the largest BC as a merge criteria directly, which is equivalent to merging with the smallest d_{Bhat} due to their strictly monotonic-inverse relation. In contrast to the distance measures presented for color image superpixel hierarchy from [141] or the geodesic approach from Section 6.4, the BC similarity from Equation 6.3 may result in all-zero similarities between a superpixel and its spatial neighbors. This is because the random walks purposefully do not cover the entire data. In this case, we do not merge this superpixel with any neighbor. (Unlike the Borůvka superpixel hierarchy algorithm [141] which we generally follow for superpixel merging.) However, the superpixel may still be merged later with newly-formed neighbors on higher levels.
While we have a general superpixel merging criterion in Equation 6.3, we still need a method to generate a transition matrix T^l on an abstraction level l. On each level, we want the superpixel descriptors to preserve the data manifold. Therefore, to obtain T^l on hierarchy level l, we merge corresponding rows and columns of the transition matrix T^{l-1}, following the same merging-pattern of superpixel components from $\mathcal{S}^{l-1}$ to $\mathcal{S}^l$ and re-normalize. Figure 6.3 (3-5) illustrates the vertex and feature merging. The merged features of the meta vertices now describe the local graph structure of all pixels contained in the corresponding superpixel.

6.5.2. Computing embeddings on each abstraction level

On the data level, we use $p^0_{\mathbf{ij}}$ directly for embedding, as they are equivalent to the high-dimensional similarities used in t-SNE. On higher abstraction levels, we convert the superpixel similarity measure BC into analogous probability distributions $p^l_{\vartheta_s|\vartheta_r}$ by applying a Gaussian kernel to the Bhattacharyya distance d_{Bhat}. For better readability we omit superscripts l for the remainder of this section, all equations are defined per level l:

$$p_{\vartheta_s|\vartheta_r} = \frac{\exp\left(-d_{Bhat}(\vartheta_r, \vartheta_s)/\sigma_{\vartheta_s}\right)}{\sum_k \exp\left(-d_{Bhat}(\vartheta_r, \vartheta_k)/\sigma_{\vartheta_s}\right)} \quad \text{with } \vartheta_s, \vartheta_k \in \mathcal{N}(\vartheta_r) \tag{6.4}$$

where ϑ_r and ϑ_s refer to superpixels on level l and $\mathcal{N}(\vartheta_r)$ is the set of superpixel directly connected to ϑ_r. σ_{ϑ_s} is determined as in Equation 6.2.

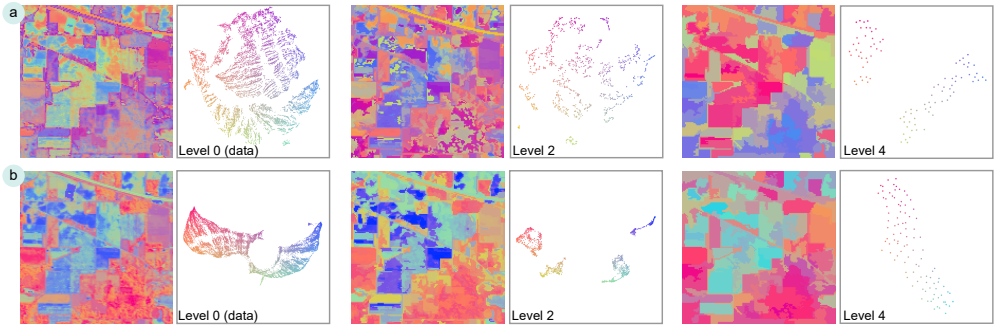


Figure 6.4. ***Indian Pines:*** *Embeddings and image space recolorings (see Figure 6.5f) for (a) t-SNE and (b) UMAP probabilities. The full hierarchy is shown in Figure C.5 in the Supplemental Material SC6.*

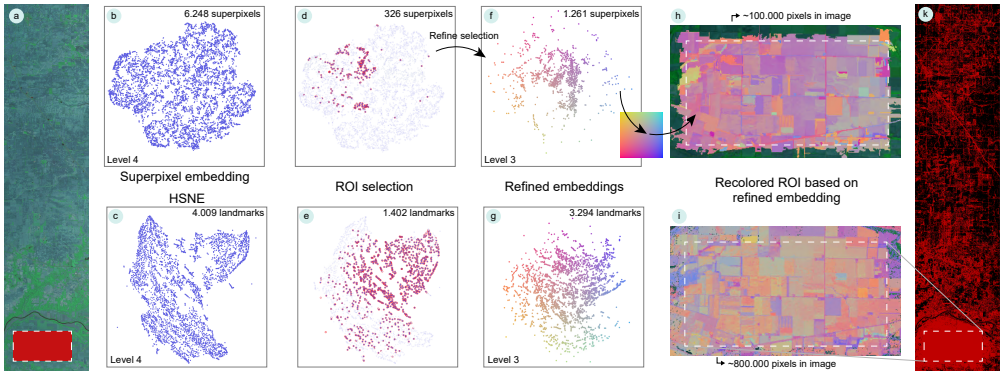


Figure 6.5. ***Indian Pines Exploration:*** *(a) a false color image based on data channels 20 (587 nm, red), 76 (1090 nm, green) and 130 (1591 nm, blue) with a ROI marked in red. (b) and (c) show the 4th abstraction level embedding of our superpixel embedding and HSNE, respectively; (d) and (e) highlight the superpixels and landmarks which correspond to the ROI. (f) and (g) show refined embeddings of the highlighted subsets on a lower abstraction level. (h) and (i) recolor the a cutout of the image based on the refined embeddings using an overlaid 2D colormap Finally, (k) indicates the pixels in the full images which are represented by the HSNE refinement, whereas our superpixel refinement extends only slightly around the ROI.*

The Bhattacharyya coefficient BC in $d_{Bhat}(\vartheta_r, \vartheta_s)$ is equivalent to the dot product of two rows from T after taking the square-root of each element. Further, the random-walk generated superpixel features T are a very sparse matrix which allows for efficient computation of a dissimilarity matrix $D = [d_{Bhat}(\vartheta_r, \vartheta_s)]$ on each level instead of computing each entry individually, as:

$$D = -\ln\left(\sqrt{T}\,\sqrt{T}^{\top}\right) \tag{6.5}$$

where $\sqrt{T} = \left[\sqrt{T(\vartheta_r, \vartheta_t)}\right]$ takes the element-wise square root of T and ln takes the element-wise natural log. D is inherently symmetric since the Bhattacharyya coefficient is symmetric as well, reducing the computational complexity further.
For the embedding layout computation we use the symmetrized conditional probabilities $P = \{p_{\mathbf{ij}}\}$ with $p_{\mathbf{ij}} = (p_{\mathbf{i}|\mathbf{j}} + p_{\mathbf{j}|\mathbf{i}})/2$ as in t-SNE [110].

Embedding with UMAP. Above, we used t-SNE as our basis for defining joint probabilities $p_{\mathbf{ij}}$ but neither the hierarchy creation nor embedding layout are limited to this choice. For example, we might choose the probability definition from UMAP:

$$p^0_{\mathbf{j}|\mathbf{i}} = \exp\left(-\frac{\delta(\mathbf{i},\mathbf{j}) - \rho_{\mathbf{i}}}{\sigma_{\mathbf{i}}}\right) \quad \text{with } \mathbf{j} \in \mathcal{N}(\mathbf{i}) \tag{6.6}$$

where $\rho_{\mathbf{i}}$ is the distance of $\mathbf{i}$ to its closest neighbor and $\sigma_{\mathbf{i}}$ is set such that $\sum_{\mathbf{i}} p_{\mathbf{j}|\mathbf{i}} = \log_2 k$ as in [28]. On the abstraction levels, the probability distributions $p_{\vartheta_s|\vartheta_r}$ are defined analogously to Equation 6.5 using the Bhattacharyya distance. Here, we perform the conditional probability symmetrization with $p_{\mathbf{ij}} = p_{\mathbf{i}|\mathbf{j}} + p_{\mathbf{j}|\mathbf{i}} - p_{\mathbf{i}|\mathbf{j}} p_{\mathbf{j}|\mathbf{i}}$. The rest of the described algorithm remains unchanged. Figs. 6.4a and 6.4b shows two abstraction levels and embeddings for the Indian Pines dataset [10] with t-SNE and UMAP probabilities respectively.

6.5.3. Subset embedding refinement

To support the exploration of ROIs in image space, it is essential to be able to request more detail, i.e., an embedding on lower abstraction level. Such a refinement operation starts with a selection of superpixels $\widetilde{S}^l$ on level l, e.g., by manually selecting points in the embedding, or more pertinently, superpixels in the image. We find the set of superpixels $\widetilde{S}^{l-1}$ which represents the same data

points as $\widetilde{\mathcal{S}}^l$. As each superpixel $s^l \in \widetilde{\mathcal{S}}^l$ was obtained by merging superpixels from level $l-1$, the refined superpixels can be looked up directly in the hierarchy. From there, we calculate a new probability matrix $\widetilde{P}^{l-1}$ from the symmetrized conditional probabilities P^{l-1} by extracting the submatrix for which all rows and columns correspond to the refined superpixels. The rows in $\widetilde{P}^{l-1}$ are then normalized to sum to 1.
Superpixel selections in the image space can easily lead to situations in which superpixels without any connections in the transition matrix are selected. The force-directed layout in t-SNE will not be able to place these superpixels close to any other, leading to isolated points occupying large peripheral parts of the embedding space. For such cases, we can relax the selection criteria of lower-level superpixels in order to obtain a better connected superpixel. To introduce additional, non-selected superpixels from level $l-1$ we define a threshold $\gamma \in [0,1]$. Any superpixel s_j^{l-1} that is connected to a selected $\widetilde{s}_i^{l-1}$ with a transition value $p_{\mathbf{ij}}^{l-1} > \gamma$ is also added to the refined superpixels.

6.5.4. Implementation

As the computation of the kNN-graph on the data level tends to be increasingly expensive for large data sets, we instead approximate the k-nearest neighbors with HNSW [106] using Faiss [144]. Using an approximated kNN-graph is common practice in virtually all contemporary neighborhood-based dimensionality reduction methods since its first introduction in A-tSNE [103] as this practice comes with small to negligible loss of embedding quality.
The random walks on the data graph with λ steps are a Markov Chain Monte Carlo technique to approximate P^λ. While it is possible to perform the sparse matrix multiplication explicitly, we chose to approximate it with the random walks. This allows us to tweak the number of repeated walks ω and step length λ to adjust the tradeoff between computation speed, and approximation accuracy and memory consumption.
We use the GPGPU approximation [103] of the t-SNE embedding layout and employ the stochastic gradient descent implemented by umappp for the UMAP embedding layout.
An implementation of our method as a standalone library and an interactive tool with coordinated views between image and embedding representation in the ManiVault framework [125] are available on GitHub.

Code available on GitHub in the repository biovault/HDILib

Code available on GitHub in the repository libscran/umappp

SpatialHierarchyLibrary and SpatialHierarchyPlugin available on GitHub in the repositories alxvth/SpatialHierarchyLibrary and alxvth/SpatialHierarchyPlugin

6.6. Validation

In this section, we first present two use cases with real-world data to illustrate the application of our method and, secondly, evaluate the superpixel hierarchy quantitatively.

6.6.1. Use Case: Exploring Hyperspectral Images

Hyperspectral images contain information about a large spectrum of light, in contrast to three color channels in RGB images. Here, we present an example based on the Indian Pines Test Site 3 [10] dataset. The image depicts fields (e.g., corn and soy), forests, roads, rivers and houses from an aerial perspective. It measures $614 \times 2,678 \approx 1.6$M pixels with 200 channels. The pixel resolution is roughly $20\,\text{m} \times 20\,\text{m}$ and the channels contain electromagnetic spectral information from 400 nm to 2400 nm sampled at 10 nm. (We exclude 20 of the original 220 channels due to their low information, as suggested by Gualtieri and Cromp [145].)
When exploring large hyperspectral images, one typically searches for interesting spatial-spectral regions, i.e., a combination of high-dimensional attribute and image layout characteristics. We start from a zoomed-out overview representation of the image and want to focus on a ROI in the lower part of the image

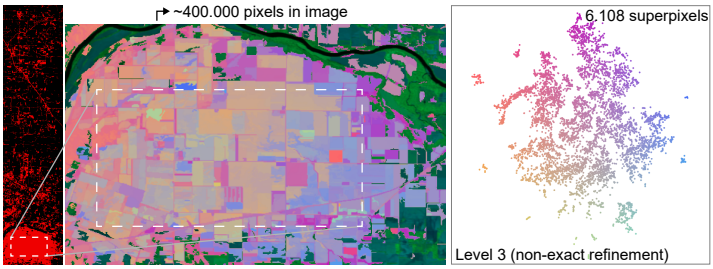


Figure 6.6. ***Non-exact refinement:*** *Instead of refining to the minimal superpixel cover of the pixel ROI in a lower abstraction level we may additional include all superpixels with a transition probability* $p_{\mathbf{ij}}^{l-1} > \gamma$ *to a superpixel* $\mathbf{j}$ *from the cover above a threshold, here* $\gamma = 0.01$.

as indicated in Figure 6.5a. We compute our superpixel hierarchy with $\omega = 50$ random walks with $\lambda = 25$ steps each. The seven superpixel levels alongside their embeddings are shown in Figure C.1 in .

We compare our superpixel embedding exploration with an image-coupled HSNE exploration [109] which proposes to drive the refinement of a hierarchical image embedding from ROIs in image space, but uses a conventional image-agnostic hierarchical data representation. Figs. 6.5b and 6.5c show the fourth abstraction level embedding of our superpixel hierarchy and the HSNE hierarchy respectively. The ROI encompasses ~100, 000 pixels. These are covered by 326 superpixels on the fourth abstraction level and 1, 402 landmarks in the respective HSNE level, see Figs. 6.5d and 6.5e. As the HSNE hierarchy has no notion of the spatial data layout, high-level landmarks likely correspond to data points scattered across the entire image, whereas all pixels in a superpixel are located close together by design. Hence, far fewer points in the superpixel embedding are used to cover the ROI.

Focusing on the ROI, we refine the selected embedding points to explore data similarities on a lower level of abstraction. To connect the embedding with the image representation we use an image recoloring based on the embedding: the embedding positions are converted to colors using a 2D colormap by Bernard et al. [126]; then, the corresponding superpixels are colored accordingly (Figs. 6.5h and 6.5i). An interactive application would not solely rely on color here, but provide a coupled selection mechanism, as in our reference implementation, see Section 6.5.4. The 1, 261 refined superpixels on level 3 cover the same image region as the selected superpixels in level 4 — slightly extending over the ROI due to the superpixels irregular shapes. By contrast, the 3, 294 refined landmarks cover a much larger area, and with ~800, 000 pixels almost half the image (see Figure 6.5j).

Notably, the refined superpixel embedding shows better distinguishable clusters in the center, where the bulk of embedded superpixels are located, but also shows more isolated superpixels in its periphery than the ROI-refined HSNE embedding. This follows from the matrix cutting scheme in Section 6.5.3 to obtain the transition matrix of the refined superpixel subset: the most similar superpixels to those isolated, peripheral superpixel are located outside the ROI and therefore not included in the refined embedding. As a result, some embedding points have few, if any, non-zero transition probabilities within the refined group and are pushed to the outside during the layout optimization.

One possible remedy to this issue is to perform a non-exact refinement. In Figure 6.6 we included additional superpixels with a transition probability $p_{\mathbf{ij}}^{l-1} > 0.01$ to a superpixel $\mathbf{j}$ already contained in the exact refinement. The resulting set of superpixels still covers only half as many image pixels as the refined HSNE landmarks. The superpixels clearly delineate borders of relevant image regions like fields, roads and rivers. Since our superpixel embedding projects a smaller cover of the total image it is able to show more detail than the HSNE embedding at the same abstraction level.

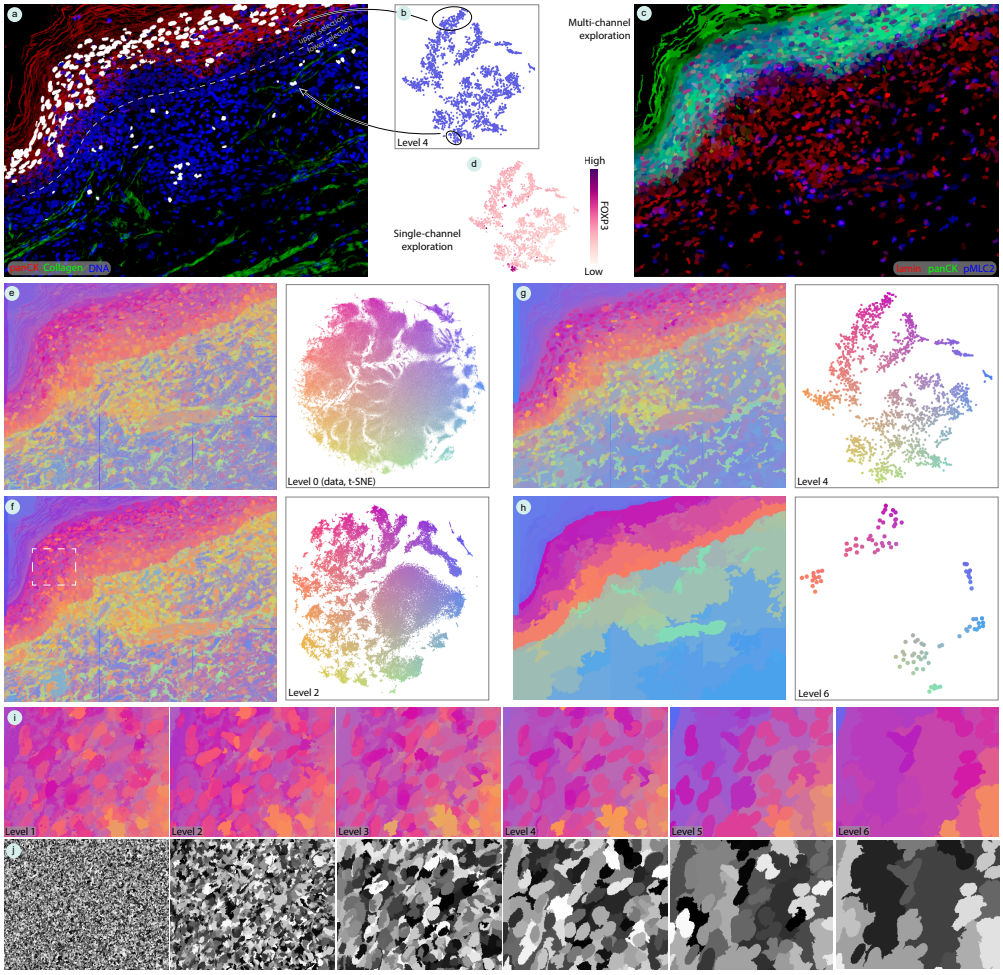


Figure 6.7. ***CyCIF Exploration:*** *(a) Image recolored with two structural markers and DNA. Highlighted in white are two separate selections, corresponding to two clusters in the fourth abstraction level superpixel embedding (b, d). (c) Recoloring based on three markers of interest, indicating a region with high intensity in multiple markers. (e-h) Embeddings on several levels of abstraction and corresponding recolored images, using the colormapping from Figure 6.5f. (i, j) Zoom in on a ROI as highlighted in (f), both recolored according to the embedding and with random gray values to better distinguish between superpixels.*

6.6.2. Use Case: Exploring CyCIF Images

Analyzing the function of and interplay between cells in tissue is of major interest in systems biology, e.g., for researching cancer. There exists a range of spatially resolved, multiplexed imaging methods that are commonly used in the domain. For this use case, we focus on cyclic immunofluorescence (CyCIF) [146], an imaging technique that enables the detection of numerous molecules in a piece of tissue. CyCIF generates high-dimensional images, each channel representing the abundance of a specific protein with a spatial resolution in the micrometer range. The characteristic profile of protein expression is used to characterize cells, and the spatial distribution of cells provides cues on interactions between cells as well as between cells and other tissue structures. Both this segmentation of cells and the subsequent analysis of their spatial neighborhood are key concerns in current research.

Here, we apply our superpixel hierarchy embedding exploration to a CyCIF data set on cancerous skin tissue published by Yapp et al. [147] which is also used

For techniques like CyCIF, *multiplexing* means the technique can detect and image many different proteins or biomarkers within the same tissue sample or cell population during a single experimental workflow.

Data available via
biovis.net/2025/biovisChallenge

in the Bio+MedVis Challenge at IEEE VIS 2025. While the original 54-channel 3D CyCIF data contains $10,908 \times 5,508 \times 194$ voxels at a resolution of $0.14\,\mu m$, we focus on a downsampled, maximum-projected subset of $2,000 \times 1,500$ with a pixel resolution of $1.12\,\mu m$. Figure C.2 in the Supplemental Material SC4 indicates the location of the cutout. Figure 6.7a shows a false-color image using markers indicative of tissue structure and general DNA abundance.

We use 27 log-normalized channels (as suggested by the original authors, and listed in the Supplemental Material SC4) as input for our superpixel hierarchy. For more memory efficient computation we use $\omega = 30$ random walks with $\lambda = 10$ steps and set the number of nearest neighbors in the data kNN-graph to $k = 90$. Figure 6.7b shows the fourth abstraction level embedding with 4,104 superpixels. Many superpixels on this level partially or fully match the outline of cells in the image.

In a first, straightforward exploration step, a practitioner might consider a single marker/channel, as cells with a large abundance of this specific marker indicate a certain role in the immune system. Figure 6.7d maps the average `FOXP3` expression of all pixels in a superpixel on the embedding. `FOXP3` indicates regulatory T cells which play a role in the immune response to cancer cells. One cluster in the embedding stands out with all superpixels showing bright pink color indicating high expression of `FOXP3`. The superpixels correspond to multiple cells scattered along the diagonal of the image, indicated as white in Figure 6.7a.

In a second exploration step, a practitioner can search for embedding clusters and image regions with high expression across multiple channels, e.g., `lamin`, `panCK` and `pMLC2`. Figure 6.7c again recolors the image, but now based on the superpixels using the average marker expression of all pixels covered by a superpixel are mapped to RGB color channels. A distinct group of superpixels in the embedding shows high expression of all three markers — and these superpixels in turn correspond largely to cells located in an upper-left stratum of the tissue.

To show the merge sequence of several superpixels across hierarchy levels, Figure 6.7i and Figure 6.7j focus on a small cutout, as indicated in Figure 6.7f. This comparison requires a degree of alignment of embeddings in several hierarchy levels. Kobak et al. [113] argue that initializing t-SNE embeddings of single-cell transcriptomics data with their first two PCA components is beneficial for preserving global structure and reproducibility. We apply this approach to the superpixel embeddings by computing the first two principal components based on the average expression of proteins within the superpixels. Figs. 6.7e, 6.7f, 6.7g and 6.7h show multiple abstraction levels, following the same embedding-based recoloring as in Figure 6.5h.

To better visually distinguish individual superpixels, Figure 6.7j assigns random shades of gray to each. On lower abstraction levels, like 1 and 2, superpixels still subdivide meaningful structures in the data. On higher abstraction levels, like 4 and 5, superpixels clearly highlight cell structure.

An in-depth analysis of the segmentation quality of the superpixel hierarchy goes beyond the scope of this work. However, these initial findings are a promising indication of a potential joint segmentation and exploration of single-cell data, which are typically two completely separate workflows.

6.6.3. Quantitative Hierarchy Evaluation

Numerous quantitative metrics for the evaluation of superpixel algorithms exist; Stutz et al. [128] give a comprehensive overview of commonly used metrics. Classical superpixel algorithms are developed for color images and aim to group perceptually similar pixels together. Most algorithms are not straightforward extensible to high-dimensional data sets, as concepts like color, texture or edges do not always have a trivial equivalent in high-dimensional images. This renders a direct comparison between color-superpixel methods and our high-dimensional-superpixel method not directly meaningful. We

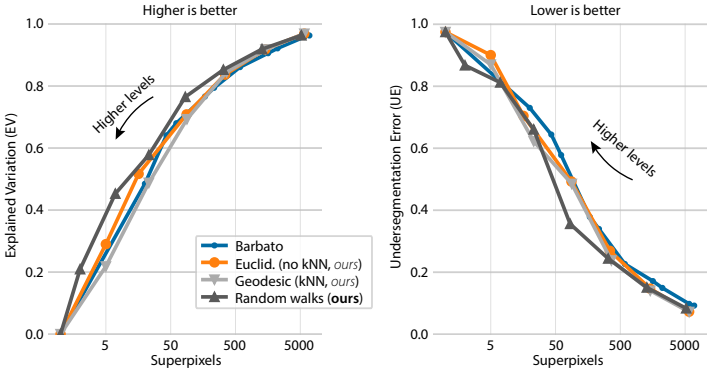


Figure 6.8. ***Explained Variation and Undersegmentation Error:*** *Numerical results are listed in Supplemental Material SC5.*

compare our method with the work by Barbato et al. [134]. It is the only method — to our knowledge — that works directly in the high-dimensional space, does not first project to three channels, and is also open-source, and, thereby, reproducible. Additionally, we compare our random-walk based method to two variations. First, the geodesic distance variation discussed in Section 6.4 that similarly tries to preserve the manifold-structure. Secondly, to a variation using the Euclidean distance between attributes instead of a distance based on the data kNN graph, ignoring the manifold structure, to contrast our approach with one that follows the same superpixel strategy but explicitly does not preserve manifold structure.

A small region of the Indian Pines data set [10] comes with ground truth labels, shown in Figure 6.9a, which we use as a ground truth segmentation for this quantitative evaluation. This region comprises $145 \times 145 \approx 21,000$ pixels with the same channels as described in Section 6.6.1 and contains 16 labels (exclusive background).

For our main method, we use $\omega = 50$ random walks with $\lambda = 25$ steps. For the geodesic variant we do not compute the exact Hausdorff distance from Equation 6.1, but take 100 sample pixels from each superpixel as discussed in Section 6.4. We set the number of nearest neighbors in the data kNN-graph to $k = 300$ for each of our methods variants.

Our methods do not have a direct input parameter that steers the number of output superpixels. Nonetheless, both our main method and its two variations yield a segmentation of around 5000 superpixels at the first abstraction level. Barbato's method steers the number of computed superpixels using an input parameter, `n_clusters`. We evaluate Barbato's method over a range of `n_clusters` that yields superpixel segmentations with similar numbers of superpixels as our abstraction levels. (See Table C.1 in Supplemental Material SC5 for a list of all settings.) We use `m_clust` $= 0.8$ as proposed by the authors as the suggested optimal setting. Additionally, we set the input parameter `m` to 0, as any non-zero value enforced box-shaped superpixels and resulted in worse metrics.

We measured undersegmentation error (UE) and explained variation (EV) for this analysis. UE measures how much a superpixel extends outside an overlapping ground truth segment. If all superpixels completely lie within ground truth segments, the UE is zero. Consequently, larger superpixels on higher abstraction levels will cause an increasing UE. Given a ground-truth segmentation $\mathbb{Q}$ and a superpixel segmentation $\mathbb{S}$, we use:

$$UE(\mathbb{Q}, \mathbb{S}) = \frac{1}{|\mathbb{S}|} \sum_{q_i \in \mathbb{Q}} \sum_{\substack{s_r \in \mathbb{S} \\ s_r \cap q_i \neq \emptyset}} \min\{|s_r \cap q_i|, |s_r \setminus q_i|\}, \tag{6.7}$$

As before, level indices are omitted for readability, i.e. $\mathbb{S}$ may be from any abstraction level l.

EV measures the superpixel quality without reference to a ground truth. It tries to capture how much of the original image's pixel variation is preserved by the

Figure 6.9. ***Indian Pines Ground Truth:*** *(a) Ground Truth segmentation and labels with map overlay. (b) and (c) Superpixels from two segmentations using Barbato's method do not merge bottom-up, i.e. borders shift.*

superpixel representation. We define the mean attribute value for a channel c of a superpixel ϑ_r, denoted $\mu_c(\vartheta_r)$, as the average of the channel attributes of all image pixels contained in the superpixel. The global channel mean is referred to as $\mu_c^{\mathcal{I}}$. As such, we use:

$$EV(\mathcal{S}) = \frac{\sum_{c\in[1,C]} \sum_{\vartheta_r\in\mathcal{S}} |\vartheta_r|(\mu_c(\vartheta_r) - \mu_c^{\mathcal{I}})^2}{\sum_{c\in[1,C} \sum_{i\in[1,n]} (a_{ic} - \mu_c^{\mathcal{I}})^2} \tag{6.8}$$

where a_{ic} is the attribute value in image channel c for data point i.
Figure 6.8 shows the EV and UE for superpixel segmentations for all four compared methods. Additionally, all numerical results are listed in Tables C.1 and C.2 in Supplemental Material SC5. Over the entire range of superpixel segmentation, our random-walks based method shows a slightly better EV than the others. At low abstraction levels, i.e., with many small superpixels, all methods show very similar results, which is expected as they all heavily oversegment the image. Similarly, observing the UE results, the random walks method scores several percentage points better than the others across a wide range of abstraction levels. The UE measurement becomes a bit noisy for low numbers of superpixels as here the ground truth will become strongly undersegmented. Surprisingly, the Euclidean variant consistently outperforms the Geodesic variant by a small margin with respect to both EV and UE.
Notably, Barbato's method does not create a superpixel hierarchy but rather compute each segmentation stand-alone. Hence, superpixels on a higher abstraction level do not precisely comprise superpixels from lower abstraction levels, see Figs. 6.9b and 6.9c.
We do not explicitly evaluate the embedding layout here. Other hierarchical embedding methods like HUMAP [53] or Multiscale PHATE [54] use Denoised embedding manifold preservation (DEMaP) [55] which calculates the Spearman correlation between geodesic distances in the high-dimensional input data and Euclidean distances in the low-dimensional embedding. But, as we do not aim to solely capture distances between high-dimensional points but also include the image layout, measuring how well geodesic distances are preserved in the embedding would not be a relevant indicator for the quality of our method. Moreover, we do not modify the embedding layout algorithms after defining high-dimensional transition probabilities in Equations 6.5 and 6.6, as we use the same gradient descent optimization as t-SNE and UMAP respectively.

6.7. Discussion

The length λ and number ω of random walks on the data kNN graph are free parameters in our methods. The ability of those random walks to create features that are representative of a local graph neighborhood is inherently tied to the graph structure, which is different per data set. We empirically found $\lambda \in [10, 50]$ and $\omega \in [20, 50]$ to yield good results, with smaller settings

requiring less compute time and memory. Other random-walk based embedding methods reason similarly, leaving these parameters free while providing defaults that work well for various scenarios: Node2vec [64] uses walks of length 80 with 10 walks from each vertex in their experiments, and the DeepWalk [148] implementation defaults to 10 walks of length 40. Other methods propose data-driven setting selections: PHATE [55] uses a heuristic measure to set the walk length, derived from the decrease in entropy of the eigenvalues of a diffusion affinity matrix based on random walk results. Kim et al. [149] establishes a connection between the recommended walk length to the concept of "cover time" — the number of steps it takes to visit all vertices of a graph. In general, since random walks on graphs have been thoroughly analyzed in prior research [150], future work might consider how these findings can inform robust data-driven choices for random walk settings.

As visible in the image recolorings in Figs. 6.4a and 6.4b, the global position of merged superpixels in the embedding is not stable across hierarchy level. This happens due to the currently implemented default behavior for initializing embeddings on each level, i.e. random placement of all points. An initialization scheme similar to the one presented in Section 5.4.4 could provide a remedy: An embedding point could be initialized at the average position of all corresponding merged points from the final lower-level embedding.

We compute random walks only on the data kNN graph and essentially coarse-grain the data graph on each abstraction level while merging the random walk vertex features. This is similar to the diffusion and coarse-graining of Multiscale PHATE [54]. Other methods like HSNE [52] and UMAP [53] start new random walks on each new abstraction level for selecting new landmarks and defining new similarities between them. One reason for new walks on each scale is that they might yield better global data structure representation: While a walk on an abstraction level graph might cover a similar portion as a walk on the data level graph, the abstraction vertices now represent a larger percentage of the data graph vertices. We did experiment with new random walks, but did not observe improved results.

The Superpixel Hierarchy method by Wei et al. [141], which we build upon, guarantees that the algorithm stops after $O(\log n)$ abstraction levels, as any given superpixel is *always* merged with at least one other. In contrast, our method is allowed to *not* merge a superpixel, if all Bhattacharyya coefficients BC between a superpixel's feature and its spatial neighbors (see Equation 6.3) is 0, such that the logarithm based d_{Bhat} is undefined. In fact, it is worth considering introducing a threshold here, ensuring that superpixels are only merged if they are considered similar enough. If the random walk length is set high, a superpixel might be merged even though their attribute vertices are located very far apart in the data kNN graph. However, defining such a threshold is not trivial and would need to depend on the data structure.

In this work, we focus on high-dimensional image data, but cutting-edge acquisition methods as showcased in Section 6.6.2 already produce high-dimensional volumetric data. In principle, an extension of our method to volumetric data is straightforward: instead of 2D pixel neighborhoods, one would need to consider 3D voxel neighborhoods, but the general random walk and supervoxel merging procedures remain unchanged.

6.8. Conclusion

We established a superpixel hierarchy for high-dimensional images which preserves the manifold structure of the high-dimensional data. Each level, and subsets thereof, can be embedded using dimensionality reduction methods to facilitate exploration. We inform the hierarchy about high-dimensional manifold using random walks on the data kNN graph, utilizing them in a modified version of Borůvka's algorithm. Our image-informed hierarchy improved on previous *"overview-first, details-on-demand"* approaches for requesting embedding detail based on ROIs in the image space.

An interesting extension of our method is the introduction of multiple levels of abstraction in a single embedding. While such a Focus+Context approach for hierarchical embedding has been applied to exploration in attribute space [104], utilizing it for ROIs in the image space could further aid image exploration.
Our method shows good potential for combining segmentation and exploration steps. Our initial showcase of single cell data exploration in Section 6.6.2 indicates that the superpixel cell structures could be used in subsequent cell-neighborhood analysis steps.
All in all, we propose a superpixel method that we demonstrate to aid with the effective exploration of high-dimensional images.

ManiVault: A Visual Analytics Framework for High-Dimensional Data

7.

So far, we focused our discussion of Visual Analytics (VA) of high-dimensional images on methodological approaches. In this chapter, we shift the focus on how such methods can be made accessible for domain experts. We present a software framework for the exploration of high-dimensional data, ManiVault. The design of the framework emphasizes extensibility, allowing it to support a broad range of high-dimensional data types. However, in line with the central theme of this thesis, we primarily focus on high-dimensional images. Each of the previously presented methods are implemented as extensions in this framework. It is important to note that ManiVault is the result of a collaborative effort, developed jointly with several contributors whose input shaped both its design and implementation.

This chapter is based on the paper "ManiVault: A Flexible and Extensible Visual Analytics Framework for High-Dimensional Data" published in IEEE Transactions on Visualization and Computer Graphics (TVCG) [125].

7.1. Introduction

Combinations of automated analysis and interactive visualizations, i.e., VA [11, 151], have proven to assist well in gaining insight for high-dimensional data. A variety of visual encodings and processing algorithms for high-dimensional data exist. At the same time, specialized application domains require specialized workflows for handling their data and often need to adapt established methods to their use case. Even though these domains encounter different domain-specific questions, they often deal with similar abstract data set types. Additionally, abstracting different domain-specific workflows regularly yields similar goals and user tasks [152, 153] which might be tackled with recurring visual encoding components like heatmaps or analytics methods such as dimensionality reduction. It is time-consuming and wastes development resources to reinvent the wheel by re-implementing, e.g., a linked selection mechanism for multiple coordinated views every time a domain-specific VA solution is needed [47, 154–157]. We developed a visual analytics framework, ManiVault, as a flexible solution for VA software developers, application designers, and practitioners to implement algorithms and visual encodings, prototype workflow-specific tool sets, and perform their data exploration and analysis respectively.

Existing VA systems for exploring general multivariate data do not meet all of these goals. Commercial products like Visplore [158, 159] or Spotfire [160, 161] come with wide feature ranges but are closed-source and not easily extensible. Older open-source frameworks like XmdvTool [162] and GGobi [163] are mostly limited to visual analysis and lack analytics functions. ParaView [164] and Inviwo [165] are capable of displaying multivariate data as well but focus on field data and the representation of spatial structures. Business intelligence solutions like Tableau [166, 167] mostly focus on dashboard creation and chart recommendations. Other fast dashboard prototyping tools, like Keshif [168], provide infrastructure like linked selections of various data visualizations but lack analytics capability. With ManiVault we propose a visual analytics framework for general high-dimensional data that is easily extendable and lets both developers and practitioners re-use algorithmic and visualization building blocks for prototyping and reusing visual analytics systems.

Growing data sizes, both in the number of items and dimensions, increasingly complicate interactive analysis. Progressive visual analytics [169] intends to overcome this issue by continuously providing intermediate results of the current data analysis step. The ability to control the analysis based on continuous feedback is crucial for progressive VA systems [170]. In ManiVault we implement a data-centric and modular framework that facilitates continuous data updates and algorithm steering out of the box. The ManiVault core application manages data sets and plugins, which provide both analysis and visualization functionality. This architecture allows for fast data changes, selection updates, and

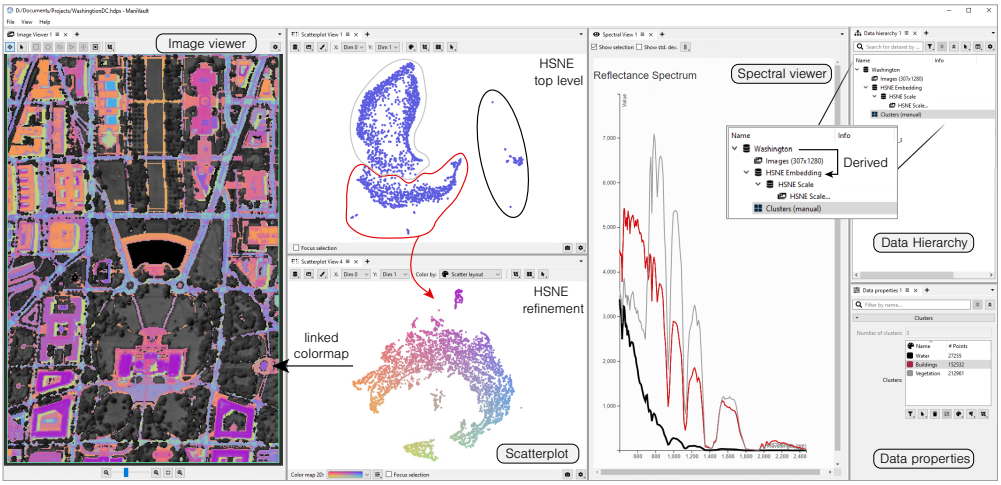


Figure 7.1. ***Example screenshot of ManiVault*** *used for the exploration of a hyperspectral imaging data set.*

overall flexible data exploration. Additionally, since each plugin is agnostic of any other, the system is easy to extend with new data types, visualizations, and analysis algorithms. ManiVault is written in C++, using the Qt framework [171] for cross-platform graphical user interface (GUI) development. OpenGL is used for high-performance rendering (e.g., our scatterplot plugin) but viewer plugins based on lower threshold JavaScript libraries like D3 [172] and Vega-Lite [173] are also possible. ManiVault is open source and can be found at github.com/ManiVaultStudio.

To summarize, in this chapter we describe:

- ManiVault, a modular and extensible visual analytics framework designed for high-dimensional data,
- several functionality extensions in the form of basic data-, viewer-, and analytics plugins, and
- three use cases ranging from plugin development to a practitioner's workflow.

7.2. Related Work

Relevant work on visual analysis of high- and multidimensional is discussed in Chapter 3.

In the following, we aim to report the work most relevant to this chapter, namely, Visual Analytics (VA) systems for multidimensional data and visualization design environments with respect to our framework.

7.2.1. Visual Analysis and Analytics Systems

VA systems for the exploration and analysis of high-dimensional data are well established both in academia and industry [176, 177]. Table 7.1 gives an overview comparison between ManiVault and visual analysis tools that we deem most similar. Most VA systems employ coordinated multiple views [3] with linked selections for data exploration, and we follow this approach with ManiVault as well. Chen et al. [178] discuss common practices and guidelines for the layout of multiple views.
Pioneering visual analysis frameworks for *multidimensional data* include XmdvTool [162], Spotfire [160], GGobi [163] and the InfoVis toolkit [179]. These frameworks mostly focused on displaying data with a variety of visual idioms and enabled exploration with brushing tools and linked selections. XmdvTool

Table 7.1. ***Comparison with other visual analysis tools*** *that are most similar to ManiVault, both open-source and closed-source (commercial, Comm.).*

	ManiVault	XmdvTool [162]	GGobi [163]	Visplore [174]	Tableau [167]	ParaView [164]	Inviwo [165]
Focus on high-dim. data	•	•	•	•	•	—	—
Focus on field data	—	—	—	—	—	•	•
Extensible	•	• [a]	•	—	—	•	•
Visual Analytics	•	•	• [b]	•	•	• [c]	— [d]
Progressive Analytics	•	—	• [b]	•	—	— [d]	— [d]
VA system authoring	•	—	—	—	• [d]	• [e]	—
Active development	•	—	—	•	•	•	•
License	LGPL-3	PD	EPL	Comm.	Comm.	BSD-3	BSD-2

[a] No dynamic extension loading [b] When used with its API, e.g., in combination with R [c] Via Trame [175] [d] The systems can be extended with Visual Analytics functionality by plugins or Python integration, but the focus is on interactive field visualization [e] Focus on dashboards with pre-populated data

was extended with several dimensionality reduction and clustering methods [121, 180, 181]. GGobi [163] integrates with the R language which enables users to apply analysis algorithms via scripting. Spotfire grew into a commercial, closed-source product with extensive analytics capabilities, while the others are open-source, albeit unmaintained. All of these tools predate *Progressive* VA and are not optimized for the specific needs of continuous updates and steering of analytics processes. ManiVault is designed around the principles of progressive VA from the start using a data-centric architecture. Data-producing and -transforming plugins can continuously update the data managed by the core, while data consumers get automatically notified about these changes. Tableau [167], building on the Polaris system [166], might be the most prominent and representative universal VA system. Marketing itself as a business intelligence tool, Tableau focuses on flexible visualization of various data types and more general analytics functions can be added via Python or R scripts. Similarly, Visplore [158, 159] implements a suit of statistical analysis and visualization methods for tabular data and aims at providing quick visual feedback for visual interactions and data queries. Its commercial offspring [174] offers a more direct integration of scripting languages to supplement built-in analysis functions.
The open-source ParaView [164], like many other analysis frameworks for spatial field data, e.g., volume data, [182–185] is based on the VTK library [186], and provides a wide range of visualization and analysis functions in an extensible framework. ParaView follows VTK's visualization pipeline and is designed around the flow of data through various transformations to their final visual presentation. Similarly, the commercial Amira Software [187, 188] offers a range of analysis functions for multidimensional volumetric data, but it is not freely extensible. Many visual analysis systems traditionally target either geometric or abstract tabular data. However, in recent years, the analysis of spatial and non-spatial data has become increasingly integrated [189]. With ManiVault we create a system for general high-dimensional data that can be extended to handle arbitrary spatial or abstract data types. Our data-centric system design enables flexible exploration workflows instead of having practitioners concerned about data flow through each step of the visualization pipeline.

7.2.2. Visualization Design Environments

Visualization design environments or similarly visualization prototyping systems are tools for creating visualizations that provide a graphical user interface for specifying visual encodings of data and interaction dynamics. Many such systems exist, and here we provide an overview of the tools most similar to ManiVault.
Lyra [190] offers fine-grained design options for single plots through handles, drop-zones, and other interaction mechanisms for graphical setup of re-usable

Vega or Vega-Lite [173] specifications. Lyra 2 [191] extends this framework by letting users define interactions like brushing and selection linkage between multiple plots. iVisDesigner [192] follows similar principles but places emphasis on collections of data visualizations in a dashboard format. Keshif [168] focuses on a novice user audience by automatically aggregating data and selecting visual representations based on pre-defined mappings for various data types. In contrast to the above design environments for single or multiple visualizations, ManiVault is a design environment for complete visual analytics systems including automated analysis methods. While the above systems are focused on abstract data, Inviwo [165] presents a visualization prototyping system for spatial field data. Its design allows users to specify visualizations on various abstraction levels, from visual (connecting functional boxes) to conventional programming. Compared to Inviwo's data-flow model, ManiVault is data-centric and focused on providing several visualizations and analytics tool building blocks. ManiVault's core system coordinates views on the data and enables linked selections between views out-of-the-box.
From a plugin-in developer's perspective, ManiVault resembles the prefuse [193] and ComVis [194] toolkits. They provide development environments and software components for building dynamic visualizations. Both focus on non-spatial data and target graph and tabular data set types. Scripting-based solutions like Dash [195] for creating dashboard applications or Voilà [196] for converting Jupyter notebooks into standalone web pages provide a GUI front-end to the wide offer of analysis libraries in the Python, R or Julia ecosystems. ManiVault is specifically laid out for progressive and high-dimensional data analysis. Our C++ implementation supports high-performance computations and interactions necessary for visual analytics.

7.3. Design Considerations

We designed ManiVault as a VA framework with multiple user groups in mind. While these groups can overlap, their requirements for the effective and convenient use of ManiVault are varied.

7.3.1. General Setting

High-dimensional data has become ubiquitous in many domains and the analysis of such data plays a pivotal role in acquiring insights into complex systems. Analytics software in different domains targeted at such data generally utilizes comparable sets of analytical and visual tools, such as dimensionality reduction, clustering algorithms, scatterplots, or parallel coordinates plots. These generic tools are then combined with data-, user-, and domain-specific tools and customizations to create a specific application. The primary motivation for developing ManiVault is to facilitate rapid construction of visual analytics applications for high-dimensional data without the need to re-implement common functionality. Modularity is a key aspect for creating reusable tools, both on a code and a user-facing abstraction level. The second main motivation for ManiVault is a need for flexible exploratory analysis, but also subsequent sharing of results, as well as the means to recreate the corresponding workflows. We learned of the target user characteristics and design requirements during multiple collaborations with practitioners in various fields [197–200] spanning several years.

7.3.2. Target Users

We identified three target user groups, each with specific requirements:

U1 Developers use ManiVault to implement new ideas and methods. These users, e.g., visualization researchers, interact with the system via code in order to create customized modules. Developers need the framework to provide a

stable API that allows for the integration of their methods with little overhead. Further, they need existing modules to focus on their specific contribution; e.g., a developer of a dimensionality reduction method might want to visualize results in an existing scatterplot module without having to implement their own.
U2 Application designers combine and adapt existing modules to create standalone applications for specific use-cases. Not all options of a view (e.g., the point size in a scatterplot) might be necessary for a specific workflow, and providing all options in the GUI can be distracting. In these scenarios, ManiVault needs to support flexible GUI customization. To minimize the burden, the framework should support such customization directly in the GUI without programming.
U3 Practitioners and domain experts use the software to analyze their high-dimensional data. Practitioners need ManiVault to allow for a flexible data exploration process, to provide responsive user interfaces, and to offer domain-specific visualization and analysis modules. Once their analysis is finished, practitioners need the ability to easily share and reproduce the results and their workflow in ManiVault. Given a well-defined workflow, they also need easy access to specified presets of visualization and analysis layouts.

The boundaries between these user groups are fluid. E.g., a skilled practitioner might want to extend a pre-bundled application with a module or develop a module themselves.

7.3.3. System Requirements

Based on the general usage setting and needs of our target users, we define the following high-level requirements for a visual analytics platform such as ManiVault. The framework must be:

R1 Extensible: ManiVault has to provide an interface for adding new functionalities. It must be possible to create modules for new
- **a** data types,
- **b** visualizations,
- **c** analytics methods,
- **d** data transformations,
- **e** loading/writing data.

R2 Flexible: ManiVault must allow for workflows in multiple domains and specifically enable straightforward workflow adaption during use.
R3 Linkable: ManiVault must provide modules with an API to easily link data selections and synchronize parameters, such that no dependencies between modules are created.
R4 Configurable: ManiVault must provide options for GUI configuration during runtime through the user interface.
R5 Distributable: ManiVault must be able to save its current state, including layout, data sets, and settings and reproduce a saved state.
R6 Performant: ManiVault must be performant when handling large data, stay responsive and provide interfaces to interact with processes during calculation to support progressive VA.

7.4. Framework Architecture

In order to ensure easy extensibility (**R1**), ManiVault is implemented as a modular system, see Figure 7.2a. The core application is a lightweight set of managers and any user-facing functionality is dynamically loaded from self-contained libraries, i.e., plugins, respectively discussed in Sections 7.4.1 and 7.4.2 (**R6**). This compartmentalization into a core and extensions provides easier maintainability, better scalability, and faster development. Together with a data-centric system structure (Section 7.4.3), this enables flexible workflows (**R2**) with various analytics and visualization techniques. ManiVault features an intricate notification and parameter sharing system to allow for communicating between plugins,

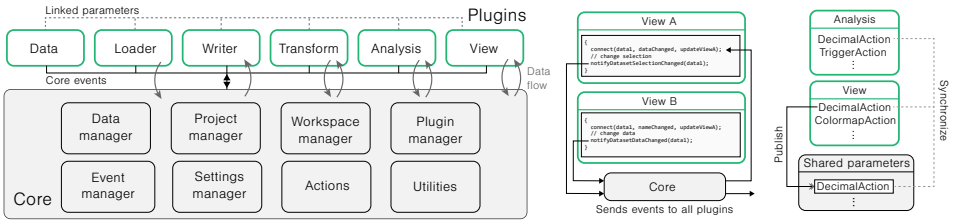


Figure 7.2. ***ManiVault's system architecture.*** *The core manages data and events, provides GUI management (actions), etc. Green borders indicate plugins, a light-grey background the core. Data flow from the core to data consumer plugins and from data producer plugins to the core is indicated with arrows. (b) View A listens to* `notifyDatasetDataChanged` *emitted by View B. View B does not listen to the* `notifyDatasetSelectionChanged` *event triggered by View A, but any plugin could. (c) a view plugin published a DecimalAction, moving the action in a shared parameters space and immediately subscribes to it. Now, an analytics plugin can connect to the shared action, enabling synchronization across plugins.*

see Section 7.4.4 (**R3**). GUI management objects, called actions (Section 7.4.5), implement a part of the communication system and the configuration and serialization system, see Sections 7.4.6 and 7.4.7 (**R4**, **R5**).

7.4.1. Core Application

ManiVault's core is modularized into a set of managers, actions, and utilities as shown in Figure 7.2a. ManiVault comprises a data-centric architecture: a data manager stores and administers access to data sets. All data sets are organized hierarchically, such that derived data sets like clusterings, embeddings, or proper subsets are marked as children of their respective source data. This enables simple access to properties of the parent data set and propagation of selections from derived to source data sets. Analysis, transformation, visualization, and loading/writing functionality as well as the definition of data types themselves are separated into plugins. A plugin manager loads plugins into the core and makes them available to the user. Each plugin can *consume data*, i.e., process existing data in the core and/or *produce data*, i.e., store a new or alter an existing data set in the core. While each plugin is self-contained, communication between plugins is made possible using two messaging systems (Section 7.4.4). An event manager in the core administers globally defines notifications while actions are used for run-time configurable notifications (see Figs. 7.2b and 7.2c).
The general application layout is handled by a workspace manager which takes care of the arrangement of all GUI widgets provided by view plugins. The core contains two main system view plugins, a data hierarchy, and a data properties viewer. The former displays the internal hierarchical data structure, while the latter shows properties of the data (number of data points, dimensions, active selections) and gives users access to the settings of analytics plugins, as discussed in more detail in Section 7.5. ManiVault provides a number of actions, GUI management objects, and administers any user-defined linking between them, see Section 7.4.5. Further, a project manager is responsible for saving and loading the current state of the application, including loaded data sets, the GUI layout, opened plugins, and linked parameters. Global settings applicable to, e.g., all plugins or the general application layout are handled by a dedicated settings manager.
Additionally, ManiVault's core supplies a set of utilities like dedicated renderers, shaders, color maps, mathematical helper classes, such as vectors and matrices, as well as common algorithms like mean shift clustering. These tools can be used to create a more coherent visualization and analysis setup across plugins. E.g., developers can rely on the availability of a standard set of color map types in every view plugin, while maintaining the ability to introduce custom ones.

7.4.2. Plugin Types

ManiVault works with six distinct plugin types that bundle various types of functionality. The system can be easily extended with new functionality by writing a new plugin that will automatically be loaded on start-up (**R1**). In combination with the data-centric core architecture, this enables a user to perform flexible workflow changes (**R2**).

Data plugins enable extending the types of data the system can handle. ManiVault provides a base data plugin class that developers can extend to define a custom data format. E.g., we provide an image data type that extends our basic point data type with image dimensions and thus a mapping of points to image coordinates. The system can generally be extended with arbitrary data formats.
View plugins provide a view on the data and allow interaction, such as selection of data elements. Views can be fully-fledged visualizations or simpler views such as lists. View plugins are primarily *data consumers*, i.e., they take a data set as input for visualization, but can also function as *data producers*, e.g., by providing means for annotating data. We provide example plugins with diverse backends, like OpenGL and D3.
Analytics plugins allow for the implementation of data analytics modules such as dimensionality reduction. As such, they are primarily *data producers* but also follow the *data consumer* API to receive the input data on which they perform calculations.
Transformation plugins resemble analytics plugins in code but are semantically different. They are also primarily *data producers*, but while analytics plugins derive new properties, e.g., an embedding, that can have an arbitrary shape, transformation plugins produce data of the same shape, i.e., with identical items and attributes. An example of such a transformation is a normalization of the original data.
Loader/Writer plugins respectively load specific types of data into the system (*data producer*) or write it back to file (*data consumer*).

7.4.3. Data Handling

The data handling in ManiVault follows a model-view pattern. Internally, the core's data manager keeps a list of raw data models, data set views, and selection views. A data plugin has to define both a raw data model and data set view — the selection view is simply another instance of the same data set view on the raw data. The raw data model holds the physical data values of a set and is never exposed directly to non-data plugins. Therefore, for most intents and purposes, the data set views can be regarded as the actual data sets present in the system. They define access to the raw data for all non-data plugins by providing, e.g., views on or copies of it. Each raw data object is associated with exactly one selection object to ensure straightforward selection sharing across all plugins that access a data set. Selection and set views can be separately requested and adjusted. This model-view pattern allows for a simple API and to create and use subsets with minimal overhead.
New data sets can be marked as derived from existing ones, e.g., when a new data set is created by an analytics plugin. The derived data also functions as the user-facing entry point through which the analytics settings can be accessed. This operation will create new data set and raw data objects but no new selection view. Instead, selection views are shared between parent and derived data sets. This simplifies the propagation of selections between views, e.g., a derived embedding shown in a scatterplot and the original data in a parallel coordinates plot. To enable selection sharing between arbitrary data sets, ManiVault lets users group data sets in the hierarchy view. Selections of any data sets within a group and with the same number of data points are then automatically synchronized.

We implemented a set of base data plugins in ManiVault, including plugins for point data, multichannel images, clusters, color, and text data. The development of ManiVault so far primarily targeted the point data type, which can store various high-dimensional integer and floating point formats. Our image data plugin shows the versatility of ManiVault's data handling and the point data type. When loading an image, two data sets are created: a point data set whose raw data object stores the actual pixel values and a child image data set whose raw data object stores metadata like image size. The image data set view provides access to the parent's raw data. This configuration ensures compatibility with analytics, transformation, and view plugins that expect point data to process multichannel images.
The implemented data handling system is lightweight. Besides the basic ManiVault core (< 90 MB), the data manager and hierarchy require < 8 MB of memory (on Windows). Each loaded data set produces less than 1.5 MB overhead in addition to its binary size, stemming from the plugin instance and core integration. More details can be found in Supplemental Material SD1.

7.4.4. Plugin Communication

Coordinated Multiple Views (CMVs) [3] are the basis for virtually any visual analytics application. While the individual views in a CMV system naturally map to modules in a modular architecture, an essential part of CMV systems is the integration of those views. This enables techniques like brushing and linking [201], where selections on the data are propagated to all views in the system, or the synchronization of parameters, like the viewport in an Overview+Detail system [202]. Enabling such linking of views, without breaking the system's modularity (**R3**) is no trivial task. A plugin should be self-contained with respect to its functionality. Yet, at the same time, plugins need to be able to communicate, such that they can inform other plugins about data changes and that their parameters can be linked and synchronized throughout the application.
We have designed and implemented two interfaces to solve the issue of inter-plugin communication. First, an event-based communication API to cover common system-wide types of events related to data set changes (**Core Events**) and second a parameter-sharing API (**Shared Parameters**) as part of our GUI building blocks (Section 7.4.5).

Core Events The ManiVault core API provides an event-based system for inter-plugin communication using the publish-subscriber pattern. Plugins send predefined events to the core, which distributes them, and all subscribers (typically plugins) can digest these events as depicted in Figure 7.2b. To efficiently support linking and brushing (**R3**), we have implemented such events for any changes of data values like addition, updates, removal, changes to data selections and several other data related changes. A plugin can choose to listen to all events of a certain type or subscribe only to certain events concerning a specific data set.
An example of a linked selection is shown in Figure 7.3. The figure shows a screenshot with three views, a scatterplot and a density plot on the left, and the properties of a clustering analysis on the right. Clicking any cluster in the clusters list (Figure 7.3a) will update the selection set attached to the data set and notify the core of these changes with the `notifyDatasetDataSelectionChanged` event. The core will then emit the `dataSelectionChanged` event with the changed data as an argument and subscribed plugins will receive a notification that triggers a refresh of the view with the updated selection (red points in Figure 7.3b).

Exemplary events:

- `notifyDatasetAdded`
- `notifyDatasetDataChanged`
- `notifyDatasetRemoved`

Shared Parameters We designed a complementary API to share parameters between modules (**R3**) using GUI actions (Section 7.4.5). With this system, a plugin parameter is exposed to other plugins by placing it in a public shared parameter pool, i.e., the parameter is *published* (Figure 7.2c). From there, other

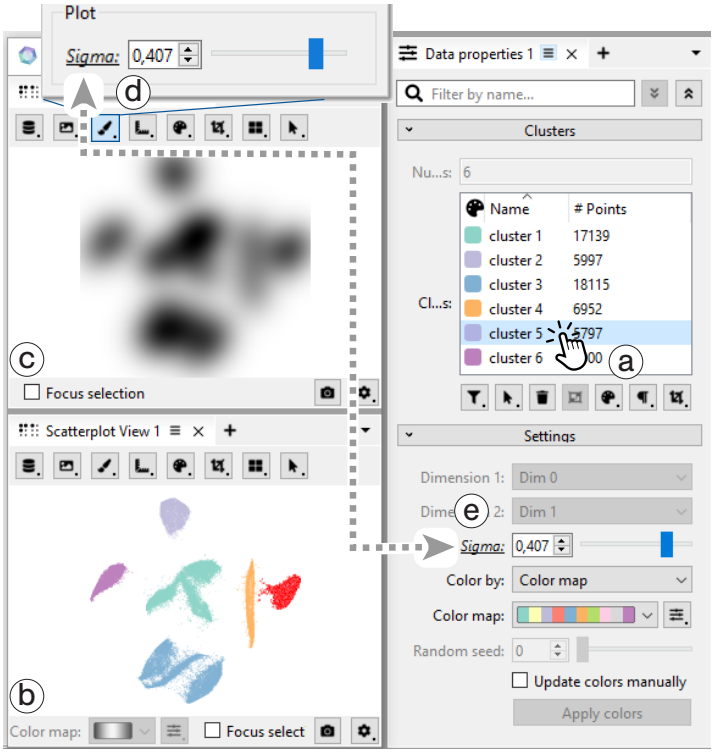


Figure 7.3. ***Parameter sharing*** *by connecting two actions of the same type in the GUI. Both, the Mean-Shift plugin and Scatterplot plugin use a DecimalAction to steer their computation and view respectively.*

plugins can *subscribe* to published parameters (provided that the parameter types match). Any change to a published parameter will be synchronized with all subscribed parameters. We provide common GUI elements with ManiVault, that developers can integrate into their plugins such that the user can publish a parameter or subscribe to any published parameter at run-time through the GUI (**R4**).
Figure 7.3 presents an example in the form of the kernel bandwidth (sigma) parameter used in kernel density estimation (KDE) employed in density plot visualizations (Figure 7.3c) but also mean-shift clustering. We have implemented plugins for both that allow real-time changes of the sigma parameter, based on Lampe and Hausers real-time KDE [203]. Linking this parameter between the density plot and the clustering module enables visually finding a suitable density estimation while the clustering is updated on-the-fly. To link the parameters the user simply clicks on the underlined label in the GUI (Figure 7.3d), e.g., in the density plot view, and chooses "publish". After defining a suitable name for the parameter, the user can then click on the corresponding label in the settings widget of the mean shift clustering plugin (Figure 7.3e) and click subscribe to be presented with a list of suitable parameters, including the just defined one. After subscribing, the connection is indicated by the italic font of the *Sigma* label.

7.4.5. Actions

To support sharing of parameters as described above, but also to make it easy to capture the state of a plugin, configure the GUI and unify the look and feel between plugins, we have devised and implemented a number of building blocks we call *actions* on top of the standard Qt GUI widgets. These include simple actions for `decimal` and `integral` values as well as `strings` but also

more complex elements such as `colors`, `color maps`, `file-pickers`, etc.. In addition to those standard GUI elements we implemented a number of custom actions targeting typical VA applications. These include a general-purpose selection action, that supports different modalities (brushing, rectangle, lasso, etc.) and Boolean combinations (replace, add, remove), and a dimension picker action that provides a consistent way to select one or multiple dimensions of a data set, e.g., to limit the input to a dimensionality reduction plugin. Although we believe that we provide large coverage of commonly required tasks with the built-in actions, we also provide an API for plugin developers to create custom actions.

By using our actions API, sharing of parameters as described in Section 7.4.4 is automatically available through the GUI. In addition, actions can also be attached to data objects, to expose their functionality to other plugins. A data producer plugin can, e.g., attach an action to trigger a calculation within the plugin. Other plugins can query these attached actions and provide the corresponding GUI elements within their scope. We showcase this in our Hierarchical Stochastic Neighbor Embedding (HSNE) [52] analytics plugin. The plugin creates a hierarchical embedding structure that can be refined interactively. We attach an action for triggering the refinement to the produced embedding data set. When viewing the embedding in a scatterplot, the scatterplot view plugin exposes the refine action and other attached actions through the context menu. The user can then trigger the refinement directly from the scatterplot visualization, even though the actual calculation is carried out by the HSNE plugin.

Besides serving as GUI building blocks, we have also implemented support for serialization in the action system. Each action can be serialized into a `QVariant` object, including its complete current state, consisting of whether it is active, visible, writable, and the parameter itself. All actions that belong to a plugin form a hierarchy that can again be serialized into a QVariant object and from there into a JSON object in memory or file on disk. As such, a plugin that has consistently been implemented with the actions API supports saving and loading of the state out-of-the-box. Currently, we use this to create presets of a plugin's configuration and to save the complete state of the application to a project file. In the future, we intend to extend this to a complete provenance mechanism.

An example of a simple decimal action is the implementation of the Sigma parameter discussed above and shown in Figure 7.3d. The GUI for this parameter consists of the label, a spinbox, and a slider. Rather than manually creating the GUI elements, the desired elements can be specified when creating the action. An example of a customization that we integrated in the decimal action is to show a spinbox or slider individually or both, as in this example. The action then creates the GUI elements on-the-fly and also makes sure they are synchronized by creating them as linked views on the parameter itself. The underlined label indicates that the parameter is publishable and/or ready to subscribe, while the *italics* font indicates that it is already linked. Clicking the label opens a GUI interface for setting up parameter linking.

7.4.6. Projects and Workspaces

To save the entire state of the application and fully restore it at a later point in time ManiVault uses projects (**R5**). Projects extend the serialization of actions, described in Section 7.4.5, to the core framework, capturing settings and the layout of the CMV system. In addition, a project contains a complete snapshot of the data hierarchy. We implemented projects as self-contained, compressed archives that are a combination of human-readable JSON files and binary files. Two JSON files are used to save the entire state of the application. A `workspace.json` contains the CMV layout and actions state and a `project.json` saves the data hierarchy and additional project metadata. The actual data sets are saved as raw binary blobs, with unique identifiers referenced in `project.json`, to minimize load and save times. As such, a project is completely self-contained

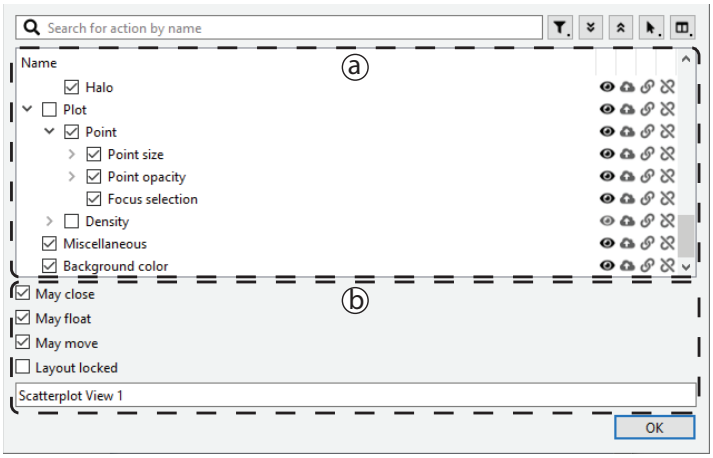


Figure 7.4. *Example of the plugin* ***GUI configuration editor*** *which allows application designers to edit the properties of the plugin actions hierarchy from within the application.*

and can be easily distributed to share findings or simply used to come back to an analysis at a later point in time.
We split the description of the project into `project.json` and `workspace.json` to add an additional feature, i.e., the definition of user-defined workspaces. As described above, the workspace contains the complete spatial arrangement of views (layout configuration) and their complete state. A workspace is used to set up a complete tailor-made CMV VA application, including customized GUI elements, but without preset data, as a project would. To enable easy tailoring of layouts and cross-plugin connections directly in the application, even without programming, we designed the *Studio Mode* for ManiVault.

7.4.7. Studio Mode

For the configuration of actions, workspaces, and complete projects, ManiVault can be put into *Studio Mode*. This mode of operation allows application designers to create complete tailor-made applications and data viewers from within the GUI of ManiVault itself.
A plugin editor, shown in Figure 7.4, enables fine-grained control over the user interface. It lists an overview of all actions that are currently available for opened plugins (Figure 7.4a). Therein each action can be enabled or disabled as a whole ☑, but also customized with respect to its visibility 👁 or whether it can be published 🌩, connected 🔗, or disconnected 🔗. Additionally, the editor lets a user configure general options like the name of a plugin instance, shown in its title bar, or whether the GUI of the plugin may be moved or closed (Figure 7.4b).
The plugin editor is an essential tool for application designers, to create a completely customized user experience for a specific application. At the same time, it provides the possibility for advanced users of the system to create presets of views. Besides saving a complete project, users can adjust the interface of an individual plugin to their needs and save the resulting configuration as a template for future instances of that plugin. Using the serialization described above, these templates can be saved to disk, providing persistent access across sessions.
For a user-definable flexible layout of the application, we incorporate the Qt advanced docking system [204] into ManiVault. The system allows users and application designers to re-arrange the entire layout according to their needs and preferences.

7.5. Implementation

Code available on GitHub in the repository ManiVaultStudio/core

The ManiVault core is implemented in C++ and the Qt [171] cross-platform application development framework. ManiVault provides a plugin API for data types, view, analytics, transformation, and writer/loader modules. For each of these types we provide template implementations to lower the entry barrier for developers. In addition, we have already implemented a number of plugins for various use cases, including some of the core functionality of ManiVault such as the basic data types, and the data hierarchy and data properties view plugins.
The **data hierarchy view** (Figure 7.5a) functions as the central access point to any data loaded or created in ManiVault. It displays the data hierarchy in a searchable tree widget where derived data, such as a clustering, are added as children to the original data. A data set can be loaded into a viewer plugin by simply dragging it from the hierarchy onto the view (Figure 7.5c). Alternatively, the user can also interact with each data set through a context menu providing access to all compatible data consumer plugins. For a fast setup of plugins that expect more than a single input, users can select multiple data sets in the hierarchy and open them through the same menu. The info panel shows additional information like an analytics progress bar, status messages from plugins or data group affiliation. If a data set is associated with an analytics plugin, selecting the hierarchy entry will open the analytics settings in the properties view.
The **data properties view** (Figure 7.5b) provides information for a data set selected in the data hierarchy. For a loaded data set this can be additional metadata created by the loader, e.g., the extents of an image data set. More importantly, the data properties view also functions as the user interface for analytics and transformation plugins. These plugins are instantiated through the context menu of a data set, which then functions as their input; their output data sets are then created as children of the input. Selecting an output data set provides access to the parameters of the analytics or transformation plugin. Figure 7.5b shows the data properties view of an embedding data set, created with our t-SNE plugin. From here, the user can at any time interact with the t-SNE algorithm, e.g., to pause the calculation, change parameters or compute more iterations.
The data hierarchy and data properties views are integral parts of the system. More specific functionality is implemented in a number of further plugins. Dimensionality reduction, integral to high-dimensional data analysis, is provided by Principal Component Analysis (**PCA**), t-distributed Stochastic Neighbor Embedding (**t-SNE**) [29], and Hierarchical Stochastic Neighbor Embedding (**HSNE**) [52] plugins. The t-SNE and HSNE plugins wrap the high-performance HDI library [205] and as such scale to millions of data points using its GPU-based

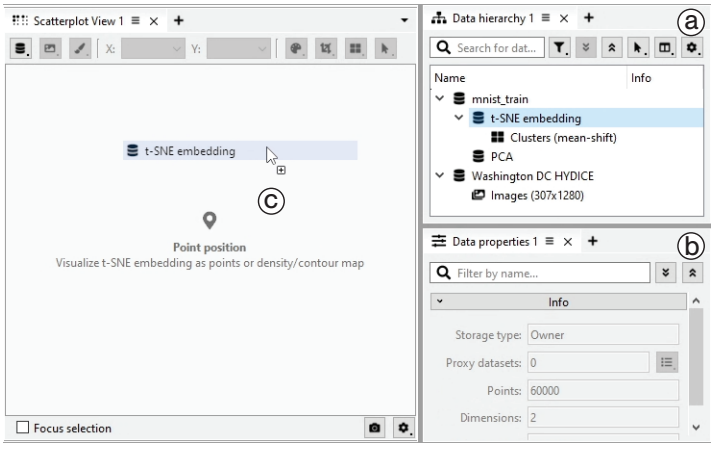
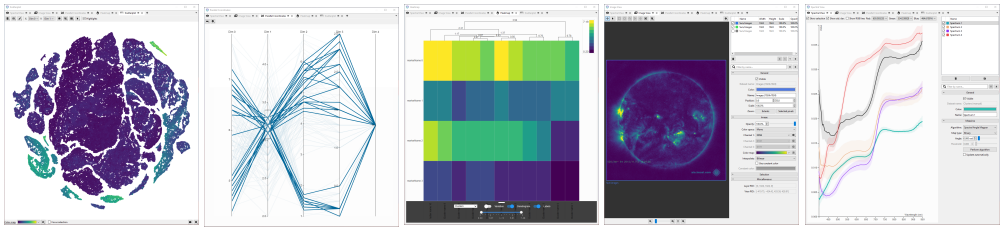


Figure 7.5. ***Data hierarchy*** *(a) and* ***data properties*** *view (b) in ManiVault. Data sets can easily be shown in views via drag and drop (c).*



(a) *Scatterplot* **(b)** *Parallel Coordinates* **(c)** *Cluster Heatmap* **(d)** *Image View* **(e)** *Spectral View*

Figure 7.6. *A selection of* ***viewer plugins*** *in ManiVault.*

implementations [105]. For clustering, we provide an interactive **mean-shift clustering** plugin, based on real-time kernel density estimation [203].
For visualization, we provide a number of plugins for common plots, including a **scatterplot** (Figure 7.6a), **parallel coordinates plot** (Figure 7.6b), and **cluster heatmap** (Figure 7.6c). If performance is not a major concern, developers can use web views in combination with Qt's webchannel API for communicating between the C++ back-end and web-technology-based front-end. This allows for easily integrating the vast amount of available visualizations in languages like D3 [172] and Vega-lite [173]. Our heatmap and parallel coordinates plot are based on this technology. While the webchannel introduces some overhead, such plugins are generally limited by the performance of the JavaScript rendering libraries. If the scalability of a visualization is of high priority, developers can implement custom high-performance views, e.g., using OpenGL. We have done so with our scatterplot and image view (Section 7.5.1) plugins. The scatterplot enables visualization and interaction with millions of points in real-time. In the default point rendering mode, the different visual channels (point size, color, opacity, etc.) are fully configurable either using fixed values or based on any fitting data available. Additionally, we implemented a density representation, to provide more visual scalability.
Finally, for data loading and writing, we currently provide support for basic formats in the form of a comma-separated value (**CSV**) loader/writer and a **binary** loader/writer.

7.5.1. High-Dimensional Imaging

Besides traditional abstract high-dimensional data analytics, we target a number of applications related to high-dimensional imaging (e.g., the workflow presented in Section 7.6.2). As such, we developed a number of plugins targeting such image data.
Central to these efforts is the **image data type** plugin. The image data type extends the point data type by the extent of the image. Consequently, the image data type is compatible with all data consumer plugins that take point data as input; e.g., this allows to calculate a t-SNE using the pixels of a high-dimensional image as input.
We implemented a sophisticated **image view** plugin (Figure 7.6d). Inspired by widely used image editors, we opted for a layer-based approach. Users can simply drag multiple data sets into the view, where they are added as layers. From here, users can define the transparency, as well as the position of each layer, e.g., to stack multiple properties of a single data set as semi-transparent layers or arrange complementing data sets next to each other. These interactions are possible through standard navigation tools for zooming and panning, while selection is implemented using the action described in Section 7.4.5. The actual visualization of the image is fully configurable: One or two attributes can be displayed by using 1D and 2D color mapping, and three attributes by directly mapping them to the three channels of *RGB*, *HSL*, or *CIELAB* color spaces.

Next to the image viewer, we also provide a **spectral view** plugin (Figure 7.6e), specifically for hyperspectral images. The viewer is based on a simple D3 line plot and shows spectra of individual pixels or, in the case of groups (e.g., selections or clusters), a mean spectrum and a variation as a band around it.
To load image data into ManiVault, we currently provide two options. The first one is a versatile general **image loader** plugin. Hyperspectral image data is commonly available as a stack of grayscale images, where each image represents a specific wavelength, also interpreted as a dimension of a high-dimensional space. Our image loader detects such stack in a folder containing common image formats (including .png, .jpg, .tiff), and also allows direct loading of other common image formats (grayscale, RGB, ARGB). Dimensions can be interactively included or excluded from the data set in the loading menu. We also support re-sampling of the data before loading and the creation of image pyramids to enable analysis at varying levels of detail, depending on the features of interest or time available for the analysis. Specific to hyperspectral images, we also provide an **ENVI loader** plugin compatible with L3Harris' geospatial analysis software ENVI [206].

7.6. Application Examples

ManiVault has already been used for several projects across four universities and several partners. Popa et al. [198] and Li et al. [200] describe the design of complete VA systems for analysis of cultural heritage and biological data, respectively. Vieth et al. [69] and Thijssen et al. [199] developed VA approaches for dimensionality reduction and explaining projections as ManiVault plugins. Here, we walk through exemplary usage scenarios for our framework from the perspective of our three target user groups (Section 7.3.2): software developers (Section 7.6.1), practitioners (Section 7.6.2) and application designers (Section 7.6.3).

7.6.1. Writing ManiVault Plugins – Developer Perspective

ManiVault provides developers of VA modules with a comprehensive API for data set access, the event notification system, and the other core managers (Section 7.4.1). Extending the functionality of ManiVault through new plugins thus comes with minimal overhead. Example code for each plugin type is available at github.com/ManiVaultStudio/ExamplePlugins.
Here, we present two examples of the necessary steps for creating basic plugins (**R1**). First, we create an analytics plugin based on the high-performance t-SNE library HDI [205]. In addition, we discuss the implementation of a parallel coordinates plot (PCP) plugin using an existing D3 implementation. Together with the existing image viewer and scatterplot, these plugins combine into a

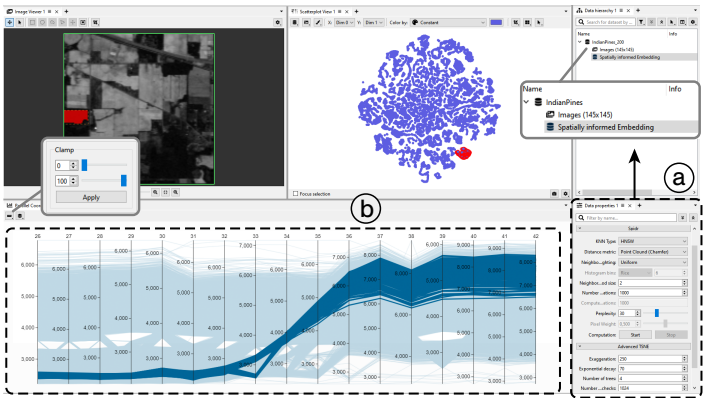


Figure 7.7. *The **Spidr analysis and parallel coordinates plot** as implemented with the plugin setups from Figs. 7.8 and 7.9.*

complete GUI-based application shown in Figure 7.7 that is usable by domain expert users without programming knowledge.
To implement the analytics plugin, we follow the steps laid out in Figure 7.8. In `step 1`, we create the output data set by deriving a new data set from the input data, for which the plugin is opened in ManiVault. In this case, we will create a two-dimensional t-SNE embedding containing x- and y-coordinates for all the points in the input data set. As such, the output data set will be a points data set that has the same number of points and two dimensions. Next, we add a settings action to the created data set and define GUI elements using ManiVault's action system. The actions are added to the output data and listed in the data properties view as shown in Figure 7.7a (`step 2`). We create `TriggerActions` which add pushbuttons to the GUI, to start, pause, and resume the calculations and a number of categorical `OptionActions` and numerical `DecimalActions`, e.g., to expose t-SNE parameters like the distance metric (`OptionAction`) or perplexity (`DecimalAction`) (**R4**). Finally, in `step 3`, calls and reactions to library functions need to be defined. Here, we notify the core and thereby other plugins about updated output data, in particular, as the t-SNE optimization iteratively progresses, we notify the core after every iteration, such that the viewer plugins can show the progress live. The result is a lightweight wrapper with no notable performance overhead. Comparing the performance to running the HDI library using its own Python wrapper showed no performance regression (Supplemental Material SD1), even when including progressive updates in ManiVault.
To implement the PCP viewer plugin, we need to set up a view widget that shows the PCP chart in addition to settings, like with the analytics plugin. Here, the settings are displayed in the same windows as the view widget (Figure 7.7b). Since we build a JavaScript-driven plot, we derive this widget from `ManiVault::WebWidget` and introduce all HTML and JavaScript resources that are used for the PCP through a Qt resource file, `pcp.qrc` (`step 1`, Figure 7.9). `Step 2` is to simply set the existing `pcp.html` file in the existing `viewWidget`. All JavaScript resources are automatically included through the HTML file. At this point, the viewer is only able to show the content of the provided HTML page. To establish interactions to and from the C++ side, we set up a `ManiVault::WebCommunicationObject`, which uses a QWebChannel. Within this communication object, we define signals and slots for communication. E.g., the `setData` signal (`step 3`) is used to send the data, provided as a `QVariantList` object, to a receiver on the JavaScript side. This receiver, i.e., the `initPlot` function is connected in `step 4` to receive the signal. Vice versa, slots defined in the communication object can be called directly in JavaScript code, e.g., here we define an `updateSelection` slot, that can be called from the JavaScript side with a list of selected items. The plugin then handles any related computations in the corresponding C++ function.

7.6.2. Data Exploration – Practitioner Perspective

Practitioners in various disciplines work with high-dimensional data sets. Here, we consider the exemplary case of exploring remote sensing data using ManiVault. Similar to other application areas, visual exploration of geospatial data is considered important but challenging [207]. While specific considerations and final insights will differ from domain to domain, we can follow the

```
void AnalyticsPlugin::init() {
  // 1. Derive output from input data set
  setOutputDataset(_core->createDerivedDataset("outData"));
  // 2. Add settings actions to output data set
  outDataset->addAction(_settings->getSettings());
  // 3. Connect GUI interactions (e.g. button press)
  //    and library callbacks (e.g. progress or finish)
  connect(_settings->getStart(), press, this, runTask);
  connect(_lib, finishedTask, this, updateCore);
}
```

Figure 7.8. *Bare bone **analytics plugin setup** for wrapping a C++ library. Notifying of output data change (`step 4`) can be called progressively during the calculation of or on finishing a task.*

```
[ViewWidget.cpp]
ViewWidget::ViewWidget() :WebWidget() {
  // 1. Init resources and communication bridge
  Q_INIT_RESOURCE(pcp);
  init(…comObj);
}
[ViewPlugin.cpp]
void ViewPlugin::init() {
  // 2. Init web widget (set HTML contents)
  viewWidget->setPage(":res/pcp.html", "qrc:/res/");
  layout->addWidget(viewWidget);
}
[CommunicationObject.h]
class ComObj :public WebCommunicationObject {
  // 3. Init signals for communication from cpp to js
  signals:
    void setData(QVariantList& data);
  // 5. Init slots for communication from js to cpp
  public slots:
    void updateSelection(QVariantList& selection);
}
[qwebchannel.tools.js]
// 4. Register signals sent by the view widget
bridge.setData.connect(function(){initPlot(arguments[0])})
```

Figure 7.9. *Bare bone **viewer plugin setup** for wrapping a JavaScript library. Some boilerplate code is left out for brevity; complete implementation is available alongside other example plug-ins online.*

task abstraction by Lam et al. [153] to create a partial workflow that will be representative of many fields (**R2**).

We want to explore a hyperspectral image data set, the HYDICE image of the National Mall [208], showing 307 by 1280 pixels, each attached to 191 spectral bands covering the 0.4-2.4 µm region of the light spectrum reflected by the objects in view. Each band can be interpreted as an image channel. A major objective when exploring hyperspectral images is the identification of surface cover classes. It is typical to manually define class labels for a small subset of pixels that afterwards are used in semi-supervised automated classification for the rest of the data. Connecting any derived features from the spectrum back to the spatial image layout is essential during these analysis steps. More specifically, our goals are now to (I) explore the data, connected to the task of *discovering and describing observations*, and to (II) explain these observations by *identifying main causes*. These steps will yield well-justified classes that can be used in downstream analysis.

First, in ManiVault, we load the HYDICE data set using an *image loader plugin*. To inspect the loaded image we can open it in an *image viewer plugin*, which provides single-channel and false-coloring visualizations based on any three channels. We additionally open a *spectral view plugin* which shows the full spectrum of a single pixel or the averaged spectrum of a selection that we define in the image viewer, resulting in the setup of Figure 7.10a. Then, to easily discover a hierarchical class structure, we use the *HSNE analytics plugin* to create a hierarchical embedding of the data employing angular distance: we open the analysis through a context-menu of the data set entry in the *data hierarchy*, select the cosine distance metric, start the embedding and display it in a *scatterplot* as seen in Figure 7.10b. Next, we manually outline three clusters that are apparent in the top-level HSNE embedding as shown in Figure 7.1 (center top). To inspect their spectra, we drag and drop the new cluster data set from the data hierarchy into the spectral viewer, Figure 7.1 (right). Additionally, we might inspect the cluster sizes in the *data properties*. Clicking on a specific cluster displayed in the data properties will select corresponding data points in the embedding and highlight corresponding pixels in the image (Figure 7.10c). Thus, we can quickly relate the cluster spectra to image positions and define the main pixel classes water, vegetation, and buildings. We want to focus on a single cluster — the one corresponding to buildings. Therefore, we refine the cluster of interest to a lower HSNE hierarchy level through a context menu opened by clicking inside the embedding — the HSNE plugin added an action to the data set that is displayed there as well as in the data properties window. To establish a visual connection between the spatial data layout and embedding, we drag the new

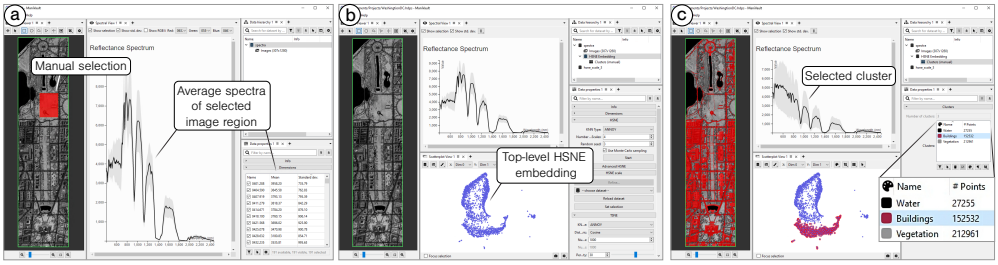


Figure 7.10. ***A typical exploration workflow with ManiVault:*** *A user can open and re-arrange views on the fly, derive new data sets using analytics plugins and connect parameters between plugins. Linked colormaps of the scatterplot and image viewer are shown in Figure 7.1.*

embedding data set to the image viewer, which automatically infers the proper image dimensions for the data subset from its parent in the data hierarchy and converts it into an additional image layer. Further, we can link the colormaps of this image layer and the embedding through the parameter-sharing system by publishing one and connecting the other to it (**R3**). Zooming into a spatial area of interest, Figure 7.1 (left), we can discriminate between several building structures like houses and streets, and even create subclasses of roofs that immediately stand out thanks to the embedding-based recoloring.
The above procedure intertwined the accomplishment of goals (I) and (II). ManiVault made it easy to connect various views on the data, i.e., a spatial layout, high-dimensional pixel attributes, and derived features in the form of embedding positions. We quickly discriminated between classes in the data and identified differing spectral characteristics as their cause.

7.6.3. Sharing Analysis Setups – Designer Perspective

ManiVault's workspace and project features can be used to save and continue an analysis session but also enable dissemination of results and complete workflows. To showcase this, we re-implemented the Cytosplore Viewer application [209] dedicated to sharing the results of Bakken et al. [210] in ManiVault, shown in Figure 7.11. Instead of having to write an entire stand-alone application to share an interactive environment alongside data to explore related insights in, we can use a ManiVault project to bundle both views and data (**R5**).
The viewer application depicts RNA sequencing data on brain cells from three vertebrate species. The viewer aims to highlight differences in the expression of genes and cell types that are shared across the species as described in the original paper. The main elements of the viewer application are three scatterplots showing t-SNE embeddings of the gene data of each species, a hierarchical cluster viewer showing cell types, and a table view showing statistical properties of the expression data. To create the viewer, we configure ManiVault's GUI from within the GUI (**R4**). We start with loading all data sets and setting up a single scatterplot plugin. We link scatterplot parameters like its colormap to a global settings panel that lets users configure all three scatterplots, like in the original application. Its settings can be saved as a preset which we use for the other two scatterplot instances. Similarly, we populate the cluster hierarchy view and table viewer with data. Figure 7.11 shows a configuration in which a user-selected entry in the table view defines the data attributes (here a gene's expression) used to recolor the scatterplot data points (here tissue samples).
ManiVault's Studio Mode allows us to lock this setup of views and parameter connections. This is achieved by simply publishing the current view layout, loaded data, and parameter linkage through the "File" menu tab. We can now share the viewer with other parties.

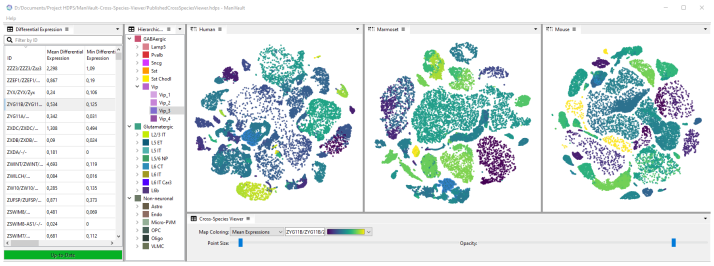


Figure 7.11. *Screenshot of a re-implementation of a* ***Cytosplore Viewer*** *for comparative cellular analysis of motor cortex in human, marmoset, and mouse [210]. The viewer shows embeddings of cells from the three species in combination with a shared cluster hierarchy and the option to calculate differential gene expression.*

7.7. Conclusion

This chapter describes the design considerations for and implementation of ManiVault, an extensible visual analytics framework for high-dimensional data. Due to its modular architecture and data-centric design, the software enables flexible exploration and analysis workflows. We presented various plugins that provide visualization and analytics functionalities to the system. To build upon these, we showed how existing libraries can be easily incorporated into the system. ManiVault's action and event systems allow users to adjust plugins and their interplay, enabling the creation of fully customized applications.
Currently, the system provides data plugins that cover a wide range of applications. New data types like multivariate graph data [58] can be introduced into the system as new data plugins without changes to the application's core. We plan to extend the current serialization mechanism, used for saving the state of the system, to handle information about interaction history and other kinds of provenance [211]. Analytics plugins that run code in interpreted languages like Python. We would like to improve this integration in order to easily integrate the vast amount of data science tools available in those languages.
We believe that ManiVault has great potential in aiding with the creation and use of visual analytics applications for visualization developers, practitioners, and application designers.

8. Conclusion

A wide variety of domains produce high-dimensional images, from single-cell biology to astrophysics. As experts in these domains acquire new data of unseen phenomena or using novel equipment, they face the need to explore and analyze these new images. However, high-dimensional images are challenging to handle: Both their size and complex structure introduce difficulties computationally, e.g., in terms of algorithmic design, as well as for interactive visual representation, since their direct visualization is incompatible with standard displaying methods and thus limits human interpretation.
In this thesis, we introduced several visual analytics techniques, i.e., the combination of visualization, interaction, and automated analysis methods, to aid exploration and analysis of high-dimensional images. Specifically, we presented several methods that bridge dimensionality reduction techniques with image data properties and a software framework for the exploration of high-dimensional data.

8.1. Contributions

One major shortcoming of applying DR techniques like t-SNE and UMAP to images is that they do not incorporate spatial information, i.e., pixel neighborhood relations. While their embedding may capture the pixel attribute structure well, any image-specific characteristics, e.g., texture information, is lost. In chapter 4 we modified the distance metric used in neighborhood-based DR techniques to address this limitation. Instead of using a distance metric between the attribute vectors of two pixels, as is the standard approach, we include the pixel's spatial neighbor's in the comparison. We explore comparisons based on high-dimensional texture features and using a point-cloud based distance metric that directly works with all attribute vectors in the pixel's neighborhood. We show that incorporating spatial information into DR techniques exposes data patterns that are not easily identifiable in standard embeddings.

Chapter 4:
Spatial Information in Dimensionality Reduction for High-Dimensional Images

Modern image data is typically high in resolution, containing up to several million data points. With these large images sizes, dimensionality reduction techniques become more computationally demanding and their output less interpretable as embeddings become cluttered. Hierarchical DR techniques address these scalability issues. Typical exploration setups for high-dimensional images consist of an image view and an embedding view. While a user can change the level of abstraction and regions of interest in image and embedding space, there exists no coupling between these interactions. In chapter 5 we proposed such a mapping from image navigation interactions to embedding space actions. We show that such a coupling allows for a natural extension of the "*overview-first, details-on-demand*" approach to exploring image and embedding space simultaneously. The automated coupling reduces the number of interactions required to reach a desired level of abstraction in both spaces.

Chapter 5:
Coupled Exploration of High-Dimensional Images and Hierarchical Embeddings

We introduced interactions for coupled image exploration and attribute exploration via hierarchical embeddings. However, existing data hierarchies used in hierarchical DR techniques are designed for abstract data, not image data, leading to non-spatially-continuous abstractions. That is, embedding points on abstraction levels often represent a wide spread of pixels, hampering a region-of-interest based image exploration. This issue would remain even when combining the hierarchical DR with spatially-informed distance measure as addressed in chapter 4. In chapter 6, we proposed a superpixel hierarchy to overcome this problem. Therein, we present a method to merge pixels spatially based on a random-walk-based attribute similarity, which preserves the high-dimensional attribute manifold. We showed that these manifold-preserving superpixels provide a suitable abstraction for high-dimensional image exploration.

Chapter 6:
Manifold-Preserving Superpixel Hierarchies

Chapter 7: ManiVault: A Visual Analytics Framework for High-Dimensional Data

As valuable as individual methods for exploration, analysis, and interaction may be, they can only be effective if they are accessible to domain experts. New imaging modalities and advances in computational methods will continuously update the best practices in data handling, requiring analysis environments to be flexible. In chapter 7 we presented the software framework ManiVault, which enables users from varying backgrounds and with diverse expertise to setup and extend visual analytics workflows. We show how ManiVault supports software developers, application designers, and practitioners to implement algorithms and visual encodings, prototype workflow-specific tool sets, and perform their data exploration and analysis, respectively. All methods presented in this thesis are also made available in or fully implemented as plugins for this framework.

Collectively, these contributions address high-dimensional image exploration by integrating spatial awareness into dimensionality reduction, developing interaction techniques, and creating manifold-preserving superpixel hierarchies. The domain-agnostic nature of these methods enables their application across diverse fields. This generality necessarily comes at the cost of leaving out any domain-specific optimizations that may be available in those use cases. However, by implementing these approaches within the ManiVault framework, they become accessible and adaptable. Domain experts can build on this foundation to tailor workflows to their specific data and analysis needs.

8.2. Challenges and Outlook

Dimensionality reduction enables the discovery of meaningful insights in data that is otherwise unfamiliar or difficult to interpret. However, especially non-linear embeddings, which this thesis builds on, can be difficult to reason about. Clusters in the data might become apparent, but explanations for why any particular data point is clustered generally require additional interpretive tools or domain knowledge. Our introduction of similarities that combine image-space and attributes in the dimensionality reduction process exacerbates this point and opens up additional questions about the feature importance of spatial pixel layout. A rich body of visual analytics research exists to aid in embedding interpretation [212, 213], and leveraging this work further presents an opportunity to refine and improve the effectiveness of the methods presented in this thesis.

Hypothesis generation and confirmatory analysis are closely linked and should be complementary during the data analysis process. Hypothesis generation focuses on gaining insight into unseen data as well as uncovering patterns and anomalies, whereas confirmatory analysis is concerned with validating predefined hypotheses using statistical methods. The methods and software framework in this thesis primarily target the former analysis stage. While confirmatory analysis remains outside the scope of this work, its tighter integration into the exploratory workflow with embeddings and inside ManiVault represents a valuable direction for future development.

Ongoing development in domains such as spatial transcriptomics indicate a trend toward significant growth in the size of high-dimensional image data — both in terms of spatial resolution and number of channels. This trend underscores the need for increasingly specialized methods for interactive exploration and analysis, capable of handling both the scale and complexity of such data, as well as inclusion of domain specific knowledge. Additionally, advances in acquisition of volumetric high-dimensional data highlight the need to generalize spatial neighborhood relationships and interaction paradigms to accommodate 3D data and corresponding embeddings techniques.

8.3. Closing Words

This thesis started out with the goal of addressing long-standing challenges in facilitating the understanding of massive and continually growing data, supporting multiple levels of data and information abstraction and providing frameworks for analysis of spatial data. Did it succeed? We presented such a software framework and new methods for abstracting and interacting with large spatial data. Yet, progress in visual analytics is difficult to quantify and the search for good solutions often open-ended. As new types of data and analytical challenges emerge, both methods and frameworks must evolve accordingly. The work presented here contributes to this ongoing process. It is another step along the path forward.

"We do not guarantee to introduce you to the "best" tools, particularly since we are not sure that there can be unique bests." John W. Tukey in *Exploratory Data Analysis*, page 1 [1]

References

[1] John W. Tukey. *Exploratory Data Analysis*. Pearson, 1977 (cited on pages 1, 75).

[2] Christian Tominski and Heidrun Schumann. *Interactive Visual Data Analysis*. AK Peters Visualization Series. CRC Press, 2020. DOI: 10.1201/9781315152707 (cited on page 1).

[3] Jonathan C. Roberts. 'State of the Art: Coordinated & Multiple Views in Exploratory Visualization'. In: *Proc. CMV*. New York: IEEE, 2007, pp. 61–71. DOI: 10.1109/CMV.2007.20 (cited on pages 2, 56, 62).

[4] Julian Heinrich and Daniel Weiskopf. 'State of the Art of Parallel Coordinates'. In: *Proc. Eurographics*. Ed. by Mateu Sbert and László Szirmay-Kalos. Eurographics Association, 2013, pp. 95–116. DOI: 10.2312/CONF/EG2013/STARS/095-116 (cited on pages 2, 5).

[5] A. Lhuillier, C. Hurter, and A. Telea. 'State of the Art in Edge and Trail Bundling Techniques'. In: *Computer Graphics Forum* 36.3 (2017), pp. 619–645. DOI: 10.1111/cgf.13213 (cited on page 2).

[6] Justin Matejka and George Fitzmaurice. 'Same Stats, Different Graphs: Generating Datasets with Varied Appearance and Identical Statistics through Simulated Annealing'. In: *Proc. CHI*. Association for Computing Machinery, 2017, pp. 1290–1294. DOI: 10.1145/3025453.3025912 (cited on page 2).

[7] F. J. Anscombe. 'Graphs in Statistical Analysis'. In: *The American Statistician* 27.1 (1973), pp. 17–21. DOI: 10.1080/00031305.1973.10478966 (cited on page 2).

[8] Laurens van der Maaten, Eric Postma, and Jaap van den Herik. *Dimensionality Reduction: A Comparative Review*. Tilburg University Technical Report, TiCC-TR 2009-005. URL: lvdmaaten.github.io/publications. 2009 (cited on pages 2, 6, 7).

[9] C. LeCun Y. Cortes and C.J.C. Burges. *The MNIST Database of Handwritten Digits*. URL: yann.lecun.com/exdb/mnist. New York, NY, USA, 1998 (cited on page 3).

[10] Marion F. Baumgardner, Larry L. Biehl, and David A. Landgrebe. *220 Band AVIRIS Hyperspectral Image Data Set: June 12, 1992 Indian Pine Test Site 3*. 2015. DOI: 10.4231/R7RX991C (cited on pages 3, 24, 36, 46, 47, 51, 99).

[11] Kristin A Cook and James J Thomas. *Illuminating the Path: The Research and Development Agenda for Visual Analytics*. osti.gov/biblio/912515. Pacific Northwest National Lab (PNNL), 2005 (cited on pages 3, 55).

[12] Leland Wilkinson and Michael Friendly. 'The History of the Cluster Heat Map'. In: *The American Statistician* 63.2 (2009), pp. 179–184. DOI: 10.1198/tas.2009.0033 (cited on page 5).

[13] J. M. Chambers. *Graphical Methods for Data Analysis*. New York: Chapman and Hall/CRC, 1983, p. 410. DOI: 10.1201/9781351072304 (cited on page 5).

[14] Charu C. Aggarwal, Alexander Hinneburg, and Daniel A. Keim. 'On the Surprising Behavior of Distance Metrics in High Dimensional Space'. In: *Proc. Database Theory — ICDT*. Ed. by Jan Van den Bussche and Victor Vianu. Springer Berlin Heidelberg, 2001. DOI: 10.1007/3-540-44503-X_27 (cited on page 5).

[15] Allison Marie Horst, Alison Presmanes Hill, and Kristen B Gorman. *palmerpenguins: Palmer Archipelago (Antarctica) penguin data*. R package version 0.1.0. 2020. DOI: 10.5281/zenodo.3960218. URL: https://allisonhorst.github.io/palmerpenguins/ (cited on page 6).

[16] Luis Gustavo Nonato and Michaël Aupetit. 'Multidimensional Projection for Visual Analytics: Linking Techniques with Distortions, Tasks, and Layout Enrichment'. In: *IEEE Transactions on Visualization and Computer Graphics* 25.8 (2019), pp. 2650–2673. DOI: 10.1109/TVCG.2018.2846735 (cited on pages 6, 11, 12, 99).

[17] Mateus Espadoto et al. 'Toward a Quantitative Survey of Dimension Reduction Techniques'. In: *IEEE Transactions on Visualization and Computer Graphics* 27.3 (2021), pp. 2153–2173. DOI: 10.1109/TVCG.2019.2944182 (cited on pages 6, 11, 22, 25, 91).

[18] Cyril de Bodt et al. *Low-dimensional embeddings of high-dimensional data*. arXiv preprint. 2025. DOI: 10.48550/arXiv.2508.15929 (cited on pages 6, 8, 11).

[19] John A. Lee and Michel Verleysen, eds. *Nonlinear Dimensionality Reduction*. 1st ed. New York, NY: Springer, 2007. DOI: 10.1007/978-0-387-39351-3 (cited on page 6).

[20] Sylvain Lespinats, Benoit Colange, and Denys Dutykh, eds. *Nonlinear Dimensionality Reduction Techniques*. 1st ed. New York, NY: Springer, 2022. DOI: 10.1007/978-3-030-81026-9 (cited on page 6).

[21] Benyamin Ghojogh et al. *Elements of Dimensionality Reduction and Manifold Learning*. 1st ed. Cham: Springer, 2023. DOI: 10.1007/978-3-031-10602-6 (cited on page 6).

[22] I. T. Jolliffe. *Principal Component Analysis*. 2nd ed. Springer, 2002. DOI: 10.1007/b98835 (cited on pages 6, 7).

[23] Ingwer Borg and Patrick J. F. Groenen. *Modern Multidimensional Scaling*. 2nd ed. Springer, 2005. DOI: 10.1007/0-387-28981-X (cited on page 7).

[24] J.W. Sammon. 'A Nonlinear Mapping for Data Structure Analysis'. In: *IEEE Transactions on Computers* C-18.5 (1969), pp. 401–409. DOI: 10.1109/T-C.1969.222678 (cited on page 7).

[25] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. URL: http://www.deeplearningbook.org. MIT Press, 2016 (cited on page 7).

[26] J. B. Tenenbaum, V. De Silva, and J. C. Langford. 'A global geometric framework for nonlinear dimensionality reduction'. In: *Science* 290.5500 (2000), pp. 2319–2323. DOI: 10.1126/science.290.5500.2319 (cited on pages 7, 13, 41).

[27] Ronald R. Coifman and Stéphane Lafon. 'Diffusion maps'. In: *Applied and Computational Harmonic Analysis* 21.1 (2006). Special Issue: Diffusion Maps and Wavelets, pp. 5–30. DOI: 10.1016/j.acha.2006.04.006 (cited on page 7).

[28] Leland McInnes, John Healy, and James Melville. *UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction*. arXiv preprint. 2018. DOI: 10.48550/arXiv.1802.03426 (cited on pages 7, 9, 11, 12, 15, 21, 29, 46).

[29] Laurens van der Maaten and Geoffrey Hinton. 'Visualizing data using t-SNE'. In: *Journal of Machine Learning Research* 9 (2008). URL: jmlr.org/vandermaaten08a, pp. 2579–2605 (cited on pages 7, 9, 11, 12, 15, 66).

[30] Laurens van der Maaten. 'Accelerating t-SNE using Tree-Based Algorithms'. In: *Journal of Machine Learning Research* 15 (2014), pp. 3221–3245 (cited on pages 7, 9).

[31] Dmitry Kobak and George C. Linderman. 'Initialization is critical for preserving global data structure in both t-SNE and UMAP'. In: *Nature Biotechnology* 39.2 (2021), pp. 156–157. DOI: 10.1038/s41587-020-00809-z (cited on page 8).

[32] Jian Tang et al. 'Visualizing Large-scale and High-dimensional Data'. In: *Proc. WWW*. 2016, pp. 287–297. DOI: 10.1145/2872427.2883041 (cited on page 9).

[33] Pak Chung Wong and R. Daniel Bergeron. '30 Years of Multidimensional Multivariate Visualization'. In: *Scientific Visualization, Overviews, Methodologies, and Techniques*. New York: IEEE, 1997, pp. 3–33 (cited on page 11).

[34] R. Fuchs and H. Hauser. 'Visualization of Multi-Variate Scientific Data'. In: *Computer Graphics Forum* 28.6 (2009), pp. 1670–1690. DOI: 10.1111/j.1467-8659.2009.01429.x (cited on pages 11, 16).

[35] Johannes Kehrer and Helwig Hauser. 'Visualization and Visual Analysis of Multifaceted Scientific Data: A Survey'. In: *IEEE Transactions on Visualization and Computer Graphics* 19.3 (2013), pp. 495–513. DOI: 10.1109/TVCG.2012.110 (cited on pages 11, 16).

[36] Shusen Liu et al. 'Visualizing High-Dimensional Data: Advances in the Past Decade'. In: *IEEE Transactions on Visualization and Computer Graphics* 23.3 (2017), pp. 1249–1268. DOI: 10.1109/TVCG.2016.2640960 (cited on pages 11, 30).

[37] Miriah Meyer et al. 'MulteeSum: A tool for comparative spatial and temporal gene expression data'. In: *IEEE Transactions on Visualization and Computer Graphics* 16.6 (2010), pp. 908–917. DOI: 10.1109/TVCG.2010.137 (cited on page 11).

[38] C Eichner et al. 'Feature-Based Visual Analytics for Studying Simulations of Dynamic Bi-Stable Spatial Systems'. In: *Proc. EuroVA*. 2013. DOI: 10.2312/PE.EuroVAST.EuroVA13.025-029 (cited on pages 11, 20).

[39] Liang Zhou and Charles Hansen. 'Transfer function design based on user selected samples for intuitive multivariate volume exploration'. In: *Proc. PacificVis*. 2013, pp. 73–80. DOI: 10.1109/PacificVis.2013.6596130 (cited on page 11).

[40] Ahmed Eldawy, Mohamed F. Mokbel, and Christopher Jonathan. 'HadoopViz: A MapReduce Framework for Extensible Visualization of Big Spatial Data'. In: *Proc. ICDE*. New York: IEEE, 2016, pp. 601–612. DOI: 10.1109/ICDE.2016.7498274 (cited on page 11).

[41] Trevor Manz et al. 'Viv: Multiscale Visualization of High-Resolution Multiplexed Bioimaging Data on the Web'. In: *Nature Methods* 19.5 (2022), pp. 515–516. DOI: 10.1038/s41592-022-01482-7 (cited on page 11).

[42] Leslie Solorzano, Gabriele Partel, and Carolina Wählby. 'TissUUmaps: Interactive Visualization of Large-Scale Spatial Gene Expression and Tissue Morphology Data'. In: *Bioinformatics* 36.15 (2020), pp. 4363–4365. DOI: 10.1093/bioinformatics/btaa541 (cited on page 11).

[43] Won-Ki Jeong et al. 'Interactive Histology of Large-Scale Biomedical Image Stacks'. In: *IEEE Transactions on Visualization and Computer Graphics* 16.6 (2010), pp. 1386–1395. DOI: 10.1109/TVCG.2010.168 (cited on page 11).

[44] Jesper Molin et al. *Scale Stain: Multi-Resolution Feature Enhancement in Pathology Visualization*. arXiv preprint. 2016. DOI: 10.48550/arXiv.1610.04141 (cited on page 11).

[45] Walid M. Abdelmoula et al. 'Data-driven identification of prognostic tumor subpopulations using spatially mapped t-SNE of Mass spectrometry imaging data'. In: *Proceedings of the National Academy of Sciences of the United States of America* 113.43 (2016), pp. 12244–12249. DOI: 10.1073/pnas.1510227113 (cited on page 11).

[46] Marina Evers, Karim Huesmann, and Lars Linsen. 'Uncertainty-aware Visualization of Regional Time Series Correlation in Spatio-temporal Ensembles'. In: *Computer Graphics Forum* 40.3 (2021), pp. 519–530. DOI: 10.1111/CGF.14326 (cited on page 11).

[47] Robert Krueger et al. 'Facetto: Combining Unsupervised and Supervised Learning for Hierarchical Phenotype Analysis in Multi-Channel Image Data'. In: *IEEE Transactions on Visualization and Computer Graphics* 26.1 (2020), pp. 227–237. DOI: 10.1109/tvcg.2019.2934547 (cited on pages 12, 29, 30, 55).

[48] Antonios Somarakis et al. 'ImaCytE: Visual Exploration of Cellular Microenvironments for Imaging Mass Cytometry Data'. In: *IEEE Transactions on Visualization and Computer Graphics* 27.1 (2019), pp. 1–1. DOI: 10.1109/TVCG.2019.2931299 (cited on pages 12, 26, 30).

[49] Mark van de Ruit, Markus Billeter, and Elmar Eisemann. 'An Efficient Dual-Hierarchy t-SNE Minimization'. In: *IEEE Transactions on Visualization and Computer Graphics* 28.1 (2022), pp. 614–622. DOI: 10.1109/TVCG.2021.3114817 (cited on page 12).

[50] Stephen Ingram, Tamara Munzner, and Marc Olano. 'Glimmer: Multilevel MDS on the GPU'. In: *IEEE Transactions on Visualization and Computer Graphics* 15.2 (2009), pp. 249–261. DOI: 10.1109/TVCG.2008.85 (cited on page 12).

[51] F.V. Paulovich and R. Minghim. 'HiPP: A Novel Hierarchical Point Placement Strategy and its Application to the Exploration of Document Collections'. In: *IEEE Transactions on Visualization and Computer Graphics* 14.6 (2008), pp. 1229–1236. DOI: 10.1109/tvcg.2008.138 (cited on pages 12, 29, 39).

[52] N. Pezzotti et al. 'Hierarchical Stochastic Neighbor Embedding'. In: *Computer Graphics Forum* 35.3 (2016), pp. 21–30. DOI: 10.1111/cgf.12878 (cited on pages 12, 29, 34, 36, 39, 43, 53, 64, 66, 99).

[53] Wilson E. Marcílio-Jr et al. 'HUMAP: Hierarchical Uniform Manifold Approximation and Projection'. In: *IEEE Transactions on Visualization and Computer Graphics* 31.9 (2025), pp. 5741–5753. DOI: 10.1109/TVCG.2024.3471181 (cited on pages 12, 29, 39, 43, 52, 53).

[54] Manik Kuchroo et al. 'Multiscale PHATE Identifies Multimodal Signatures of COVID-19'. In: *Nature Biotechnology* 40.5 (2022), pp. 681–691. DOI: 10.1038/s41587-021-01186-x (cited on pages 12, 52, 53).

[55] Kevin R. Moon et al. 'Visualizing Structure and Transitions in High-Dimensional Biological Data'. In: *Nature Biotechnology* 37.12 (2019), pp. 1482–1492. DOI: 10.1038/s41587-019-0336-3 (cited on pages 12, 52, 53).

[56] Stéphane Lafon and Ann B. Lee. 'Diffusion Maps and Coarse-Graining: A Unified Framework for Dimensionality Reduction, Graph Partitioning, and Data Set Parameterization'. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 28.9 (2006), pp. 1393–1403. DOI: 10.1109/TPAMI.2006.184 (cited on pages 12, 13, 41, 43).

[57] Donatello Conte et al. 'How and Why Pattern Recognition and Computer Vision Applications Use Graphs'. In: *Applied Graph Theory in Computer Vision and Pattern Recognition*. Ed. by Abraham Kandel, Horst Bunke, and Mark Last. Springer Berlin Heidelberg, 2007, pp. 85–135. DOI: 10.1007/978-3-540-68020-8_4 (cited on page 13).

[58] Andreas Kerren, Helen C. Purchase, and Matthew O. Ward, eds. *Multivariate Network Visualization*. 1st ed. Vol. 8380. Lecture Notes in Computer Science. Cham, Switzerland: Springer International Publishing, 2014, pp. XVI, 237. DOI: 10.1007/978-3-319-06793-3 (cited on pages 13, 72).

[59] Helen Gibson and Paul Vickers. *graphTPP: A multivariate based method for interactive graph layout and analysis*. arXiv preprint. 2017. DOI: 10.48550/arXiv.1712.05644 (cited on page 13).

[60] Yao Yang Leow, Thomas Laurent, and Xavier Bresson. 'GraphTSNE: A Visualization Technique for Graph-Structured Data'. In: *Proc. ICLR Workshop on Representation Learning on Graphs and Manifolds*. arXiv document: 1904.06915. 2019 (cited on page 13).

[61] Rafael M Martins et al. 'MVN-Reduce: Dimensionality Reduction for the Visual Analysis of Multivariate Networks'. In: *Proc. EuroVis*. 2017. DOI: 10.2312/eurovisshort.20171126 (cited on page 13).

[62] Shiyu Chang et al. 'Heterogeneous Network Embedding via Deep Architectures'. In: *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2015). DOI: 10.1145/2783258 (cited on page 13).

[63] Megha Khosla, Vinay Setty, and Avishek Anand. 'A Comparative Study for Unsupervised Network Representation Learning'. In: *IEEE Transactions on Knowledge and Data Engineering* 33.5 (2021), pp. 1807–1818. DOI: 10.1109/TKDE.2019.2951398 (cited on pages 13, 41).

[64] Aditya Grover and Jure Leskovec. 'Node2vec: Scalable Feature Learning for Networks'. In: *Proc. KDD*. 2016, pp. 855–864. DOI: 10.1145/2939672.2939754 (cited on pages 13, 41, 53).

[65] J. F. Kruiger et al. 'Graph Layouts by T-SNE'. In: *Computer Graphics Forum* 36.3 (2017), pp. 283–294. DOI: 10.1111/cgf.13187 (cited on pages 13, 41).

[66] John Aldo Lee and Michel Verleysen. 'Nonlinear Dimensionality Reduction of Data Manifolds with Essential Loops'. In: *Neurocomputing*. Geometrical Methods in Neural Networks and Learning 67 (2005), pp. 29–53. DOI: 10.1016/j.neucom.2004.11.042 (cited on pages 13, 41, 43).

[67] F. Göbel and A. A. Jagers. 'Random Walks on Graphs'. In: *Stochastic Processes and their Applications* 2.4 (1974), pp. 311–336. DOI: 10.1016/0304-4149(74)90001-5 (cited on pages 13, 41).

[68] Keenan Crane et al. *A Survey of Algorithms for Geodesic Paths and Distances*. arXiv preprint. 2020. DOI: 10.48550/arXiv.2007.10430 (cited on pages 13, 41).

[69] A. Vieth et al. 'Incorporating Texture Information into Dimensionality Reduction for High-Dimensional Images'. In: *Proc. PacificVis*. New York: IEEE, 2022, pp. 11–20. DOI: 10.1109/pacificvis53943.2022.00010 (cited on pages 15, 36, 68).

[70] Thomas Höllt et al. 'Cytosplore: Interactive Visual Single-Cell Profiling of the Immune System'. In: *Eurographics 2019 - Dirk Bartz Prize*. Ed. by Stefan Bruckner and Steffen Oeltze-Jafra. The Eurographics Association, 2019. DOI: 10.2312/egm.20191032 (cited on page 15).

[71] R. A. Leite et al. 'EVA: Visual Analytics to Identify Fraudulent Events'. In: *IEEE Transactions on Visualization and Computer Graphics* 24.1 (2018), pp. 330–339. DOI: 10.1109/TVCG.2017.2744758 (cited on page 15).

[72] Michael Sedlmair, Tamara Munzner, and Melanie Tory. 'Empirical Guidance on Scatterplot and Dimension Reduction Technique Choices'. In: *IEEE Transactions on Visualization and Computer Graphics* 19.12 (2013), pp. 2634–2643. DOI: 10.1109/TVCG.2013.153 (cited on page 15).

[73] Matthias Alfeld et al. 'Joint Data Treatment for Vis–NIR Reflectance Imaging Spectroscopy and XRF Imaging Acquired in the Theban Necropolis in Egypt by Data Fusion and t-SNE'. In: *Comptes Rendus Physique* 19.7 (2018), pp. 625–635. DOI: 10.1016/j.crhy.2018.08.004 (cited on page 15).

[74] Natasja L. de Vries et al. 'Unravelling the complexity of the cancer microenvironment with multidimensional genomic and cytometric technologiess'. In: *Frontiers in Oncology* (2020). DOI: 10.3389/fonc.2020.01254 (cited on page 15).

[75] Anna Halladin-Dąbrowska, Adam Kania, and Dominik Kopeć. 'The t-SNE Algorithm as a Tool to Improve the Quality of Reference Data Used in Accurate Mapping of Heterogeneous Non-Forest Vegetation'. In: *Remote Sensing* 12.1 (2020). DOI: 10.3390/rs12010039 (cited on pages 15, 16).

[76] Boyd Kenkhuis et al. 'Iron Loading is a Prominent Feature of Activated Microglia in Alzheimer's Disease Patients'. In: *Acta Neuropathologica Communications* (2021). DOI: 10.1186/s40478-021-01126-5 (cited on page 15).

[77] Barbara Zitová and Jan Flusser. 'Image registration methods: a survey'. In: *Image and Vision Computing* 21.11 (2003), pp. 977–1000. DOI: 10.1016/S0262-8856(03)00137-9 (cited on pages 15, 17, 18).

[78] Haidekker M. 'Texture Analysis'. In: *Advanced Biomedical Image Analysis*. Wiley, 2010, pp. 236–275. DOI: 10.1002/9780470872093.CH8 (cited on page 16).

[79] Anne Humeau-Heurtier. 'Texture feature extraction methods: A survey'. In: *IEEE Access* 7 (2019), pp. 8975–9000. DOI: 10.1109/ACCESS.2018.2890743 (cited on pages 16, 19).

[80] Christoph Palm. 'Color texture classification by integrative Co-occurrence matrices'. In: *Pattern Recognition* 37.5 (2004), pp. 965–976. DOI: 10.1016/j.patcog.2003.09.010 (cited on page 16).

[81] Chandan Singh, Ekta Walia, and Kanwal Preet Kaur. 'Color texture description with novel local binary patterns for effective image retrieval'. In: *Pattern Recognition* 76 (2018), pp. 50–68. DOI: 10.1016/J.PATCOG.2017.10.021 (cited on page 16).

[82] Sylvain Lefebvre and Hugues Hoppe. 'Appearance-space texture synthesis'. In: *ACM Transactions on Graphics* 25.3 (2006), pp. 541–548. DOI: 10.1145/1141911.1141921 (cited on page 16).

[83] Xuejiao Luo, Leonardo Scandolo, and Elmar Eisemann. 'Texture Browser: Feature-based Texture Exploration'. In: *Computer Graphics Forum* 40.3 (2021), pp. 99–109. DOI: https://doi.org/10.1111/cgf.14292 (cited on page 16).

[84] Mathieu Fauvel et al. 'Advances in spectral-spatial classification of hyperspectral images'. In: *Proceedings of the IEEE* 101.3 (2013), pp. 652–675. DOI: 10.1109/JPROC.2012.2197589 (cited on page 16).

[85] Pedram Ghamisi et al. 'Advances in Hyperspectral Image and Signal Processing: A Comprehensive Overview of the State of the Art'. In: *IEEE Geoscience and Remote Sensing Magazine* 5.4 (2017), pp. 37–78. DOI: 10.1109/MGRS.2017.2762087 (cited on pages 16, 24).

[86] Mauro Dalla Mura et al. 'Morphological attribute Profiles for the Analysis of Very High Resolution Images'. In: *IEEE Transactions on Geoscience and Remote Sensing* 48 (2010), pp. 3747–3762. DOI: 10.1109/TGRS.2010.2048116 (cited on page 16).

[87] Hong Huang et al. 'Spatial-spectral local discriminant projection for dimensionality reduction of hyperspectral image'. In: *ISPRS Journal of Photogrammetry and Remote Sensing* 156 (2019), pp. 77–93. DOI: 10.1016/j.isprsjprs.2019.06.018 (cited on page 16).

[88] Guolan Lu and Baowei Fei. 'Medical hyperspectral imaging: a review'. In: *Journal of Biomedical Optics* 19.1 (2014), p. 010901. DOI: 10.1117/1.jbo.19.1.010901 (cited on page 16).

[89] Dalton Lunga and Okan Ersoy. 'Spherical stochastic neighbor embedding of hyperspectral data'. In: *IEEE Transactions on Geoscience and Remote Sensing* 51.2 (2013), pp. 857–871. DOI: 10.1109/TGRS.2012.2205004 (cited on pages 16, 17).

[90] A. Ardeshir Goshtasby. 'Similarity and Dissimilarity Measures'. In: *Image Registration*. Springer Link, 2012, pp. 7–66. DOI: 10.1007/978-1-4471-2458-0_2 (cited on page 18).

[91] Rajiv Kapoor, Deepak Sharma, and Tarun Gulati. 'State of the art content based image retrieval techniques using deep learning: a survey'. In: *Multimedia Tools and Applications 2021* (2021), pp. 1–23. DOI: 10.1007/S11042-021-11045-1 (cited on page 18).

[92] Chengcai Leng et al. 'Local Feature Descriptor for Image Matching: A Survey'. In: *IEEE Access* 7 (2019), pp. 6424–6434. DOI: 10.1109/ACCESS.2018.2888856 (cited on page 18).

[93] Jiayi Ma et al. 'Image Matching from Handcrafted to Deep Features: A Survey'. In: *International Journal of Computer Vision 2020 129:1* 129.1 (2020), pp. 23–79. DOI: 10.1007/S11263-020-01359-2 (cited on page 18).

[94] Naomi Altman and Martin Krzywinski. 'The curse(s) of dimensionality'. In: *Nature Methods* 15.6 (2018), pp. 399–400. DOI: 10.1038/s41592-018-0019-x (cited on page 18).

[95] Adrien Depeursinge, Omar S. Al-Kadi, and J. Ross Mitchell. *Biomedical texture analysis: Fundamentals, tools and challenges*. Elsevier, 2017, pp. 1–415. DOI: 10.1016/C2016-0-01903-4 (cited on page 19).

[96] Manfred M. Fischer and Arthur Getis, eds. *Handbook of Applied Spatial Analysis: Software Tools, Methods and Applications*. 1st ed. Springer Berlin Heidelberg, 2010, pp. XV, 811. DOI: 10.1007/978-3-642-03647-7 (cited on page 19).

[97] David J. Heeger and James R. Bergen. 'Pyramid-Based Texture Analysis/Synthesis'. In: *Proc. SIGGRAPH*. 1995, pp. 229–238. DOI: 10.1145/218380.218446 (cited on page 19).

[98] Leon Gatys, Alexander Ecker, and Matthias Bethge. 'A Neural Algorithm of Artistic Style'. In: *Journal of Vision* 16.12 (2016). arXiv document: 1508.06576, pp. 326–326. DOI: 10.1167/16.12.326 (cited on page 19).

[99] Will Equitz et al. 'Efficient Color Histogram Indexing for Quadratic Form Distance Functions'. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 17.7 (1995), pp. 729–736. DOI: 10.1109/34.391417 (cited on page 19).

[100] Euisun Choi and Chulhee Lee. 'Feature extraction based on the Bhattacharyya distance'. In: *Pattern Recognition* 36.8 (2003), pp. 1703–1709. DOI: 10.1016/S0031-3203(03)00035-9 (cited on page 20).

[101] Daniel P. Huttenlocher, Gregory A. Klanderman, and William J. Rucklidge. 'Comparing Images Using the Hausdorff Distance'. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 15.9 (1993), pp. 850–863. DOI: 10.1109/34.232073 (cited on page 20).

[102] Haoqiang Fan, Hao Su, and Leonidas Guibas. 'A point set generation network for 3D object reconstruction from a single image'. In: *Proc. CVPR*. 2017, pp. 2463–2471. DOI: 10.1109/CVPR.2017.264 (cited on page 20).

[103] Nicola Pezzotti et al. 'Approximated and user steerable tSNE for progressive visual analytics'. In: *IEEE transactions on visualization and computer graphics* 23.7 (2017), pp. 1739–1752. DOI: 10.1109/TVCG.2016.2570755 (cited on pages 21, 23, 47).

[104] T. Höllt et al. 'Focus+Context Exploration of Hierarchical Embeddings'. In: *Computer Graphics Forum* 38.3 (2019), pp. 569–579. DOI: 10.1111/cgf.13711 (cited on pages 22, 31, 32, 35, 36, 38, 54, 99).

[105] Nicola Pezzotti et al. 'GPGPU Linear Complexity t-SNE Optimization'. In: *IEEE Transactions on Visualization and Computer Graphics* 26.1 (2020), pp. 1172–1181. DOI: 10.1109/tvcg.2019.2934307 (cited on pages 23, 67, 107).

[106] Yu A. Malkov and D. A. Yashunin. 'Efficient and Robust Approximate Nearest Neighbor Search Using Hierarchical Navigable Small World Graphs'. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 42.4 (2020), pp. 824–836. DOI: 10.1109/TPAMI.2018.2889473 (cited on pages 23, 47).

[107] Alexander Vieth. *Spatial Information in Dimensionality Reduction (Spidr)*. URL: github.com/biovault/Spidr. 2022. DOI: 10.5281/zenodo.6120879 (cited on page 23).

[108] Charlotte Giesen et al. 'Highly multiplexed imaging of tumor tissues with subcellular resolution by mass cytometry'. In: *Nature Methods* 11.4 (2014), pp. 417–422. DOI: 10.1038/nmeth.2869 (cited on page 25).

[109] Alexander Vieth et al. 'Interactions for Seamlessly Coupled Exploration of High-Dimensional Images and Hierarchical Embeddings'. In: *Proc. Vision, Modeling, and Visualization*. Ed. by Michael Guthe and Thorsten Grosch. The Eurographics Association, 2023. DOI: 10.2312/vmv.20231227 (cited on pages 29, 39, 48).

[110] Laurens van der Maaten. 'Accelerating T-SNE Using Tree-Based Algorithms'. In: *Journal of Machine Learning Research* 15.93 (2014), pp. 3221–3245 (cited on pages 29, 44, 46).

[111] Marc Vermeulen et al. 'Application of Uniform Manifold Approximation and Projection (UMAP) in Spectral Imaging of Artworks'. In: *Spectrochimica Acta - Part A: Molecular and Biomolecular Spectroscopy* 252 (2021), p. 119547. DOI: 10.1016/j.saa.2021.119547 (cited on page 29).

[112] Hong Huang et al. 'Dimensionality Reduction of Hyperspectral Imagery Based on Spatial–Spectral Manifold Learning'. In: *IEEE Transactions on Cybernetics* 50.6 (2020), pp. 2604–2616. DOI: 10.1109/TCYB.2019.2905793 (cited on page 29).

[113] Dmitry Kobak and Philipp Berens. 'The Art of Using T-SNE for Single-Cell Transcriptomics'. In: *Nature Communications* 10.1 (2019), p. 5416. DOI: 10.1038/s41467-019-13056-x (cited on pages 29, 36, 44, 50).

[114] B Shneiderman. 'The Eyes Have It: A Task by Data Type Taxonomy for Information Visualizations'. In: *Proc. Visual Languages*. 1996, pp. 336–343. DOI: 10.1109/VL.1996.545307 (cited on page 29).

[115] David A. Ellsworth, Christopher E. Henze, and Bron C. Nelson. 'Interactive Visualization of High-Dimensional Petascale Ocean Data'. In: *Proc. LDAV*. 2017, pp. 36–44. DOI: 10.1109/LDAV.2017.8231849 (cited on page 30).

[116] France Rose et al. 'PySpacell: A Python Package for Spatial Analysis of Cell Images'. In: *Cytometry Part A* 97.3 (2020), pp. 288–295. DOI: 10.1002/cyto.a.23955 (cited on page 30).

[117] Giovanni Palla et al. 'Squidpy: A Scalable Framework for Spatial Omics Analysis'. In: *Nature Methods 2022 19:2* 19.2 (2022), pp. 171–178. DOI: 10.1038/s41592-021-01358-2 (cited on page 30).

[118] Jared Jessup et al. 'Scope2Screen: Focus+Context Techniques for Pathology Tumor Assessment in Multivariate Image Data'. In: *IEEE Transactions on Visualization and Computer Graphics* 28.1 (2022), pp. 259–269. DOI: 10.1109/TVCG.2021.3114786 (cited on page 30).

[119] Denis Schapiro et al. 'histoCAT: analysis of cell phenotypes and interactions in multiplex image cytometry data'. In: *Nature Methods* 14.9 (2017), pp. 873–876. DOI: 10.1038/nmeth.4391 (cited on page 30).

[120] Dominik Sacha et al. 'Visual Interaction with Dimensionality Reduction: A Structured Literature Analysis'. In: *IEEE Transactions on Visualization and Computer Graphics* 23.1 (2017), pp. 241–250. DOI: 10.1109/TVCG.2016.2598495 (cited on page 30).

[121] Jing Yang, Matthew O. Ward, and Elke A. Rundensteiner. 'Interactive Hierarchical Displays: A General Framework for Visualization and Exploration of Large Multivariate Data Sets'. In: *Computers & Graphics* 27.2 (2003), pp. 265–283. DOI: 10.1016/S0097-8493(02)00283-2 (cited on pages 31, 57).

[122] Mark Sifer. 'User Interfaces for the Exploration of Hierarchical Multi-dimensional Data'. In: *Proc. VAST*. 2006, pp. 175–182. DOI: 10.1109/VAST.2006.261422 (cited on page 31).

[123] Wilson E. Marcílio-Jr et al. 'ExplorerTree: A Focus+Context Exploration Approach for 2D Embeddings'. In: *Big Data Research* 25 (2021). DOI: 10.1016/J.BDR.2021.100239 (cited on page 31).

[124] Niklas Elmqvist and Jean-Daniel Fekete. 'Hierarchical Aggregation for Information Visualization: Overview, Techniques, and Design Guidelines'. In: *IEEE Transactions on Visualization and Computer Graphics* 16.3 (2010), pp. 439–454. DOI: 10.1109/TVCG.2009.84 (cited on pages 31, 34, 37, 99).

[125] Alexander Vieth et al. 'ManiVault: A Flexible and Extensible Visual Analytics Framework for High-Dimensional Data'. In: *IEEE Transactions on Visualization and Computer Graphics* 30.1 (2024), pp. 175–185. DOI: 10.1109/TVCG.2023.3326582 (cited on pages 34, 47, 55).

[126] Jürgen Bernard et al. 'A Survey and Task-Based Quality Assessment of Static 2D Colormaps'. In: *Proc. SPIE 9397, Visualization and Data Analysis*. Ed. by David L. Kao et al. Vol. 9397. SPIE, 2015, p. 93970M. DOI: 10.1117/12.2079841 (cited on pages 36, 48).

[127] Murong Wang et al. 'Superpixel Segmentation: A Benchmark'. In: *Signal Processing: Image Communication* 56 (2017), pp. 28–39. DOI: 10.1016/j.image.2017.04.007 (cited on page 40).

[128] David Stutz, Alexander Hermans, and Bastian Leibe. 'Superpixels: An Evaluation of the State-of-the-Art'. In: *Computer Vision and Image Understanding* 166 (2018), pp. 1–27. DOI: 10.1016/j.cviu.2017.03.007 (cited on pages 40, 50).

[129] Isabela Borlido Barcelos et al. 'A Comprehensive Review and New Taxonomy on Superpixel Segmentation'. In: *ACM Comput. Surv.* 56.8 (2024), 200:1–200:39. DOI: 10.1145/3652509 (cited on page 40).

[130] Pedro F. Felzenszwalb and Daniel P. Huttenlocher. 'Efficient Graph-Based Image Segmentation'. In: *International Journal of Computer Vision* 59.2 (2004), pp. 167–181. DOI: 10.1023/B:VISI.0000022288.19776.77 (cited on page 40).

[131] David R. Thompson et al. 'Superpixel Endmember Detection'. In: *IEEE Transactions on Geoscience and Remote Sensing* 48.11 (2010), pp. 4023–4033. DOI: 10.1109/TGRS.2010.2070802 (cited on page 40).

[132] Xiang Xu et al. 'Regional Clustering-Based Spatial Preprocessing for Hyperspectral Unmixing'. In: *Remote Sensing of Environment* 204 (2018), pp. 333–346. DOI: 10.1016/j.rse.2017.10.020 (cited on page 40).

[133] Radhakrishna Achanta et al. 'SLIC superpixels compared to state-of-the-art superpixel methods'. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 34.11 (2012), pp. 2274–2281. DOI: 10.1109/TPAMI.2012.120 (cited on page 40).

[134] Mirko Paolo Barbato et al. 'Unsupervised segmentation of hyperspectral remote sensing images with superpixels'. In: *Remote Sensing Applications: Society and Environment* 28 (2022), p. 100823. DOI: https://doi.org/10.1016/j.rsase.2022.100823 (cited on pages 40, 51).

[135] Ming-Yu Liu et al. 'Entropy Rate Superpixel Segmentation'. In: *Proc. CVPR*. 2011, pp. 2097–2104. DOI: 10.1109/CVPR.2011.5995323 (cited on page 40).

[136] Yiwei Tang, Liaoying Zhao, and Lang Ren. 'Different Versions of Entropy Rate Superpixel Segmentation For Hyperspectral Image'. In: *Proc. ICSIP*. 2019, pp. 1050–1054. DOI: 10.1109/SIPROCESS.2019.8868344 (cited on page 40).

[137] Ya-Ru Fan. 'Robust Superpixel Segmentation for Hyperspectral-Image Restoration'. In: *Entropy* 25.2 (2023), p. 260. DOI: 10.3390/e25020260 (cited on page 40).

[138] L. Grady. 'Random Walks for Image Segmentation'. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 28.11 (2006), pp. 1768–1783. DOI: 10.1109/TPAMI.2006.233 (cited on page 41).

[139] Jianbing Shen et al. 'Lazy Random Walks for Superpixel Segmentation'. In: *IEEE Transactions on Image Processing* 23.4 (2014), pp. 1451–1462. DOI: 10.1109/TIP.2014.2302892 (cited on page 41).

[140] Xuejing Kang, Lei Zhu, and Anlong Ming. 'Dynamic Random Walk for Superpixel Segmentation'. In: *IEEE Transactions on Image Processing* 29 (2020), pp. 3871–3884. DOI: 10.1109/TIP.2020.2967583 (cited on page 41).

[141] Xing Wei et al. 'Superpixel Hierarchy'. In: *IEEE Transactions on Image Processing* 27.10 (2018), pp. 4838–4849. DOI: 10.1109/TIP.2018.2836300 (cited on pages 41, 42, 45, 53).

[142] Tingman Yan, Xiaolin Huang, and Qunfei Zhao. 'Hierarchical Superpixel Segmentation by Parallel CRTrees Labeling'. In: *IEEE Transactions on Image Processing* 31 (2022), pp. 4719–4732. DOI: 10.1109/TIP.2022.3187563 (cited on page 41).

[143] Andrei C. Jalba, Michel A. Westenberg, and Jos B. T. M. Roerdink. 'Interactive Segmentation and Visualization of DTI Data Using a Hierarchical Watershed Representation'. In: *IEEE Transactions on Image Processing* 24.3 (2015), pp. 1025–1035. DOI: 10.1109/TIP.2015.2390139 (cited on page 42).

[144] Matthijs Douze et al. *The Faiss library*. arXiv preprint. 2024. DOI: 10.48550/arXiv.2401.08281 (cited on page 47).

[145] J. Anthony Gualtieri and Robert F. Cromp. 'Support Vector Machines for Hyperspectral Remote Sensing Classification'. In: *Proc. 27th AIPR Workshop: Advances in Computer-Assisted Recognition*. Vol. 3584. SPIE, 1999, pp. 221–232. DOI: 10.1117/12.339824 (cited on pages 47, 99).

[146] Jia-Ren Lin et al. 'Highly multiplexed immunofluorescence imaging of human tissues and tumors using t-CyCIF and conventional optical microscopes'. In: *eLife* 7 (2018), e31657. DOI: 10.7554/eLife.31657 (cited on page 49).

[147] Clarence Yapp et al. 'Highly multiplexed 3D profiling of cell states and immune niches in human tumors'. In: *Nature Methods* 22.10 (2025), pp. 2180–2193. DOI: 10.1038/s41592-025-02824-x (cited on page 49).

[148] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 'DeepWalk: Online Learning of Social Representations'. In: *Proc. KDD*. New York, NY, USA: Association for Computing Machinery, 2014, pp. 701–710. DOI: 10.1145/2623330.2623732 (cited on page 53).

[149] Jinwoo Kim et al. 'Revisiting Random Walks for Learning on Graphs'. In: *Proc. ICLR*. arXiv document: 2407.01214. 2025, pp. 82497–82547 (cited on page 53).

[150] László Lovász. 'Random Walks on Graphs: A Survey'. In: *Combinatorics, Paul Erdős is eighty*. Ed. by D. Miklós, V. T. Sós, and T. Szőnyi. Vol. 2. János Bolyai Mathematical Society, 1993, pp. 1–46 (cited on page 53).

[151] Daniel Keim et al. 'Visual Analytics: Definition, Process, and Challenges'. In: *Information Visualization: Human-Centered Issues and Perspectives*. Ed. by Andreas Kerren et al. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2008, pp. 154–175. DOI: 10.1007/978-3-540-70956-5_7 (cited on page 55).

[152] Matthew Brehmer and Tamara Munzner. 'A Multi-Level Typology of Abstract Visualization Tasks'. In: *IEEE Transactions on Visualization and Computer Graphics* 19.12 (2013), pp. 2376–2385. DOI: 10.1109/TVCG.2013.124 (cited on page 55).

[153] Heidi Lam, Melanie Tory, and Tamara Munzner. 'Bridging from Goals to Tasks with Design Study Analysis Reports'. In: *IEEE Transactions on Visualization and Computer Graphics* 24.1 (2018), pp. 435–445. DOI: 10.1109/TVCG.2017.2744319 (cited on pages 55, 70).

[154] Yuxin Ma et al. 'Explaining Vulnerabilities to Adversarial Machine Learning through Visual Analytics'. In: *IEEE Transactions on Visualization and Computer Graphics* 26.1 (2020), pp. 1075–1085. DOI: 10.1109/TVCG.2019.2934631 (cited on page 55).

[155] Dong Sun et al. 'PlanningVis: A Visual Analytics Approach to Production Planning in Smart Factories'. In: *IEEE Transactions on Visualization and Computer Graphics* 26.1 (2020), pp. 579–589. DOI: 10.1109/TVCG.2019.2934275 (cited on page 55).

[156] Mingyu Pi et al. 'Visual Cause Analytics for Traffic Congestion'. In: *IEEE Transactions on Visualization and Computer Graphics* 27.3 (2021), pp. 2186–2201. DOI: 10.1109/TVCG.2019.2940580 (cited on page 55).

[157] Jiang Wu et al. 'TacticFlow: Visual Analytics of Ever-Changing Tactics in Racket Sports'. In: *IEEE Transactions on Visualization and Computer Graphics* 28.1 (2022), pp. 835–845. DOI: 10.1109/TVCG.2021.3114832 (cited on page 55).

[158] Harald Piringer, Wolfgang Berger, and Helwig Hauser. 'Quantifying and Comparing Features in High-Dimensional Datasets'. In: *Proc. IV*. New York: IEEE, 2008, pp. 240–245. DOI: 10.1109/IV.2008.17 (cited on pages 55, 57).

[159] Harald Piringer et al. 'A Multi-Threading Architecture to Support Interactive Visual Exploration'. In: *IEEE Transactions on Visualization and Computer Graphics* 15.6 (2009), pp. 1113–1120. DOI: 10.1109/TVCG.2009.110 (cited on pages 55, 57).

[160] Christopher Ahlberg. 'Spotfire: An Information Exploration Environment'. In: *ACM SIGMOD Record* 25.4 (1996). URL: tibco.com, archived webpage, pp. 25–29. DOI: 10.1145/245882.245893 (cited on pages 55, 56).

[161] Christopher Ahlberg and Ben Shneiderman. 'Visual Information Seeking: Tight Coupling of Dynamic Query Filters with Starfield Displays'. In: *Proc. CHI*. New York: ACM, 1994, pp. 313–317. DOI: 10.1145/191666.191775 (cited on page 55).

[162] M.O. Ward. 'XmdvTool: Integrating Multiple Methods for Visualizing Multivariate Data'. In: *Proc. VIS*. URL: davis.wpi.edu/xmdv, archived webpage. New York: IEEE, 1994, pp. 326–333. DOI: 10.1109/VISUAL.1994.346302 (cited on pages 55–57).

[163] Deborah F Swayne et al. 'GGobi: Evolving from XGobi into an Extensible Framework for Interactive Data Visualization'. In: *Computational Statistics & Data Analysis*. Data Visualization 43.4 (2003), pp. 423–444. DOI: 10.1016/S0167-9473(02)00286-4 (cited on pages 55–57).

[164] JAMES Ahrens, BERK Geveci, and CHARLES Law. 'ParaView: An End-User Tool for Large-Data Visualization'. In: *Visualization Handbook*. Ed. by Charles D. Hansen and Chris R. Johnson. Burlington, MA, USA: Butterworth-Heinemann, 2005, pp. 717–731. DOI: 10.1016/B978-012387582-2/50038-1 (cited on pages 55, 57).

[165] Daniel Jönsson et al. 'Inviwo - A Visualization System with Usage Abstraction Levels'. In: *IEEE Transactions on Visualization and Computer Graphics* 26.11 (2020), pp. 3241–3254. DOI: 10.1109/TVCG.2019.2920639 (cited on pages 55, 57, 58).

[166] C. Stolte, D. Tang, and P. Hanrahan. 'Polaris: A System for Query, Analysis, and Visualization of Multidimensional Relational Databases'. In: *IEEE Transactions on Visualization and Computer Graphics* 8.1 (2002), pp. 52–65. DOI: 10.1109/2945.981851 (cited on pages 55, 57).

[167] Tableau Software, LLC. *Tableau*. URL: tableau.com, archived webpage (cited on pages 55, 57).

[168] Mehmet Adil Yalçın, Niklas Elmqvist, and Benjamin B. Bederson. 'Keshif: Rapid and Expressive Tabular Data Exploration for Novices'. In: *IEEE Transactions on Visualization and Computer Graphics* 24.8 (2018), pp. 2339–2352. DOI: 10.1109/TVCG.2017.2723393 (cited on pages 55, 58).

[169] Charles D. Stolper, Adam Perer, and David Gotz. 'Progressive Visual Analytics: User-Driven Visual Exploration of In-Progress Analytics'. In: *IEEE Transactions on Visualization and Computer Graphics* 20.12 (2014), pp. 1653–1662. DOI: 10.1109/TVCG.2014.2346574 (cited on page 55).

[170] Sriram Karthik Badam, Niklas Elmqvist, and Jean-Daniel Fekete. 'Steering the Craft: UI Elements and Visualizations for Supporting Progressive Visual Analytics'. In: *Computer Graphics Forum* 36.3 (2017), pp. 491–502. DOI: 10.1111/cgf.13205 (cited on page 55).

[171] The Qt Company. *Qt*. URL: qt.io, archived webpage (cited on pages 56, 66).

[172] Michael Bostock, Vadim Ogievetsky, and Jeffrey Heer. 'D^3 Data-Driven Documents'. In: *IEEE Transactions on Visualization and Computer Graphics* 17.12 (2011), pp. 2301–2309. DOI: 10.1109/TVCG.2011.185 (cited on pages 56, 67).

[173] Arvind Satyanarayan et al. 'Vega-Lite: A Grammar of Interactive Graphics'. In: *IEEE Transactions on Visualization and Computer Graphics* 23.1 (2017), pp. 341–350. DOI: 10.1109/TVCG.2016.2599030 (cited on pages 56, 58, 67).

[174] Visplore GmbH. *Visplore*. URL: visplore.com, archived webpage (cited on page 57).

[175] Kitware. *Trame*. URL: kitware.github.io/trame, archived webpage (cited on page 57).

[176] Wenqiang Cui. 'Visual Analytics: A Comprehensive Overview'. In: *IEEE Access* 7 (2019), pp. 81555–81573. DOI: 10.1109/ACCESS.2019.2923736 (cited on page 56).

[177] Aindrila Ghosh et al. 'A Comprehensive Review of Tools for Exploratory Analysis of Tabular Industrial Datasets'. In: *Visual Informatics* 2.4 (2018), pp. 235–253. DOI: 10.1016/j.visinf.2018.12.004 (cited on page 56).

[178] Xi Chen et al. 'Composition and Configuration Patterns in Multiple-View Visualizations'. In: *IEEE Transactions on Visualization and Computer Graphics* 27.2 (2021), pp. 1514–1524. DOI: 10.1109/TVCG.2020.3030338 (cited on page 56).

[179] J.-D. Fekete. 'The InfoVis Toolkit'. In: *Proc. INFOVIS*. New York: IEEE, 2004, pp. 167–174. DOI: 10.1109/INFVIS.2004.64 (cited on page 56).

[180] J. Yang et al. 'Visual Hierarchical Dimension Reduction for Exploration of High Dimensional Datasets'. In: *Proc. VisSym*. Eindhoven, NL: The Eurographics Association, 2003. DOI: 10.2312/VisSym/VisSym03/019-028 (cited on page 57).

[181] Qingguang Cui et al. 'Measuring Data Abstraction Quality in Multiresolution Visualizations'. In: *IEEE Transactions on Visualization and Computer Graphics* 12.5 (2006), pp. 709–716. DOI: 10.1109/TVCG.2006.161 (cited on page 57).

[182] L. Bavoil et al. 'VisTrails: Enabling Interactive Multiple-View Visualizations'. In: *Proc. VIS*. New York: IEEE, 2005, pp. 135–142. DOI: 10.1109/VISUAL.2005.1532788 (cited on page 57).

[183] Slicer Community. *3D Slicer*. URL: slicer.org, archived webpage (cited on page 57).

[184] Hank Childs et al. *VisIt: An End-User Tool for Visualizing and Analyzing Very Large Data*. Chapman and Hall/CRC, 2012, pp. 395–410. DOI: 10.1201/b12985-29 (cited on page 57).

[185] Naohisa Sakamoto and Koji Koyamada. 'KVS: A Simple and Effective Framework for Scientific Visualization'. In: *Journal of Advanced Simulation in Science and Engineering* 2.1 (2015), pp. 76–95. DOI: 10.15748/jasse.2.76 (cited on page 57).

[186] Will Schroeder, Ken Martin, and Bill Lorensen. *The visualization toolkit*. 4th. URL: gitlab.kitware.com/vtk/textbook, archived pdf. Kitware, 2006 (cited on page 57).

[187] Detlev Stalling, Malte Westerhoff, and Hans-Christian Hege. 'amira: A Highly Interactive System for Visual Data Analysis'. In: *Visualization Handbook*. Ed. by Charles D. Hansen and Chris R. Johnson. Burlington: Butterworth-Heinemann, 2005, pp. 749–767. DOI: 10.1016/B978-012387582-2/50040-X. (Visited on 02/07/2023) (cited on page 57).

[188] Thermo Fisher Scientific. *Amira*. URL: thermofisher.com, archived webpage (cited on page 57).

[189] Johannes Sorger et al. 'A Taxonomy of Integration Techniques for Spatial and Non-Spatial Visualizations'. In: *Proc. VMV*. Eindhoven, NL: The Eurographics Association, 2015. DOI: 10.2312/vmv.20151258 (cited on page 57).

[190] Arvind Satyanarayan and Jeffrey Heer. 'Lyra: An Interactive Visualization Design Environment'. In: *Computer Graphics Forum* 33.3 (2014), pp. 351–360. DOI: 10.1111/cgf.12391 (cited on page 57).

[191] Jonathan Zong et al. 'Lyra 2: Designing Interactive Visualizations by Demonstration'. In: *IEEE Transactions on Visualization and Computer Graphics* 27.2 (2021), pp. 304–314. DOI: 10.1109/TVCG.2020.3030367 (cited on page 58).

[192] Donghao Ren, Tobias Höllerer, and Xiaoru Yuan. 'iVisDesigner: Expressive Interactive Design of Information Visualizations'. In: *IEEE Transactions on Visualization and Computer Graphics* 20.12 (2014), pp. 2092–2101. DOI: 10.1109/TVCG.2014.2346291 (cited on page 58).

[193] Jeffrey Heer, Stuart K. Card, and James A. Landay. 'Prefuse: A Toolkit for Interactive Information Visualization'. In: *Proc. CHI*. New York: ACM, 2005, pp. 421–430. DOI: 10.1145/1054972.1055031 (cited on page 58).

[194] Kresimir Matkovic et al. 'ComVis: A Coordinated Multiple Views System for Prototyping New Visualization Technology'. In: *Proc. IV*. New York: IEEE, 2008, pp. 215–220. DOI: 10.1109/IV.2008.87 (cited on page 58).

[195] Plotly Technologies Inc. *Dash*. URL: dash.plotly.com, archived webpage (cited on page 58).

[196] Thomas Kluyver et al. *Jupyter Notebooks-a publishing format for reproducible computational workflows*. URL: jupyter.org, archived webpage, github.com/voila-dashboards/voila. 2016 (cited on page 58).

[197] T. Höllt et al. 'Cytosplore: Interactive Immune Cell Phenotyping for Large Single-Cell Datasets'. In: *Computer Graphics Forum* 35.3 (2016), pp. 171–180. DOI: 10.1111/cgf.12893 (cited on page 58).

[198] Andra Popa et al. 'Visual Analysis of RIS Data for Endmember Selection'. In: *Proc. GCH*. Eindhoven, NL: The Eurographics Association, 2022. DOI: 10.2312/gch.20221233 (cited on pages 58, 68).

[199] Julian Thijssen, Zonglin Tian, and Alexandru Telea. 'Scaling Up the Explanation of Multidimensional Projections'. In: *Proc. EuroVA*. Ed. by Marco Angelini and Mennatallah El-Assady. The Eurographics Association, 2023. DOI: 10.2312/eurova.20231098 (cited on pages 58, 68).

[200] Chang Li et al. 'SpaceWalker enables interactive gradient exploration for spatial transcriptomics data'. In: *Cell Reports Methods* 3.12 (2023). DOI: 10.1016/j.crmeth.2023.100645 (cited on pages 58, 68).

[201] A. Buja et al. 'Interactive Data Visualization Using Focusing and Linking'. In: *Proc. VIS*. New York: IEEE, 1991, pp. 156–163. DOI: 10.1109/VISUAL.1991.175794 (cited on page 62).

[202] C. Plaisant, D. Carr, and B. Shneiderman. 'Image-browser taxonomy and guidelines for designers'. In: *IEEE Software* 12.2 (1995), pp. 21–32. DOI: 10.1109/52.368260 (cited on page 62).

[203] Ove Daae Lampe and Helwig Hauser. 'Interactive visualization of streaming data with Kernel Density Estimation'. In: *Proc. PacificVis*. New York: IEEE, 2011. DOI: 10.1109/pacificvis.2011.5742387 (cited on pages 63, 67).

[204] githubuser0xFFFF. *Advanced Docking System for Qt*. URL: github.com/Qt-Advanced-Docking-System, archived webpage (cited on page 65).

[205] Nicola Pezzotti. *High Dimensional Inspector*. URL: github.com/Nicola17/High-Dimensional-Inspector. 2018. DOI: 10.5281/zenodo.1303855 (cited on pages 66, 68, 107).

[206] Jason D. Wolfe and Sarah R. Black. *Hyperspectral Analytics in ENVI Target Detection and Spectral Mapping Methods*. Tech. rep. URL: l3harrisgeospatial.com/Whitepaper.pdf, archived pdf. Harris Corporation, 2018, p. 40 (cited on page 68).

[207] Mark Gahegan. 'Visual Exploration and Explanation in Geography Analysis with Light'. In: *Geographic Data Mining and Knowledge Discovery*. Ed. by Harvey J. Miller and Jiawei Han. 2nd. CRC Press, 2009. Chap. 11, pp. 291–324. DOI: 10.1201/9781420073980-11 (cited on page 69).

[208] David A. Landgrebe. *HYDICE image of Washington DC Mall*. URL: engineering.purdue.edu, archived webpage (cited on page 70).

[209] Jeroen Eggermont et al. *Cytosplore Viewer*. URL: viewer.cytosplore.org, archived webpage (cited on page 71).

[210] Trygve E. Bakken, Nikolas L. Jorstad, and Qiwen Hu et al. 'Comparative Cellular Analysis of Motor Cortex in Human, Marmoset and Mouse'. In: *Nature* 598.7879 (2021), pp. 111–119. DOI: 10.1038/s41586-021-03465-8 (cited on pages 71, 72).

[211] Eric D. Ragan et al. 'Characterizing Provenance in Visualization and Data Analysis: An Organizational Framework of Provenance Types and Purpose'. In: *IEEE Transactions on Visualization and Computer Graphics* 22.1 (2016), pp. 31–40. DOI: 10.1109/TVCG.2015.2467551 (cited on page 72).

[212] Z. Huang et al. 'VA + Embeddings STAR: A State-of-the-Art Report on the Use of Embeddings in Visual Analytics'. In: *Computer Graphics Forum* 42.3 (2023), pp. 539–571. DOI: 10.1111/cgf.14859 (cited on page 74).

[213] Hyeon Jeon et al. 'Unveiling High-dimensional Backstage: A Survey for Reliable Visual Analytics with Dimensionality Reduction'. In: *Proc. CHI 2025*. Proc. CHI. New York, NY, USA: Association for Computing Machinery, 2025, pp. 1–24. DOI: 10.1145/3706598.3713551 (cited on page 74).

[214] G. Bradski. 'The OpenCV Library'. In: *Dr. Dobb's Journal of Software Tools* (2000). URL: opencv.org, archived webpage (cited on page 91).

[215] Leland McInnes et al. 'UMAP: Uniform Manifold Approximation and Projection'. In: *The Journal of Open Source Software* 3.29 (2018), p. 861 (cited on page 91).

[216] F. Pedregosa et al. 'Scikit-learn: Machine Learning in Python'. In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830 (cited on page 91).

[217] Joshua Tenenbaum. 'Mapping a Manifold of Perceptual Observations'. In: *Proc. NIPS*. Ed. by M. Jordan, M. Kearns, and S. Solla. Vol. 10. URL: scikit-learn.org, official pdf. MIT Press, 1997, pp. 682–688 (cited on page 107).

[218] Samer A. Nene, Shree K. Nayar, and Hiroshi Murase. *Columbia Object Image Library (COIL-20)*. Tech. rep. CUCS-005-96. URL: cs.columbia.edu. Department of Computer Science, Columbia University, 1996 (cited on page 107).

[219] Y. LeCun et al. 'Gradient-based learning applied to document recognition'. In: *Proceedings of the IEEE* 86.11 (1998). URL: yann.lecun.com/exdb/mnist, pp. 2278–2324. DOI: 10.1109/5.726791 (cited on page 107).

[220] Han Xiao, Kashif Rasul, and Roland Vollgraf. *Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms*. URL: github.com/zalandoresearch/fashion-mnist. arXiv preprint. 2017. DOI: 10.48550/arXiv.1708.07747 (cited on page 107).

[221] Grace X. Y. Zheng, Jessica M. Terry, and Jason H. Bielas. 'Massively Parallel Digital Transcriptional Profiling of Single Cells'. In: *Nature Communications* 8.1 (2017). URL: file.biolab.si/opentsne/benchmark, p. 14049. DOI: 10.1038/ncomms14049 (cited on page 107).

Appendix

A. Supplement: Spidr

SA1: Computation settings

For all t-SNE computations we set used the following HDLib parameters:

- `Exaggeration=250`
- `exponential decay=40`
- `number of trees=4`
- `number of checks=1024`

When computing approximated nearest neighbors with HNSWlib we use the default parameters `M=16` and `ef_construction=200` as well as the random seed 0.
We apply bilateral filtering using OpenCV [214] with the settings

- `sigmaColor=75`
- `sigmaSpace=75`
- `d=5`

For the quantitative analysis, we compute the k-nearest neighbor hit, as described by Espadoto et al [17]. In brief, for labelled data, for every point in the low-dimensional embedding, we compute the fraction of the k nearest neighbors in the low-dimensional embedding have the same label as the probed point. This fraction is then averaged for all points in the dataset. For the synthetic data, we define the ground truth by separating the checkered areas and homogeneous areas as shown in Figure A.2a. For the Indian Pines dataset, we use the 16-class ground truth data provided with the original data. For the synthetic data, we limit k to $k = [1..63]$, as the inner, homogeneous squares in the image cover 64 pixels, meaning larger values for k would include more neighbors than pixels existing for the given label. For the Indian Pines data, we compute the k-nearest neighbor hit for $k = [1..100]$. For all, we do not include the probed point in the k-nearest neighbors.

SA2: Weighted feature computation and weighted Chamfer distance

With weights $\mathbf{w}$ that sum to 1 and weighted $\mu^* = [\mu_1^*, \ldots, \mu_C^*]$, an entry σ_{jk} of the covariance matric $\mathbf{\Sigma_i}$ is given by:

$$\sigma_{jk} = \sum_{\mathbf{q} \in \mathcal{N}_\mathbf{i}^{S,\eta}} \mathbf{w}(\mathbf{i}-\mathbf{q})(a_{\mathbf{q}j} - \mu_j^*)(a_{\mathbf{q}k} - \mu_k^*)^T. \tag{A.1}$$

where the weighted means are $\mu_c^* = \sum_{\mathbf{q} \in \mathcal{N}_\mathbf{i}^{S,\eta}} \mathbf{w}(\mathbf{i}-\mathbf{q}) a_{\mathbf{q}c}$.
For weighting the covariance matrix feature, including the channel-wise means, one only needs to introduce the weights $\mathbf{w}$ in the calculation of the expected value as probabilities.
The Chamfer point cloud distance from Equation 4.6 can be extended by weighting the minimal distances from each point in the first to the second neighborhood as shown in Equation:

$$\begin{aligned} d_s^{PC}(\mathbf{Z_i}, \mathbf{Z_j}) = & \frac{1}{|\mathcal{N}_\mathbf{i}^{S,\eta}|} \sum_{\mathbf{q} \in \mathcal{N}_\mathbf{i}^{S,\eta}} w(\mathbf{i}-\mathbf{q}) \min_{\mathbf{p} \in \mathcal{N}_\mathbf{j}^{S,\eta}} \|\mathbf{a_q} - \mathbf{a_p}\|_2^2 + \\ & \frac{1}{|\mathcal{N}_\mathbf{j}^{S,\eta}|} \sum_{\mathbf{p} \in \mathcal{N}_\mathbf{j}^{S,\eta}} w(\mathbf{j}-\mathbf{p}) \min_{\mathbf{q} \in \mathcal{N}_\mathbf{i}^{S,\eta}} \|\mathbf{a_q} - \mathbf{a_p}\|_2^2. \end{aligned} \tag{A.2}$$

SA3: Texture-aware UMAP and MDS embeddings

It is possible to use the spatially informed distances between image patches of high-dimensional images in any distance-based dimensionality reduction method. Here, we show spatially informed UMAP and metric MDS embeddings for the synthetic data set from Section 4.4.
We use the umap-learn [215] implementation for UMAP and scikit-learn [216] metric MDS. Note, that in the MDS Bhattacharyya example, the central cluster is actually two: the upper part corresponds to the upper left

area in the image and the lower part to the lower right. Between the two clusters are the border points between the checkered regions (and the pixels on the vertical border between the homogeneous areas).

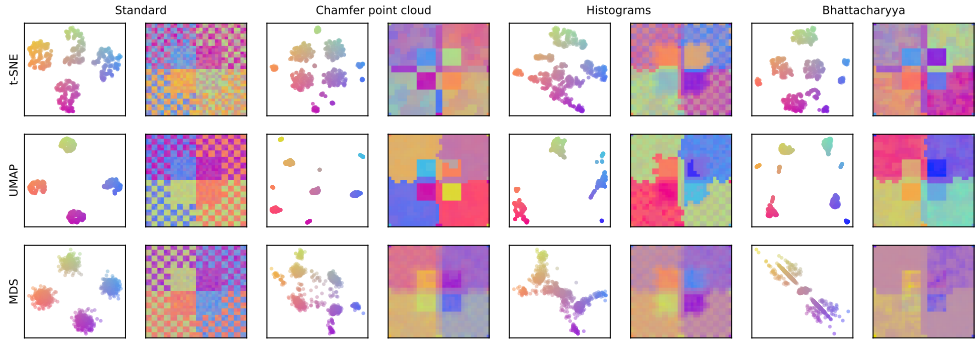
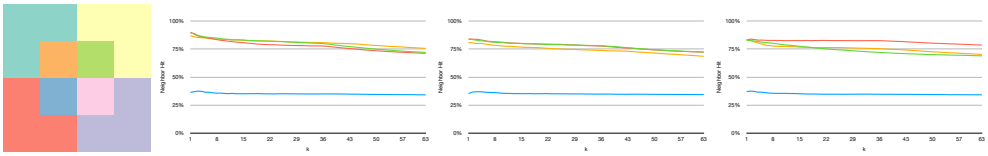


Figure A.1. *Spatially informed and standard embeddings with t-SNE, UMAP and MDS. Embeddings and recolored images as described in Figure 4.4 in the main chapter.*



(a) *Ground truth.* **(b)** *t-SNE* **(c)** *UMAP* **(d)** *MDS*

Figure A.2. *Ground truth for the synthetic dataset, classifying the four checkered areas on the outside and the four homogeneous regions in the center (a). Nearest neighbor hit for the t-SNE (b), UMAP (c), and MDS (d) embeddings from Figure A.1. — standard version, — covariance matrix and mean, — point cloud distance and — histogram.*

SA4: Other point cloud distances

The Chamfer point cloud distance, see Equation 4.6, belongs to the broader family of distances related to the Hausdorff distance. These distances build on finding the nearest point for each point in one set to the other. Instead of averaging the minima, the Hausdorff distance takes their maximum instead:

$$d_{Haus}^{PC}(\mathbf{Z_i}, \mathbf{Z_j}) = \max\left\{ \max_{\mathbf{q} \in \mathcal{N}_\mathbf{i}^{S,\eta}} \left(\min_{\mathbf{p} \in \mathcal{N}_\mathbf{j}^{S,\eta}} \|\mathbf{a_q} - \mathbf{a_p}\|_2^2 \right), \max_{\mathbf{q} \in \mathcal{N}_\mathbf{j}^{S,\eta}} \left(\min_{\mathbf{p} \in \mathcal{N}_\mathbf{i}^{S,\eta}} \|\mathbf{a_q} - \mathbf{a_p}\|_2^2 \right) \right\} \tag{A.3}$$

Another variant that might be more robust against outliers in the data might take the median, instead of the average, like the Chamfer distance:

$$d_{HM}^{PC}(\mathbf{Z_i}, \mathbf{Z_j}) = \frac{1}{2} \left(\underset{\mathbf{q} \in \mathcal{N}_\mathbf{i}^{S,\eta}}{\text{median}} \left(\min_{\mathbf{p} \in \mathcal{N}_\mathbf{j}^{S,\eta}} \|\mathbf{a_q} - \mathbf{a_p}\|_2^2 \right) + \underset{\mathbf{q} \in \mathcal{N}_\mathbf{j}^{S,\eta}}{\text{median}} \left(\min_{\mathbf{p} \in \mathcal{N}_\mathbf{i}^{S,\eta}} \|\mathbf{a_q} - \mathbf{a_p}\|_2^2 \right) \right) \tag{A.4}$$

Sum of squared differences: When taking the average instead of the minimum of point-wise distances in the Chamfer distance, ending up an average of averages.

$$
\begin{aligned}
d_{SSD}^{PC}(\mathbf{Z_i}, \mathbf{Z_j}) &= \frac{1}{|\mathcal{N}_\mathbf{i}^{S,\eta}|} \sum_{\mathbf{q} \in \mathcal{N}_\mathbf{i}^{S,\eta}} \frac{1}{|\mathcal{N}_\mathbf{j}^{S,\eta}|} \sum_{\mathbf{p} \in \mathcal{N}_\mathbf{j}^{S,\eta}} \|\mathbf{a_q} - \mathbf{a_p}\|_2^2 + \frac{1}{|\mathcal{N}_\mathbf{j}^{S,\eta}|} \sum_{\mathbf{q} \in \mathcal{N}_\mathbf{j}^{S,\eta}} \frac{1}{|\mathcal{N}_\mathbf{i}^{S,\eta}|} \sum_{\mathbf{p} \in \mathcal{N}_\mathbf{i}^{S,\eta}} \|\mathbf{a_q} - \mathbf{a_p}\|_2^2 \\
&= \frac{2}{|\mathcal{N}_\mathbf{i}^{S,\eta}| \cdot |\mathcal{N}_\mathbf{j}^{S,\eta}|} \sum_{\mathbf{q} \in \mathcal{N}_\mathbf{j}^{S,\eta}} \sum_{\mathbf{p} \in \mathcal{N}_\mathbf{i}^{S,\eta}} \|\mathbf{a_q} - \mathbf{a_p}\|_2^2
\end{aligned} \tag{A.5}
$$

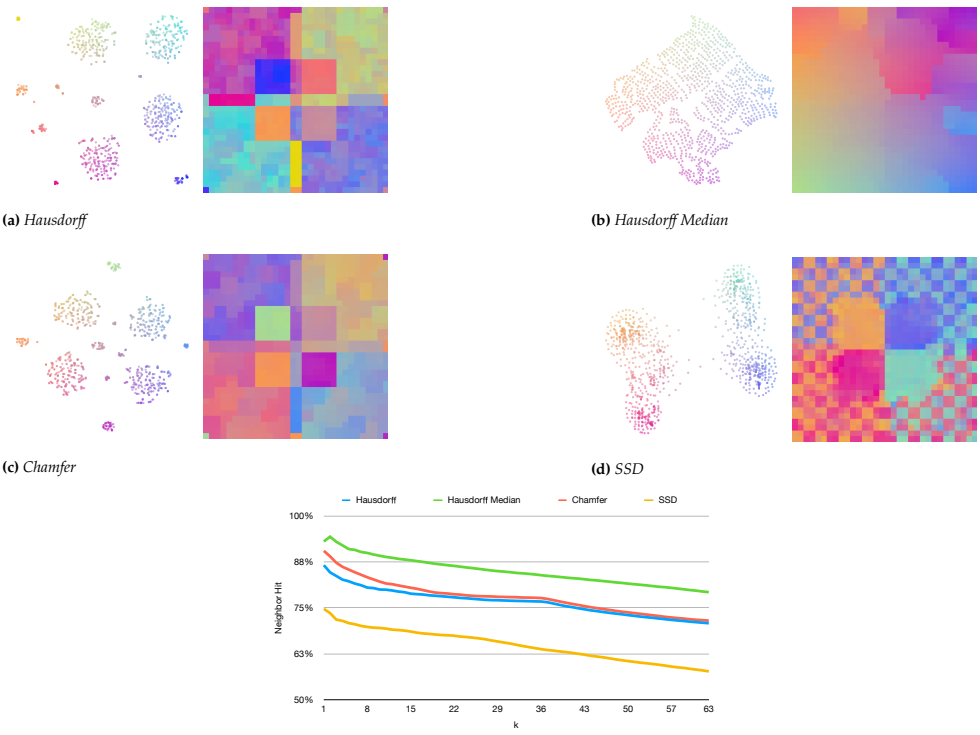
Weighted versions analogous to Section 4.3.6:

$$
d_{Haus}^{PC}(\mathbf{Z_i}, \mathbf{Z_j}) = \max\left\{ \max_{\mathbf{q} \in \mathcal{N}_\mathbf{i}^{S,\eta}} \left(w_\mathbf{q} \min_{\mathbf{p} \in \mathcal{N}_\mathbf{j}^{S,\eta}} \|\mathbf{a_q} - \mathbf{a_p}\|_2^2 \right), \max_{\mathbf{q} \in \mathcal{N}_\mathbf{j}^{S,\eta}} \left(w_\mathbf{q} \min_{\mathbf{p} \in \mathcal{N}_\mathbf{i}^{S,\eta}} \|\mathbf{a_q} - \mathbf{a_p}\|_2^2 \right) \right\} \tag{A.6}
$$

$$
d_{HM}^{PC}(\mathbf{Z_i}, \mathbf{Z_j}) = \frac{1}{2} \left(\underset{\mathbf{q} \in \mathcal{N}_\mathbf{i}^{S,\eta}}{\text{median}} \left(w_\mathbf{q} \min_{\mathbf{p} \in \mathcal{N}_\mathbf{j}^{S,\eta}} \|\mathbf{a_q} - \mathbf{a_p}\|_2^2 \right) + \underset{\mathbf{q} \in \mathcal{N}_\mathbf{j}^{S,\eta}}{\text{median}} \left(w_\mathbf{q} \min_{\mathbf{p} \in \mathcal{N}_\mathbf{i}^{S,\eta}} \|\mathbf{a_q} - \mathbf{a_p}\|_2^2 \right) \right) \tag{A.7}
$$

$$
d_{SSD}^{PC}(\mathbf{Z_i}, \mathbf{Z_j}) = \frac{2}{|\mathcal{N}_\mathbf{i}^{S,\eta}| \cdot |\mathcal{N}_\mathbf{j}^{S,\eta}|} \sum_{\mathbf{q} \in \mathcal{N}_\mathbf{j}^{S,\eta}} \sum_{\mathbf{p} \in \mathcal{N}_\mathbf{i}^{S,\eta}} (w_\mathbf{q} + w_\mathbf{p}) \|\mathbf{a_q} - \mathbf{a_p}\|_2^2 \tag{A.8}
$$

See Figure A.3 for a comparison of these point cloud distances for the synthetic data set from the main chapter.

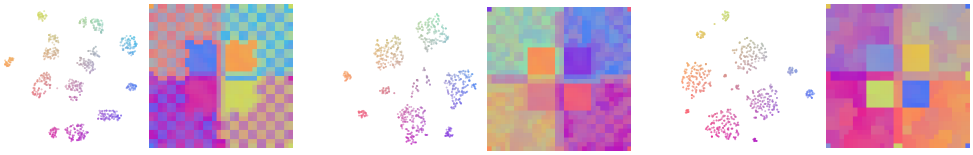


(a) *Hausdorff* **(b)** *Hausdorff Median* **(c)** *Chamfer* **(d)** *SSD* **(e)** *Nearest neighbor hit*

Figure A.3. *All spatially informed embeddings are computed with a 3x3 neighbourhood and their respective nearest neighbor hit.*

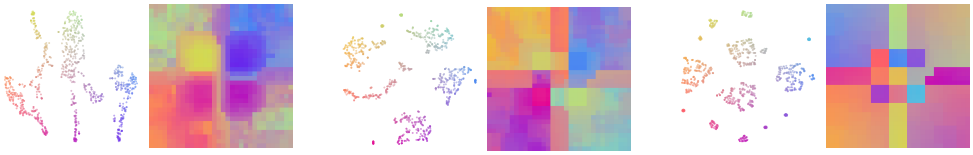
SA5: Varying neighborhood sizes and spatial weighting

See Figs. A.4, A.5, A.6, A.7 and A.8 for an overview of of the effect of different neighborhood sizes for the synthetic data set from the main chapter.



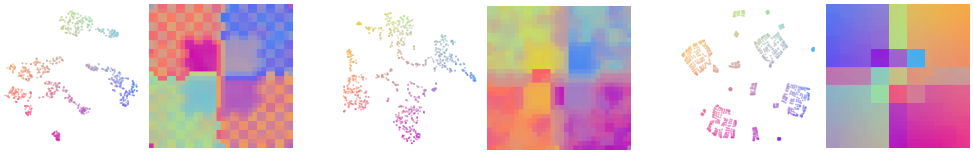
(a) *Local histograms* **(b)** *Covariance matrix and means* **(c)** *Chamfer point cloud distance*

Figure A.4. *All spatially informed embeddings are computed with a 3x3 neighbourhood and Gaussian weighting.*



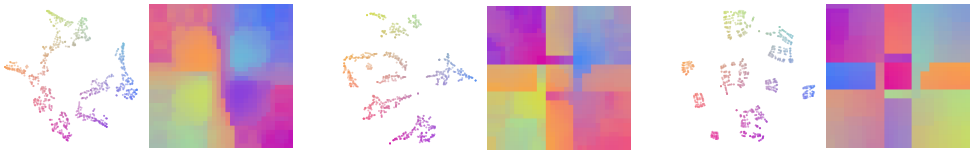
(a) *Local histograms* **(b)** *Covariance matrix and means* **(c)** *Chamfer point cloud distance*

Figure A.5. *All spatially informed embeddings are computed with a 5x5 neighbourhood.*



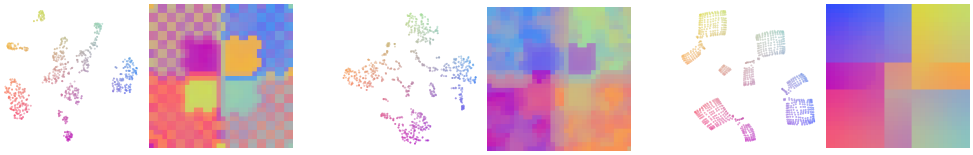
(a) *Local histograms* **(b)** *Covariance matrix and means* **(c)** *Chamfer point cloud distance*

Figure A.6. *All spatially informed embeddings are computed with a 5x5 neighbourhood and Gaussian weighting.*



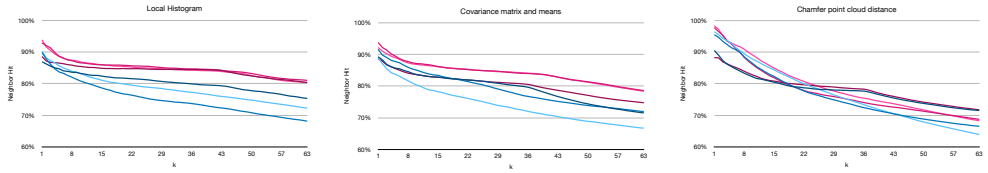
(a) *Local histograms* **(b)** *Covariance matrix and means* **(c)** *Chamfer point cloud distance*

Figure A.7. *All spatially informed embeddings are computed with a 7x7 neighbourhood.*



(a) *Local histograms* **(b)** *Covariance matrix and means* **(c)** *Chamfer point cloud distance*

Figure A.8. *All spatially informed embeddings are computed with a 7x7 neighbourhood and Gaussian weighting.*



(a) *Local histograms* **(b)** *Covariance matrix and means* **(c)** *Chamfer point cloud distance*

Figure A.9. *Neighbor hit values for the local histogram (Figure 4.4b), covariance (Figure 4.4c), and point cloud (Figure 4.4d) -based embeddings and their different neighborhood size versions (Figs. A.4, A.5, A.6, A.7, and A.8). — 3x3 neighborhood, — 5x5 neighborhood, — 7x7 neighborhood, — 3x3 neighborhood Gaussian weighted, — 5x5 neighborhood Gaussian weighted, — 7x7 neighborhood Gaussian weighted.*

SA6: Computation time evaluation

Figs. A.10 and A.11 show the computation time for the distance computation (including feature computation) and subsequent embedding time. All measurements were conducted on a computer with an Intel i5-9600K processor and a NVIDIA GeForce RTX 2080 SUPER graphics cards. A corresponding theoretical complexity analysis of each distance is presented in Section 4.3.5. Some measurements show the influence of hardware optimizations implemented in the used libraries, which influences the computation time, see for example the time behaviour of the Bhattacharyya distance for various neighborhood sizes in Figure A.10.

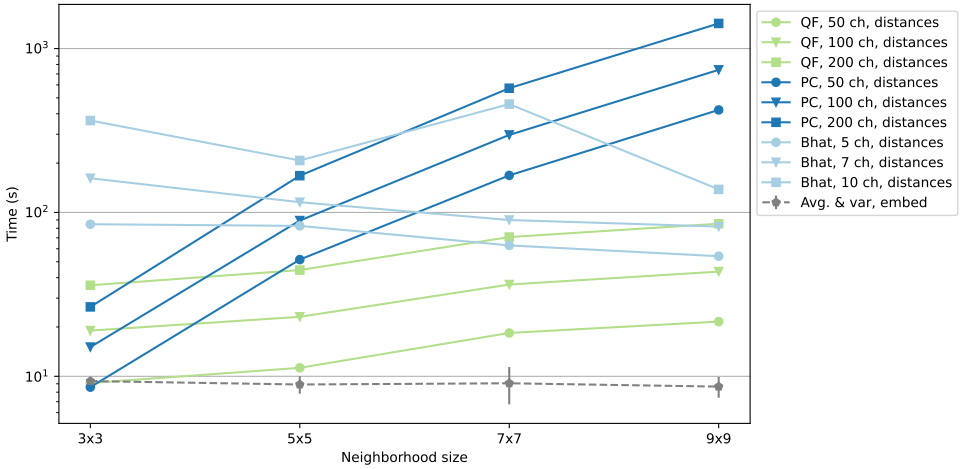


Figure A.10. *Computation time of the distance computation for varying neighborhood sizes: local histogram comparison with the quadratic form distance (QF), covariance matrix feature comparison with the Bhattacharyya distance (Bat) and the Chamfer point cloud distance (PC). The Indian Pines data set with* 21.025 *data points and 200 channels was used for computation. The same random channel subsets were used for runs with less than 200 channels. The Bhattacharyya distance is listed for fewer channels since its runtime grows impractically large for higher channel counts, as shown in Figure A.11. As mentioned in the main chapter, when using the QF distance we use the Rice rule to set the number of histogram bins. This results in 5, 6, 8 and 9 bins for the various neighborhood sizes respectively. The embedding time is not influenced by the distance metric and shown as an average of all measurements with variance bars.*

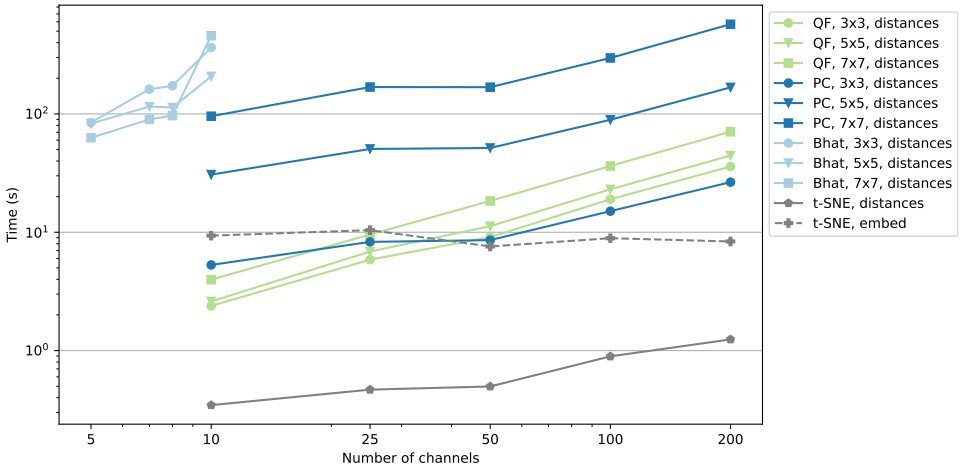


Figure A.11. *Computation time of the distance computation for channel numbers: local histogram comparison with the quadratic form distance (QF), covariance matrix feature comparison with the Bhattacharyya distance (Bat) and the Chamfer point cloud distance (PC). The Indian Pines data set with* 21.025 *data points and 200 channels was used for computation. The same random channel subsets were used for runs with less than 200 channels. The Bhattacharyya distance is listed for fewer channels since its runtime grows impractically large for higher channel counts. The histogram bin number for the QF distance is set as described in Figure A.10. The embedding time is not influences by the distance metric and the shown embedding times for the standard t-SNE procedure is representative for the embeddings times of all runs.*

SA7: Indian Pines - additional figures

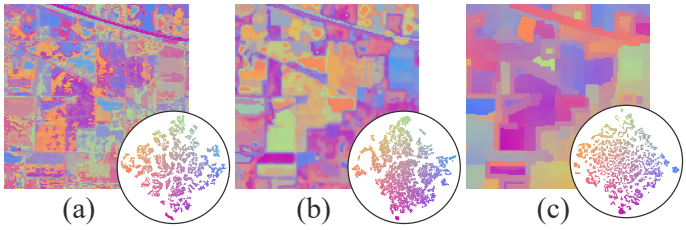
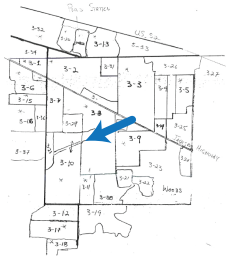
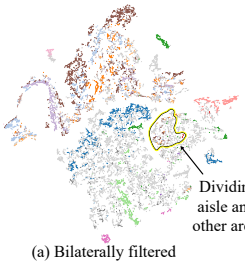


Figure A.12. ***Overview of the Indian Pines dataset*** *with different embedding methods. Coloring based on colormapping embedding coordinates, as discussed for Figure 4.4. (a) standard t-SNE, (b) standard t-SNE applied to a bilaterally filtered version of the image, and (c) our point cloud-based t-SNE.*

(a) *Highlights in the ground truth.*



(b) *Dividing aisle.*



(a) Bilaterally filtered (b) Annotated map

(a) *Attempted selection of the dividing aisle in the t-SNE embedding based on the bilaterally filtered data. In contrast to Figure 4.5d, it is not possible to find a cluster that only corresponds to the aisle.*

Figure A.13. *Ground truth of Indian Pines Site 3 with highlighted areas as further discussed in Figure A.15 and an annotated map of the Site without a color-coded background.*

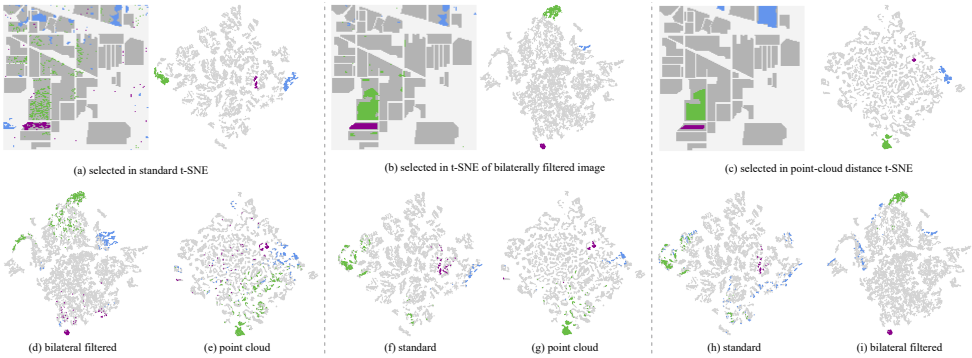


Figure A.15. ***Indian Pines: Comparison of spatially-aware and standard t-SNE embeddings.*** *The top row (a-c) shows a standard t-SNE embedding, an embedding of the bilaterally filtered data set and our spatially-aware embedding (based on the Chamfer point cloud distance). We tried to select the three highlighted regions from Figure A.13a in each embedding and show the respective pixel on the ground truth. The lower row (d-i) highlights the selections made in the above embeddings in the two other embeddings, for example highlighted points in (d) and (e) correspond to points selected in (a).*

B. Supplement: Interactive Image HSNE

SB1: Background - Hierarchical Embeddings

A *hierarchical embedding* method typically extends existing DR techniques by creating a hierarchical representation of the original data items and projecting only elements from individual hierarchy levels instead of all data [16]. Using the notation for hierarchical embeddings from Höllt et al. [104], the hierarchical data structure consists of *landmarks*, within m levels, namely the sets $\mathscr{L}^0 \dots \mathscr{L}^{m-1}$. The lowest hierarchy level $\mathscr{L}^0$ contains all data points. Each landmark $L_i^{k+1} \in \mathscr{L}^{k+1}$ in a higher hierarchy level represents ($\rightarrow$) a set of landmarks from the lower level $\mathscr{L}^k = \{L_i^k \mid L_i^k \leftarrow L_j^{k+1}\}$. Thus, the higher embedding levels provide more abstract representations of the original data. Here, for simplicity, we assume the hierarchy to be a proper tree: each landmark L_i^k is represented by only one landmark in $\mathscr{L}^{k+1}$. In practice this is not necessarily the case [52].
Given a selection of landmarks $\mathscr{K}^k$ on a level k, we are interested in their relation to landmarks in other hierarchy levels. Following the terminology for visualizations of hierarchically structured data from Elmqvist et al. [124], we describe coarsening the level of detail, i.e., the process of finding all landmarks $\mathscr{K}^{k+1}$ on the more abstract level $k+1$ that well represent them, as *rolling-up*. Analogously, refining the level of detail, that is finding the landmarks $\mathscr{K}^{k-1}$ on the more detailed level $k-1$ that are represented by $\mathscr{K}^k$ is the result of *drilling-down*.

SB2: Timings

Timings for regular HSNE and interactive HSNE, all taken on a machine equipped with an Intel Core i5-9600K CPU and a NVIDIA GeForce RTX 2080 SUPER GPU. The total embedding time is approximately equal to the data structure initialization and gradient descent time. Our hierarchy traversal step adds a comparably small overhead to the entire embedding computation. In most scenarios, the user is presented with iteratively updating embeddings within 300ms (hierarchy traversal plus data structure initialization times).

Table B.1. *Indian Pines: Durations of embedding update step connected to an image interaction. ROIs as indicated in Figure 5.4a. Times, in ms, are averages over 10 runs with sample standard deviation.*

	ROI Ⓘ		ROI Ⓘⓘ		ROI Ⓘⓘⓘ	
Scale	2		2		3	
Landmarks	9,738		9,533		3,979	
Data points in view	57,424		58,212		50,691	
Hierarchy traversal [our]	48.0	(2.6)	39.3	(2.9)	39.2	(4.2)
Embedding	848.1	(78.2)	862.2	(17.8)	619.8	(33.2)
└ Data structure initialization	244.6	(11.8)	243.8	(4.7)	40.4	(2.5)
└ Gradient descent	598.8	(80.0)	614.0	(20.4)	545.5	(32.2)

Note: Times in ms, standard deviation in parentheses.

SB3: Additional Indian Pines Information and Figures

The Indian Pines Test Site 3 [10] data are a set of hyperspectral 145×145 pixel images with known class information for each pixel. This well analyzed data set is only a small part of a larger, unlabeled measurement of $614 \times 2,678 \approx 1.6$M pixels, depicting an area around the Purdue University Agronomy Center in Indiana, USA, made up of fields (e.g. corn and soy), forests, roads, rivers and houses. The pixel resolution is roughly 20m x 20m and contains electromagnetic spectral information from from 400-2400 nm sampled at 10 nm. We excluded 20 channels of the 220 channel data set due to their low information quality since they cover spectral water absorption bands, as suggested in [145]. This results in 200 samples, that we interpret as dimensions, forming our high-dimensional attribute space used as input for the dimensionality reduction. We normalized the data by clamping each channel to the 99.999th percentile and scaling it to the range [0, 1].

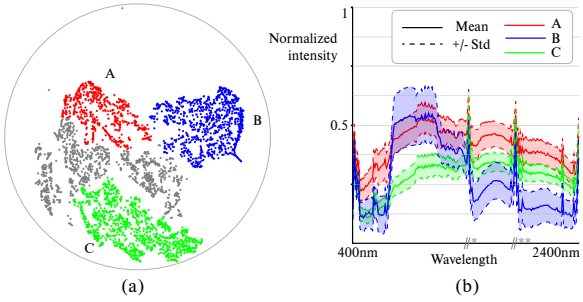


Figure B.2. ***Indian Pines cluster characteristics:*** *(a) shows the top level (4th) HSNE embedding of the Indian Pines data. (b) displays channel-wise intensity values for three cluster of different types of fields (A and C) and forest (B). * and ** indicate frequency bands that were omitted from the data. The three regions of interest discussed in Figure 5.4 are mostly represented with landmarks from the three clusters above.*

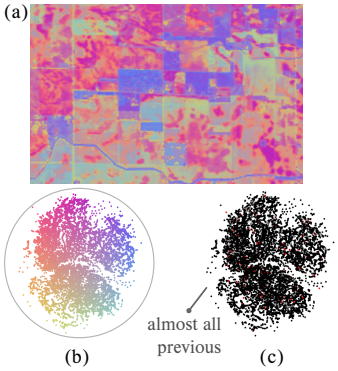


Figure B.3. ***Drill-down*** *of the embedding for ROI 3 from Figure 5.4d: the recolored image region (a), new embedding (b) and initializations (c).*

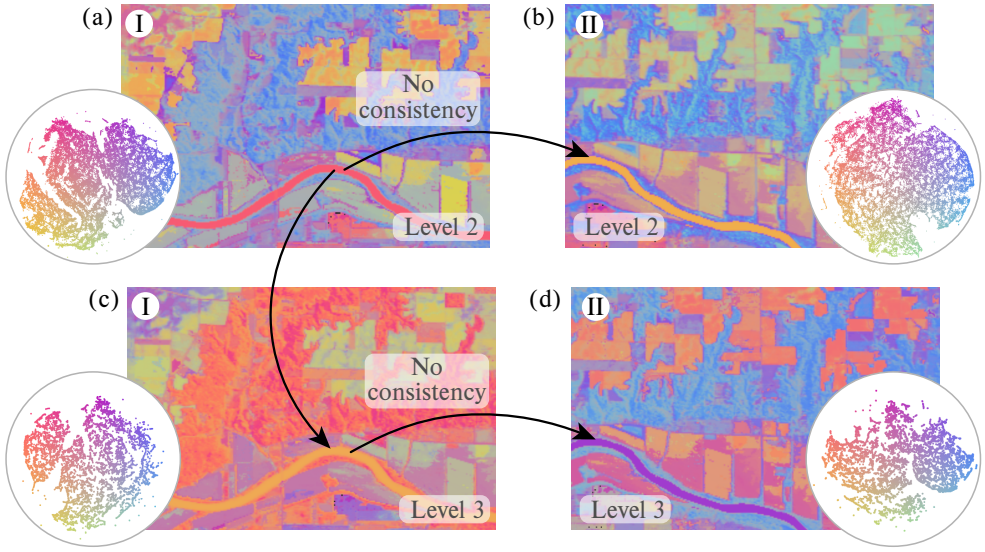


Figure B.4. ***Regular HSNE embeddings:*** *(a) and (b) as in Figure 5.4f and Figure 5.4g. (c) and (d) show the intermediate refinements on level 3 between the top level embedding and the level 2 embeddings. They contain 11,857 and 11,207 landmarks respectively while the level 2 embeddings contain 50,116 and 46,461 landmarks respectively.*

C. Supplement: Superpixels

SC1: Complexity of geodesic approach

Both the construction of a search index in HNSW and the actual nearest neighbor search is rather feasible with complexities of respectively $O(n \log n)$ and $O(\log n)$. But computing the geodesic distance $d_{\mathcal{G}}$ between two graph nodes is intrinsically complex and increases the typical complexity of distance computation for, e.g. Euclidean distances, of $O(1)$ to $O((n+|\mathcal{E}|) \log n)$ with the A* search algorithm or $O(n \log n + |\mathcal{E}|)$ with Dijkstra's algorithm*. This distance needs to be computed $|\wp_p| \cdot |\wp_q|$ times for the Hausdorff-based set comparison. On higher levels the components contain numbers of pixels in the same order of magnitude as the entire image, such that computing a single set comparison with the Hausdorff distance will become intractable, as well.

SC2: Symmetrized and connected kNN-graph

We convert the directed kNN-graph into an undirected graph by making each edge bidirectional, compute its connected components and add new edges that connect the components. For that purpose, we first compute the mean value of the high-dimensional data points in each connected component. Secondly, we set up a fully-connected helper graph, each vertex corresponding to one connected component and edge weights being the δ distances between their means. We then compute its minimum spanning tree (MST). Finally, we find the smallest δ distance between any two points from two components connected by an edge in the above MST and introduces a bidirectional connection into the symmetrized kNN-graph. If a symmetrized kNN-graph already only has one connected component, no new edges are introduced.

SC3: Indian Pines

Settings for the superpixel hierarchy discussed in Section 6.6.1: kNN algorithm: HNSW (faiss implementation), 300 nearest neighbors, Number of random walks: 50, Length of random walks: 25.
Settings for the HSNE hierarchy : kNN algorithm: HNSW (hnswlib implementation), perplexity of 30 (yielding 90 nearest neighbors),
Figure C.1 shows the superpixel hierarchy from Section 6.6.1 for all level from 0 to 7. It is possible to continue with more abstraction levels until there is only one superpixel, which covers the entire image, but the very large superpixles on high levels do not create interesting subdivision of the image anymore and are not shown here.

SC4: CiCYF

The marker channels used for the superpixel hierarchy and corresponding embeddings for the CiCYF discussed in Section 6.6.2 (as per personal correspondence with the data paper's authors): 5hmc, b-actin, CD103, CD11b, CD11c, CD163, CD20, CD206, CD31, CD3E, CD4, CD8a, FOXP3, HLA-AB, IRF1, Ki67, laminABC, MART1, panCK, PD1, podoplanin, S100A, S100B, SOX10, SOX9, pMLC2, yH2AX.
The channels Collagen and Hoechst (a marker used to stain DNA and labeled as "DNA" in our figure) used for recoloring in Figure 6.7 are not part of the input markers to the superpixel hierarchy.
Figure C.2 shows the full image data and indicates the region we focus on in Section 6.6.2, with $x_{begin} = 2060$, $x_{size} = 2000$, $y_{begin} = 450$, $y_{size} = 1500$.
Figure C.3 shows the superpixel hierarchy from Section 6.6.2 for all level from 0 to 6.

SC5: Quantitative results: Indian Pines dataset

Table C.1 and Table C.2 list the numerical values for the graph in Figure 6.8, showing the quantitative evaluation results discussed in Section 6.6.3.
Even though UE is constrained to $[0, 1]$ and a value of 1 can be expected for a worst-case segmentation, a "segmentation" of the Indian Pines data into a single segment yields a UE of 0.975. This is due to the

* see A* on boost.org/doc/libs/1_86_0/libs/graph/doc/astar_search.html or en.wikipedia.org/wiki/A*_search_algorithm and Dijkstra's algorithm on boost.org/doc/libs/1_86_0/libs/graph/doc/dijkstra_shortest_paths.html or en.wikipedia.org/wiki/Dijkstra%27s_algorithm

term $\min\{|\vartheta_r \cap q_i|, |\vartheta_r \setminus q_i|\}$ in Equation 6.7, which was introduced in this measure to improve in earlier formulations of undersegmentation error which tend to penalize small overlap of large superpixels with a ground truth segment. However, since the ground-truth label "background" covers just-so more than half the image pixels, this is the only term in which $|\vartheta_r \setminus q_i|$ is considered instead of $|\vartheta_r \cap q_i|$. This edge case of a ground-truth segment covering more than 50% does not impact the evaluation though.

SC6: Miscellaneous

Figure C.4 shows the full superpixel hierarchy for the bus image from Figure 6.2.
Figure C.5 shows the full superpixel hierarchy for small Indian Pines image from Figure 6.4.

Table C.1. ***Undersegmentation error (UE)****: Values for the plot in Figure 8. Lower is better.*

Level		1	2	3	4	5	6	7	8	9	10	11
Euclid.	Val.	0.073	0.147	0.269	0.493	0.704	0.900	0.975	—	—	—	—
(ours)	#	5763	1441	345	88	16	5	1	—	—	—	—
Geo.	Val.	0.072	0.141	0.238	0.485	0.622	0.870	0.975	—	—	—	—
(ours)	#	5769	1447	364	89	23	5	1	—	—	—	—
Barbato	Val.	0.092	0.098	0.149	0.171	0.227	0.340	0.378	0.578	0.644	0.730	0.975
	#	6901	5769	2227	1590	587	234	169	61	43	20	1
*k*NN	Val.	0.084	0.151	0.244	0.356	0.661	0.812	0.867	0.975	—	—	—
(ours)	#	5191	1282	325	84	23	7	2	1	—	—	—

Barbato's method does not know an explicit level distinction. The levels correspond to several `n_clusters` settings: 1, 5, 10, 25, 50, 100, 250, 500, 1000, 2500, 5000

Table C.2. ***Explained variation (EV):*** *Values for the plot in Figure 8. Higher is better. EV is 1 when the number of superpixels is equal the number of pixels, i.e., no abstraction.*

Level		1	2	3	4	5	6	7	8	9	10	11
Euclid.	Val.	0.970	0.918	0.837	0.709	0.517	0.290	0.0	—	—	—	—
(ours)	#	5763	1441	345	88	16	5	1	—	—	—	—
Geo.	Val.	0.969	0.918	0.845	0.693	0.488	0.218	0.0	—	—	—	—
(ours)	#	5769	1447	364	89	23	5	1	—	—	—	—
Barbato	Val.	0.963	0.958	0.922	0.906	0.861	0.794	0.766	0.680	0.641	0.485	0.0
	#	6901	5769	2227	1590	587	234	169	61	43	20	1
*k*NN	Val.	0.963	0.919	0.853	0.765	0.578	0.453	0.209	0.0	—	—	—
(ours)	#	5191	1282	325	84	23	7	2	1	—	—	—

Barbato's method does not know an explicit level distinction. The levels correspond to several `n_clusters` settings: 1, 5, 10, 25, 50, 100, 250, 500, 1000, 2500, 5000

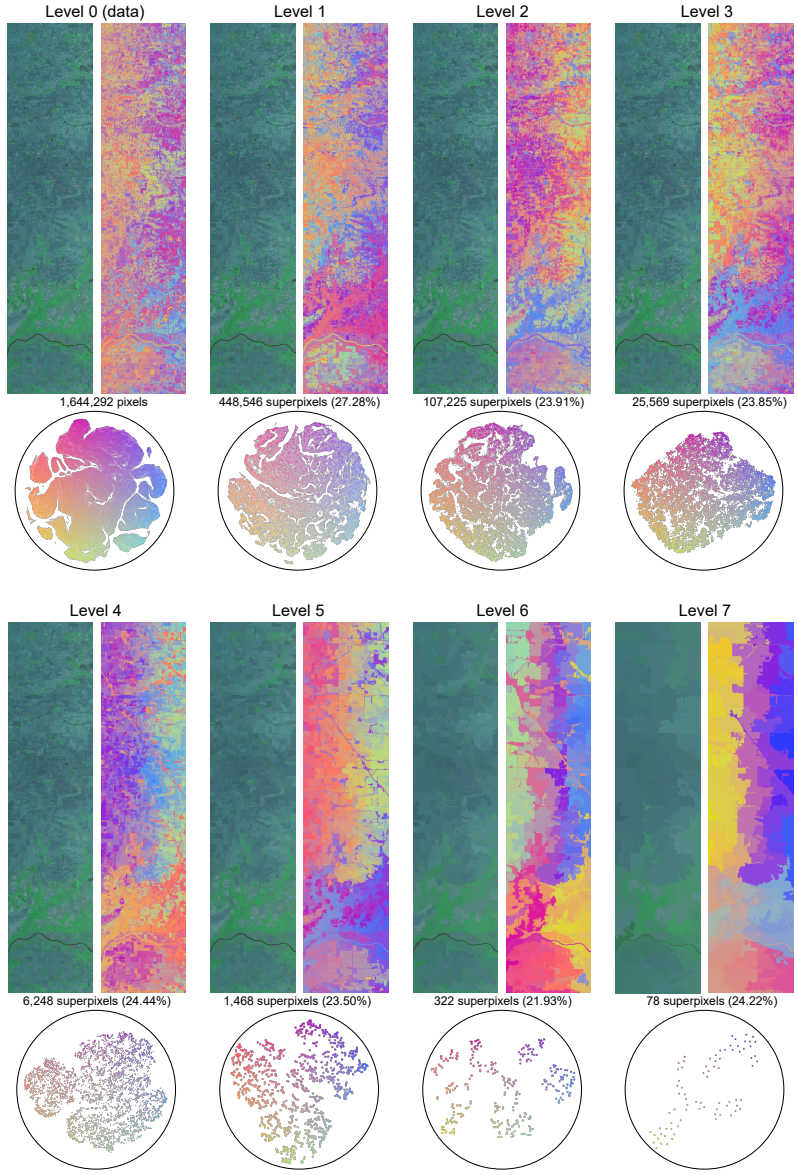


Figure C.1. ***Indian Pines superpixel hierarchy:*** *Data level embedding alongside seven superpixel abstraction level embedding. Each level shows a false color image based on the average spectrum per superpixel of channel 20 (587 nm, red), 76 (1090 nm, green) and 130 (1591 nm, blue). Next to them are recolorings of the superpixel based on the embedding layout using a 2D colormap which is superimposed on the respective embeddings (as shown in Figure 6.5. Superpixels are hardly visible in these down-scaled version of the originally* $614 \times 2,678$ *(w x h) images, but more abstract levels clearly show more and more high-level structure. The percentages indicates the reduction of components, i.e., the number of components (superpixels) in level 1 reduces to 27.28% of the previous level.*

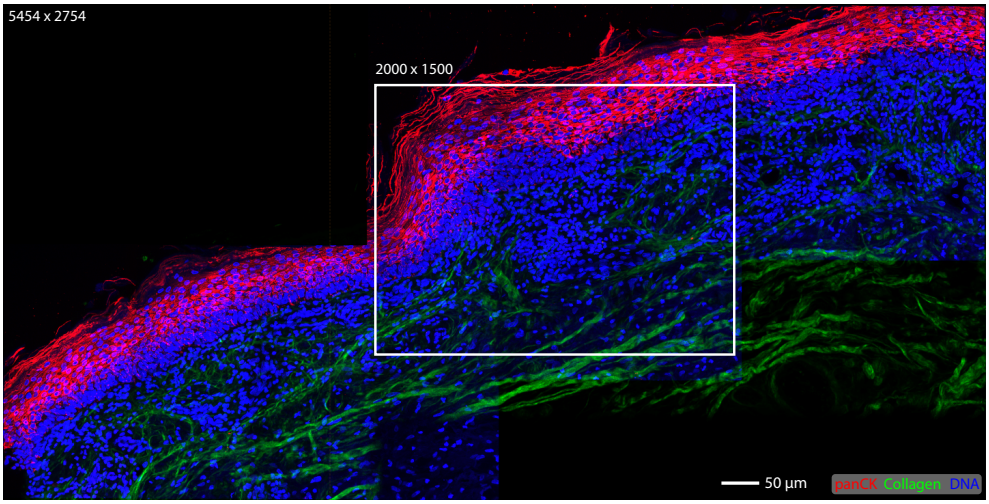


Figure C.2. ***CyCIF Focus Region:*** *As used in Section 6.6.2. DNA corresponds to the Hoechst channel.*

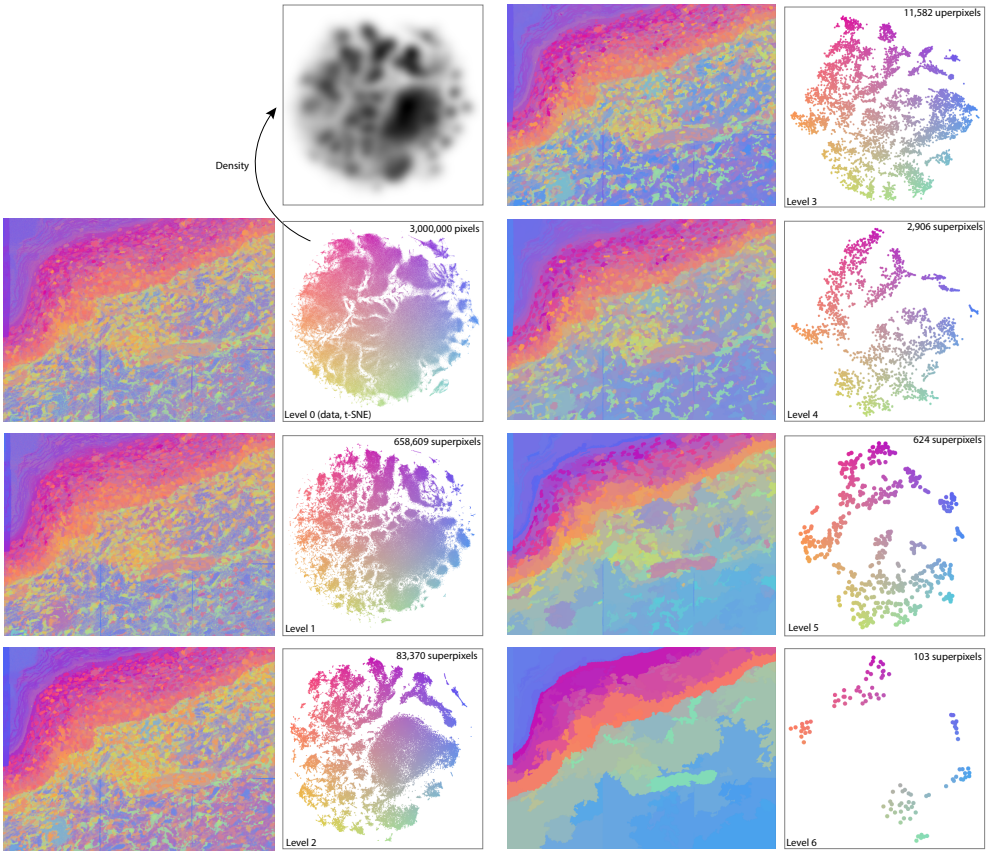


Figure C.3. ***CyCIF Superpixel Hierarchy:*** *Full superpixel hierarchy for the CyFIC data from Section 6.6.2.*

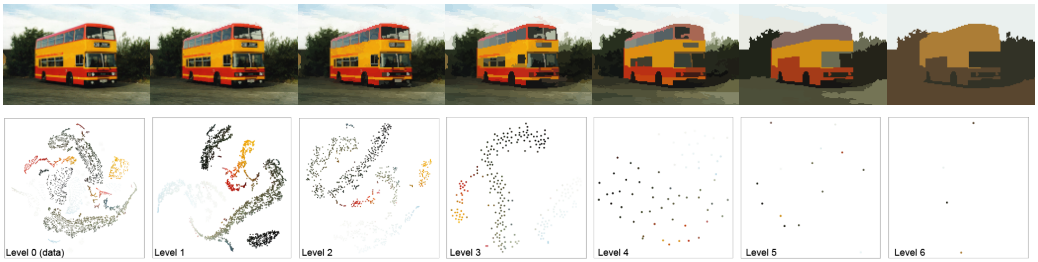


Figure C.4. *RGB image of a bus (top left) and 6 levels of abstraction. Superpixels are recolored with the average color of all the image pixels they contain. Below, embeddings of each level using the same coloring. Numbers of components: 15300 (150x102 pixels), 4094, 1054, 271, 64, 14, 4.*

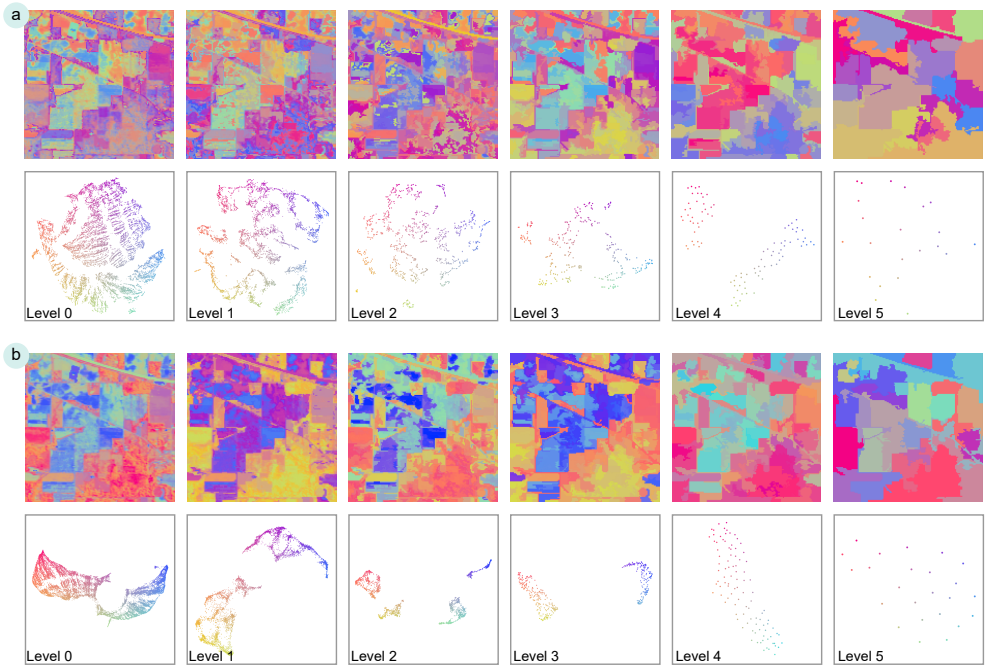


Figure C.5. ***Indian Pines:*** *Embeddings and image space recolorings for (a) t-SNE and (b) UMAP probabilities.*

D. Supplement: ManiVault

SD1: Benchmarks

Speed ManiVault can show progressive updates of analytics plugins with only small additional computational penalties. To show this, we compute t-SNE embeddings with a ManiVault analytics plugin that uses the HDI library GPGPU implementation of t-SNE [105, 205]. First, we compute embeddings non-progressively, and then, in a second setting, we show intermediate embeddings every 10 gradient descent iterations (respectively "no updated" and "with updates" in Table D.1. Additionally, we compare these runs with a lightweight python wrapper * around the same t-SNE library. Every embedding is laid out over 500 gradient descent iterations. The non-progressive computation is slightly faster than the Python wrapper around the same library calls. The difference between the total runtime of the t-SNE embeddings in ManiVault with and without updates is explained by the difference in the gradient descent time: In the former setting, the analytics plugin notifies ManiVault's core about the current embedding layout. All measurements were taken on a machine equipped with an NVIDIA GeForce RTX 2080 SUPER GPU and an Intel Core i5-9600K CPU and running Windows 11 22H2.

Table D.1. *Duration of t-SNE embedding computations with the same implementation, invoked via a Python wrapper and ManiVault, once showing only the final embedding and once progressively updating a scatterplot. Times, in seconds, are averages over 10 runs with sample standard deviation.*

Data set	Swiss Roll 3D [217]	COIL-20 [218]	MNIST [219]	Fashion-MNIST [220]	10x Mouse [221]
# points	1,500	1,440	70,000	70,000	1,306,127
# dimensions	3	16,384	784	784	50 (first PCs)
nptsne * [a]	0.30 (0.02)	2.32 (0.08)	23.31 (0.14)	20.58 (0.01)	268.38 (2.21)
ManiVault [b]	0.58 (0.05)	2.37 (0.05)	22.51 (0.11)	20.20 (0.27)	258.60 (5.76)
ManiVault [c]	0.59 (0.07)	2.46 (0.09)	22.85 (0.15)	20.24 (0.11)	257.91 (4.02)

Note: Times in seconds, sample standard deviation in parentheses. [a] Python wrapper, [b] no updates, [c] with updates

Memory After starting ManiVault, with the Data Hierarchy and Data Property Viewer open, the software consumes around 87 MB of memory (on Windows). Loading data sets comes with a small memory overhead. Here, we loaded various data sets, as listed in Table D.1, and compared their binary size on disk with the growing memory footprint of ManiVault after loading them. We observe a $0.7 - 1.5$ MB overhead per data set, compared to their binary size, when utilizing the point data type plugin. For larger data set, it can be useful to trade off precision for lower memory uptake. We can employ a bfloat16 floating point implementation † to store large data set, and thereby effectively half the memory ManiVault requires: e.g. the 10x Mouse data will take up 126.15 MB instead of 249.87 MB.

Table D.2. *Memory consumption of loaded data sets, as listed in Table D.1, in ManiVault compared to their binary size on disk. Values are averages over 4 loaded data sets.*

Data set	[217]	[218]	[219]	[220]	[221]
Binary type	float32	uint8	uint8	uint8	float32
Raw binary	0.017	22.5	52.34	52.34	249.12
ManiVault (float32)	0.97	-	-	-	249.87
ManiVault (uint8)	-	23.77	53.5	54.1	-

Note: Values in MB. Slight deviations might occur due to Qt's memory management on Windows 11, e.g., the difference between MNIST and Fashion-MNIST.

* Github: biovault/nptsne

† Github: biovault/biovault_bfloat16

Publications

- **A. Vieth**, A. Vilanova, B. Lelieveldt, E. Eisemann and T. Höllt, "*Incorporating Texture Information into Dimensionality Reduction for High-Dimensional Images*," 2022 IEEE 15th Pacific Visualization Symposium (PacificVis), 2022, pp. 11-20, doi: 10.1109/PacificVis53943.2022.00010.

- **A. Vieth**, B. Lelieveldt, E. Eisemann, A. Vilanova and T. Höllt, "*Interactions for Seamlessly Coupled Exploration of High-Dimensional Images and Hierarchical Embeddings*," Vision, Modeling, and Visualization (VMV), 2023, pp. 63-70, doi: 10.2312/vmv.20231227. Best Paper Honorable Mentions VMV 2023

- **A. Vieth**, T. Kroes, J. Thijssen, B. van Lew, J. Eggermont, S. Basu, E. Eisemann, A. Vilanova, T. Höllt and B. Lelieveldt, "*ManiVault: A Flexible and Extensible Visual Analytics Framework for High-Dimensional Data*," IEEE Transactions on Visualization and Computer Graphics, vol. 30, no. 1, pp. 175-185, 2024, doi: 10.1109/TVCG.2023.3326582. Best Paper Honorable Mentions IEEE VIS 2023

- **A. Vieth**, B. Lelieveldt, E. Eisemann, A. Vilanova and T. Höllt, "*Manifold-Preserving Superpixel Hierarchies for Exploration of High-Dimensional Images*" [under review]

Curriculum Vitæ

Alexander Vieth

31 December 1994 Born in Münster, Germany

Education

2006 - 2013	Abitur at Gymnasium Arnoldinum in Steinfurt, Germany
2013 - 2017	Bachelor Sc. Electrical Engineering, Information Technology and Computer Engineering at RWTH Aachen University, Germany
2017 - 2019	Master Sc. Electrical Engineering, Information Technology and Computer Engineering at RWTH Aachen University, Germany
2019 - 2026	PhD in the Computer Graphics and Visualization group at TU Delft, The Netherlands

Acknowledgments

Roughly ten years ago, during my Bachelors, a couple of friends and I confidently declared that we would soon leave the academic path and embrace industry. Planning for or speculating about the future is always both exciting and important. Surely, we had a good grasp on what our future would hold. Fast forward to now, and each of us is finishing up their PhD. A good moment to pause and ponder: What went wrong? And, of course, what went right? First and foremost, I was lucky to cross paths with many amazing people who I want to thank for their constant support over the last years. It took me quite some time to get here, but without you it would have been far less enjoyable, and surely taken even longer.

Thank you, Thomas and Anna, for without you I could not have started or finished this PhD. I can only wish for everyone to experience their scientific socialization with colleagues who display integrity in and joy for research as much as you do. Thank you, Boudewijn, for the enthusiasm that you bring to every project. And thank you, Elmar, for your display of immense scientific creativity. You are great supervisors.

I'm glad to have been part of the Computer Graphics and Visualization group at the TU Delft. The variety of research topics and people made for a consistently interesting environment. Thank you to all of the PhDs and Postdocs: Jerry, Mathijs, Lukas, Jackson, Mijael, Guowei, Xuejiao, Ali, Amir, Yang, Chen-Chi, Benno, Soumyadeep, Christoph, Mika, Mrinal, Fengshi, Celine, Priyanka, Nasikun and Leo. And thank you to the staff members as well: Rafa, Klaus, Riccardo, Petr, Michael, and Martin. Of course, thank you, Lauretta and Ruud for always helping out. Faizan and Marcos, you were the first vis-related PhD buddies in a mostly graphics-centered group, thank you for starting the journey with me. Nicolas, thank you for the many conversations over coffee and dinner, as well as unforgettable travel experiences! Thank you, Mark, Annemieke, Baran and Ruben for defeating gruesome villains with the power of friendship. And thanks, Berend, for constantly reminding me that invasions are not socially acceptable anymore.

I was lucky to be involved with the Imaging Genetics group in the Division of Image Processing at the Leiden University Medical Center and the Visualization cluster at the TU Eindhoven throughout the course of my PhD. I truly appreciate having gotten to know all of you. Thank you, Thomas, Baldur and Jeroen for the fruitful ManiVault collaboration. Thanks, Astrid and Julian, for a great after-conference road trip. And of course, thank you Sanne, Vidya, Kirsten, Linhao, Dennis, Bram, and Chang for many fun and vaguely vis-related meetups! Thanks, Silvia, for consistently motivating people to join reading groups. Thanks, Jenia and Faeze, for being great office mates in the new LKEB cubicles. And thanks Moody, for joining me in pushing through the final steps of finishing our PhDs.

Thank you, Linda, for only sometimes losing patience with me over the last years. Thanks, Bean, simply for being the cutest cat, except at five in the morning. Thank you, my sisters, parents, family and all friends who accompanied me through this journey. You are the best.

And finally, thank you, dear reader, for making it this far, or jumping here straight away. After all, you are one of the select few to open this thesis in the first place.

There is no justice in the laws of nature,
no term for fairness in the equations of motion.
The universe is neither evil, nor good, it simply does not care.
The stars don't care, or the sun, or the sky.
But they don't have to! ***We*** care! There ***is*** light in the world, and it is ***us***!"

– from *Harry Potter and the Methods of Rationality* by Eliezer Yudkowsky