Utrecht University

Augmenting Reduction Plots with Localized Explanations

Student:
Gijs van Steenpaal
g.j.vansteenpaal@students.uu.nl

Supervisor:
Alexandru C. Telea
a.c.telea@uu.nl

May 27, 2021

Abstract

Multidimensional projection have been a common technique to aid the analysis high dimensional data, techniques exist that attempt to explain the projection using per point or global techniques. Neither can convey to the user *why* points are placed in a certain neighborhood. We will implement a localized explanation technique which achieves this based on the attributes [1] and dimensionality [2] and extend them to 3D. We aim to make a tool which is both correct and works at interactive rates, by doing so making the techniques more accessible. We will also use this tool to apply the techniques to a series of real world dataset to provide examples of the types of insights possible. We also compare 2D and 3D for 29 projections techniques to investigate the additional value of the 3rd dimension.

Keywords: Visualization, Multidimensional projections, Localized explanation

Contents

Chapter 1

Introduction

Visual exploration of high-dimensional data is a key component of modern data science. We live in an era of *big data* where it is cheaper to collect more data points with more attributes that to actually think about it[1]. This brings with it the need for techniques to reduce this data again to something humans can interpret. Many visualization methods have been proposed for this type of data, such as parallel coordinates [3], table lensing [4], [5], and scatterplot matrices [6]. Multidimensional projections, also known as dimensionality reduction, is one of these methods, arguable one of the most important ones. Numerous implementation have already been proposed by the information visualization and machine learning communities [7]–[10] and the topic is still being worked on as no perfect solution exists. This method helps battle the curse of dimensionality by projecting entries of a dataset into typically screenspace coordinates. They are able to handle datasets which are large both in terms of observations (i.e. points or samples) and attribute (i.e. dimensions or variables). Doing so allow this users to gain insight into relative point distances, structures like clusters and finding similar points or outliers. However explaining in terms of attributes *why* these structures appear in the projection and thus the dataset is not trivial. This brings us to the topic of explanatory techniques which empower the user to solve this problem.

Global explanations, such as biplot axis [11], [12] and axis legends [13], [14] provide the insight into how the dimensions influence all the points but as a consequence provide no information about local structures. conversely *Local explanations* such as tooltips provide an annotation for each point in the dataset but still require the user to select *which* point to explain. Last we have *image based* or *localized explanations*, these techniques provide an annotations for each point which is so simple it can easily be encoded into a color in the visualization. As a consequence we no longer need the user to select a single point, we can display the local explanation everywhere. The user can now perceive the annotation for numerous points at once by looking at the image. These techniques often scale well both visually, computationally and can generally handle high dimensional data well.

Da Silva [1] conceived of a image based technique which find within a user defined neighborhood size which attribute best explained the commonality based

[1]Paraphrased from Prof A. Siebes during the Big Data course at Utrecht University

on either euclidean distance or variance. Later Van Driel [2], [15] build upon this idea of neighborhood explanations by encoding color based on the intrinsic dimensionality for a point and its surroundings. With this the user can separate regions based on the underlying dimensionality and identify areas that are difficult to explain (i.e. high dimensionality). This set of techniques offer very insightful additions to the classical techniques, they are also agnostic to which projection technique is being used and theoretically scale well since the dependency between points for the explanations is rather low. Sadly the practical aspects of these works are lacking. The actual explorations of real world datasets and different projection techniques are sparse and only the 2D dimensional case is considered. The reference implementation are also limited, it can be used to generate static explanations and the given performance figures are bleak. The works also provide arguably radically different techniques which makes them very difficult to relate to classical methods, and with it to adopt into data analysis pipelines. We aim to find find ways lower the barrier of entry, to achieve this we offer this work to outline the issues and systematically combat them.

In short we set out the following goals for this work and the accompanying tool:

- Build a correct, scalable and general implementation. It needs to produce the same images as the reference implementation at fast rates for any projection and dataset.
- Ensure produced images are interactive and subject configurable rendering settings.
- Include support for displaying and explaining 3D dimensional projection and discuss the challenges that arise from the rendering and interaction with it.
- Perform an quantitative and qualitative comparison between 2D and 3D projections.
- Explore 8 real datasets in the form of a series of case studies.

All this packaged together will form a solid basis for data analysts to explore these novel techniques, justify their use and incorporate them into their toolbox!

To start our work we will create an extensive overview of theory behind the techniques in chapters 2. Here we outline all the concepts and intuition behind the explanations, selection of projections and related research. Moving on we will outline our own implementation and what makes it tick in chapter 3. Here we talk through all the requirements focussing on the gap between what was originally proposed and the real world and how we meet them. We will also outline how the user can interact with the data and how we validated that the produced images are correct. In chapter 4 we will use our own implementation to explain 8 real world datasets and provide more in depth analysis of the input parameters for the explanations. This will allow the future users to learn by example and intuition instead of dense math. Then in the chapter 5 we will discuss the difficulties surroundings a possible a user study and other issues we encountered, here we also include the conclusion for a summary of our most important findings.

Chapter 2

Background

In this chapter I will discuss related works that form the basis of my thesis and outline its context. Here I will also discuss related techniques and provide overviews of projection methods (sec 2.1), quality metrics (sec 2.2), explanation procedures (sec 2.3) and interaction techniques (sec 2.4). Each will be accompanied by formal definition where applicable.

2.1 Multidimensional Projections

The context in which our area of research exists is that of multidimensional projections , also referred to as dimensionality reduction. This is a set of mechanisms that compute a projection of typically high-dimensional data points into a low-dimensional space, often two dimension, where visual exploration can be performed. More formally we can consider a dataset of size N to be defined as $D = \{p_1, ..., p_N\} \in \mathbb{R}^n$ where each element p is an n-dimensional point $p_i = (p_i^1, ..., p_i^n)$. Here a projection transforms each point in D from being n-dimensional to a typically lower m-dimensional point, using this transformation we can also define the projected dataset D^P.

$$P : \mathbb{R}^n \to \mathbb{R}^m \tag{2.1}$$

$$D^P = \{P(p_i) \mid p_i \in D\} \tag{2.2}$$

This function P has many different implementations, each has benefits in different areas. We would like to outline some of the main classes of techniques as it can help us reason about generated visualizations later on. Do note that the techniques we will use are agnostic to which P is selected. For our research we will consider techniques that are suited well towards 2 and 3 dimensional projections and use a variety of different underlying methods.

In the following section we will elaborate on which set methods we used. After we will dive into two classes of methods for which the context is important for reasoning about results or explanations later on.

2.1.1 Selected projection techniques

The works which propose the explanation techniques both claim that the method of projection does not matter and reason using the general projection P. We have no way to validate that it is truly general but we can explore a broad range of techniques and find which techniques perform better. For the selection process we relied on a recent survey done by Espadoto et al [10]. Here a set of 44 techniques were used in a 2D benchmark over numerous datasets. For our work we select 29 projections which, out-of-the-box, can be used for both 2D and 3D projections and are open source.

Table 2.1 lists, for these techniques, whether they are (non)linear, accept samples or sample-pair distances as input, project local neighborhoods differently or work globally the same for the entire dataset, their computational complexity, whether they have out-of-sample quality, whether they are deterministic or stochastic, and the public source of their implementation (for replication purposes). Complexity is a function of the number of dimensions n, number of samples N, number of iterations i (for iterative methods), and number of weights w (for deep learning methods). As visible, the selected projections cover a wide spectrum of methods.

2.1.2 Principal Component Analysis (PCA)

The first set of projection techniques we would like to pick out is the concept of Principal Component Analysis (PCA). This technique is especially relevant since it is also at the basis of the dimensionality based explanation, for this reason we will discuss it in more detail than all others. It is technique well known in the data analysis space as it is one of the simplest methods available, the dimensions under consideration are transformed into a new set of axis which are a linear combination of the input. They ordered in such a manner that the first component captures for as much variation from the original data, the second follows up with the same goal. Each additional component will explain less or equal amount of the variance until all component explain 100% of the variance. The way to utilize this property in a reduction can be done with thresholding in two manners. The first and most common is limiting the amount of components, here we take the top 2 or 3 components, we effectively maximized the amount of variance explained by this limited amount of axis available through linear combination [41]. While quality of these projections has been shown to be limited [42] it is a relatively simple, fast and predictable method. This makes it very a powerful tool in the validation of our explanation techniques, we can generate structures in 3D about which we can easily reason. Then using PCA project it into 2D where the relative positions of points is preserved.

Another method of thresholding is on the amount of variance, here we reduce the dataset removing components that explain little of the variance. We simply collect the top n components until the cumulative proportion of variation explained exceeds a threshold. This is also commonly used to find the *intrinsic dimensionality* of a dataset, i.e. how many latent variables are needed to explain 95% of the variance. This concept will come back later on in the localized explanation methods.

Another interesting application of PCA is finding the general shape and orientation of objects in 2 and 3 dimensions. Intuitively if all of the variance is

Projection	**linearity**	**Input**	**Neighborhood**	**Complexity**	**Out-of-sample**	**Deterministic**	**Implementation**
AE [16]	nonlinear	samples	global	$O(iNw)$	yes	no	Keras
DM [17]	nonlinear	samples	local	$O(N^3)$	no	yes	Tapkee
FA [18]	linear	samples	global	$O(n^3)$	yes	yes	scikit-learn
F-ICA [19]	linear	samples	global	$O(n^3)$	yes	yes	scikit-learn
G-RP [20]	nonlinear	samples	global	$O(Nn^3)$	yes	no	scikit-learn
H-LLE [21]	nonlinear	samples	local	$O(N^3)$	yes	no	scikit-learn
I-PCA [22]	linear	samples	global	$O(n^3)$	yes	no	scikit-learn
ISO [23]	nonlinear	samples	local	$O(N^3)$	yes	yes	scikit-learn
K-PCA-P [24]	nonlinear	samples	global	$O(N^3)$	yes	no	scikit-learn
K-PCA-R [24]	nonlinear	samples	global	$O(N^3)$	yes	no	scikit-learn
K-PCA-S [24]	nonlinear	samples	global	$O(N^3)$	yes	no	scikit-learn
LE [25]	nonlinear	distances	local	$O(N^3)$	no	no	scikit-learn
LLE [26]	nonlinear	samples	local	$O(N^3)$	yes	no	scikit-learn
L-LTSA [27]	linear	samples	local	$O(N^3)$	no	no	Tapkee
L-MDS [28]	nonlinear	distances	global	$O(N^3)$	no	no	Tapkee
LPP [29]	linear	samples	global	$O(N^3)$	yes	yes	Tapkee
LTSA [30]	nonlinear	samples	local	$O(N^3)$	yes	no	scikit-learn
MDS [31]	nonlinear	distances	global	$O(N^3)$	no	no	scikit-learn
M-LLE [32]	nonlinear	samples	local	$O(N^3)$	yes	no	scikit-learn
N-MDS [33]	nonlinear	samples	global	$O(iN^2)$	no	no	scikit-learn
NMF [34]	linear	samples	global	$O(n^2)$	yes	no	scikit-learn
NPE [35]	nonlinear	samples	local	$O(N^3)$	yes	no	Tapkee
PCA [18]	linear	samples	global	$O(n^3)$	yes	yes	scikit-learn
S-PCA [36]	linear	samples	global	$O(N^3)$	yes	yes	scikit-learn
SPE [37]	nonlinear	samples	global	$O(N^2)$	no	no	Tapkee
S-RP [20]	nonlinear	samples	global	$O(Nn^3)$	yes	no	scikit-learn
T-SNE [38]	nonlinear	distances	local	$O(iN^2)$	no	no	Multicore TSNE
T-SVD [39]	linear	samples	global	$O(N^2)$	yes	no	scikit-learn
UMAP [40]	nonlinear	distances	local	$O(iN^2)$	yes	no	umap-learn

Table 2.1: Selected projection techniques for evaluation and characteristics (Sec. 2.1).

explained along a single Component we can argue that the points of the input object are only spread along a line, with 2 component we have a flat surface and with 3 components we would have a spherical shape. This characteristic is referred to as *eccentricity*, it is defined by the dividing the largest Eigen value by the smallest one. The orientation of the component can also help you align objects with a xyz coordinate system, a common preprocessing step in 3D mesh classification [43].

2.1.3 Stochastic methods

To illustrate the wide variety of these technique I would also like to bring up another class of techniques, namely non deterministic methods. One of these is Multidimensional scaling (MDS) which works with the distances between each point as input and outputs point on an abstract cartesian space, commonly 2/3 when used for visualization. The core idea behind the technique is to minimize a loss function based on the stress between the actual distances and the distances in the current arrangement, here the data is randomly initialized and the in an iterative process the stress is lowered until you get the resulting projection.

t-SNE is another stochastic approach which differs in that it does not try to project data by looking at the pairwise distances, instead if makes tha assumption that there exists a low-dimensionality manifold in the high dimensional data along which the points lie. In other words it assumes that there is a limited amount of continuous latent variables which describe can be used to segregate the data. t-SNE tries to find this manifold and unrolls it to display the full data in two or three dimensions. It has a parameter called perplexity is an adjustable parameter that controls the size of the neighborhood around each object that it attempts to preserve. Here the data is also randomly initialized and the solution is found using a gradient descent method.

Both these methods can have identical inputs and parameters and have varying outputs, this is important to keep in mind when considering the reproducibility of our research. For this reason we will also provide all the precomputed projections on all the datasets that we will use in the later chapters, this allows third parties to replicate our results with relative ease. For more in-depth analysis of all the techniques I would like to refer back to the other works which include significant efforts to surveys the problem space [7]–[10], [44]–[50].

2.2 Quality metrics

As mentioned earlier, there is no one projection method that can *perfectly* embed the original nD data into a lower dimensional space. We have many methods what vary wildly in how they function, an important topic here of course is how to compare these. This is exactly where quality metrics come into play, when performing a survey or introducing a new projection these can be used to objectively analyse the results. Many surveys exist which introduce such metrics [42], [51]–[54] but personally I will consider a very recent entry made by Espadoto et al [10]. The general definition for these metrics is given by equation 2.3

$$M : \{(D, D^P)\} \to \mathbb{R}_+ \in [0, 1] \tag{2.3}$$

A metric M measures how well the projection D^P captures specific properties of the dataset D, the underlying idea being that a good projection will preserve relative distances between points. Here 0 denotes the minimal quality and 1 the maximal value. In the following section we discuss a series of these metrics, we will use these later on to compare between projections and between 2D and 3D.

2.2.1 Trustworthiness and Continuity

The first metric we will discuss is the trustworthiness and continuity of a projection. The core idea of dimensionality reduction is to lower the amount of dimensions while attempting to keep the relative distance between all the points. Trustworthiness measures the proportion of points that are close in D that are also close in D^P. We call this measure *Trustworthiness* because if can give an idea how much you can trust that the observed patterns actually represent ones in the dataset.

Continuity is closely related as it measures the amount of points that are close in D^P that are also close in D. It gives you an idea of missing neighbors, if this ratio is low you know that displayed neighborhoods are actually might not be a good representation of the original data. This metric is important when considering the explanation techniques from section 2.3. There we *assume* that neighborhoods in D^P are directly linked to neighborhoods in D, this metric is key when using our analysis tool. If we see that the continuity is poor we know that the explanation technique cannot be trusted either.

Both of these techniques are defined on a global scale in the works of Mateus et al [10], it they yield a single scalar value between 0 and 1 for the entire projection. This metric also lends itself to being used in local manner, we simply use the value per point by looking only at it's neighborhood and computing the proportion there. This can then be used to color code each point and give the user an idea of where the projection should not be trusted.

For the formal definition we first define U_i^K to be the set of points that are among the K nearest neighbors of point i in the projected space but not in $\mathbb{R}^n$. Conversely we have V_i^K for points that are neighbors in $\mathbb{R}^n$ but not in the projected space. We also use $r(i,j)$ to denote the rank of the projected point in the order set of neighbors in U_i^K, for the nD space we define $\hat{r}(i,j)$ which is the same for V_i^K. Finally we use N to denote the amount of samples in D. We can now combine these into equations 2.4 and 2.5 to get the formal definition of both metrics, these are taken from Table 5 of the survey paper [10].

$$\text{Trustworthiness} = 1 - \frac{2}{NK(2n - 3K - 1)} \sum_{i=1}^{N} \sum_{j \in U_i^K} (r(i,j) - K) \tag{2.4}$$

$$\text{Continuity} = 1 - \frac{2}{NK(2n - 3K - 1)} \sum_{i=1}^{N} \sum_{j \in V_i^K} (\hat{r}(i,j) - K) \tag{2.5}$$

2.2.2 Shepard diagram correlation

To access the so called goodness of fit you can utilize the shepard diagram, it plots the distance between points before and after a transform / projection in a scatter plot. points close to a diagonal indicate good distance preservation [55]. Points below, respectively above, the diagonal tell distance ranges for which false neighbors, respectively missing neighbors, occur. To compress this concept into a single scalar value we use the Spearman rank correlation of the plot, here 1 indicates perfect positive correlation and 0 denotes a complete lack thereof.

2.2.3 Stretching and compression

Stretching and compressions is a measures that illustrates the relation between distances for point $p \in D$ and all other points versus the same distances in the projected space. Compression is when these distances shrink in the projected space and respectively the stretching is when the distances grow. We might be able to apply this method using the neighborhoods in v_i^P and their original points v_i.

2.3 Explanation techniques

When explaining projected different types of metrics can be used to convey information to the end user. Simplest method is to just color code the points conveying the value of a point for a certain dimension, this sadly does not scale well. There also exist *global* metrics that provide information over all the given points, an example of this would be drawing the bi-plot axis [11], [12]. These can be used to find relevant view ports but are lacking when looking at subsets of the data. Conversely you have local explanation than only provide information about a single or subset of point(s), the most common example of this would be labeling each point.

The works of Da Silva [1] which was later extended by Van Driel [2] propose a local method to explain neighborhoods of points. They calculate this local metric for all points in the dataset and color code based on global rankings. Da Silva annotates the points based on the dimension that contributes most to making points in a neighborhood similar. Van Driel extents this idea by annotating points based on their local intrinsic dimensionality. Both these methods are originally designed to work in 2D but can easily be lifted to 3D. An interesting aspect of this metric is that we can also compute it in local region with different neighborhood sizes to find more information. It also provides information about which regions of a point cloud can be explained well, a property which could be useful when augmenting the interactions. In the rest of this section I would like to outline the formal definitions of both these methods.

Explanation techniques attempt to convey information about D while the user interacts with D^P combined with output of the explanations. Both works consider the neighborhoods of points in D^P and encode their color based on some metric in this neighborhood. These neighborhoods are defined by either an absolute radius value (r-nearest) or a certain amount of neighbors (k-nearest). In the papers radii are normalized using the projection width. We can now define a neighborhood in the projected space as follows:

$$v_i^P = \{q \in D^P \mid \|q - p_i\| \leq r\} \tag{2.6}$$

Here we will also define v_i which consist of the nD points that form a neighborhood in around p_i in the projected space. Using these neighborhoods we compute metrics that look at either which attribute contribute most or the local dimensionality.

$$v_i = \{q \in D \mid P(q) \ \in \ v_i^P\} \tag{2.7}$$

2.3.1 Attributes based explanation

For the attribute based explanation we will be using the aforementioned neighborhoods to determine which dimension (i.e. attribute) for the original points contributes the most to that points location. Here we will consider the euclidean distance or variance along said dimensions locally and normalize it used the global values.

For the euclidean method we will need to fist define the contribution $C^j_{p,r}$ of dimension j as the squared distance between two nD points p and r. Note that we will use superscript to denote the dimension and the subscript to denote indexes.

$$C^j_{p,r} = \frac{\|p^j - r^j\|_1^2}{\|p - r\|_n^2} \tag{2.8}$$

Here we use $\|\cdot\|_n$ to denote the n-dimensional euclidean distance. With the definition of the contribution between two points we can now move to finding the contribution of each dimension between a point $p_i \in D$ and all its neighbors $r \in v_i$.

$$\overline{C^j_i} = \frac{\sum_{r \in v_i} C^j_{p_i,r}}{|v_i|} \tag{2.9}$$

Note that we use $|\cdot|$ to denote the size of a set. This can be translated to finding the average euclidean distance per dimension from the point to its neighborhood. The authors also propose to use the variance in a similar manner for an alternative metric, one that is shown to be more resistant to noise. For this we define V^j_i:

$$V^j_i = var(r^j \ \in \ v_i) \tag{2.10}$$

For both these cases we will need a global normalization term because the distribution and scale might not be equal across all the dimensions. For this we first need to find the global equivalent of both metrics, for the variance it is quite simple, we just take the variance over each dimension for all points in the dataset. With the euclidean based contribution we take will use the distance to the nD centroid of the dataset. The formula for both methods can be generalized to one, consider here that (l^j_i), a local term, either C^j_i or V^j_i and g^j_i is the global term as described in this paragraph. The result if a ranking r^j_i for dimension j and the neighborhood v_i.

$$r^j_i = \frac{l^j_i / g^j}{\sum_{d=1}^{n} (l^d_i / g^d)} \tag{2.11}$$

For each index i in the dataset we will sort all the ranking values on descending order, we will use the top dimension j, i.e. attribute, to annotate the point. The last thing to do is the confidence encoding, here we simply look at all the annotations found in a points neighborhood. The confidence is defined by the amount of neighbors that share the same annotation divided by the neighborhood size.

2.3.2 Dimensionality based explanation

The work of Van Driel [2] builds upon the same core idea of using local neighborhoods and their respective *nD* points to explain the reduction, where it differs is in how it uses this neighborhood. Instead of seeking to explain the variance or distance contribution within a single dimensions it turns to principal component analysis to find how many components are needed to explain a certain amount of variance in the data. They propose a set of thresholding methods for these components and with it varying ways to find the confidence. The actual variance threshold, referred to as θ, can be defined by the user.

There are three different methods proposed to create annotations based on the output of the PCA. They all only consider the so called Eigen values found with the analysis, each value denotes the portion of the variance explain along the principal component. We sort these values in descending order and then use the three different methods to determine the dimensionality i.e.

$$E_i = \{E_i^1, \dots, E_i^n \mid E_i^j \geq E_i^{j+1}\} \tag{2.12}$$

Here the subscript denotes the index for the vector of Eigen values in the dataset and the superscript is used for indexing said vector. Using this set of values we can start discuss the methods, the first being a **Cumulative** approach. The user gives a threshold θ and then we annotate based on how many of the components are needed to exceed θ when added together. Formally the annotation a_i^{sum} can be found using the Eigen values E computed from v_i as follows:

$$a_i^{sum} = \min_k \left\{ \left(\frac{\sum_{j=1}^{k} E_i^j}{\sum_{j=1}^{n} E_i^j} \right) \geq \theta \right\} \tag{2.13}$$

For encoding the confidence we know that the k first Eigen values explain at least the portion θ of the variance by construction. Using this we can define find that the confidence is high if we explain exactly θ and the more we overshoot it, the lower the confidence.

$$c_i^{sum} = 1 - \left(\frac{\sum_{j=1}^{a_i^{sum}} E_i}{\sum_{j=1}^{n} E_i} - \theta \right) \tag{2.14}$$

It is crucial to note that the this definition differs from what is mentioned in the source paper, we do this because we believe that explanation is very difficult to reason about and slightly ill defined. Our method of encoding does penalize choosing low θ values, if you have a cumulative threshold on 50% of the variance and one area is dominated by 2 variables each explaining 40% you will see a *very* low confidence, especially with normalization on the confidence bounds. If this is observed the users will need to fine tune the θ which might be counter intuitive as that area is actually explained very well by these 2 variables.

The next method is better equipped to handle this case and is based on a **Minimal** approach, here the annotation is based by checking each Eigen value individually and counting how many exceed θ. This can be summarized as:

$$a_i^{min} = |\{e_i \geq \theta \mid e_i \in E_i\}| \tag{2.15}$$

The confidence encoding is defined by how much of the variance the top a_i^{min} Eigen values cover of the total. With it the definition is almost trivial, we simple sum the top a_i^{min} values and divide it by the total, i.e.

$$c_i^{min} = \frac{\sum_{j=1}^{a_i^{min}} E_i}{\sum_{j=1}^{n} E_i} \tag{2.16}$$

The final method of annotating based on the PCA output is to use a **ratio** based technique. Here the ratio between E_i and E_{i+1} is used, you count the number of steps i you can make before the ratio dips below a certain threshold. This method is very hard to interpret, it was omitted from the published paper and only defined in the thesis of van Driel [15]. Since it shows mixed results I will also not consider it in my work.

2.4 Interaction techniques

Since we will be building a tool from the ground up we can also consider how the user interacts with the data. Instead of producing a static image the users should be able to navigate it and explore what *they* deem interesting. They can guide the tool to regions they want to know more about and ask the tool to provide information on said area. We can split up the interactions into two different categories.

First we have the interaction used for viewpoint selection, when navigating the projection the user can use these to move what they actually see. Most tools provide basic camera controls that can be used to select the view points but with the move to 3D more complex navigation becomes available. An example of this is proposed in the works of Zhai [56], it allows you to rotate the entire cloud around structures in visible points. It works based by taking the screen space representation of the point cloud, calculating the skeleton and rotating around these skeleton vertices. Intuitively this is similar to holding a point of the object with you hands and rotating it. They show that this method of interaction is very intuitive and fast. implementing this technique is out of scope for our current version of the tool but might pose an interesting extension in the future.

The second type of interaction would be for selection of data, providing the user with ways to highlight a subsection of points. It can be beneficial to zoom into a certain area of interest and recompute the metrics on this subset of the original data. Local patterns might can clearer and the computations faster. It could also be used for navigation by finding the centroid of a set of selected points and moving the cameras focus there. To use these techniques you first need a robust way to select points using a 2D screen bound input method. A common solution for this are lasso's, simply draw an figure around the points you want to select. This works very well in 2D but can fail in 3D because then you are actually selecting a cylinder. You could solve this using multiple selections from different view ports and combining these using boolean operations, but another option exists, namely LassoNet [57]. Its goal is to provide an easier and more intuitive way to do this. It is based in deep learning and provides a model that is trained using 2D lasso selections combined with the set of correct points that should be selected. It has been proven as a powerful data selection tool. Another way to aid selection process is to have the use select a single point and have the tool expand this outwards from this point to select a region that it thinks it can explain well, a good way to achieve this is to use the quality metrics discussed in section 2.2. For our use case neither will be applied, we will only allow selection of single points which can be found with some simple algebra.

Chapter 3

Implementation

This chapter will focus on our implementation which build upon the techniques discussed in chapter 2. Previously the focus was on individual techniques and now we will combine these into a real world application. For this application we set 3 key requirements namely, correctness, interactive usage and ease of use. We will elaborate on this in section 3.1. Section 3.2 is more related to system architecture, we illustrate and explain how data flows through our tool. Next we will dive into more details, first we discuss how we achieve interactive rates in section 3.3, then in section 3.4 we outline the aspects of the visualization itself. Finally in section 3.5 we will validate that the produced results are correct using primarily synthetic datasets.

3.1 Requirements

When moving from theory to practice a range of new consideration come into question, we aim to build a real world system which people can, and want to use after reading the works. As with any system it is important to outline the focus and key requirements of the system before building. In this section I will enumerate all the requirements.

Correctness: First and most important point is that our system actually produces images that are correct with the definition in the source papers and with similar visuals. Without this the tool is not usable, all other aspects build upon this. We will focus on this topic in section 3.5.

Interactive: Users should be able to interact with projections at interactive rates, this includes changing rendering settings, navigating and running actual explanation algorithms. Section 3.3 will outline how our tool handles this.

Accessibility: Our application should be simple to install and run for general data analysts, ideally the program fits within more traditional work flows. The works provide visualization that strongly differ from the norm, our tool needs to include well known explanations as well.

3.2 Data processing pipeline

The point processing pipeline can be split into 2 sections, an offline section and an online section. Offline implies that this section will be run while the program is not in use, i.e. before starting the viewer. The online section work in realtime while the viewer is active. A complete high level overview of the pipeline can be found in figure 3.1. In this section we will talk through the elements of this pipeline.

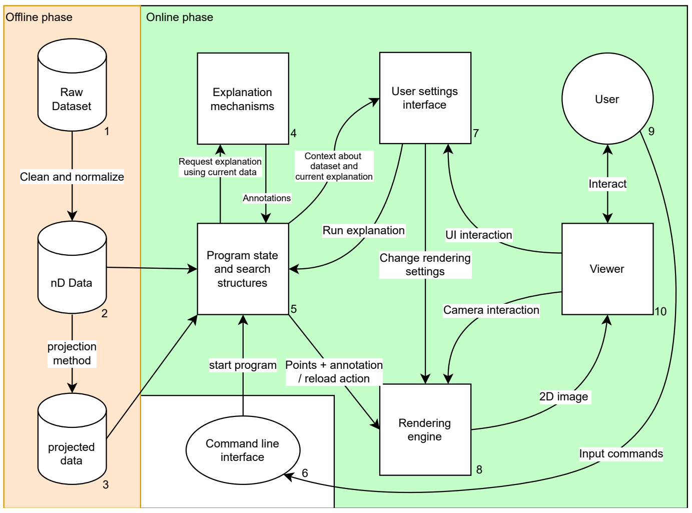
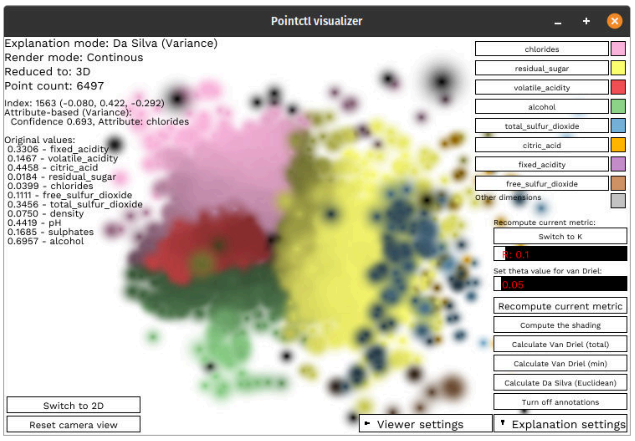


Figure 3.1: High level overview of the structure of our tool

The offline part of the pipeline consists of only the projection method. The program is agnostic to what reduction method is used, an assumption is made that the end user want to use this program to annotate a reduction plot not to create one. It is possible to run the explanation module without starting the viewer itself, this would make it part of the offline phase as this will only output the annotations. To do this you can start it via the command line interface. The main reason this is included is for testability and benchmarking purposes, it allows us to run part of the online phase with a deterministic input which is nice when comparing versions of the program.

Using the the original and reduced data we can enter the online phase of the pipeline. The user can start it using 2D and/or 3D data, you can switch between the 2 while running the program if both are loaded. Both will be stored in an data structure optimized for nearest neighbor searches, in the diagram this is labeled with 5. Using this data the rendering engine is initialized without any annotations (8) and UI is also loaded with default parameters (7). The output of the render is shown in the viewer with the UI as an overlay (12), together the are shown to the user. The UI allows the user to modify the current gamma, point size, rendering method and also recompute the current metric using different parameters (see figure 3.2). UI delegates these tasks to either the rendering engine or the program state. When running a calculation the the program state will borrow the data structure to the explanation module, this will return the annotations which are cached in the state and then passed on to the rendering engine again. During this process we display progress bars if the computation is not completely realtime. The caching allows the user to quickly switch between two different explanations without having to recompute the whole metric.



```
Explanation mode: Da Silva (Variance)
Render mode: Continous
Reduced to: 3D
Point count: 6497

Index: 1563 (-0.080, 0.422, -0.292)
Attribute-based (Variance):
  Confidence 0.693, Attribute: chlorides

Original values:
 0.3306 - fixed_acidity
 0.1467 - volatile_acidity
 0.4458 - citric_acid
 0.0184 - residual_sugar
 0.0399 - chlorides
 0.1111 - free_sulfur_dioxide
 0.3456 - total_sulfur_dioxide
 0.0750 - density
 0.4419 - pH
 0.1685 - sulphates
 0.6957 - alcohol
```

```
Recompute current metric:
R: 0.1
Set theta value for van Driel:
0.05
```

Figure 3.2: Generic screenshot of the viewer with the user interface enlarged

3.3 Interactive computation

Arguably the most difficult constraint on our entire program is the fact that we want it to run these explanations in realtime, this lifts the tool from just another novel implementation for the exploration technique into one that can be used to explore and tune varying explanation. The core bottle neck in the entire computation are the explanation algorithms themselves, the rough goal we set for ourselves is that we can process *100.000* points and get results within a minute (given reasonable neighborhood sizes). The thesis of Van Driel [15] only shortly mentions computational scalability, for this he utilizes the wine quality dataset containing 6773 observations and 12 dimensions. His implementation could compute the PCA_{sum} metric for that dataset with radius 0.1 in *1 minute and 8 seconds* excluding precomputing of distances. Our program can run the same explanation including computation of distances in **0.28 seconds**. This computation also scales into reasonable run times for datasets which are an magnitude larger. In this section I will discuss what we did to make this possible.

3.3.1 Programming language (Rust)

The previous works had a strong focus on a proof of concept for the explanation techniques, they relied on Python code to compute the annotations and visualization. A good choice given that the goal of just generating results of the paper, one down side is that while easy to get running Python is notoriously slow for pure compute. For this specific reason we opted to look at a lower level language, here we get more control over data layouts and memory management. The obvious choice in this case would be to employ C++, it has a strong community and presence in among info visualization program. It also provides good ways to handle graphics bindings and has a lot of resources on how to us them. But it is not all good news, the manual control of memory can lead to data races and can make it challenging to parallelize part of your program.

This is where our language of choice comes in, Rust, a young language which aims to provide safety guarantees like high-level languages and performance efficiency like low-level language. At the core stands a set of strict safety principals which are enforced at memory safety at compile time. This so called ownership model allows users to exactly track where data is mutated and in turn makes parallelization easier to attain.

With Rust we will able to build a fast implementation using modern build tool that are easy to use for others. It also has a robust testing suite we can use to write tests for properties we implement. These are the first step towards meeting our requirements.

3.3.2 Data structures

An important consideration for our program is the algorithmic complexity, if we want to achieve interactive rates large datasets our the computations need to be optimized. The first high level consideration is choosing data structures that scale well of the operations needed by the explanation algorithms. The core problem we need to solve is a fast nearest neighborhood search in both two and three dimensions. For this we need to consider a class of data structures called spatial indices.

Structure	**Memory usage**	**Search**	**Construction cost**	**K nearest neighborhood**
Z-order [58]	$O(n)$	$O(n)$	$O(n)$	O(n)
kd-tree [59]	$O(n)$	$O(\log n)$	$O(n \log n)$	$O(k \log n)$
VP-tree [60]	$O(n)$	$O(\log n)$	$O(n \log n)$	$O(k \log n)$
R*-Tree [61]	$O(n)$	$O(\log n)$	$O(n \log n)$	$O(k \log n)$

Table 3.1: Comparison of spatial indexing methods.

In table 3.1 we enumerate a series of structures for consideration, we can already determine that the first option is not suitable from the search costs. Two closely related methods are KD and R* trees, each has different benefits but from my research there exist more optimizations for searching in the later. The main reason for this is that the KD variant works with strictly disjoint partition (one point per region) where they can overlap in R trees [62]. R-trees and by extension R*-trees are also flexible in what is stored, instead of points it can store rectangles or polygons as well [61].

The last contender are vantage point trees, they work based on using position as vantage point and then segregating the rest of the points based on a distance threshold. This allows for a nice abstraction from the dimensionality of data and it can also be used for different distance metrics (e.g. earth mover distance between vectors [63]). While these are nice they are not requirements in our cases, this leaves us with one important final consideration, namely, availability.

As it turn out no solid implement exists for VP trees in rust, we have one version[1] that has seemingly been abandoned and thus poses a liability to the maintainability of our program. Conversely we have broadly used and actively maintained implementation[2] for R*-trees, with this in mind we choose this data structure for our program. Another advantage this library has is tight and exact packing of data, this means more of our computations can be preformed on the stack and this will lead to significant speed gains [64]. A set of benchmarks for this final data structure can be found in table 3.2

[1]https://github.com/kornelski/vpsearch
[2]https://docs.rs/crate/rstar

benchmark	Tree size	Timing
bulk loading	2000	112.50 us
nearest neighbor (bulk loaded tree)	100k	1.22 us
successful point lookup	100k	167.32 ns
unsuccessful point lookup	100k	263.ns

Table 3.2: Benchmarks for the Rust R*-Tree implementation for 2D points. Computed using criterion on a AMD Ryzen 3900X @ 4.2GHz over averaged 1000 runs

3.3.3 Parallelism

Moore's law tells us that transistor density in computers doubles roughly every 1.5 years [65]. This is a trend we still see today and besides seeing more complex cores we are also seeing more cores per computer. This increase in cores does not directly translate into faster programs in practice, it requires developers to consider working with concurrent tasks and data instead of sequential ones. Running our computation in parallel also does not allow you to ignore bad algorithmic complexity, at best we can expect linear increases in performance over the sequential variant. Amdahl's law [66] computes the maximal amount of performance increase we can expect given which proportion $P \in (0, 1]$ of the program is run in parallel and the speed up gained by parallelization said portion S (i.e. the amount of cores).

$$\frac{1}{1 - P + \frac{S}{P}} \tag{3.1}$$

Our plan of approach is to parallelize the computation of metrics over the points, so all the computations listed in the source works except the creation of the search structures, this is roughly 90% of the programs run time. Using the aforementioned formula we can extrapolate a best case 10x performance increase with an arbitrary amount of cores, and 3x performance using 4 cores. A plot of this behavior can be seen in figure 3.3. We argue that this increase in performance outweighs the additional time spend on implementation.

Lets us now discuss how we made the program make use of the cores available in the system, To achieve this we need to consider two main problems, concurrency of both tasks and data. On a high level this problem is actually quite simple, we first create a search structure based on the 2D and 3D point clouds. From then on this search structure is immutable and thus safe to read from using multiple threads. We then start create parallel tasks per point, each task computes the neighborhood and then uses it to compute the metric. We then collect the results. After the search structure is mutated from a single thread to store the data again. Using Rusts strong ownership model this approach becomes feasible.

With this implement we can verify our hypothetical gains, sadly we have no machine huge amount of cores available but we can use a workstation with 24 threads[3]. For this we will run our the variance based explanation for the TSNE projection of 3 datasets. We use default credit rate for its size along the sample axis, reuters for the amount of dimensions and finally wine for having neither. We will use 3 different radii settings averaged over 5 runs. We will

[3]tested on a AMD Ryzen 3900x

plot the percentage of performance gained over the single threaded baseline by moving to 2, 4, 8, 12, 18 and 24 threads. The result is shown in figure 3.3 and raw results in table 3.3. We averaged based on radius as this was the one factor which determined the ratio of runtime between loading in the dataset and computing the metric.

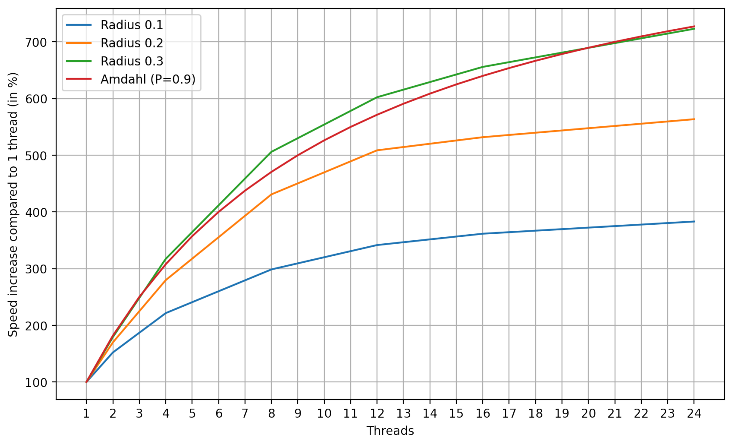


Figure 3.3: Benchmarks for various datasets averaged by radius and Amdahl's law $P = 0.9$ over core count. Each benchmark ran 5 times

We can conclude that the hypothetical gain was inline with the real world application, we see an increase of almost 250% using a 4 cores machine and 500% using 24 thread workstation. An great step towards our interactive requirement and improving the scale of datasets that could be handled. We find that that the as more of the runtime is spend on the metric (i.e. large neighborhood size) the use of more cores because increasingly more important. With Amdahl's law in mind we can also extrapolate that that running the program on specialized, highly threaded, hardware is not worth it unless you have very large neighborhoods. There is more to gain by speeding up the sequential section with better single core performance.

3.3.4 Accessability

From my personal experience I found that implementing these algorithms in a fast manner can be a time consuming process, implementations exist but are not easily accessible which make further research cumbersome. With this in mind I will create and publish a Python 3 package and make the Rust code publicly available. While I am a strong advocate for Rust I simply cannot deny the strong presence of Python in the Data Science landscape. This work can serve as a know good and slightly optimized implementation of the explanation metrics. With this I aim to make further research easier and adoption into other tools more attainable. Links to these can be found in section 3.6.

	Dataset	Radius	Threads	Timing (seconds)	Increase over single thread
0	Default cc	0.1	1	2.371	100.00%
1	Default cc	0.1	2	1.539	154.06%
2	Default cc	0.1	4	1.092	217.12%
3	Default cc	0.1	8	0.869	272.84%
4	Default cc	0.1	12	0.798	297.12%
5	Default cc	0.1	16	0.767	309.13%
6	Default cc	0.1	24	0.739	320.84%
7	Default cc	0.2	1	6.940	100.00%
8	Default cc	0.2	2	3.895	178.18%
9	Default cc	0.2	4	2.307	300.82%
10	Default cc	0.2	8	1.482	468.29%
11	Default cc	0.2	12	1.233	562.85%
12	Default cc	0.2	16	1.127	615.79%
13	Default cc	0.2	24	1.016	683.07%
14	Default cc	0.3	1	17.098	100.00%
15	Default cc	0.3	2	9.206	185.73%
16	Default cc	0.3	4	4.957	344.93%
17	Default cc	0.3	8	2.846	600.77%
18	Default cc	0.3	12	2.168	788.65%
19	Default cc	0.3	16	1.954	875.03%
20	Default cc	0.3	24	1.699	1006.36%
21	Reuters	0.1	1	10.685	100.00%
22	Reuters	0.1	2	6.071	176.00%
23	Reuters	0.1	4	3.548	301.16%
24	Reuters	0.1	8	2.310	462.55%
25	Reuters	0.1	12	1.894	564.15%
26	Reuters	0.1	16	1.758	607.79%
27	Reuters	0.1	24	1.626	657.13%
28	Reuters	0.2	1	35.901	100.00%
29	Reuters	0.2	2	19.874	180.64%
30	Reuters	0.2	4	10.845	331.04%
31	Reuters	0.2	8	6.315	568.50%
32	Reuters	0.2	12	5.234	685.92%
33	Reuters	0.2	16	5.377	700.68%
34	Reuters	0.2	24	6.530	710.79%
35	Reuters	0.3	1	79.763	100.00%
36	Reuters	0.3	2	42.159	189.20%
37	Reuters	0.3	4	22.845	349.15%
38	Reuters	0.3	8	13.984	570.39%
39	Reuters	0.3	12	12.835	621.45%
40	Reuters	0.3	16	11.709	681.21%
41	Reuters	0.3	24	11.124	717.04%
42	Wine	0.1	1	0.250	100.00%
43	Wine	0.1	2	0.198	126.26%
44	Wine	0.1	4	0.171	146.20%
45	Wine	0.1	8	0.156	160.26%
46	Wine	0.1	12	0.153	163.40%
47	Wine	0.1	16	0.149	167.79%
48	Wine	0.1	24	0.146	171.23%
49	Wine	0.2	1	0.466	100.00%
50	Wine	0.2	2	0.309	150.81%
51	Wine	0.2	4	0.225	207.11%
52	Wine	0.2	8	0.182	256.04%
53	Wine	0.2	12	0.168	277.38%
54	Wine	0.2	16	0.167	279.04%
55	Wine	0.2	24	0.157	296.82%
56	Wine	0.3	1	0.798	100.00%
57	Wine	0.3	2	0.490	162.86%
58	Wine	0.3	4	0.310	257.42%
59	Wine	0.3	8	0.230	346.96%
60	Wine	0.3	12	0.201	397.01%
61	Wine	0.3	16	0.194	411.34%
62	Wine	0.3	24	0.179	445.81%

Table 3.3: Raw benchmark results for parallelism test, each row is averaged over 5 runs.

3.4 Point cloud visualization

One more practical aspect of working with point clouds is how these should be visualized. After reducing the data and running an explanation techniques we are left with a set of annotated discrete points. Points per definition have no surface area, to display them on a screen we need to change this. In section 3.4.1 I will discuss how to resolve this. Following this I will discuss a method to add approximate shading to three dimensional points which can be used convey shape in section 3.4.2.

3.4.1 Rendering methods

The easiest and most common approach is to simply draw a small square around the point, the size of these points can be configured at an early stage of the rendering pipeline and the size proportional to each other is handled by the render as well. Points further away are smaller with a lower bound on a minimal size, point closer appears larger with an upper bound. This basic technique has the advantage of being very fast but has a few drawbacks. The most relevant ones are that it has poorly handles differing density areas in the dataset, when using small point the sparse areas can look poor and vice versa for the dense area with large points. Another key aspect is that these points can not convey occlusion in three dimensions, point drawn directly next to each other in screen space might have wildly varying Z position. From now I will refer to this approach as the discrete method.

A slightly more advanced approach to this problem is rendering each point as disk with an opaque center and transparent edges. Using this technique will allow point to blend together at the edges, this becomes relevant when we start color coding all the points based on the annotations. It will offer an approximate explanation for the area between the points as well. This is a crude but a common solution to the problem. This approach is relatively simple in 2 dimensions but the element of opacity will cause problems in 3 dimensions. The reason of this is that in 2 dimensions the depth or Z value is constant for all the points relative to the camera position, in 3 dimensions this is not the case. The solution for this is to simply sort the points based on the screen space Z coordinate and rendering the points from back to front, each point will be over layed on all currently rendered points. In this order the point closed to the camera will always be visible and can (partially) occlude other points behind it. I will refer to this method as the continuous rendering method. A side by side comparison of the described methods can be found in figure 3.4.

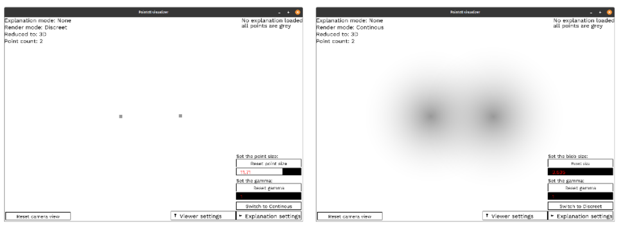

```
Explanation mode: None
Render mode: Discreet
Reduced to: 3D
Point count: 2
No explanation loaded
all points are grey
Set the point size: 11.71
Set the gamma: 1
```
```
Explanation mode: None
Render mode: Continous
Reduced to: 3D
Point count: 2
No explanation loaded
all points are grey
Set the blob size: 3.636
Set the gamma: 1
```

Figure 3.4: Two nearby points rendered in a discrete (left) and continuous (right) manner

One aspect that I glanced over while discussing both methods is the orientation of the *flat* shapes we used to represent the point. This is no problem in two dimension since the camera orientation is static, this is not the case in three dimensions. Here we need to ensure that the normal of these shapes faces the camera. Luckily this can easily be solved in the vertex shader of the rendering pipeline.

To achieve this effect we will render a two triangles forming a square consisting of a total of 6 points (of which 4 are unique) on a 2D plane. These 6 points are all offset by consistent amount determined by the size of the blob or point. These 2D dimensional offset along the $x-$ and $y-$axis will be applied to the point in the the *view* space, i.e. the coordinates system with the camera at the origin. With this we know that the normals are in parallel with the $z-$axis. The final step is to project these into the screen space using the projection matrix, since we are using a pinhole camera this will be a perspective projection. Summarized we have the transformations in equation 3.2.

$$\begin{aligned} P_{world} &= \begin{bmatrix} x_{world} \\ y_{world} \\ z_{world} \\ 1 \end{bmatrix} \\ P_{view} &= M_{view} \times P_{world} \\ P'_{view} &= P_{view} + \text{Offset} \\ P_{screen} &= P'_{view} \times P_{projection} \end{aligned} \tag{3.2}$$

This trick is referred to as bill-boarding and can often be found in older video games to save on resources. It is fast and easy to implement which is great for our use case. Two real world examples can be found in figure 3.5, The continuous technique allows you to more easily spot regions of related points and can also provide proper occlusion in 3D where the discrete method lacks. Sadly the user can still ont distinguish between the convex and concave areas. The solution for this problem, shading, will be discussed in the following section.

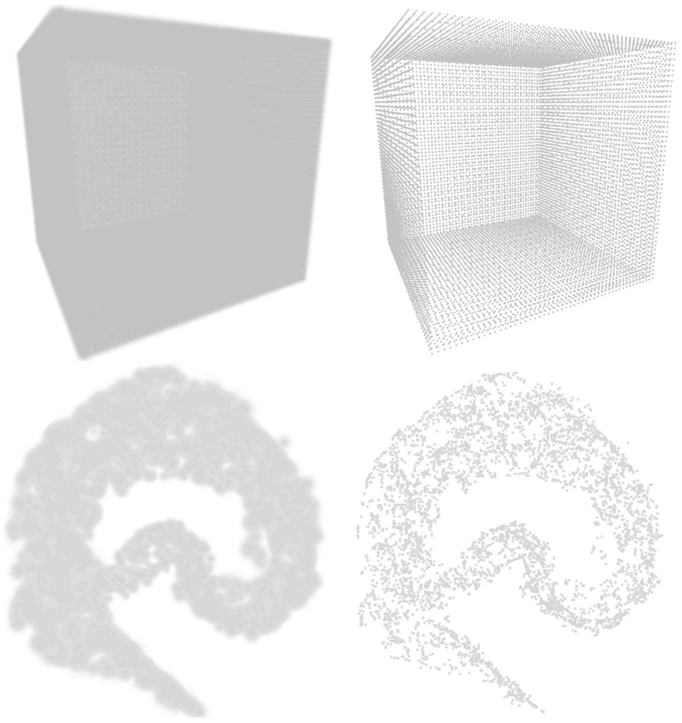


Figure 3.5: Comparison of discrete rendering (left) with continuous rendering (right)

3.4.2 Pseudo shading the point cloud

While viewing 3D datasets it is natural for the viewer to relate their perception of the point cloud to what it would look like in real life. You try to perceive which part occludes what other section and which areas are concave or convex. The way to convey this to the user when rendering objects is to use a light source and shading, it can be used to convey to orientation of a surface while in the 2 dimensional screen space.

The first option that might come to mind is recreating a triangle mesh from the point cloud, with such a mesh shading the dataset would become trivial. Sadly this is an approach which has two significant drawbacks. First being that I would give us information that we gain information about the normals of the faces, not the vertexes. We could approximate this via by looking at the adjacent faces and edges. Sadly this leaves us again with just an approximation while it costs a lot of compute to find, this cost is also our second large drawback. reconstruction of a mesh can easily take minutes to compute when the point set is large enough. This would be disastrous for the useability of our tool. Instead of a full mesh reconstruction we could also look into doing one based on the current viewport, one of these methods is proposed in the works of Katz et al [67]. It is shown to be significantly faster but the main downside is that we would have to recompute it every time the viewport changes. This makes this interesting work unusable as well. With all of this in mind we can conclude that we would need to look in another avenue.

A completely different approach to the same problem is to try and find the orientation and shape local areas. This might seem very familiar to what we already did for the dimensionality based explanation, here we will again use a neighborhood based explanation method in only the 3 dimensional space. Within the neighborhood we perform PCA to find out along which axis *the least* amount of variance occurs, i.e. the minor Eigen vector. Intuitively the cloud of points is thinnest along this axis, you could argue that this vector can be seen as a normal along the flattest part of the neighborhood. This in combination with the aforementioned eccentricity ratio we can also say how confident we are that this is a proper normal, we know that if the eccentricity is 1 the variance point cloud is perfectly distributed along the axis i.e. it is round. When this is the case the normal we found is not very useful. This is often an indication that the point is actually under the surface of the mesh, these points should not be included in the shading process. Conversely when we find a low eccentricity we know that the point cloud is shaped more like a line of disk, this also indicates that the point is on a surface and that shading it would be beneficial.

With this method we can relatively quickly find an approximate normal and we can find how confident we are in that it is a proper normal. These values can be used later on in the rendering pipeline to shade all the points in the continuous and discrete representation. Here we will use the camera position as a light source, we can cast a ray from our camera to the faces of the points. For each we find the angle between the normal and said ray. If the angle is small much of the light would be reflected back at the camera so it will result in a bright color, if the angle is closer to perpendicular then all the light would be reflected away leading to a dark color. During this process we will use a certain upper bound for the eccentricity between 0 and 1, the higher the value the further under the surface of the cloud points will be shaded.

Application on point clouds

Let us now look at some concrete applications of this, we start by looking at the shading computed for a plane in 3D. The goal is to get a general idea of the orientation of that plane. For this we use a synthetic dataset in where the original data is along the faces of a cube in 3D and projected into 2D using PCA (a linear transformation). For this test we pad an extra z coordinate to the 2D projection with value 0 everywhere. Shading over this dataset is shown in figure 3.6. While explanations are included here it is advisable to tune the parameters (i.e. neighborhood and shading intensity) on the grey points.

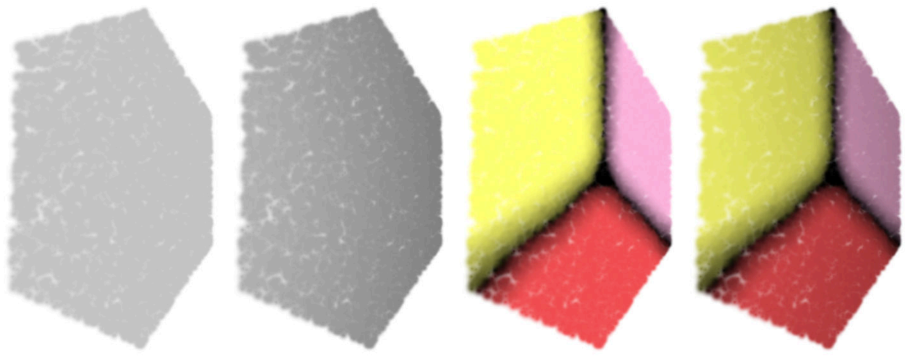


Figure 3.6: Shading approximation for points on a plane. Without explanation on left, with on the right. Right in the couples is with shading.

Another example of this method in action is a synthetic dataset where all the points align with the cases of a cube. Without shading it is impossible to determine the shape without moving the shape around and piecing it together by looking at silhouettes. This dataset is depicted in figure 3.7, here the faces and edges of the cube shape become apparent. In figure 3.8 you will find a slightly modified version of the same dataset, here the entire filled up with points. The technique will shade the points under the surface for 50%, the layer underneath is visible but it fact that these points are not on the surface is conveyed well. The last basic shape used during validation is a set of points along a sphere, this is displayed in figure 3.9. Again the benefits of the shading process are apparent, you can make out that this is a concave shape instead of a flat area.

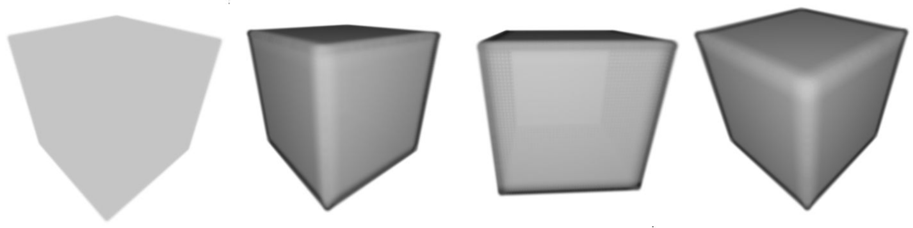


Figure 3.7: Shading approximation of point on a hollow cube, first left without any shading.

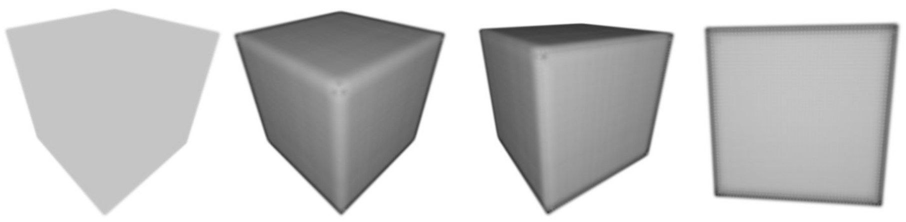


Figure 3.8: Shading approximation of point on a full cube, first left without any shading.

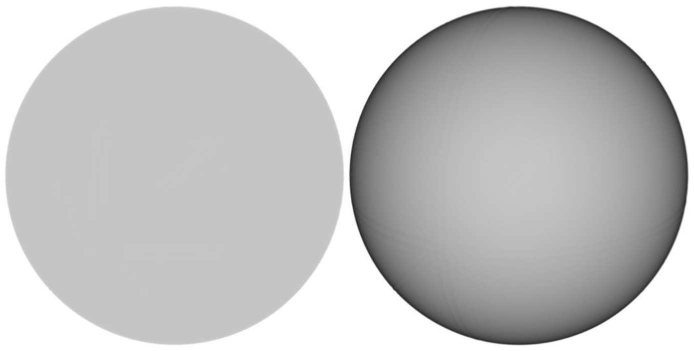


Figure 3.9: Shading approximation of point on a sphere

Besides looking at basic shapes I also considered more complicated shapes, for this I used the infamous *Stanford Bunny* [68]. This is a mesh originated from a dataset of scanned objects, as a result it contains only points on the surface. The results are shown in figure 3.10, a another clear example where shading gives insight into which areas are concave or convex, the area around the ears especially. The final test for this method is a real world dataset, for this I turned to the TSN-e reduction of the Wine quality dataset. The reasoning here is that here the points are roughly aligned along a 2 dimensional manifold in 3D. This lends itself nicely to shading, the result of this are shown in figure 3.11.

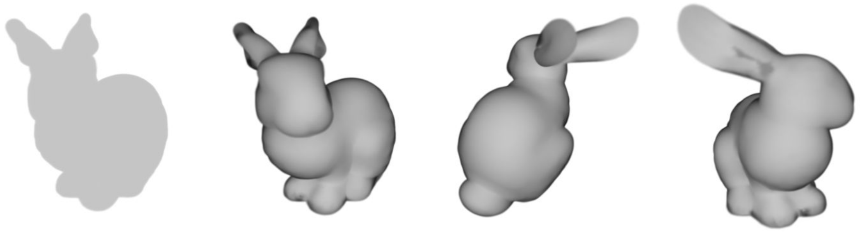


Figure 3.10: Shading approximation of faces of the Stanford bunny [68], first left without any shading.

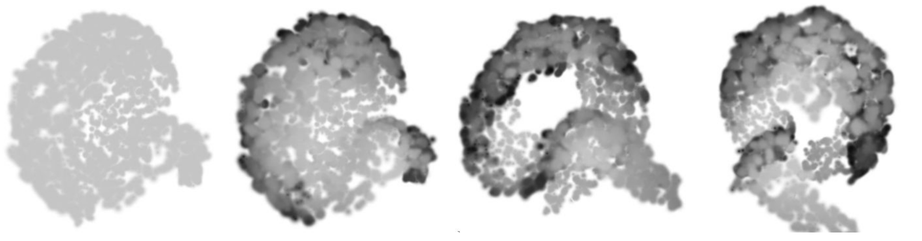


Figure 3.11: Shading approximation of TSN-e reduced Wine quality dataset [69], first left without any shading.

From working with this shading process it has become very clear that the neighborhood size required to get a clean normals differs per dataset used. I used a default radius of 10% of the projection width which works well for the basic shapes but produced to much noise in an actual dataset. It is advisable to tune this parameter on the grey points before running the explanation techniques, namely because these will also use the brightness to convey information about confidence and this might interfere with finding the correct shading.

3.4.3 Color encoding

A key aspect of displaying the outcome of our explanations is how we encode these results into a color. Within the tool we have 3 different types of color maps, categorical, ordinal and single value.

The single value color map is the simplest one, here we only have one input value between 0 and 1 and convert this into a shade between red and green. Here the perceived brightness of colors is not taken into consideration. Next we have the categorical version, here we use the color to encode the top-8 categories of points and then the brightness for the confidence. Since brightness is also relevant the colors we use for the categories need to appear equally bright, this property is called isoluminant. To find such a set of colors we turn to research done on the topic of perceived distance between colors [70]. With such a color map we can simply encode the confidence via V in the HSV color space and pick the color based on the dimension. Lastly we have the ordinal color maps, here instead of categories the explanation output has both confidence and an attribute subject to a natural ordering (i.e. number of dimensions). Here we combine the two previous ideas, we create a shade between blue and yellow to plot the ordinal attribute on, using the brightness we then encode the confidence again.

3.5 Validation of explanations

An important step for out tool is verifying that the produced images are correct, as stated in section 3.1 this is the most important part of our tool. Via unit tests we can test if small computations have been implement correct but for end to end testing we will need datasets for which we already have explanations. To get such a dataset we need to generate them, this way we can control the underlying variance for separate areas of the projection. In this section we will talk through the key explanation methods and reason about their correctness given the synthetic datasets.

3.5.1 Attribute based explanation

The work of Da Silva provides an attributes based local explanation that is computed for every point in the dataset, it is displayed using a color map based on the global ranking. Two metrics are proposed, one that measures the euclidean distance contribution of each dimension within a neighborhood and another that does the same for variance. The paper mentions that the variance based metric is more resistant to noise and faster to compute, I implement both and observed very similar results for the cube data set. This dataset consists of 10.000 points along the faces of a cube and introduced a spatial noise of max 5%, the reduction is done using PCA [71] giving us a very predictable outcome. The expected explanation is that there are 3 regions that are close because they are roughly along a plane in the original data. At the border of these regions the confidence drops as the neighborhood now contains points from 2 planes. Here employ neighborhoods using a radius that is normalized using the projection width. The results of this explanation for euclidean metrics can be found in figure 3.12, the variance based approach in figure 3.13. We can very clearly see that both approaches result are comparable with the ones from the source paper figure 1 [1]. Here you can see that the borders are clearer for euclidean, the larger the radius the larger the uncertainty is along the border of the regions.

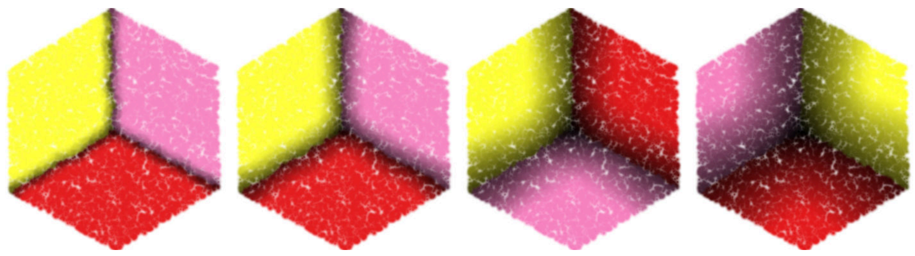


Figure 3.12: Da Silva's distance metric on a PCA reduced synthetic cube with 5% spatial noise. Neighborhood is radius based with 5%, 10%, 20% and 30% of the projection width.

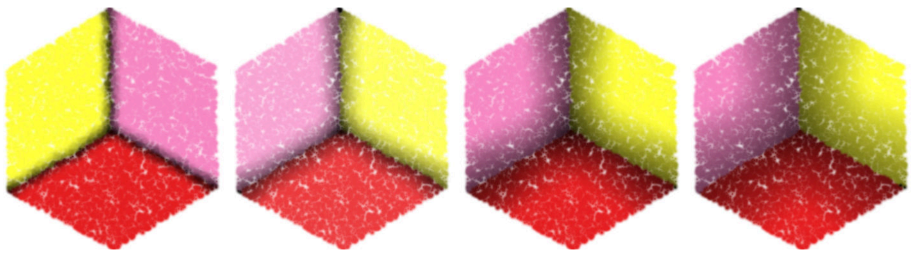


Figure 3.13: Da Silva's variance metric on a PCA reduced synthetic cube with 5% spatial noise. Neighborhood is radius based with 5%, 10%, 20% and 30% of the projection width

An extension we did to the explanation the introduction of a per attribute view, here you can see for each point how well variance along said attribute explains it. For the PCA cube example these views are shown in figure 3.14. Here we can clearly see the expected regions are green, the areas along the face of the cube are green and the rest is red.

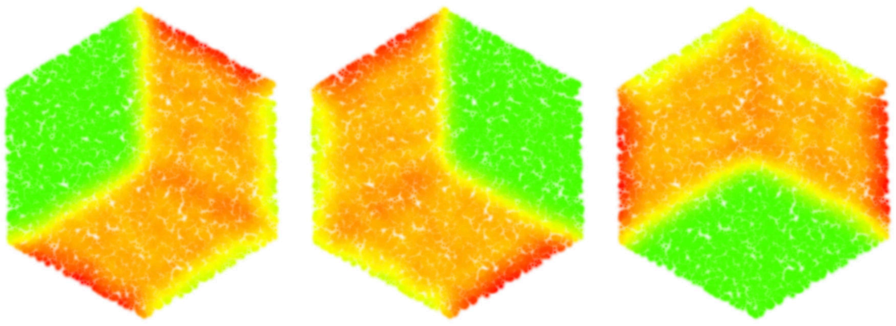


Figure 3.14: Single attribute views for cube from figure 3.13, attribute Y, Z then X. $R = 10\%$

3.5.2 Dimensionality based explanation

Next we will discuss the dimensionality based extension as proposed by Van Driel. This work will look at the intrinsic dimensionality of a point in its neighborhood. For the formal definition I would like to refer back to section 2.3, here we will show our own implement and argue that the results produced are correct. Similar to the attribute based technique starts by checking our own results against images produced in the source paper, there a series of synthetic datasets were used that we will also produced.

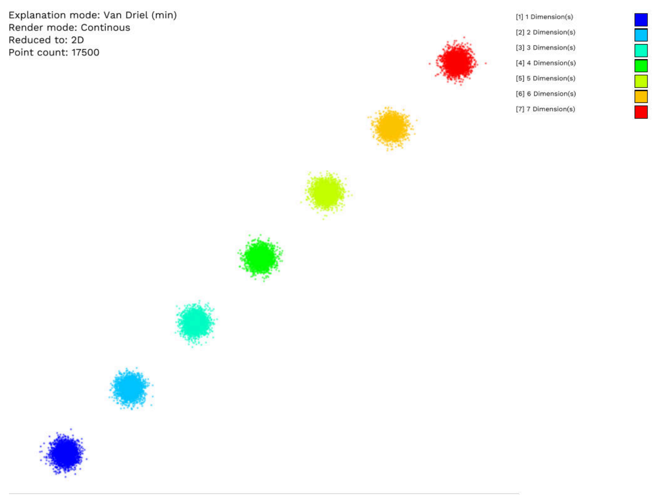

Explanation mode: Van Driel (min)
Render mode: Continous
Reduced to: 2D
Point count: 17500

Figure 3.15: Dimensionality metric (min) for the nD clusters synthetic dataset

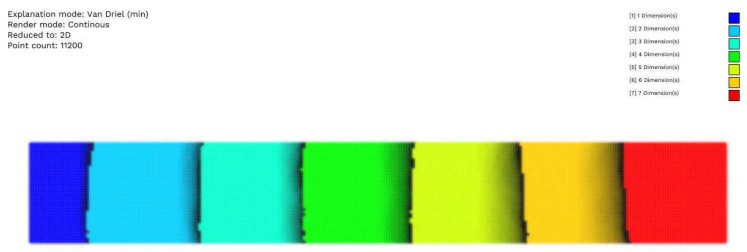

Explanation mode: Van Driel (min)
Render mode: Continous
Reduced to: 2D
Point count: 11200

Figure 3.16: Dimensionality metric (min) for the nD fade synthetic dataset

The first generated dataset I use for validation is the so called *7D-clusters* set. Here we start by generating 7 non overlapping clusters using gaussian blur, each with about 2500 points. Then per cluster of points we generate the 7D "unprojected" point with gaussian noise across a set amount of dimensions. For the first cluster we only use 1 dimension and we increment it for each cluster until we have the noise across all 7 dimensions. For this dataset we *know* the intrinsic dimensionality per cluster. Using the dimensionality based explanation method we should be able to find how many dimensions are needed to explain the variance in said cluster. The results of this can be found in figure 3.15. This dataset will provides us with a basic sanity check. The next dataset I would like to discuss is the *7D-fade*, here instead of distinct clusters all points are arranged in perfect squares directly adjacent to each other. in this use case we should start seeing the confidence encoding doing its work at lines where the dimensionality jumps. This explanation is shown in figure 3.16. Both show that the outputted image shows a clean heat map like color scale for the dimensionality of the data.

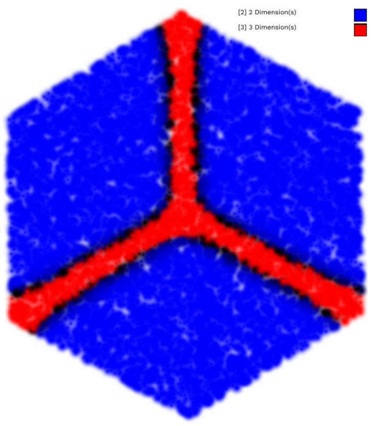


Figure 3.17: Dimensionality metric on a PCA reduced synthetic cube with 5% spatial noise. Radius is 0.1 and minimal threshold is 0.05

The next basic case we need to consider is the cube dataset we used in the attribute based explanation. Here we expect that the variance along the faces of the cube can be explained using 2 dimensions and the edges along 3. InThe results of this are shown in figure 3.17. We can clearly see the that the explanation fits our expectation, one observation to make is that the confidence in the 3 dimensional area does not drop towards the 2 dimensional areas, in the inverse case we do observe lower confidence along the edge. This is to be expected with the way the confidence function works for the minimal approach (see sec 2.3.2). We can also directly compare these images to the one in the thesis of Van Driel [15], more specificity figure 4.13 and 4.14. We can observe that method is visually similar. We now have high confidence in our explanation and believe we can start using it in real world cases, which is exactly the main topic of our next chapter.

3.6 Source Code

The tool developed and the source code for this thesis is publicly accessible via the GitLab server of Utrecht University via `https://git.science.uu.nl/vig/mscprojects/Pointctl`. Within this repository all the 8 datasets and 27 projection for each are also stored.

Chapter 4

Application to real-world data

With an understanding of the techniques and our own implementation validated it is time to explore the range of inputs for the techniques and walk through the results. We will start in section 4.1 again with simple synthetic datasets and use them to explore the parameters that can be set and reason about how to set them. Following this in section 4.2 and 4.3 will elaborate on our choice of datasets and the process of selecting the correct projection method. Finally in section 4.4 we explore each dataset and explain how our tool can be used to preform an analysis.

4.1 Parameter analysis

Our tool combines various input together to find generate the image the user eventually sees, we have the dataset, projection, explanation, viewport, shading, gamma, normalization, point size and more. In this section I would like to classify the inputs and also outline their effects. The later two

Source data: To start we have the static stored normalized dataset and projection, these can only be set when starting up the program. There is no way to compute a projection in the *online* phase of the pipeline (see sec 3.2). We will analyse various datasets selected in section 4.2 in section 4.4.

Rendering parameters: These are variables that determine the way colors are shown on screen and the size of points. Here you can think about fields like gamma and confidence encoding (i.e. how it maps to brightness) but also the camera. All of these parameters can be changed realtime via the UI.

Explanation parameters: As discussed in the preliminaries both explanation techniques are subject to inputs besides the dataset and the projection, here we have the neighborhood size used in the algorithm and for the dimensionality based explanation the thresholding method and value θ. There we explained the consequences of changing them, here we will elaborate using examples.

Dimensionality: We will can also consider the dimensionality of the projected space as a parameter, moving from 2D to 3D has large implications for how the projection is explored.

We will explore each class of parameters one by one, freezing all the parameters from the other classes in the process. If we were to consider the combination of all possible values we would have to discuss thousands if not millions of combinations (the problem suffers from the curse of dimensionality).

4.1.1 Rendering parameters

Like discussed in the previous section these parameters control the shader pipeline, table 4.1 provides an overview of the inputs to this pipeline which are determined by the user. In this section we will elaborate on each.

Parameter	**Scale**	**Range**	**Interaction**	**Occurs when**
Gamma	Real	$[0, 1]$	Slider	Always
Point size	Integer	$[1, 16]$	Slider	Using discrete rendering
Splat size	Real	$\left[\frac{1}{4}D, 4D\right]$ [1]	Slider	Using continuous rendering
Camera position	Real matrix	$()$	Mouse	Always
Shading intensity	Real	$[0, 1]$	Slider	In 3D when shading has been computed
Confidence normalization	Real tuple	$[0, 1]$	Double slider	When explanation is being displayed
Color map override	Ordinal map		Drop down	When explanation is being displayed
Render mode	Binary	$\{0, 1\}$	Button	Always

Table 4.1: Overview of rendering parameters

Brightness correction

Fist we have gamma correction, it is a commonly used non linear transformation used for encoding and decoding the luminance of colors. Given an sRGB color space where each value is between 0 and 1 this correction works by raising the value of each channel to a power γ. The formulas used in the shader can be found in equation 4.1. Note that here light colors (i.e. low values) change very little with this operation, the reasoning here is that we humans perceive more detail in the light areas.

$$(R_{out}, G_{out}, B_{out}) = (R_{in}^{\gamma}, G_{in}^{\gamma}, B_{in}^{\gamma}) \tag{4.1}$$

A value $\gamma < 1$ is often referred to as an encoding gamma value, conversely $\gamma > 1$ is used for decoding. Our tool allows the user to adjust this γ using a slider and can set a value between 1.0 and 3.4, the effect of this on the colors of an image is displayed in figure 4.1.

[1]Here D denotes the average first nearest neighborhood distance

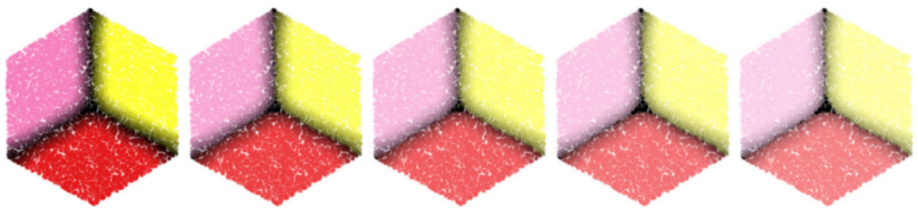


Figure 4.1: Gamma parameter setting, left to right $1.0, 1.6, 2.2, 2.8$ and 3.4.

This allows our users to choose which part of the color space they want to focus on. Since every explanation can yield varying ranges of confidence values we need to give our user control so they can find where *they* perceive the nuances in regions of interest best. It is very important to note that this operation is not linear, the power-law comes into play. If one channel shifts more than another the hue of a color will shift with it. Perceptually this effect is not very strong but it become more apparent with high values of γ, figure 4.1 captures this well.

To aid the user towards this goal even further we also provide a way to adjust the normalization of the confidence encoding. Since we are normalizing using the *global* minimal and maximal confidence outliers can form trouble. A common case is points with no neighbors within a radius that we need to explain, here we fall back to a confidence of 0. If all other points have confidence values ≥ 0.8 all nuance in this top end is lost. The user can then shift the lower bound to *focus* on the confidence range of interest and use the full brightness range to show it. The effects of these bounds and the aforementioned principle is illustrated in figure 4.2.

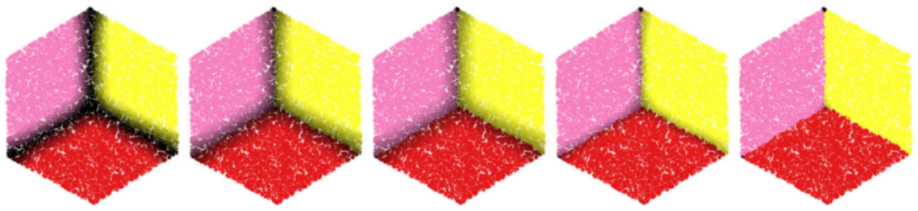


Figure 4.2: Differing normalization bound settings. Left to right the bounds are $(50\%, 100\%), (25\%, 100\%), (0\%, 100\%), (0\%, 75\%)$ and $(0\%, 50\%)$.

Point and splat size

The user can also configure the *size* of a points in the discrete rendering mode and splats in the continuous mode. While the concept is very self explanatory why we implement it is not, the main reason is the differing sparsity between projections and regions within a projection. This allows the user to choose the optimal value when inspecting a region of interest. It can be set using a slider and changes in real time. For the discrete representation the point size variable in openGL is used, the effects are shown in figure 4.3.

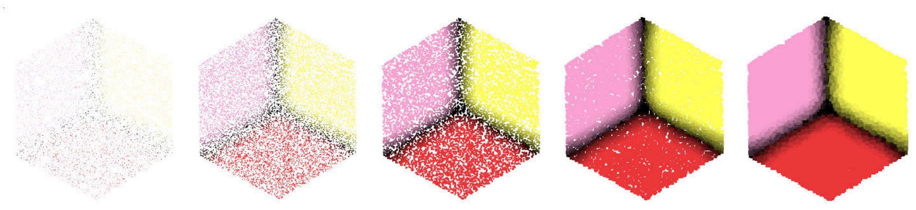


Figure 4.3: Point size for discrete rendering, left to right size is $1, 3, 6, 10$ and 16.

For the continuous rendering method this concept changes a little, the splat is rendered using world coordinates. For this we use the average first nearest neighbor distance as a default value and make the setting relative to this, figure 4.4 shows the effects. One known short coming of this technique is that some projections have overlapping points as a result, this can yield a default so small nothing appear to be rendered. It is a rare case but to work around this users will need to switch to the discrete method.

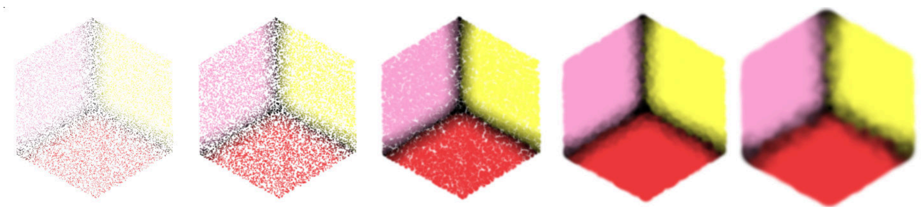


Figure 4.4: Splat size for continuous rendering, left to right size is $d\frac{1}{4}, d\frac{1}{2}, d, d2$ and $d4$ where d is default size based on average first nearest neighbor.

Shading intensity

Finally for 3 dimensional projects we have can tune the intensity of the shading. The idea behind this is explained in section 3.4.2, however there we do not consider the problem that we are encoding both point orientation and confidence into brightness. Using this parameter you can turn of the shading and also shift which proportion of the brightness encodes orientation. In figure 4.5 we use a sphere to show increasing intensity of shading.

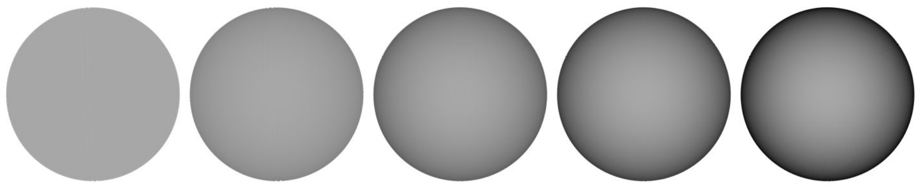


Figure 4.5: hading intensity for a sphere, left to right $0\%, 25\%, 50\%, 75\%$ and 100%.

4.1.2 Explanation parameters

Besides the rendering setting we can also change the inputs that go into our explanations techniques, while there are fewer parameters these are arguably more important to fully understand. For our metrics we have 2 key input variables, the neighborhood size and the thresholding method / value for dimensionality based explanation.

Neighborhood size

For our tooling we have defined two different types of neighborhood, a radius r based variant where the neighborhood vary in neighbor count by have a constant max distance and a count k variant here the inverse is true. We found that the radius version is easier to reason about since the regions it concerns are consistent, the variably density from area to area within a projection make the k variant complicated. One clear upside of this variant is the compute required, we know the exact upper bound on neighborhood size. Here an absolute radius can cause *very* large neighborhood sizes making the explanations difficult to compute. In a pervious section we used figure 3.13. Here in figure 4.6 we show what happens when very large radii are used, here the metrics failed to be useful and at 100% every point becomes each others neighbor and we get a perfect single attribute explanation.

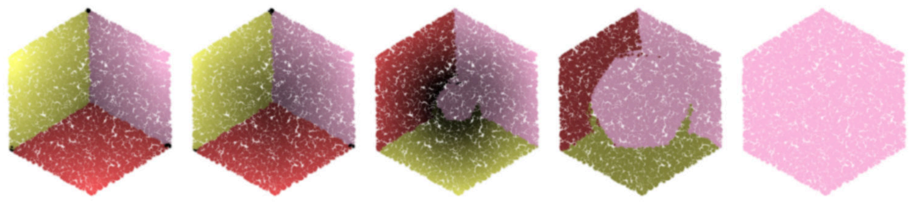


Figure 4.6: Variance metric on a PCA reduced synthetic cube with radius $r = 40\%, 50\%, 60\%, 75\%$ and 100%

Using the same dataset we can also explore the effects of differing theta values, sadly since the intrinsic dimensionality throughout this dataset is rather low the nuances are lacking. Figure 4.7 shows the differing θ values when using a minimal treshold and figure 4.8 does the same of the total method. Note that the color map changes between the images, it captures the dimensions currently being displayed and this range changes with the differing θ values. The explanation behind the metrics can be found in section 2.3.2.

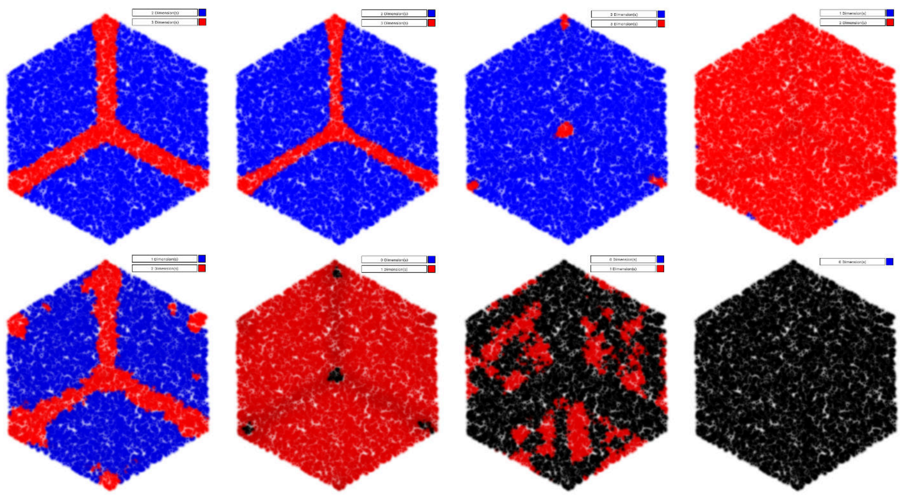


Figure 4.7: Dimensionality metric on a PCA reduced synthetic cube with minimal tresholding and $\theta = 2.5\%, 5\%, 10\%, 20\%, 30\%, 50\%, 75\%$ and 100%

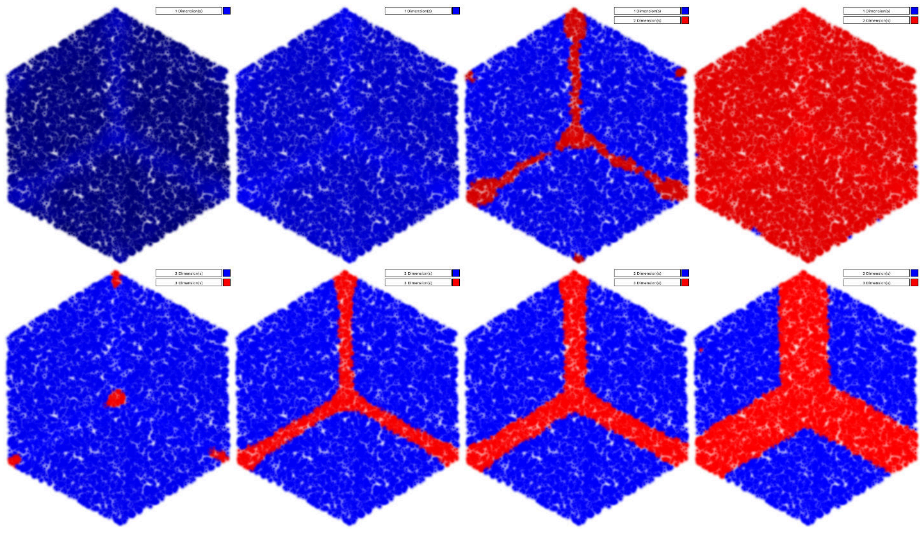


Figure 4.8: Dimensionality metric on a PCA reduced synthetic cube with total tresholding and $\theta = 0\%, 40\%, 60\%, 80\%, 90\%, 95\%, 97.5\%$ and 100%

4.2 Selected real-world datasets

We have selected 8 datasets with differing characteristics to compare during our comparison between 2D and 3D explanations. Table 4.2 lists their details, including their sparsity ratio $\gamma_n = 1 - \frac{u}{nN}$, $\gamma_n \in [0, 1]$, where u is the number of non-zero data values; and intrinsic dimensionality $\rho_n \in [0, 1]$, defined as the percentage of principal components (of the total n), computed by PCA, needed to explain 95% of the data variance. These values show that our selected datasets cover quite different characteristics, in line with those selected in the 2D benchmark. Using more datasets is definitely desirable. However, this would be too expensive, given that we aim next to project each of them by several techniques, both in 2D and 3D, and compute several quality metrics for each combination.

Dataset	**Type**	**Samples N**	**Dimensions n**	**Intrinsic dimensionality ρ_n**	**Sparsity γ_n**
Air quality [72]	tables	9357	13	5	0.1372
Breast cancer,[73]	tables	569	30	10	0.0059
City pollution,[74]	tables	32681	10	6	0.0052
Concrete [75]	tables	1030	8	6	0.1773
DefaultCC [76]	tables	30000	23	8	0.1070
Reuters,[77]	text	8432	1000	696	0.9488
Software,[78]	tables	6773	12	7	0.0818
Wine [69]	tables	6497	11	8	0.0023

Table 4.2: Selected datasets and their trait values.

The low dimensionality across the first datasets was chosen on purpose, we argue that if our explanation mechanisms cannot explain data with few dimension well it will have also be ill equipped to handle data with high (intrinsic) dimensionality. We did include the Reuters dataset as a control for this statement. One characteristic the first 7 datasets have in common is that all the dimensions are named, this makes interpreting the attribute explanation easier, this helps reasoning about the produced images.

4.3 Selection of projection and dimensionality

To analyse datasets it is also crucial to decide which projections and its dimensionality you decide to use. The tool we developed has already been deployed in a research paper which provides a quantitative and qualitative comparison of 2D and 3D projection techniques. There we compare the quality of all the projection techniques from table 2.1 for the datasets from table 4.2 using trustworthiness, continuous and spearman correlation as defined in section 2.2. Additionally we also consider the quantitative approach and used domain experts to mark which projections worked well which do not. It is important to note that this is still a very limited subset the datasets available, we cannot give a definitive answer to which you projection and dimensionality you should use when exploring an arbitrary dataset. All we can do is provide this comparison and highlight the considerations we make, it is up to researcher to make their own considerations with other datasets.

Figure 4.9 shows the results per quality metrics for all the projections in 2D and 3D summarized over all datasets, the techniques are sorted ascending on trustworthiness. It is important to note that some datasets are results are missing, here the quality metrics was not 0 but instead the projection technique failed to execute here. The figure illustrates that except one outlier in the form of N-MDS the measured quality does not vary much over the techniques. Another observation is that the gain from adding an additional dimension seems marginal at best, the average increase in quality for trustworthiness is 5%, with

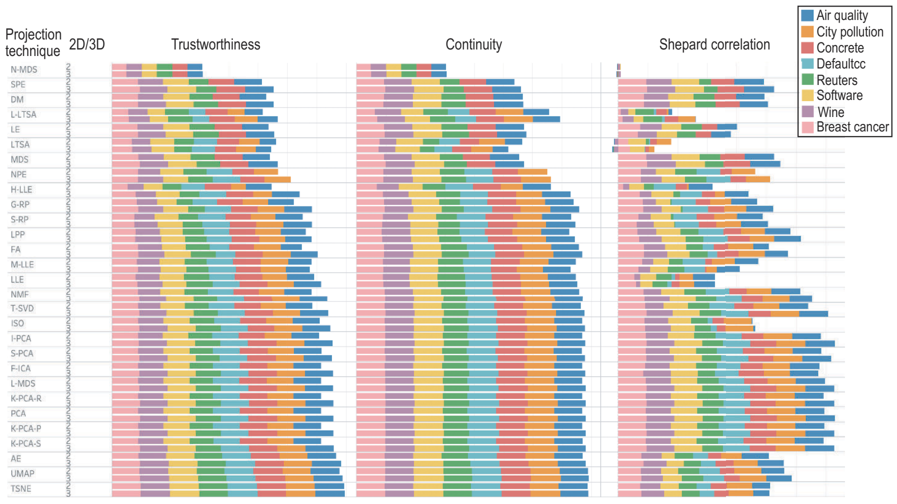


Figure 4.9: Quality metrics per projection technique (rows), dataset (colors), and projection dimension (2D *vs* 3D, second column), sorted ascendingly on technique trustworthiness.

continuity it is only 2% and finally for the sheppard correlation we get 3% but with a much higher variance. Based on these measurements the gain does not seem worthwhile when the additional complications of 3D are taken into account. What we can do here is discard a series of projection techniques from even being considered, if the trustworthiness is low the assumption, neighborhoods in D^P are the same as in D, which is the base of our localized explanation technique does not hold. This means the resulting visualization also can not be trusted.

The quantitative experiment is best summarized by figure 4.10 attempt to capture the perceived quality difference. We manually merged the color mapping so the attributes retain the same color when moving from 2D to 3D, this makes observing similar structures a lot easier. Here again we can observe that the that the quality across 2D and 3D are closely correlated. In this image we have included the projections that have clear structure visible above the horizontal line and conversely the ones without below. We find that projections that perform poorly in 2D dimension benefit little from a 3D dimension, for example N-MDS, it has issues with a region of point surrounding all others. With the extra dimension the projection explanation is still subject to this problem, only now occlusion hinders the user more. On the contrary projections like FA, MDS or T-SNE the extra projection allows it to show more nuance in the borders between regions and clusters. While regions might be interleaved in 2D they can be separated in 3D. In short we can see that the measured correlation also holds for the perceived quality. For a more in-depth analysis of this topic can be found in the source work [79] together with additional considerations on the topic of useability. For our research here we will use the dimensionality which best suites the explanation and only consider exploring using the projections techniques in the top 10 of figure 4.9.

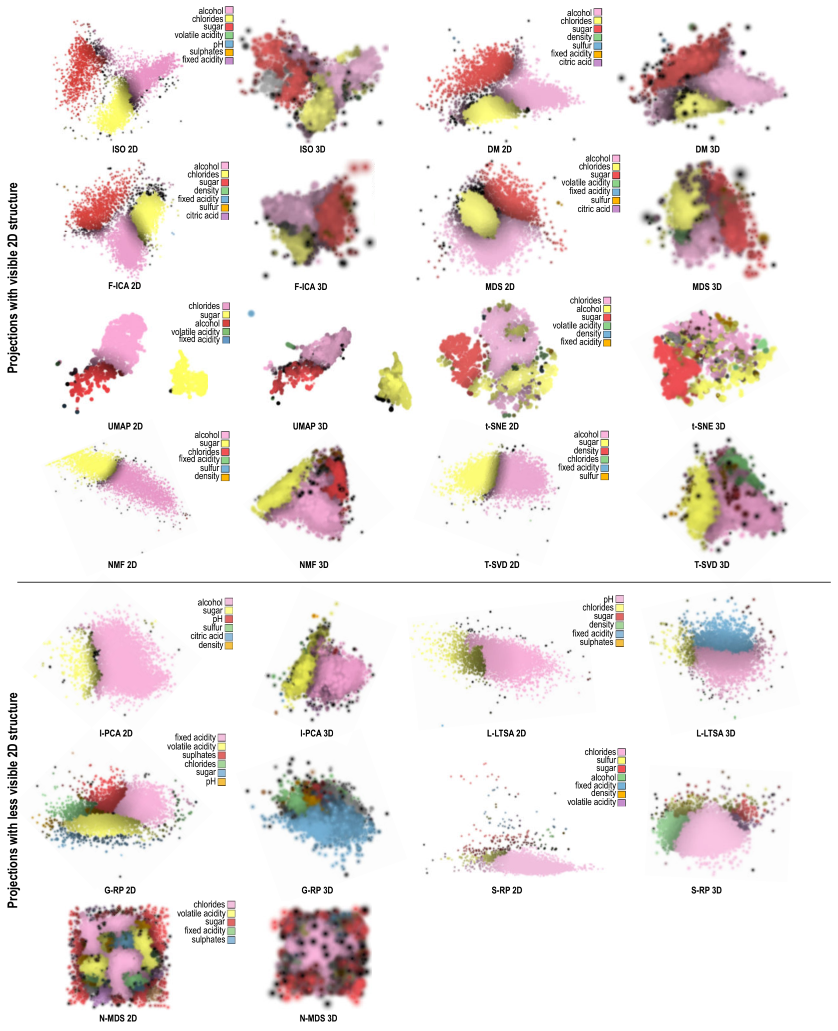


Figure 4.10: Comparison of 2D and 3D projections of the Wine dataset explained by dimension variance.

4.4 Dataset analysis

In this section we will go through the 8 selected datasets and for each show how our tool can be used to find insights. We will start with an in-depth look at datasets which are well known to us and then move on the lesser known ones.

4.4.1 Wine quality

The *vinho verde* wine quality dataset [69], this consists of 12 measurements for 6497 wines. This dataset includes a LAMP [80] based projection that was also used in the metrics paper, with this we can now compare using a real dataset but with an identical projection. We will start by recreating this view and move on with our observation from our own extensions from there. The projection produces a single clump of points with varying density, this variance can be explained by the fact that the this data set contains 4898 red and 1599 white whines. In the reduction these 2 sets are projected in 2 barely overlapping clusters as shown in figure 4.11.

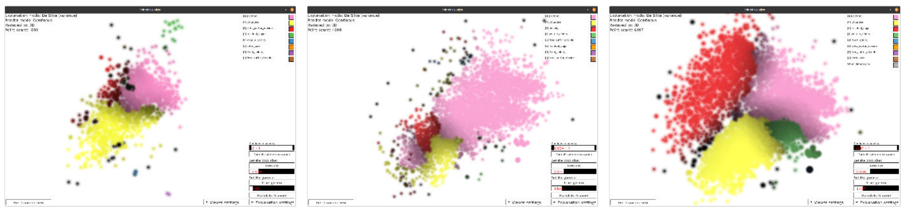


Figure 4.11: Attribute based variance metric using for the wine quality dataset [69]. Left only the white wines, middle only red wines and right all wines ($r = 10\%$)

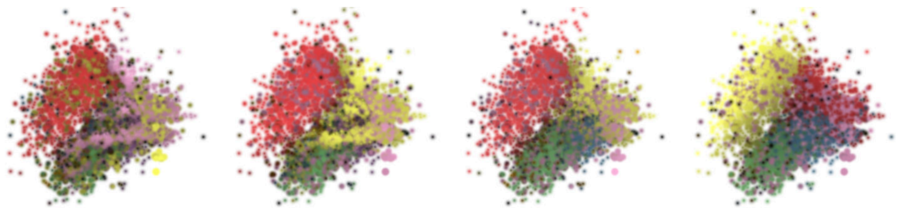


Figure 4.12: Da Silva's euclidean metric on the a LAMP projection of the Wine quality dataset [69]. Neighborhood is radius based with 10% and 20% of the projection width and then KNN based using $k = 75$ and $k = 250$

Lets first look at the euclidean in figure 4.12, it produces very noise results regardless of what radius is used. Here absolute radius and neighbor count based neighborhoods where used, the reasoning behind the KNN approach is that the absolute size can vary to handle the varying density. Sadly both radii types produced poor results, this confirms that this metrics is more noisy. The source papers briefly mentions this as well but never elaborate, in the rest of the paper they only use the variance based explanation, we will do the same for our further analysis.

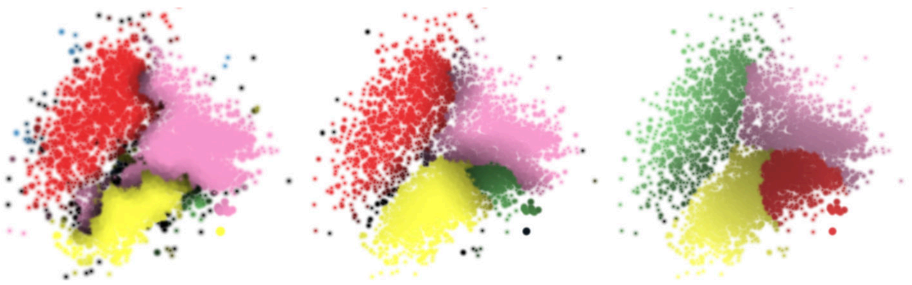


Figure 4.13: Variance metric on the a LAMP projection of the wine quality dataset [69]. Neighborhood is radius based with 5%, 10% and 20% of the projection width

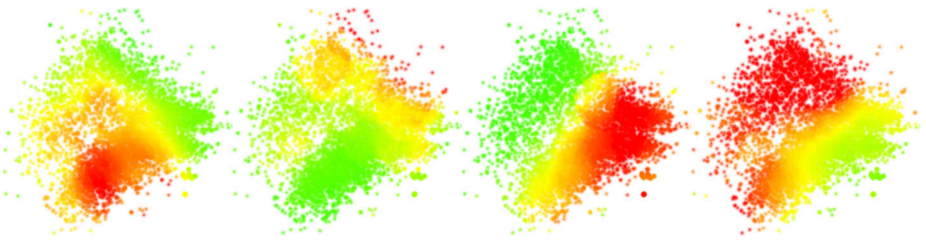


Figure 4.14: Variance metric per attribute, low contribution is lower variance. In reading order we have alcohol, chlorides, residual sugar and volatile acidity

Using the variance based metric for the wine quality dataset did prove useful, producing the results in figure 4.13. Here the middle images uses the exact same parameters as shown in figure 2 of the source paper for the attribute based explanation [1]. The exact same regions can be observed, top left we find the *Residual sugar*, right we find *Alcohol* and at the bottom *Sodium chloride*. Between each region there is a lower confidence transition area, this indicates that the area can not be explained by a single dimension. Between the *Alcohol* and *Sodium chloride* regions we also find a small region where the variance is best explained by *Volatile acidity*. Our tool extends this with the single attribute explanations view displayed in figure 4.14. Here we again find the distinct green areas where the attribute explains the dataset to the best of it's ability. We can also notice some overlap between the green areas, here the area is well explained by *both* attributes. Note that the green is also subject to normalization, so for sugar the green area denotes contribution lower that 0.001 while for the alcohol it means lower than 0.005. This explain why the full image uses the *sugar* explanation for this overlapping region.

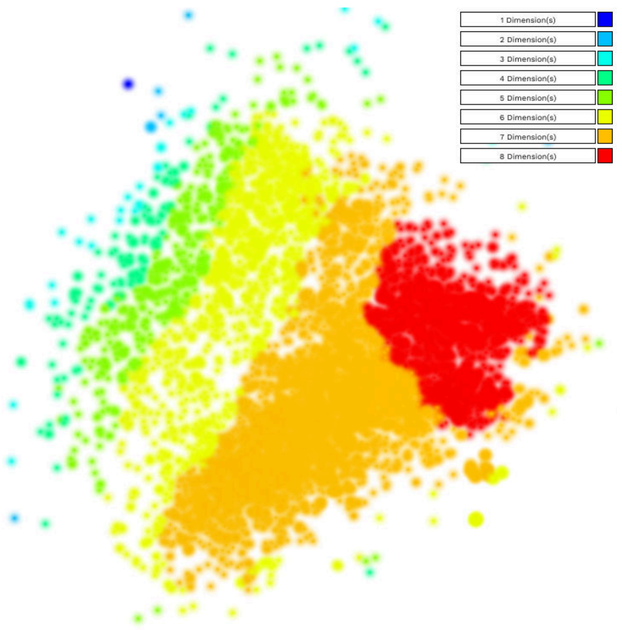


Figure 4.15: Dimensionality total metric on LAMP projection reduced Wine datasets [69]. Radius is 0.1 and threshold $\theta = 0.9$

Finally we can utilize the dimensionality view to highlight the localized intrinsic dimensionality as shown in figure 4.15, here we utilize the total threshold of 90%, so by running PCA how many of the top n principal components are required to capture 90% of the variance. Here we can say that the left of the image only 1 or 2 such components are needed, while on the right there is a certain hot spot where up to 8 components exceed this threshold. This knowledge can tell us that even though *Residual sugar* has the least variance in the left area there are probably more attributes that vary very little in that region, the variance is limited to 1 or 2 components. Conversely the highly dimensional area likely has few other attributes that vary little.

4.4.2 Software attractiveness

The software quality dataset aims to link metrics between code metrics over 6773 free software projects from the SourceForge.net repository and the attractiveness towards contributors [78]. For this dataset we will again use the T-SNE projection in 2 dimensions.

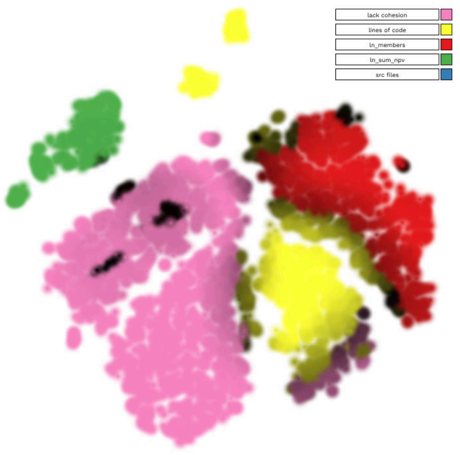


Figure 4.16: Variance based metric on TSNE projection reduced Software datasets [78]. Radius is 0.1.

Unlike the wine quality here in figure 4.16 we observe more distinct clustering for this dataset, we can see 3 larger clusters and a series of smaller ones above. Interesting to note is that our method is able to give high confidence explanations for each. Lack of cohesion explains the largest regions in pink, then we have the number of methods in red and between the lines of code in yellow.

With the knowledge of which attributes vary least in the clusters we can then use the tooltip to find the actual values. An interesting observation is that by a rough estimate most points share the *exact* same value for lack of cohesion, namely 0.2501. For the green region we can make a similar observation, these code bases all have exactly zero public attributes / values (*ln_sum_npv*).

4.4.3 Air Quality

The Air quality dataset contains 9358 instances of hourly averaged response form an set of 5 metal oxide chemical sensors. These sensors are placed by roadsides and in Italian villages.

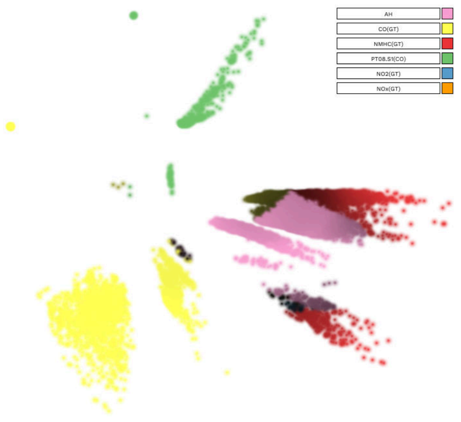


Figure 4.17: Attribute variance metric on 2D AE projection air quality dataset. Radius is 0.1.

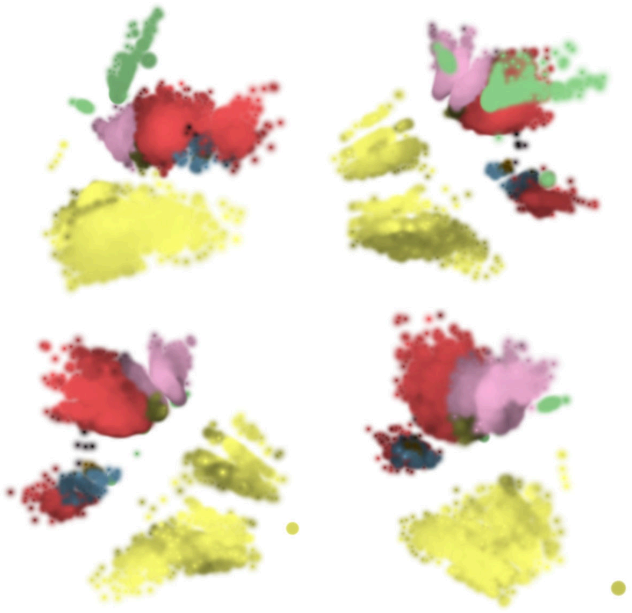
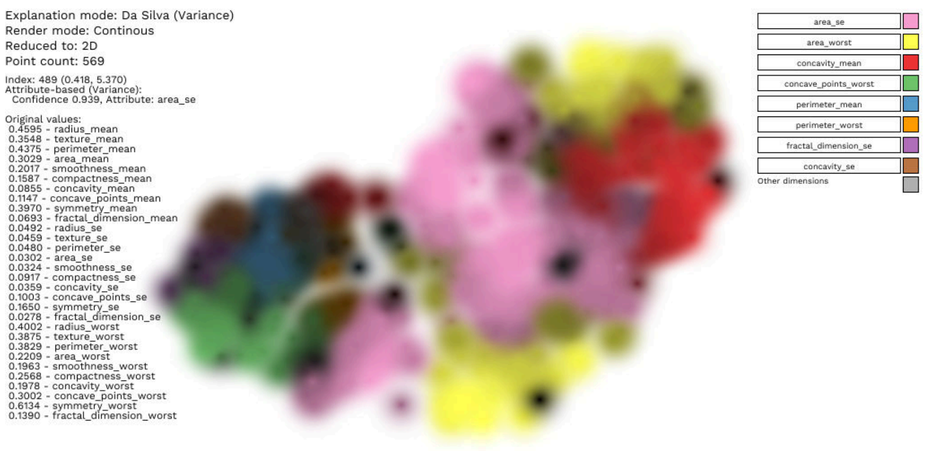


Figure 4.18: Attribute variance metric on 3D AE projection air quality dataset, same legend as figure 4.17. Radius is 0.1.

This time we use the auto encoded projection with the same explanation method for both 2D and 3D, I argue here that the 3D version is easier to interpret as the overlapping clusters are now just neighbors when using the right viewport. A good example is the pink and red clusters, in 2D these two have regions that appear to intersect while in 3D there is a clearer separation between the 2. Besides this overlap the other clusters are spaced far apart and can then, with high confidence, be explained by a single attribute. Here the researcher could progress by focusing on the clusters that have multiple attribute with lower variance.

4.4.4 Wisconsin Breast Cancer

The wisconsin dataset is an interesting addition here as it a relatively small dataset with only 569 entries and a total of 30 attributes [73]. These 30 features are extracted from digitalized images of a fine needle aspirate of a breast mass. Here the target variable is binary, the tumor is either malignant or benign.



```
Explanation mode: Da Silva (Variance)
Render mode: Continous
Reduced to: 2D
Point count: 569

Index: 489 (0.418, 5.370)
Attribute-based (Variance):
  Confidence 0.939, Attribute: area_se

Original values:
 0.4595 - radius_mean
 0.3548 - texture_mean
 0.4375 - perimeter_mean
 0.3029 - area_mean
 0.2017 - smoothness_mean
 0.1587 - compactness_mean
 0.0855 - concavity_mean
 0.1147 - concave_points_mean
 0.3970 - symmetry_mean
 0.0693 - fractal_dimension_mean
 0.0492 - radius_se
 0.0459 - texture_se
 0.0480 - perimeter_se
 0.0302 - area_se
 0.0324 - smoothness_se
 0.0917 - compactness_se
 0.0359 - concavity_se
 0.1003 - concave_points_se
 0.1650 - symmetry_se
 0.0278 - fractal_dimension_se
 0.4002 - radius_worst
 0.3875 - texture_worst
 0.3829 - perimeter_worst
 0.2209 - area_worst
 0.1963 - smoothness_worst
 0.2568 - compactness_worst
 0.1978 - concavity_worst
 0.3002 - concave_points_worst
 0.6134 - symmetry_worst
 0.1390 - fractal_dimension_worst
```

Figure 4.19: Dimensionality total metric on T-SNE projection reduced Wisconsin Breast cancer dataset [73]. Radius is 0.1.

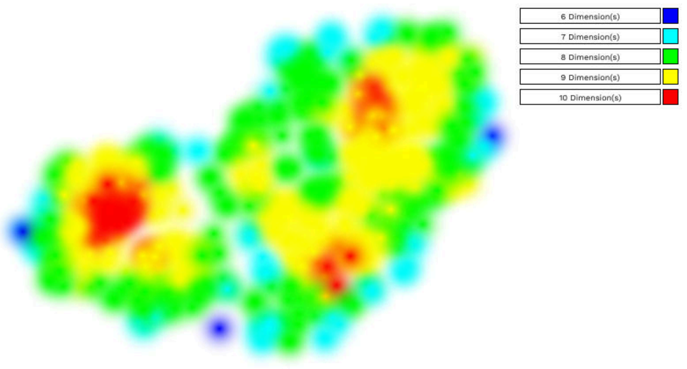


Figure 4.20: Dimensionality total metric on T-SNE projection reduced Wisconsin Breast cancer dataset [73]. Radius is 0.1 and threshold $\theta = 0.9$

We can quickly observe that the insights from the variance based explanation in figure 4.19 are lacking, we get a single structure but no clear regions in that cluster. We argue that here there are to numerous attributes that vary little across the entire dataset, because of this no single explanation prevails. The dimensionality explanation in figure 4.20 does give us some more interesting insights, we need at most 10 dimensions to explain 90% of the variance in the center of the blob but at the edges only 7 dimensions are sufficient.

4.4.5 City pollution

City pollution is a dataset that described the pollution levels in cities over 10 attributes with over 32681 samples. For this dataset we choose to explore the 3D projection as there are more ways the regions can neighbor each other.

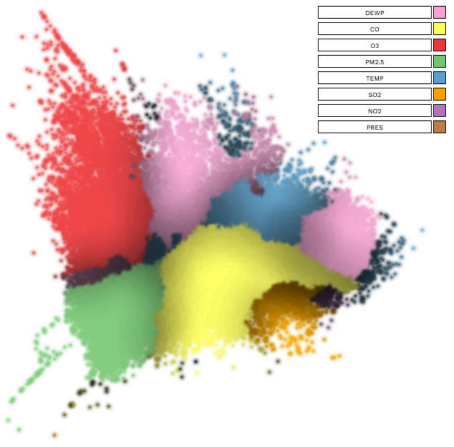


Figure 4.21: Attribute variance metric on 2D ISO projection reduced City pollution dataset. Radius is 0.1.

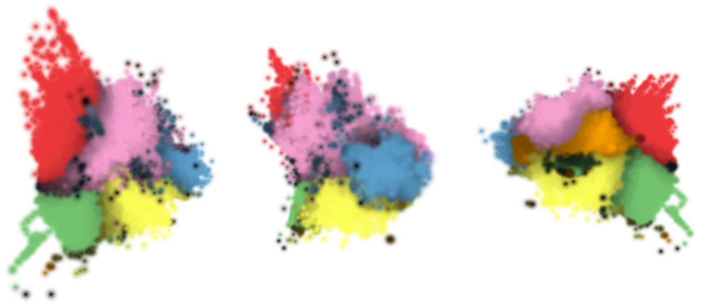


Figure 4.22: Attribute variance metric on 3D ISO projection reduced City pollution dataset, same legend as figure 4.21. Radius is 0.1.

Like the other datasets without any explanations we simply get a grey blob which conveys very little information, the 2D explanation shown in figure 4.21 shows a blue region for Temperature to to be enclosed by 2 pink regions marked by *DEWP*. With the extra degree of freedom provided by 3D we find out that these two separate clusters are actually one, figure 4.22 shows a single pink regions next to the blue one. This is a clear example of where 2D projections might fall short, here the researcher could continue their exploration of the dataset considering that there is one nD cluster that can be explained well by Temperature.

4.4.6 Defaulting Credit Cards

The default credit card dataset is aimed at determining which clients are and are not credible, it does this based on a series of measurements and boolean attributes. Interesting to note for the projections and our explanations here is the strong effect of these binary variables.

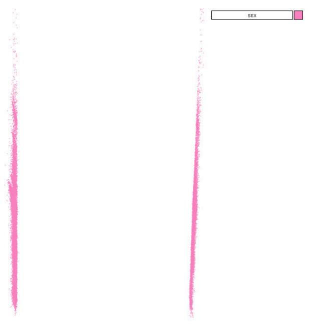


Figure 4.23: Attribute variance metric on 2D K-PCA-P projection of the Default cc dataset. Radius is 0.1.

Let us first look at how kernel PCA and most other projections handle this dataset, shown in figure 4.23 is the kernel PCA projection. As we can note here the projection produces 2 line like clusters which are *perfectly* explained by the sex of the subject. This is of course an interesting insight, the dataset appears to have 2 clusters that are clearly separate by this binary variable, but that is the only conclusion we can get from this.

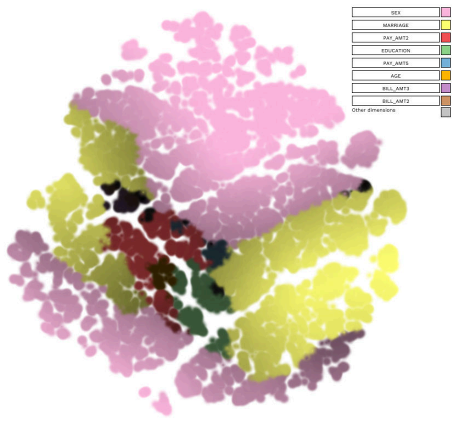


Figure 4.24: Attribute variance metric on 2D T-SNE projection of the Default cc dataset. Radius is 0.2.

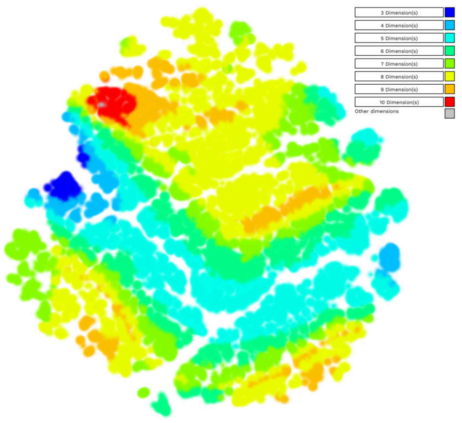


Figure 4.25: Attribute variance metric on 2D T-SNE projection of the Default cc dataset. Radius is 0.2, theta $\theta = 0.8$

To investigate further we will look at T-SNE as it does not suffer from the same problem, here we have the attribute based explanation in figure 4.24 and dimensionality explanation in figure 4.25. Instead of two lines we get a series of 3 clusters that are close other. Again sex is the most prominent explanation for followed by married (another binary variable). Interesting here is that the clusters are still divided by these binary attributes, the top cluster from the bottom two is determined by sex and the bottom two by marriage. The reasoning that the binary attributes are an amazing explanation since the variance will 0 zero per cluster still stands. Using the dimensionality based explanation we can also see that these areas have a higher intrinsic dimensionality as well, but it also shows the area surrounding the border of the clusters has low dimensionality.

With a focus on this region we can also see that *PAY_ATM2* and *education* are also factors that explain areas around this border well. An intresting lead for researchers, another option worth exploring is leaving out the binary values al together before projecting and seeing if you can get a more general explanation. We did not explore this option however.

4.4.7 Reuters

Contrary to the other datasets Reuters [77] has a high amount of attributes, namely because it is a *bag of words* representation of news articles where the top 1000 most used words are kept. For each of the 8432 articles the row contains the occurrences of words within it. We include this dataset because even though our tool was not build for these high dimensional use cases we would like to show an example of why.

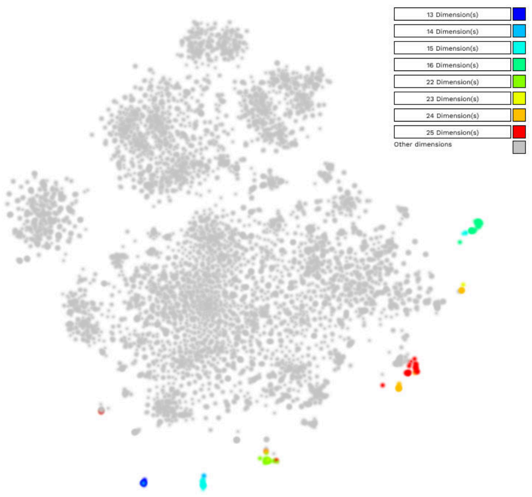


Figure 4.26: Attribute variance metric on 2D T-SNE projection reduced Reuters dataset. Radius is 0.1.

Figure 4.26 shows the results of the dimensionality based explanation, from the wisconsin breast cancer dataset we already know that using the attribute based explanation for high dimensionality does not work well. The dimensionality explanation here is also very lacking, our tool simply has a color map which it to limited to get any meaningfully insight at a glance. Besides this fact it is also extremely hard to compute localized metrics here since they are based PCA. With a 1000 dimensions we will have a 1000x1000 covariance matrix for each point that you need to preform PCA for, for this dataset our tool hardly computes metrics in realtime. It took about 4 minutes to compute the annotations. For the Reuters dataset our explanation add very little to classic global explanations.

Chapter 5

Discussion and conclusion

Chapter that closes of the thesis, here we will reflect on our research process, shortcomings and outline the concrete results.

5.1 Discussion

Within the research area of data visualization one inherent problem is complications surrounding the subjectivity of the matter. We can argue that a user might need certain configurable aspects and use sane defaults in other places but all of these are all subjective to the person who need to interpret the image. One might prefer darker shades with discrete points and another vise verse with light shades and continuous points. All we can do is argue for and against and provide the comparisons that a user can use to form their own view on the matter.

This is theme that is most prevalent in chapter 4, here we provide examples of how to use our tool in the form of case studies. While these are more extensive that what has been done in the previous work it is by no means a complete view. We make claims about generality but extrapolate this from testing 8 datasets with 29 projection methods. Besides this all the interpretation are performed by domain experts, especially in the case where we perform direct comparisons this problem shows. Here a user study for the exploration process would be in order and strengthen our claims that these insights are attainable for data analysts.

Another very fundamental problem with the explanation techniques is the fact that they assume the neighborhoods in projected space matches the one in the ND. In other words, the projected technique is perfect. In reality this assumption never hold. Conversely if we were to use the ND neighborhoods the image would be hard to interpret. Using varying neighborhoods would break the technique entirely. We could color encode the neighborhood hit rate between these spaces but our tool currently has no support for this, a shortcoming of the current tool.

The final shortcoming I would like to outline is how our tool handles the color encoding, in 3D we currently encode the confidence of the explanation and normal. When looking at still images there is no way to differentiate between the two and can lead to hard to interpret images. In hindsight we should have explored encoding the confidence into for instance the saturation.s

5.1.1 User evaluation process

One goal that we initially had for ourselves was to perform a user study based on our implementation so we can argue if this type of analysis is valuable for data analysts when compared to classical techniques. Sadly we did not have time to formally set up a full user study due to too many open questions on our side and time constraints. We did have a discussion about two different users studies that would be beneficial for this research.

First of we have a quantitative comparison between our technique and classic methods. Here the key goal would be to show our tool can aid analysts find areas of interest with regards to variance fast. To tests this we planned to setup a task based study that provides researchers with a set of tasks for the dataset and different sets of tools. The largest problem we encountered here is the potential bias included in tasks, tasks needs to be constructed from an analysts perspective and as such can possibly be solved by both new and classical techniques. This is complicated because our explanation might provide questions and answers about the dataset that have not been explored using classical methods and vice versa. Some tasks might be steer to what is possible with our work but very difficult with existing methods. We argue that in the case of users study with bias it would be more valuable to find *why* the possibility of bias exists instead of executing a time consuming study only to find expected results.

Another angle to the user study we can take is using the same explanation techniques but using a qualitative comparison between 2 and 3 dimensional projections. This is a more well defined subsection of our problem space and has fewer parameters. Here the core idea is to again use a task based evaluation over a series of datasets and then use differing dimensions. Sadly we had no time to explore this option but we did do an expert study for a separate paper [79].

5.2 Conclusion

For our work we build tooling around the mostly theoretical local explanation proposed by Da Silva [1] and extended by Van Driel [15] were non existed. While these works only consider the 2 dimensional cases, we included support for explaining 3D projections as well. We also provide a direct comparison between 2D and 3D. We thoroughly validated the *correctness* of the produced explanations at *interactive* speeds and then utilized this in real world case studies. We also make this code publicly *accessible* to everybody with complete instruction so everybody can use it.

We consider **8** datasets each projected using **29** techniques to both 2D and 3D, using this large collection of data we explored two main questions. Is there an advantage to using 3D over 2D and are how do explanation perform when facing real datasets. The first is explored on a superficial level, we find that 3D does provide a small advantage if and only if the 2D projection is sound while coming at the cost of a small interaction effort. The code opened the opportunity for more research spearheaded by A. Telea in the form of an article [79] which is still under submission. In chapter 4 we answer the latter question, here we explore how the explanation works given all the varying inputs. Here the effects of all parameters are outlined and using this knowledge we provide a case study for for 8 datasets to show how the tool and explanations can be utilized. We show that it meets the aim of finding new novel insights into actual data!

5.2.1 Future work

From a practical point of view the tool could be extended in various way, most important is that the explanations are easily accessible for other researchers. One could of instance produce a stand alone small program that produces only the explanations so future users could implement into their pipeline.

A more theoretical take would be the exploration of more types of localized explanations, we only explore two general metrics that use a points neighborhood but more might exists. I also argue that preforming a formal users study into 2D versus 3D using explanations might generate more interest into the sparsely explored topic of 3D projections. The last points from the discussion also come to mind, ensuring the luminance is not encoded by 2 values and visualizing local quality metrics can elevate the quality of the tool even further.

Bibliography

[1] R. R. da Silva, P. E. Rauber, R. M. Martins, R. Minghim, and A. C. Telea, "Attribute-based visual explanation of multidimensional projections", in *EuroVA@ EuroVis*, 2015, pp. 31–35.

[2] D. van Driel, X. Zhai, Z. Tian, and A. Telea, "Enhanced attribute-based explanations of multidimensional projections", 2020.

[3] A. Inselberg and B. Dimsdale, "Parallel coordinates: A tool for visualizing multidimensional geometry", in *Proc. IEEE VIS*, 1990, pp. 361–378.

[4] R. Rao and S. K. Card, "The table lens: Merging graphical and symbolic representations in an interactive focus+context visualization for tabular information", in *Proc. ACM SIGCHI*, 1994, pp. 318–322.

[5] A. C. Telea, "Combining extended table lens and treemap techniques for visualizing tabular data", in *Proc. EuroVis*, 2006, pp. 120–127.

[6] R. Becker, W. Cleveland, and M. Shyu, "The visual design and control of trellis display", *Journal of Computational and Graphical Statistics*, vol. 5, no. 2, pp. 123–155, 1996.

[7] L. Nonato and M. Aupetit, "Multidimensional projection for visual analytics: Linking techniques with distortions, tasks, and layout enrichment", *IEEE TVCG*, 2018. DOI: 10.1109/TVCG.2018.2846735.

[8] C. Sorzano, J. Vargas, and A. Pascual-Montano, *A survey of dimensionality reduction techniques*, arXiv:1403.2877 [stat.ML], 2014.

[9] L. van der Maaten and E. Postma, "Dimensionality reduction: A comparative review", Tilburg University, Netherlands, Tech. Rep., 2009, Tech. report TiCC TR 2009-005.

[10] M. Espadoto, R. Martins, A. Kerren, N. Hirata, and A. Telea, "Toward a quantitative survey of dimension reduction techniques", *IEEE TVCG*, vol. 27, no. 3, pp. 2153–2173,

[11] J. C. Gower and D. J. Hand, *Biplots*. CRC Press, 1995, vol. 54.

[12] M. J. Greenacre, *Biplots in practice*. Fundacion BBVA, 2010.

[13] D. B. Coimbra, R. M. Martins, T. T. Neves, A. C. Telea, and F. V. Paulovich, "Explaining three-dimensional dimensionality reduction plots", *Information Visualization*, vol. 15, no. 2, pp. 154–172, 2016.

[14] B. Broeksema, T. Baudel, and A. Telea, "Visual analysis of multidimensional categorical datasets", *Computer Graphics Forum*, vol. 32, no. 8, pp. 158–169, 2013.

[15] D. V. van Driel, *Automatic visualized explanations of multidimensional projections*, 2017.

[16] G. E. Hinton and R. R. Salakhutdinov, “Reducing the dimensionality of data with neural networks”, *Science*, vol. 313, no. 5786, pp. 504–507, 2006.

[17] R. R. Coifman and S. Lafon, “Diffusion maps”, *Applied and computational harmonic analysis*, vol. 21, no. 1, pp. 5–30, 2006.

[18] I. T. Jolliffe, “Principal component analysis and factor analysis”, in *Principal Component Analysis*, Springer, 1986, pp. 115–128.

[19] A. Hyvarinen, “Fast ICA for noisy data using Gaussian moments”, in *Proc. IEEE ISCAS*, vol. 5, 1999, pp. 57–61.

[20] S. Dasgupta, “Experiments with random projection”, in *Proc. UAI*, Morgan Kaufmann, 2000, pp. 143–151.

[21] D. L. Donoho and C. Grimes, “Hessian eigenmaps: Locally linear embedding techniques for high-dimensional data”, *Proc. of the National Academy of Sciences*, vol. 100, no. 10, pp. 5591–5596, 2003.

[22] D. A. Ross, J. Lim, R.-S. Lin, and M.-H. Yang, “Incremental learning for robust visual tracking”, *International Journal of Computer Vision*, vol. 77, no. 1-3, pp. 125–141, 2008.

[23] J. . Tenenbaum, V. De Silva, and J. C. Langford, “A global geometric framework for nonlinear dimensionality reduction”, *Science*, vol. 290, no. 5500, pp. 2319–2323, 2000.

[24] B. Schölkopf, A. Smola, and K.-R. Müller, “Kernel principal component analysis”, in *Proc. ICANN*, Springer, 1997, pp. 583–588.

[25] M. Belkin and P. Niyogi, “Laplacian eigenmaps and spectral techniques for embedding and clustering”, in *Proc. NIPS*, 2002, pp. 585–591.

[26] S. T. Roweis and L. K. Saul, “Nonlinear dimensionality reduction by locally linear embedding”, *Science*, vol. 290, no. 5500, pp. 2323–2326, 2000.

[27] T. Zhang, J. Yang, D. Zhao, and X. Ge, “Linear local tangent space alignment and application to face recognition”, *Neurocomputing*, vol. 70, no. 7-9, pp. 1547–1553, 2007.

[28] V. De Silva and J. B. Tenenbaum, “Sparse multidimensional scaling using landmark points”, Stanford University, Tech. Rep., 2004.

[29] X. He and P. Niyogi, “Locality preserving projections”, in *Proc. NIPS*, 2004, pp. 153–160.

[30] Z. Zhang and H. Zha, “Principal manifolds and nonlinear dimensionality reduction via tangent space alignment”, *SIAM journal on scientific computing*, vol. 26, no. 1, pp. 313–338, 2004.

[31] W. S. Torgerson, *Theory and Methods of Scaling.* Wiley, 1958.

[32] Z. Zhang and J. Wang, “MLLE: Modified locally linear embedding using multiple weights”, in *Proc. NIPS*, 2007, pp. 1593–1600.

[33] J. B. Kruskal, “Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis”, *Psychometrika*, vol. 29, no. 1, pp. 1–27, 1964.

[34] D. D. Lee and H. S. Seung, “Algorithms for non-negative matrix factorization”, in *Proc. NIPS*, 2001, pp. 556–562.

[35] X. He, D. Cai, S. Yan, and H. Zhang, “Neighborhood preserving embedding”, in *Proc. IEEE ICCV*, 2005, pp. 1208–1213.

[36] H. Zou, T. Hastie, and R. Tibshirani, “Sparse principal component analysis”, *Journal of Computational and Graphical Statistics*, vol. 15, no. 2, pp. 265–286, 2006.

[37] D. K. Agrafiotis, “Stochastic proximity embedding”, *Journal of Computational Chemistry*, vol. 24, no. 10, pp. 1215–1221, 2003.

[38] L. v. d. Maaten and G. Hinton, “Visualizing data using t-sne”, *Journal of machine learning research*, vol. 9, no. Nov, pp. 2579–2605, 2008.

[39] N. Halko, P.-G. Martinsson, and J. A. Tropp, *Finding structure with randomness: Stochastic algorithms for constructing approximate matrix decompositions*, arXiv:0909.4061 [math.NA], 2009.

[40] L. McInnes, J. Healy, and J. Melville, *UMAP: Uniform manifold approximation and projection for dimension reduction*, arXiv:1802.03426v2 [stat.ML], 2018.

[41] H. Hotelling, “Analysis of a complex of statistical variables into principal components.”, *Journal of educational psychology*, vol. 24, no. 6, p. 417, 1933.

[42] B. Rieck and H. Leitte, “Agreement analysis of quality measures for dimensionality reduction”, in *Topological Methods in Data Analysis and Visualization*, Springer, 2015, pp. 103–117.

[43] H. Eidenberger, *Handbook of multimedia information retrieval.* Books on Demand, 2012.

[44] I. K. Fodor, “A survey of dimension reduction techniques”, US Dept. of Energy, Lawrence Livermore National Labs, Tech. Rep., 2002, Tech. report UCRL-ID-148494.

[45] P. Hoffman and G. Grinstein, “A survey of visualizations for high-dimensional data mining”, *Information Visualization in Data Mining and Knowledge Discovery*, vol. 104, pp. 47–82, 2002.

[46] H. Yin, “Nonlinear dimensionality reduction and data visualization: A review”, *Intl. Journal of Automation and Computing*, vol. 4, no. 3, pp. 294–303, 2007.

[47] J. Cunningham and Z. Ghahramani, “Linear dimensionality reduction: Survey, insights, and generalizations”, *JMLR*, vol. 16, pp. 2859–2900, 2015.

[48] K. Bunte, M. Biehl, and B. Hammer, “A general framework for dimensionality reducing data visualization mapping”, *Neural Computation*, vol. 24, no. 3, pp. 771–804, 2012.

[49] D. Engel, L. Hüttenberger, and B. Hamann, “A survey of dimension reduction methods for high-dimensional data analysis and visualization”, in *Proc. IRTG Workshop*, vol. 27, Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2012, pp. 135–149.

[50] S. Liu, D. Maljovec, B. Wang, P.-T. Bremer, and V. Pascucci, “Visualizing high-dimensional data: Advances in the past decade”, *IEEE TVCG*, vol. 23, no. 3, pp. 1249–1268, 2015.

[51] T. Schreck, T. Von Landesberger, and S. Bremm, “Techniques for precision-based visual analysis of projected data”, *Information Visualization*, vol. 9, no. 3, pp. 181–193, 2010.

[52] P. Pagliosa, F. V. Paulovich, R. Minghim, H. Levkowitz, and L. G. Nonato, “Projection inspector: Assessment and synthesis of multidimensional projections”, *Neurocomputing*, vol. 150, pp. 599–610, 2015.

[53] R. M. Martins, D. B. Coimbra, R. Minghim, and A. C. Telea, “Visual analysis of dimensionality reduction quality for parameterized projections”, *Computers & Graphics*, vol. 41, pp. 26–42, 2014.

[54] E. F. Vernier, R. Garcia, I. d. Silva, J. L. D. Comba, and A. C. Telea, “Quantitative evaluation of time-dependent multidimensional projection techniques”, in *Computer Graphics Forum*, Wiley Online Library, vol. 39, 2020, pp. 241–252.

[55] P. Joia, D. Coimbra, J. A. Cuminato, F. V. Paulovich, and L. G. Nonato, “Local affine multidimensional projection”, *IEEE TVCG*, vol. 17, no. 12, pp. 2563–2571, 2011.

[56] X. Zhai, X. Chen, L. Yu, and A. Telea, “Interactive axis-based 3d rotation specification using image skeletons.”, in *VISIGRAPP (1: GRAPP)*, 2020, pp. 169–178.

[57] Z. Chen, W. Zeng, Z. Yang, L. Yu, C.-W. Fu, and H. Qu, “Lassonet: Deep lasso-selection of 3d point clouds”, *IEEE Transactions on Visualization and Computer Graphics*, vol. 26, no. 1, pp. 195–204, 2019.

[58] G. M. Morton, “A computer oriented geodetic data base and a new technique in file sequencing”, 1966.

[59] J. L. Bentley, “Multidimensional binary search trees used for associative searching”, *Communications of the ACM*, vol. 18, no. 9, pp. 509–517, 1975.

[60] P. N. Yianilos, “Data structures and algorithms for nearest neighbor search in general metric spaces”, in *Proceedings of the fourth annual ACM-SIAM Symposium on Discrete algorithms*, 1993, pp. 311–321.

[61] N. Beckmann, H.-P. Kriegel, R. Schneider, and B. Seeger, “The r*-tree: An efficient and robust access method for points and rectangles”, in *Proceedings of the 1990 ACM SIGMOD international conference on Management of data*, 1990, pp. 322–331.

[62] A. Guttman, “R-trees: A dynamic index structure for spatial searching”, in *Proceedings of the 1984 ACM SIGMOD international conference on Management of data*, 1984, pp. 47–57.

[63] E. Levina and P. Bickel, “The earth mover’s distance is the mallows distance: Some insights from statistics”, in *Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001*, IEEE, vol. 2, 2001, pp. 251–256.

[64] P. R. Wilson, M. S. Johnstone, M. Neely, and D. Boles, “Dynamic storage allocation: A survey and critical review. in”, in *In Proc. of International Workshop on Memory Management*, 1995.

[65] R. R. Schaller, “Moore’s law: Past, present and future”, *IEEE spectrum*, vol. 34, no. 6, pp. 52–59, 1997.

[66] M. D. Hill and M. R. Marty, “Amdahl’s law in the multicore era”, *Computer*, vol. 41, no. 7, pp. 33–38, 2008.

[67] S. Katz, A. Tal, and R. Basri, “Direct visibility of point sets”, in *ACM SIGGRAPH 2007 papers*, 2007, 24–es.

[68] G. Turk and M. Levoy, “Zippered polygon meshes from range images”, in *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, 1994, pp. 311–318.

[69] P. Cortez, A. Cerdeira, F. Almeida, T. Matos, and J. Reis, “Modeling wine preferences by data mining from physicochemical properties”, *Decision Support Systems*, vol. 47, no. 4, pp. 547–553, 2009.

[70] D. Borland and R. M. Taylor II, “Rainbow color map (still) considered harmful”, *IEEE Computer Architecture Letters*, vol. 27, no. 02, pp. 14–17, 2007.

[71] D. Frey and R. Pimentel, “Principal component analysis and factor analysis”, 1978.

[72] S. De Vito, E. Massera, M. Piga, L. Martinotto, and G. Di Francia, "On field calibration of an electronic nose for benzene estimation in an urban pollution monitoring scenario", *Sensors and Actuators B: Chemical*, vol. 129, no. 2, pp. 750–757, 2008, ISSN: 0925-4005. DOI: `https://doi.org/10.1016/j.snb.2007.09.060`. [Online]. Available: `https://www.sciencedirect.com/science/article/pii/S0925400507007691`.

[73] *Wisconsin breast cancer dataset*, `https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Diagnostic)`, 2021.

[74] S. Zhang, B. Guo, A. Dong, J. He, Z. Xu, and S. Chen, "Cautionary tales on air-quality improvement in Beijing", *Proc Royal Society A*, vol. 473, no. 2205, p. 20 170 457, 2017, `https://archive.ics.uci.edu/ml/datasets/Beijing+Multi-Site+Air-Quality+Data`.

[75] I.-C. Yeh, "Modeling of strength of high-performance concrete using artificial neural networks", *Cement and Concrete research*, vol. 28, no. 12, pp. 1797–1808, 1998.

[76] I.-C. Yeh and C.-h. Lien, "The comparisons of data mining techniques for the predictive accuracy of probability of default of credit card clients", *Expert Systems with Applications*, vol. 36, no. 2, pp. 2473–2480, 2009.

[77] *Reuters dataset*, `https://keras.io/api/datasets/reuters`, 2021.

[78] P. Meirelles, C. Santos, J. Miranda, F. Kon, A. Terceiro, and C. Chavez, "A study of the relationships between source code metrics and attractiveness in free software projects", in *Proc. Brazilian Symposium on Software Engineering (SBES)*, 2010, pp. 11–20.

[79] Z. Tian, X. Zhai, G. van Steenpaal, M. Espadoto, L. Yu, E. Dimara, and A. Telea, "Quantitative and qualitative comparison of 2d and 3d projection techniques for high-dimensional data, article in preparation.", *Information*,

[80] P. Joia, D. Coimbra, J. A. Cuminato, F. V. Paulovich, and L. G. Nonato, "Local affine multidimensional projection", *IEEE Transactions on Visualization and Computer Graphics*, vol. 17, no. 12, pp. 2563–2571, 2011.