

# DaRt: Generative Art using Dimensionality Reduction Algorithms

## Abstract

Dimensionality Reduction (DR) is a popular technique that is often used in Machine Learning and Visualization communities to analyze high-dimensional data. The approach is empirically proven to be powerful for uncovering previously unseen structures in the data. While observing the results of the intermediate optimization steps of DR algorithms, we coincidentally discovered the artistic beauty of the DR process. With enthusiasm for the beauty, we decided to look at DR from a generative art lens rather than their technical application aspects and use DR techniques to create artwork. Particularly, we use the optimization process to generate images, by drawing each intermediate step of the optimization process with some opacity over the previous intermediate result. As another alternative input, we used a neural-network model for face-landmark detection, to apply DR to portraits, while maintaining some facial properties, resulting in abstracted facial avatars. In this work, we provide such a collection of such artwork.

## Authors Keywords

dimensionality reduction; generative art.

## Introduction

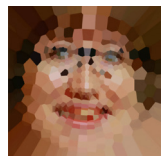
Dimensionality Reduction (DR) is a popular technique that is often used in Machine Learning and Visualization communities to analyze high-dimensional data. The target lower dimension is normally either 2 or 3. A scatterplot (2D or 3D) is then usually used to visualize the outcome of the DR technique. The approach is empirically proven to be powerful for uncovering previously unseen structures in the data.

Technically, given an analysis task, a DR algorithm aims to preserve certain properties of the original dataset in the lower dimensional embedding, for instance, properties such as the neighborhood of data points or the similarity or dissimilarity among the data points. For some DR algorithms, those properties are formulated as a cost function for an iterative optimization process. While developing the DR JavaScript library `DruidJS` [1], we gathered a collection of datasets to test the implementations of the DR techniques. Working with those algorithms and observing the results of the intermediate steps from the iterative process, we coincidentally discovered the artistic beauty of the DR plots. Inspired by this find-

Rene Cutura  
TU Wien & University of Stuttgart  
rene.cutura@tuwien.ac.at



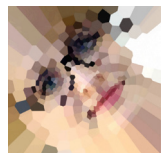
Katrin Angerbauer  
University of Stuttgart  
Katrin.Angerbauer@visus.uni-stuttgart.de



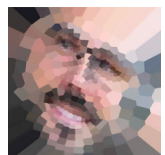
Frank Heyen  
University of Stuttgart  
Frank.Heyen@visus.uni-stuttgart.de



Natalie Hube  
University of Stuttgart  
Natalie.Hube@visus.uni-stuttgart.de



Michael Sedlmair  
University of Stuttgart  
Michael.Sedlmair@visus.uni-stuttgart.de



ing and also by the work of Tyler Hobbs<sup>1</sup>, we decided to look at DR from a generative art lens rather than their technical application aspects and use DR techniques to create artwork.

On one hand, by observing the inner working optimization process of DR algorithms one can gain insightful information about

1. <https://tylerxhobbs.com/essays/2020/how-to-hack-a-painting>

what properties and how the original data gets projected. On the other hand, visualizing the intermediate results of the process surprisingly provides aesthetic patterns that are artsy. Furthermore, the human face by its beautiful nature is the main objective topic in painting. Humans also can process faces very fast, which makes faces be reliable ground truth for detecting artifacts a DR method can produce. We further explore the artistic potential of DR algorithms by generating DR paintings from visualization and face recognition datasets.

DR also could be seen as a technique that takes this year’s VISAP’s theme literally on a higher non-technical level. DR connects data in the way it transforms and preserves certain properties of the original data in lower dimensional embeddings. At the same time,

DR disconnects data by causing some distortions that do not reflect the ground truth data. Particularly, DR creates “False neighbors” and “Missing neighbors” [2]. A “False neighbor” is a point that is a neighbor point in the lower dimensional embedding but not a neighbor in the original high-dimensional space. A “Missing neighbor” is a point that is not a neighbor point in the lower dimensional embedding but a neighbor point in the original space.

Given the coincident discovery in DR and the fact that DR matches with this year’s VISAP’s theme, we decide to exhibit our art collection generated by DR in this portfolio.

### Related Work

According to Boden and Edmonds [3], generative art had its beginnings in the late 1950s, with the rise of computer technology.

They coined “computer generated art” as

“[an artwork resulting] from some computer program being left to run by itself, with minimal or zero interference from a human being”.

Nowadays, neural networks [4] or other intelligent models [5][6] are used for generating art.

As the ways a computer program could generate art grew more and more diverse over time, Dorin et al. [7] defined a more fine-grained framework. The table below shows our work within that framework.

### Methods

We used the iterative DR methods implemented in DruidJS v0.3.14 – t-SNE [8], UMAP [9], and SAMMON’ mapping [10] – with their default parameters and the string “cellar door” as seed for the random number

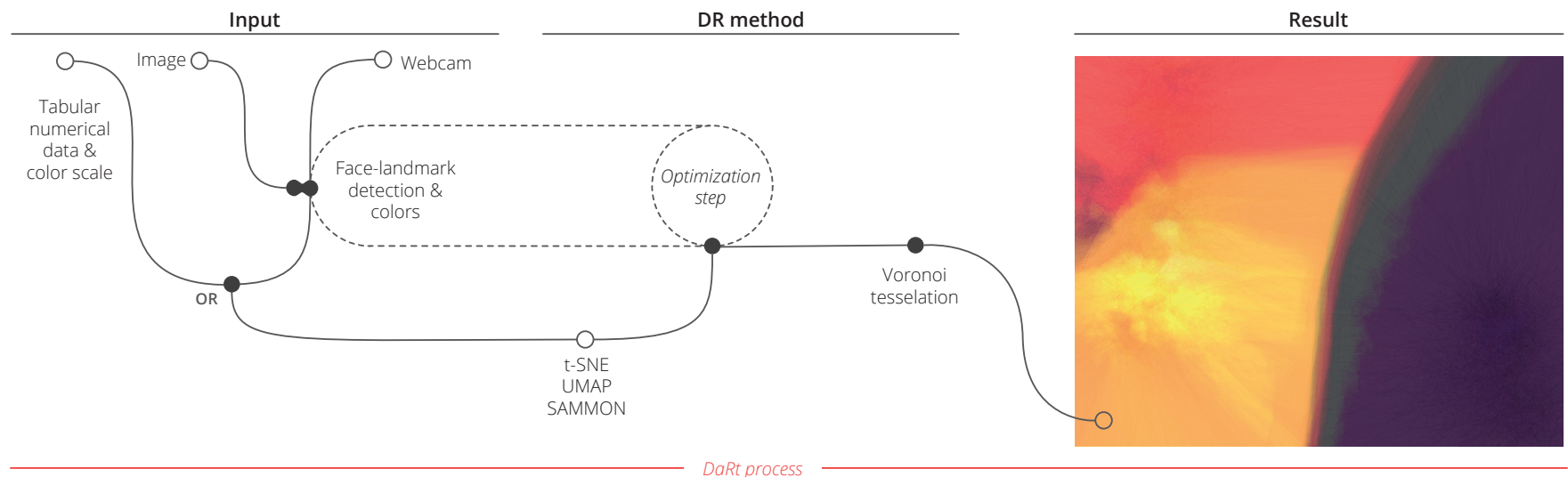
Entities	Initialization, termination	Processes	Environmental interaction	Sensory outcomes
Points, polygons (voronoi cells)	Random initialization, termination determined by a user, or by the stopping criteria of the used DR method.	The positions of the projected datapoints, defining the polygons, get optimized by a optimization method. The change in position, change the appearance of the polygons. After each optimization step, the polygons get drawn onto the canvas with a user defined opacity.	User defineable parameters. With webcam input, reactivity to facial expression.	Either, an image with hints to its origins in data, or a image, i.e. for a personalized avatar .

generator. We use the available continuous colorscales from d3-scale-chromatic [11] v3.0.0. For the Voronoi tessellation we use d3-delaunay [12] in version 6.0.2.

To generate the following images, we use the intermediate results of the respective optimization process. We then compute the Voronoi tessellation [13] of the intermediate result. We use the polygons from this tessellation to color all pixels of the image with the color of their closest data point. Then we draw each voronoi cell with a color from the selected color scale with a user-defined opacity onto the canvas.

Important for an aesthetic result is the ordering of the points. Unordered points create turbulent images. Almost all of the datasets we used are pre-labelled and sorted accordingly.

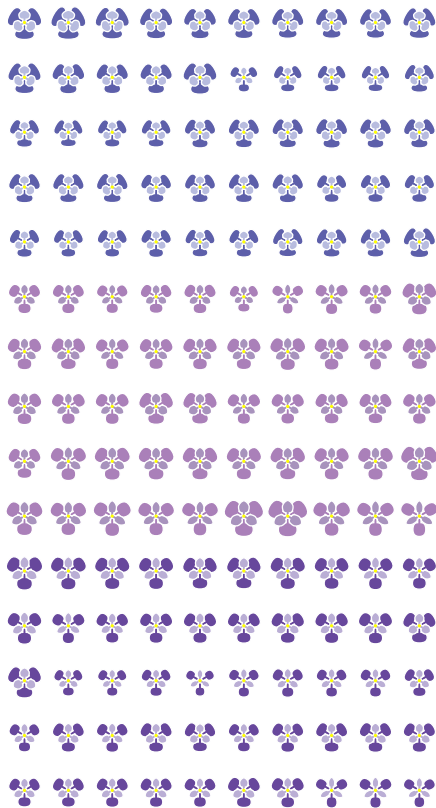
As alternative input we used images of faces, or a webcam input, where we use the face-landmark detection model [14] (v0.0.3 in combination with tfjs [15] v3.7.0), which detects specific landmarks on the face and returns those as 3D coordinates. We then project this 3D data with the iterative DR methods, and draw them as with the previous input.



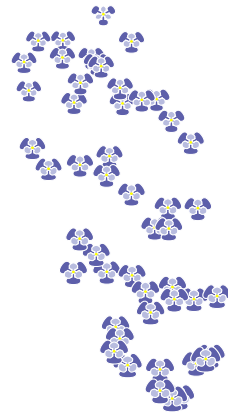
# Example

## THE IRIS DATASET.

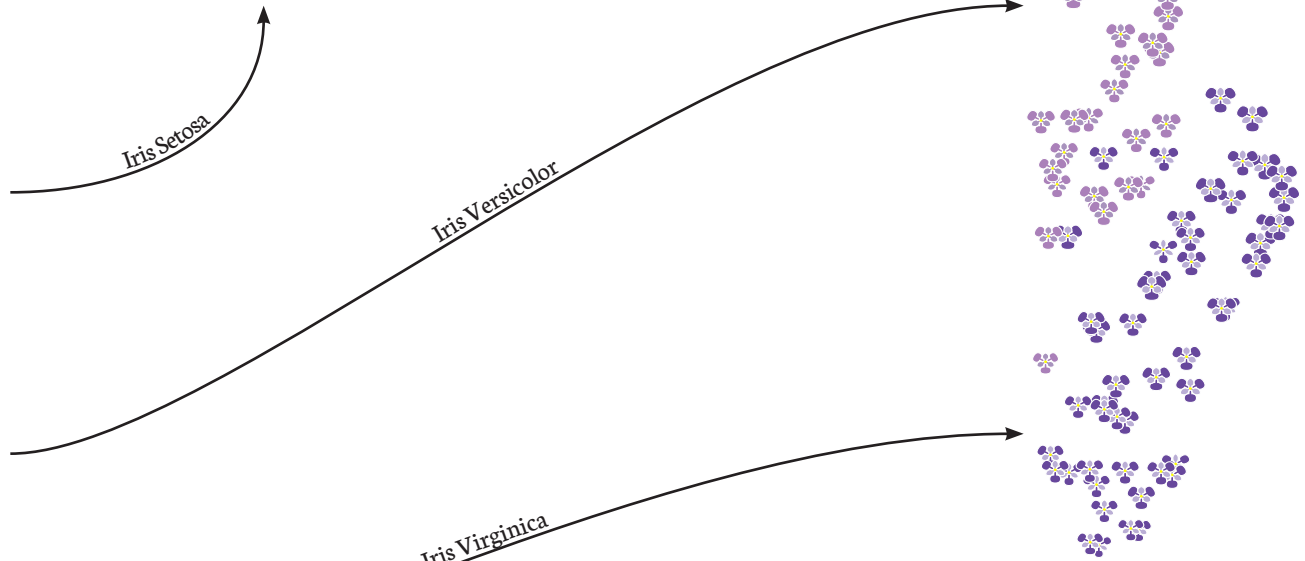
The dataset used of the previous image is the IRIS dataset [16]. This dataset consists of measurements of 50 exemplars of the 3 species “Iris Setosa”, “Iris Versicolor”, and “Iris Virginica”. The measurements consists of the sepal length, the sepal width, the petal length, and the petal width.



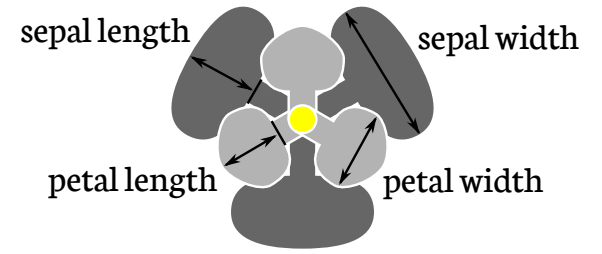
Glyphs for all 150 exemplars of the IRIS dataset..



UMAP projection of the IRIS dataset. (default parameters, seed: “cellar door”, 350 iterations)



The dataset consists of two clusters, where the exemplars of the Iris Setosa are distant to the other two species, because the sepal- and the petalwidth is bigger compared to the other two species. Iris Versicolor und Iris Virginica have exemplars which have similar measurements, therefore those are close to each other.

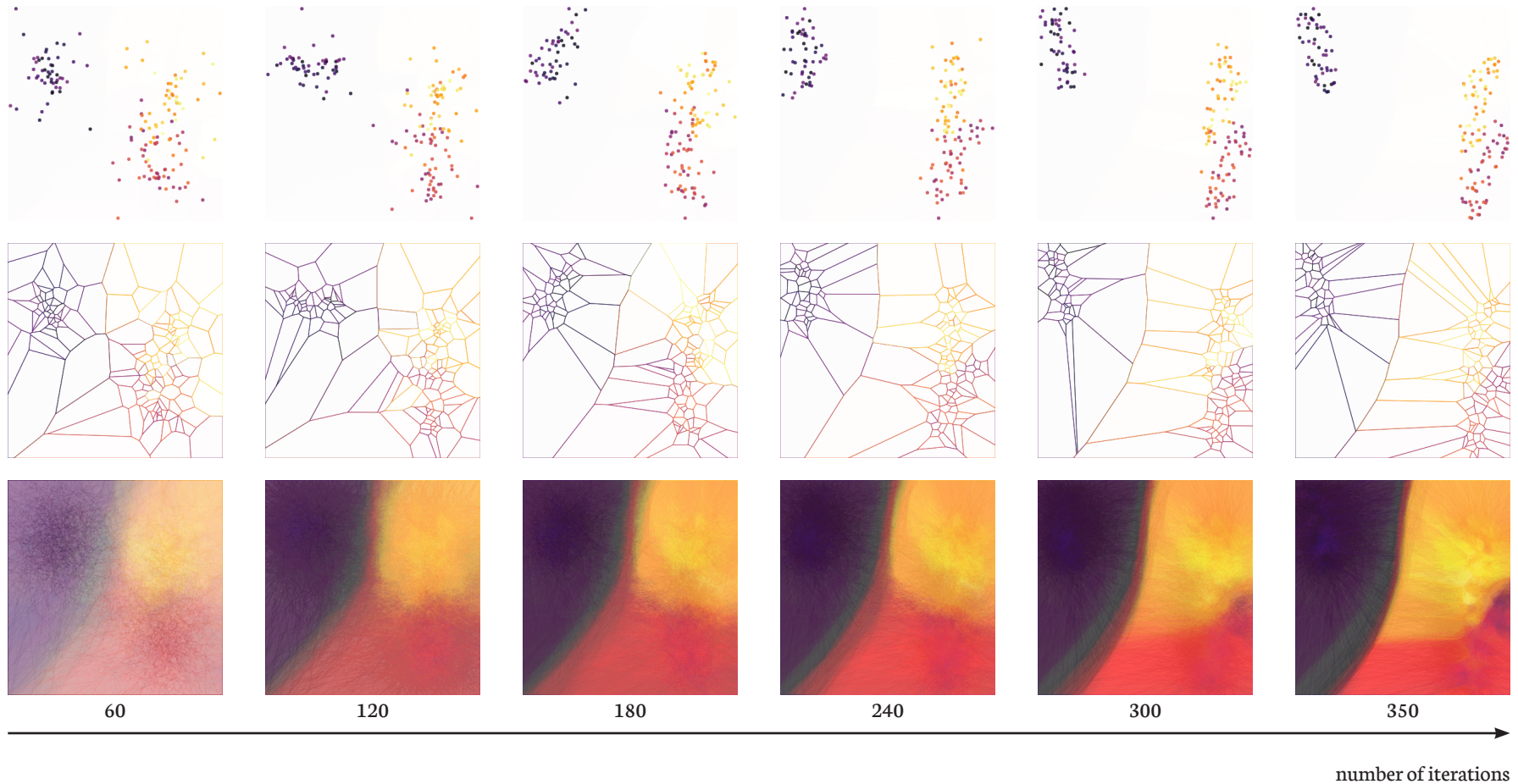


Glyph design.



# Generation

THE IRIS DATASET.

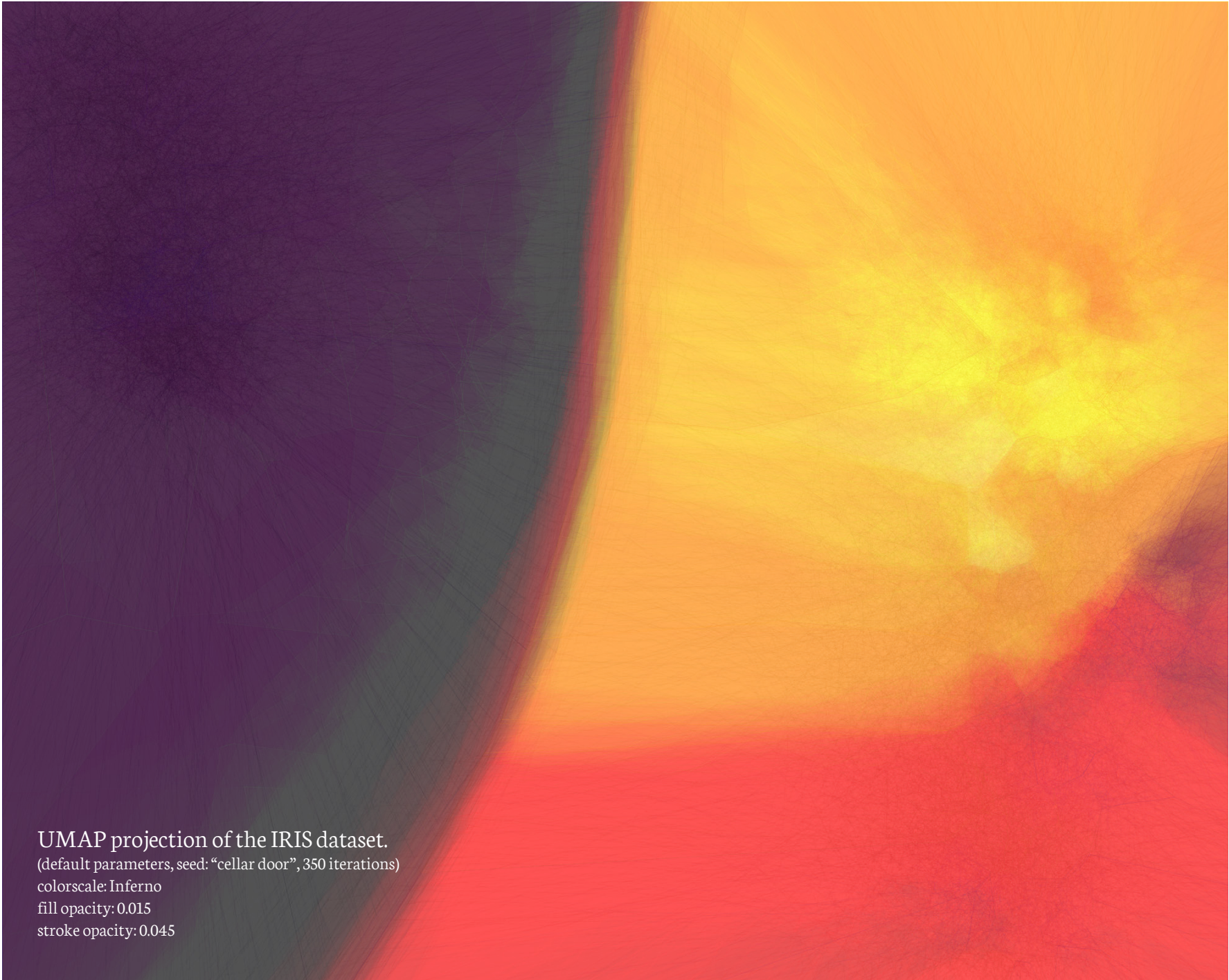


VISAP'21, Pictorials and annotated portfolios.

For generating the images, we use an iterative DR method – here UMAP – to compute a projection of the dataset. In this case, we project the 4-dimensional Iris dataset to 2 dimensions. The first row shows the intermediate results as classical scatterplot. The second row shows the voronoi cells of the intermediate result.

The last row shows the artistic generation of its image. For this example we used the colorscale “Inferno” and colored the points and voronoi cells with the colors from the colorscale. After initializing the projection with random point placement, UMAP “finds” the “two cluster” structure in the data very early in the opti-

mization process and pushes those two apart.



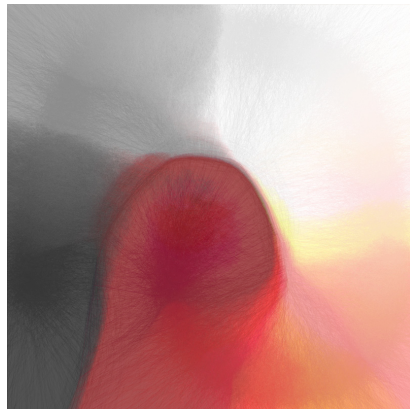
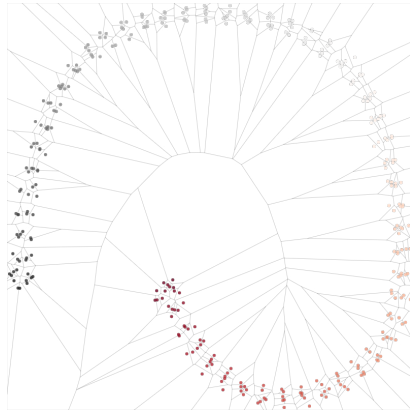
UMAP projection of the IRIS dataset.  
(default parameters, seed: "cellar door", 350 iterations)  
colorscale: Inferno  
fill opacity: 0.015  
stroke opacity: 0.045



# Examples

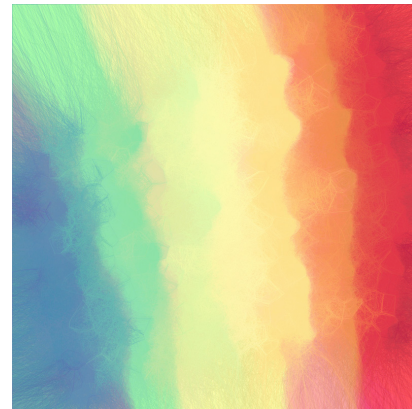
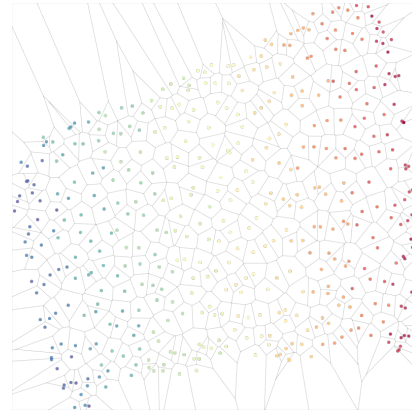
Here, we show further examples of some artistic images generated, together with their “classical” scatterplot counterpart and the voronoi tessellation.

The images of the Mammoth dataset (a) shows an example for False and Missing neighbors. A foot of the mammoth is disconnected and gets crossed by another foot. This becomes also visible in the artistic image, where the green area cuts through the cyan part.



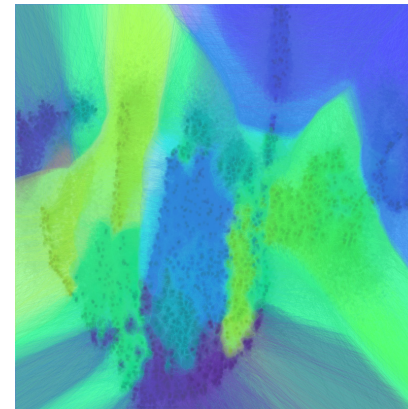
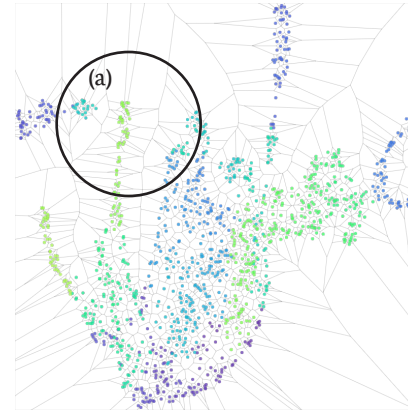
**S-shape dataset.**

UMAP, default parameters, seed: “cellar door”, 400 iterations, fill opacity: 0.015, stroke opacity: 0.045, colorscale: “RdGy”



**Swissroll dataset.**

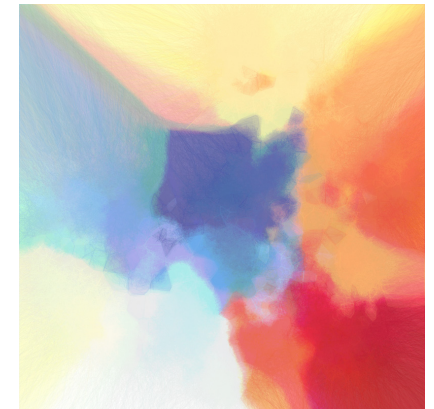
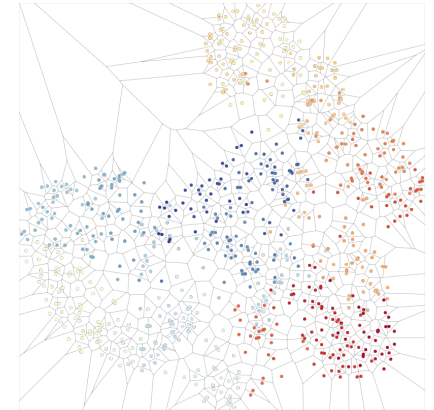
UMAP, default parameters, seed: “cellar door”, 400 iterations, fill opacity: 0.015, stroke opacity: 0.045, colorscale: “Spectral”.



**Mammoth dataset.**

UMAP, default parameters, seed: “cellar door”, 400 iterations, fill opacity: 0.015, stroke opacity: 0.045, colorscale: “Cool”.

Here we also drew the points slightly darker and with opacity 0.09 onto the canvas.

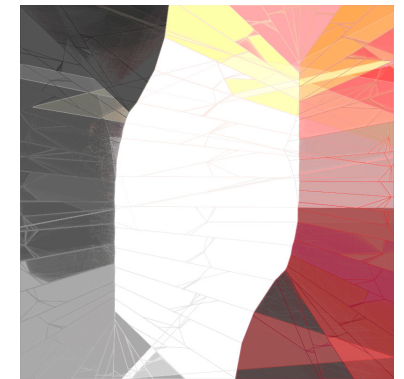
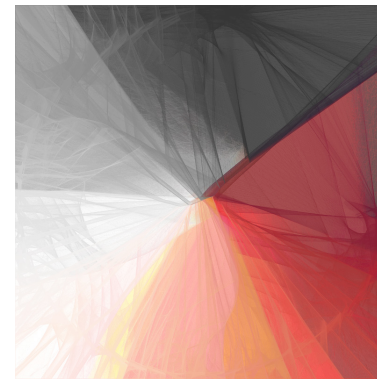
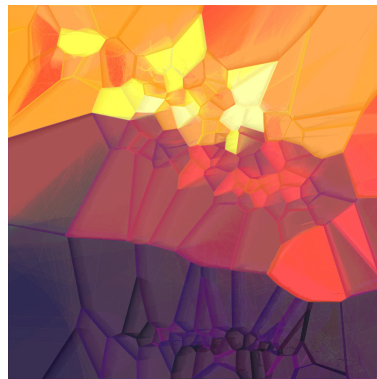
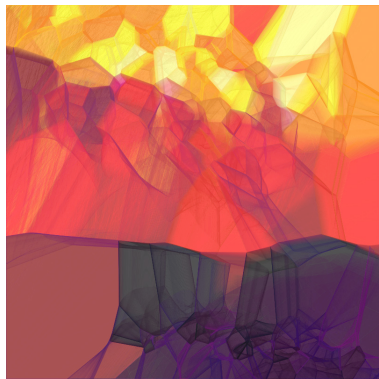
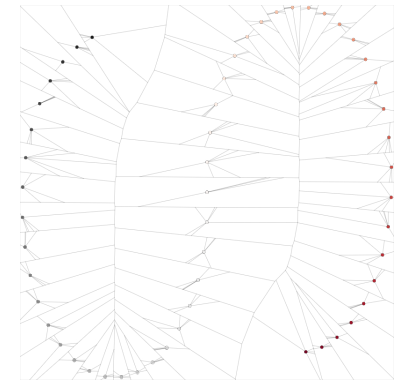
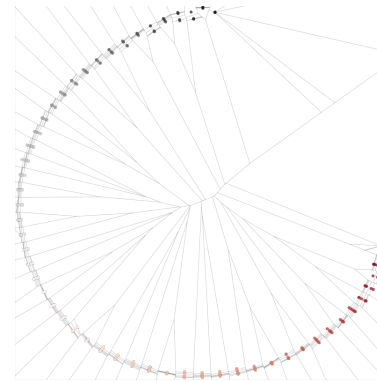
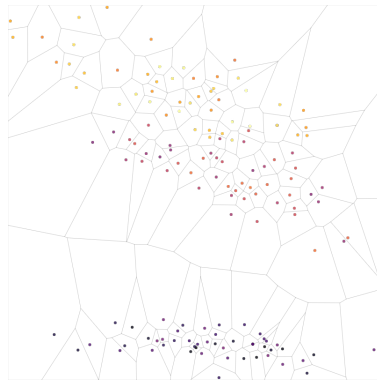
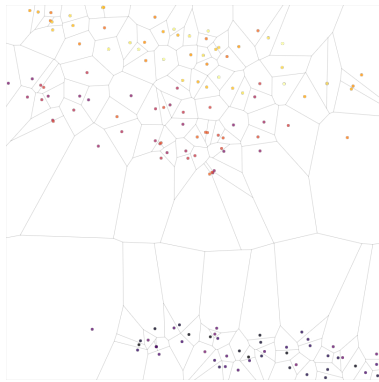


**Fashion-MNIST dataset.**

UMAP, default parameters, seed: “cellar door”, 400 iterations, fill opacity: 0.015, stroke opacity: 0.045, colorscale: “RdYlBu”

# Other DR techniques

## SAMMONS MAPPING AND T-SNE



t-SNE

SAMMON

t-SNE

SAMMON

### Iris dataset.

both with default parameters, seed: "cellar door", 400 iterations, fill opacity: 0.015, stroke opacity: 0.045, colorscale: "Inferno"

### S-shape dataset.

both with default parameters, seed: "cellar door", 400 iterations, fill opacity: 0.015, stroke opacity: 0.045, colorscale: "RdGy"

t-SNE and SAMMON use a different optimization technique (gradient descent) than UMAP (stochastic gradient descent) which runs "smoother". With that optimization technique only small changes occur between the inter-

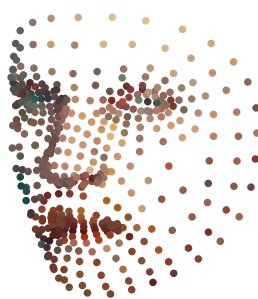
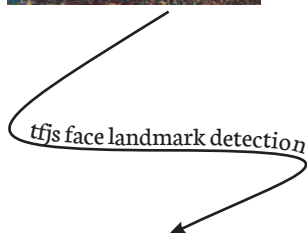
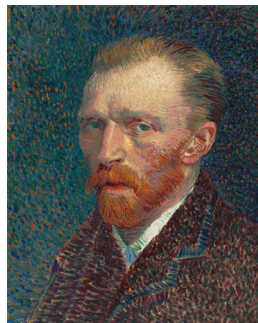
mediate results. The smaller changes then lead to a more stable voronoi tessellation, which is visible in the artistic images as more visible voronoi cells. t-SNE and SAMMON have also different optimization goals. SAMMON, for

instance tries to maintain all distances between the points, t-SNE maintains the neighborhood of the datapoints, which helps unfolding potential manifolds in the original data.

# Face Landmarks I

DR APPLIED ON PORTRAITS

Here, we run a face landmark detection model using tensorflowjs, to get 3d points of faces detected in images. We then use this 3D data as input for a DR algorithm and draw the intermediate result like before. Instead of using a colorscale we use the colors of the image.



UMAP projection of the face landmarks of the portrait of  
Vincent van Gogh.  
(default parameters, seed: "cellar door", 350 iterations)  
fill opacity: 0.15  
stroke opacity: 0.3



# More Examples

## DR APPLIED ON PORTRAITS

We selected these portraits because they are public domain and known. Other than that, the face-landmark detection model needs to work on the portrait. Portraits from Picasso for instance do not work, because the model cannot find the face landmarks. Interesting is the last image “Vertumnus” because the model found the face, although the image shows only fruits and vegetables.

Original portrait

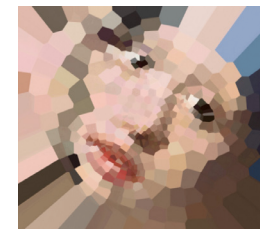
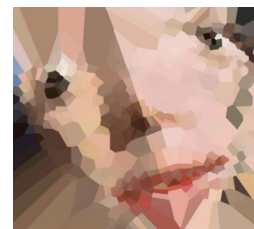
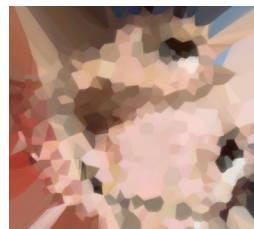
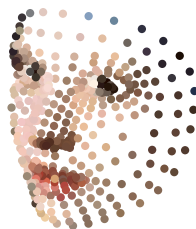
Face landmarks

UMAP projection

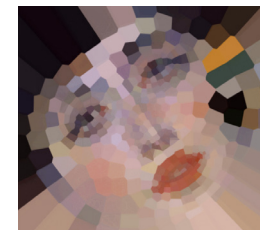
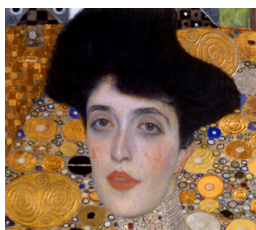
t-SNE projection

Sammons mapping

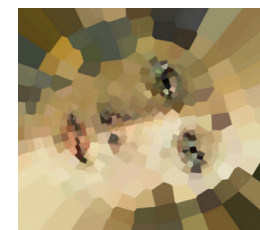
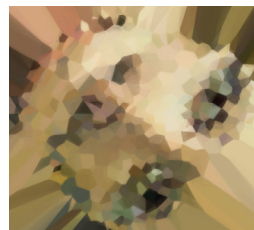
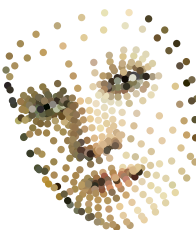
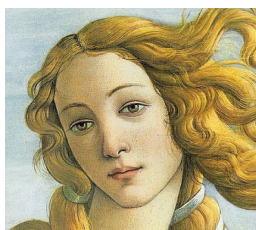
“Girl with a Pearl Earring”  
Johannes Vermeer, 1665



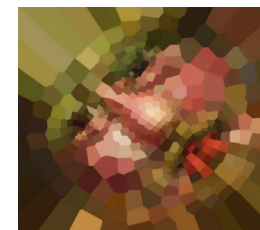
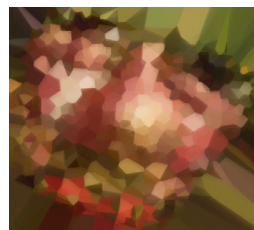
“The Lady in Gold”  
Gustav Klimt, 1907



“The Birth of Venus”  
Sandro Botticelli, 1480



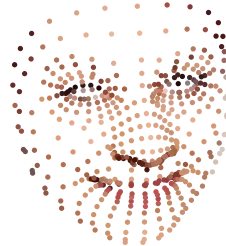
“Vertumnus”  
Giuseppe  
Arcimboldo, 1591



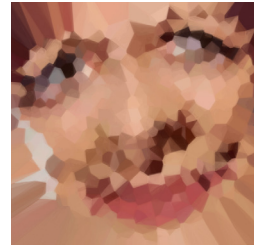
# Face Landmarks II

## DR APPLIED ON PHOTOS

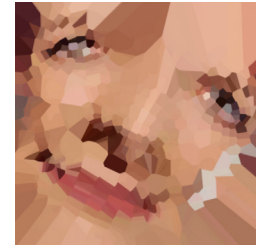
The method works also on real photos. Also, the three DR methods, UMAP, t-SNE, and SAMMON's mapping show different "artefacts", based on their optimization process and goals.



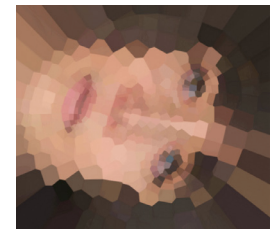
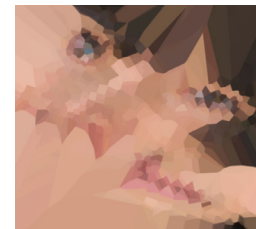
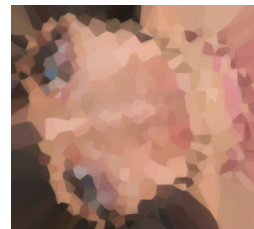
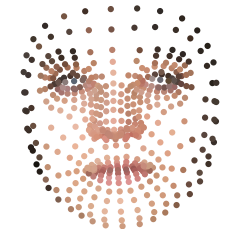
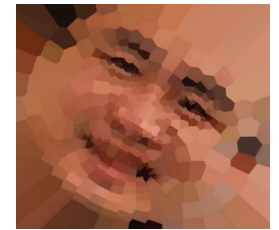
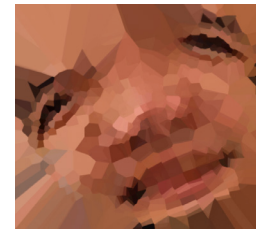
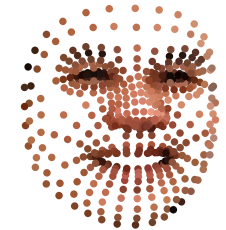
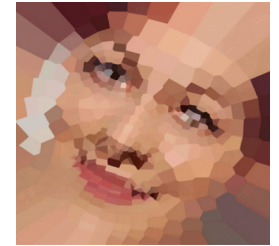
UMAP projection



t-SNE projection



SAMMON's mapping



### UMAP

This method generates more "blurry" images than the others, because the points get moved stronger in UMAP's optimization process.

### t-SNE

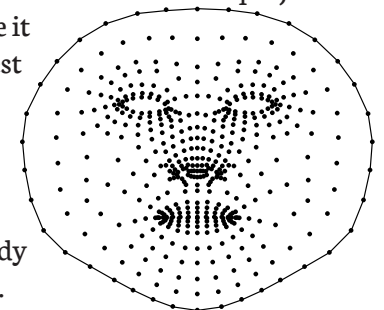
The method t-SNE computes for each point the probabilities for the other points to be a neighbor. The optimization tries to maintain this probabilities in the lower-dimensional space. With this formulation large distances are more or less ignored, which is well known

for t-SNE [17]. The keypoints which get detected in the face have some dense areas with close distances, which then get preferred or focused by t-SNE. The other connections are more irrelevant for this technique. This is observable by the eye distances in the projections of the first two rows. The projection in the third row contains some Missing and False neighbors in the nose and mouth area.

### SAMMON

SAMMON's mapping tries to maintain all

distances as good as possible. This leads to the "round shapes" artefact. From a projection perspective it produces the best results, because the datapoint given by the face-landmark model are already on a 2D surface.



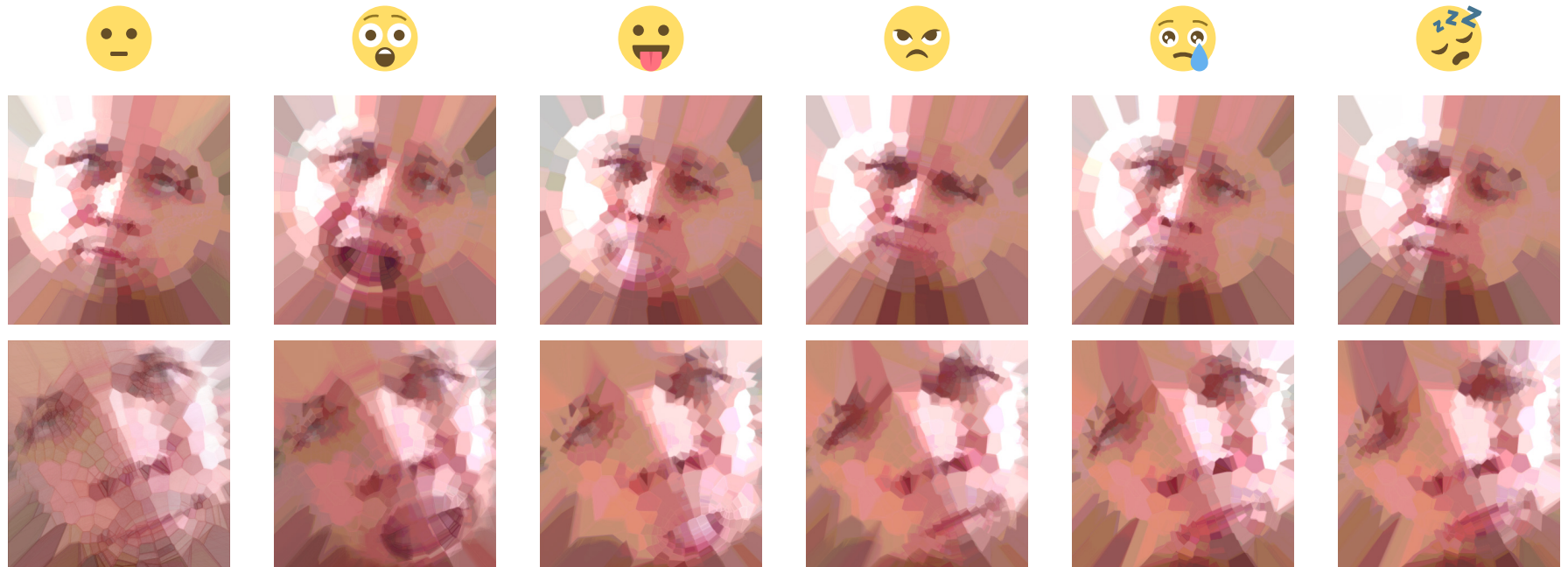
# Face Landmarks III

## DR APPLIED ON WEBCAM INPUT

As another alternative input we use the webcam.

The face-landmark detection model keeps predicting the face-landmarks. The 3D data updates the running DR method, by exchanging the data on which the DR method is optimizing the 2D projection. So, if distances change, for instance one opens

the mouth, or closes an eye, then the DR method captures this change in the original data in the next optimization steps.

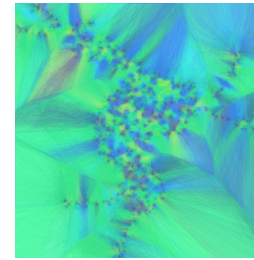


The emojis show the facial expression. Each row was taken from one webcam stream, where each of the columns show the projection after adapting to the new facial expression. For the first row SAMMON's mapping was used, and for the second row t-SNE. Only with these two DR method implementations it is possible to run the optimization process without stopping criteria.

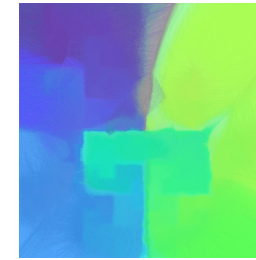


# Discussion

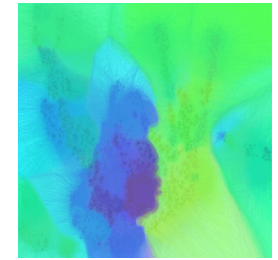
The generated images are strongly influenced by the ordering of the datapoints, because the ordering defines which color from the colorscale gets used. The expected behaviour of DR is to put similar points close in the projection. If the points are ordered by similarity, and ending up close in the projection by the DR method, then DaRt generates rather large areas with smooth color transitions. Otherwise, DaRt produces more unsteady images with a lot of changes in color.



shuffled

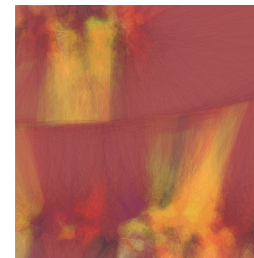


shuffled & hilbert  
curve ordering

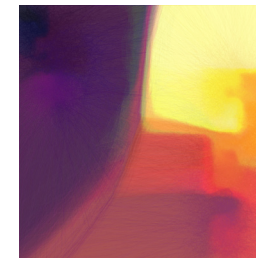


shuffled & ordering with  
hierarchical clustering

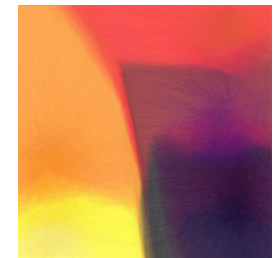
**Mammoth dataset**



shuffled



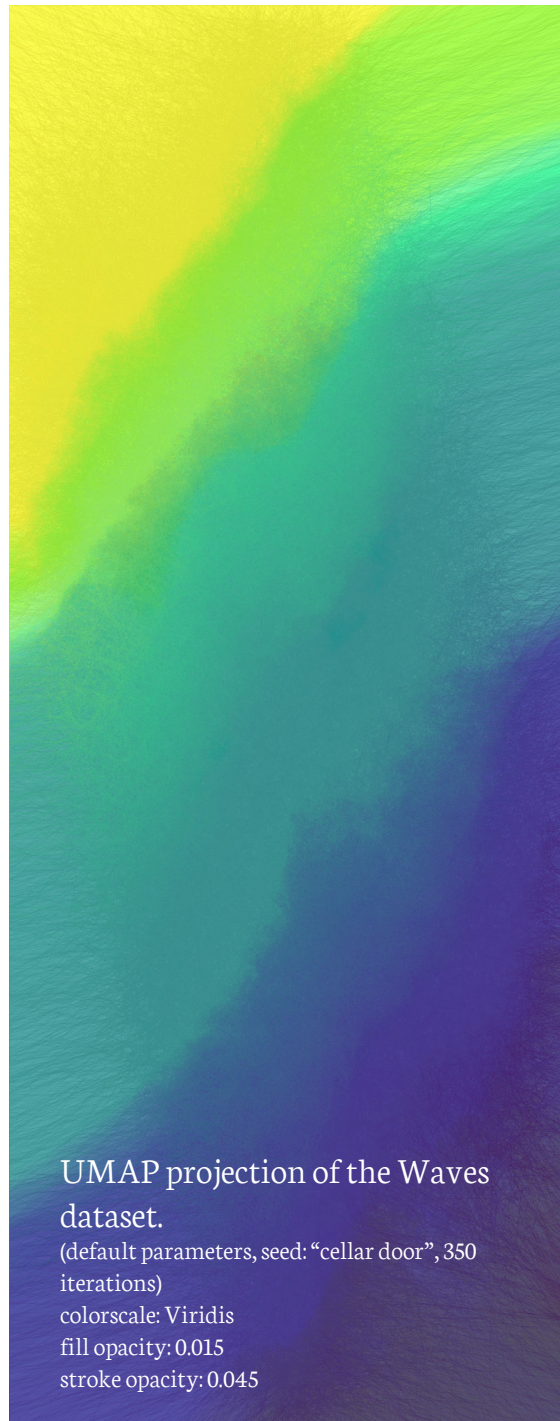
shuffled & hilbert  
curve ordering



shuffled & ordering with  
hierarchical clustering

**Iris dataset**

Each row shows three UMAP projections, but we shuffled the ordering of the points. **Left:** Here we applied DaRt on randomly shuffled dataset, which results in a turbulent image. **Middle:** For the second image, we ordered each intermediate result with the help of a hilbert curve [\[18\]](#), which leads to rectangular artefacts following the hilbert curve. **Right:** We ordered the points with hierarchical clustering (optimal leaf ordering) after shuffling and before applying DR, which orders the points by similarity, in this case the euclidean distance.



## References

1. Cutura, R., Kralj, C., & Sedlmair, M. (2020, October). DRUID JS—A JavaScript Library for Dimensionality Reduction. In 2020 IEEE Visualization Conference (VIS) (pp. 111-115). IEEE. <https://doi.org/10.1109/VIS47514.2020.00029>
2. Heulot, N., Aupetit, M., & Fekete, J. D. (2013, June). Proxilens: Interactive exploration of high-dimensional data using projections. In VAMP: EuroVis Workshop on Visual Analytics using Multidimensional Projections. The Eurographics Association. <https://dx.doi.org/10.2312/PE.VAMP.VAMP2013.011-015>
3. Boden, M. A., & Edmonds, E. A. (2009). What is generative art?. *Digital Creativity*, 20(1-2), 21-46. <https://doi.org/10.1080/14626260902867915>
4. Fernando, C., Eslami, S. M., Alayrac, J. B., Mirowski, P., Banarse, D., & Osindero, S. (2021). Generative Art Using Neural Visual Grammars and Dual Encoders. arXiv preprint arXiv:2105.00162.
5. de Andrade, D., Fachada, N., Fernandes, C. M., & Rosa, A. C. (2020). Generative Art with Swarm Landscapes. *Entropy*, 22(11), 1284. <https://doi.org/10.3390/e22111284>
6. Wu, T. (2018, February). Saliency-aware generative art. In Proceedings of the 2018 10th International Conference on Machine Learning and Computing (pp. 198-202). <https://doi.org/10.1145/3195106.3195143>
7. Dorin, A., McCabe, J., McCormack, J., Monro, G., & Whitelaw, M. (2012). A framework for understanding generative art. *Digital Creativity*, 23(3-4), 239-259. <https://doi.org/10.1080/14626268.2012.709940>
8. Van der Maaten, L., & Hinton, G. (2008). Visualizing data using t-SNE. *Journal of machine learning research*, 9(11).
9. McInnes, L., Healy, J., & Melville, J. (2018). Umap: Uniform manifold approximation and projection for dimension reduction. arXiv preprint arXiv:1802.03426.
10. Sammon, J. W. (1969). A nonlinear mapping for data structure analysis. *IEEE Transactions on computers*, 100(5), 401-409. <https://doi.org/10.1109/T-C.1969.222678>
11. d3-scale-chromatic. Retrieved June 8th, 2021. <https://github.com/d3/d3-scale-chromatic/releases/tag/v3.0.0>
12. d3-delaunay. Retrieved June 8th, 2021. <https://github.com/d3/d3-delaunay/releases/tag/v6.0.2>
13. Voronoi, G. (1908). Nouvelles applications des paramètres continus à la théorie des formes quadratiques. Premier mémoire. Sur quelques propriétés des formes quadratiques positives parfaites. *Journal für die reine und angewandte Mathematik (Crelles Journal)*, 1908(133), 97-102. <https://doi.org/10.1515/crll.1908.133.97>
14. face-landmarks-detection. Retrieved June 8th, 2021. <https://github.com/tensorflow/tfjs-models/releases/tag/face-landmarks-detection-v0.0.3>
15. tensorflowjs. Retrieved June 8th, 2021. <https://github.com/tensorflow/tfjs/releases/tag/tfjs-v3.7.0>
16. Fisher, R. A. (1936). 138: The Use of Multiple Measurements in Taxonomic Problems. <https://doi.org/10.1111/j.1469-1809.1936.tb02137.x>
17. Wattenberg, M., Viégas, F., & Johnson, I. (2016). How to use t-SNE effectively. *Distill*, 1(10), e2. <http://doi.org/10.23915/distill.00002>
18. Hilbert, D. (1935). Über die stetige Abbildung einer Linie auf ein Flächenstück. In *Dritter Band: Analysis· Grundlagen der Mathematik· Physik Verschiedenes* (pp. 1-2). Springer, Berlin, Heidelberg. [https://doi.org/10.1007/978-3-662-38452-7\\_1](https://doi.org/10.1007/978-3-662-38452-7_1)