Algorithms for generating and evaluating visually sorted grid layouts

Kai Uwe Barthel

Visual Computing Group, HTW Berlin
visual-computing.com

Hochschule für Technik
und Wirtschaft Berlin

University of Applied Sciences

Visual Computing @ HTW Berlin

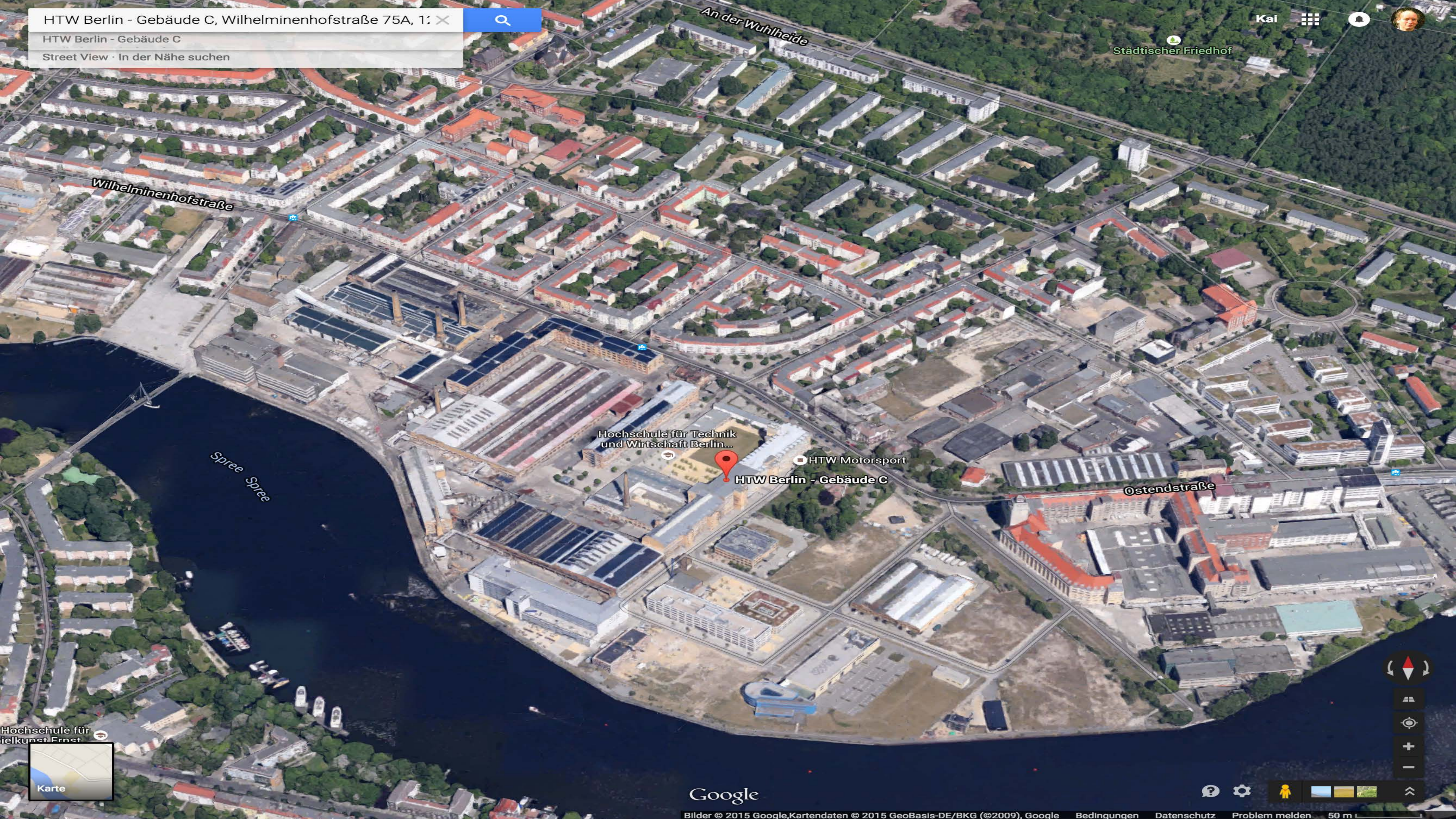
Prof. Dr. Kai Barthel

Prof. Dr. Klaus Jung

M.Sc. Nico Hezel

M.Sc. Konstantin Schall

Research areas: Information Retrieval, Machine Learning, Computer Vision, Visualization, and Visual Sorting


HTW Berlin - Gebäude C, Wilhelminenhofstraße 75A, 1:
HTW Berlin - Gebäude C
Street View · In der Nähe suchen
Kai
An der Wuhlheide
Städtischer Friedhof
Wilhelminenhofstraße
Hochschule für Technik
und Wirtschaft Berlin...
HTW Motorsport
HTW Berlin - Gebäude C
Ostendstraße
Spree
Spree
Karte
Google
Bilder © 2015 Google,Kartendaten © 2015 GeoBasis-DE/BKG (©2009), Google
Bedingungen
Datenschutz
Problem melden
50 m

Tutorial Overview I

- Motivation
- Principle of sorting
- Visual feature vectors
- Dimensionality reduction
- Image sorting algorithms
- Metrics for evaluating sorted arrangements
- A new quality metric for sorted grid layouts

Tutorial Overview II

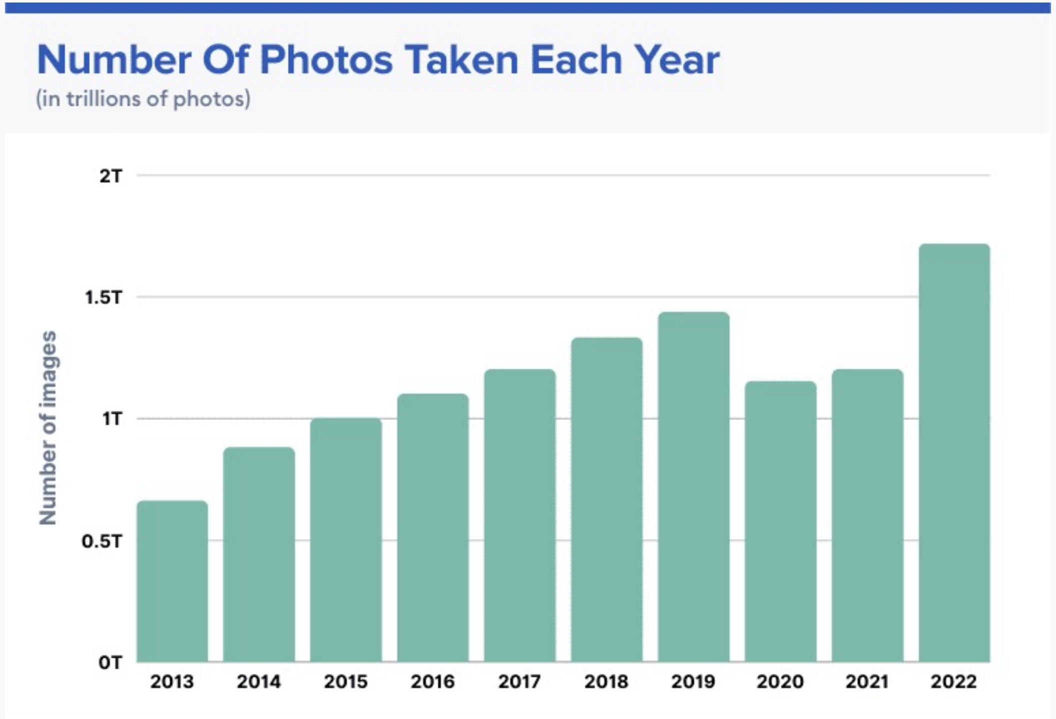
- Human evaluation of sorted arrangements
- Linear Assignment Sorting
- Performance evaluation for visually sorted grid layouts
- Sorting with spatial constraints
- Visual exploration & navigation
- Summary Q&A

Motivation for Sorting Images

Increasing Numbers of Photos

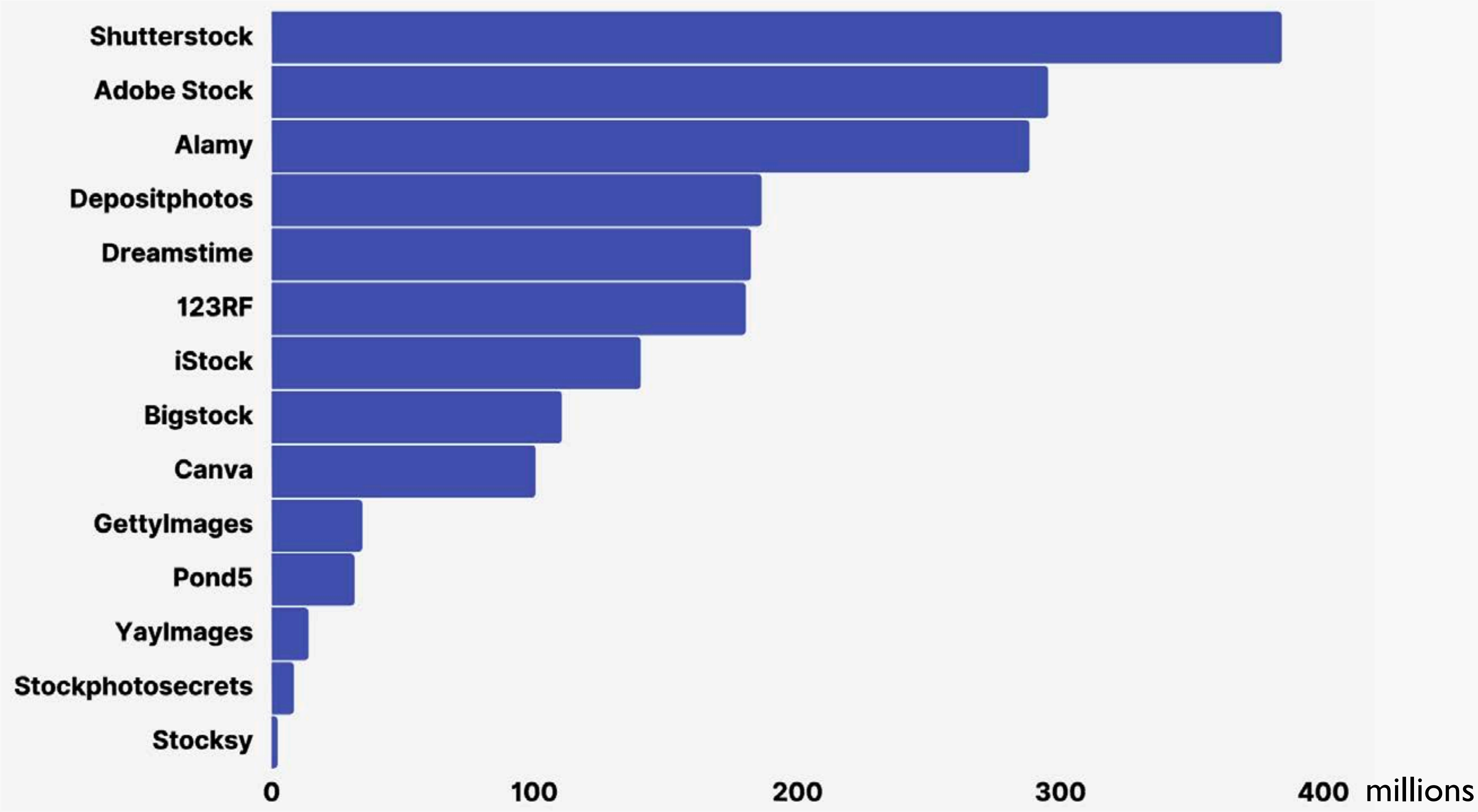
1,720,000,000,000 photos taken worldwide in 2022

The average user has around **2,100** photos on the smartphone



photutorial.com

Stock Agencies with Millions of Images



photutorial.com

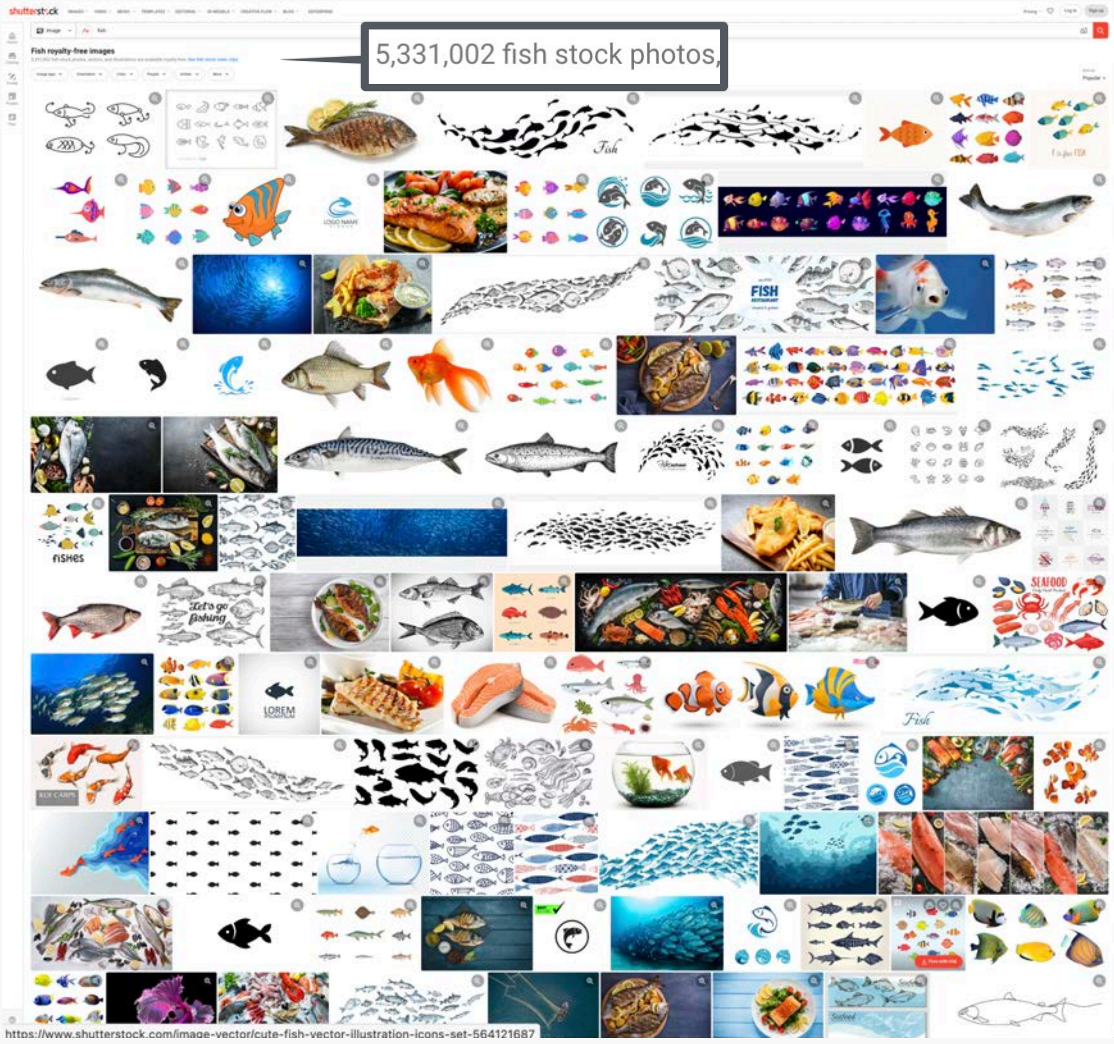
350,000 Images (uploaded to Flickr per day)

24HRS in Photos by Erik Kessels

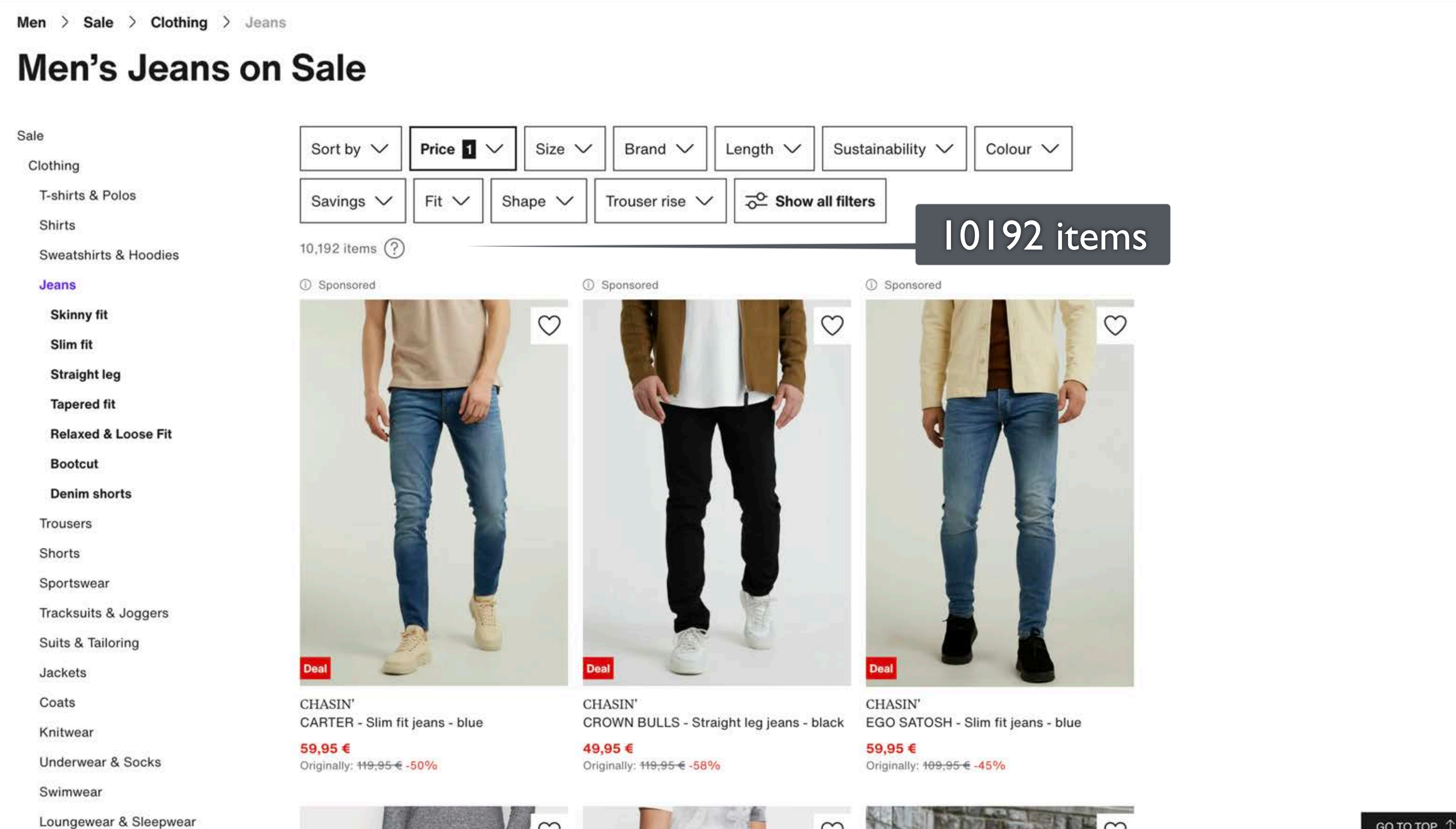
photo: www.schabel-kultur-blog.de

Agencies with Millions of Images

- No one has ever seen all the images.
- Impossible to get an overview
- "Exploring" a search result
  =
  Scrolling through endless, unstructured lists of images



Only a tiny fraction of a product type is shown on e-commerce websites



Human perception is limited to few images

800 Images (CD covers)

18 Images

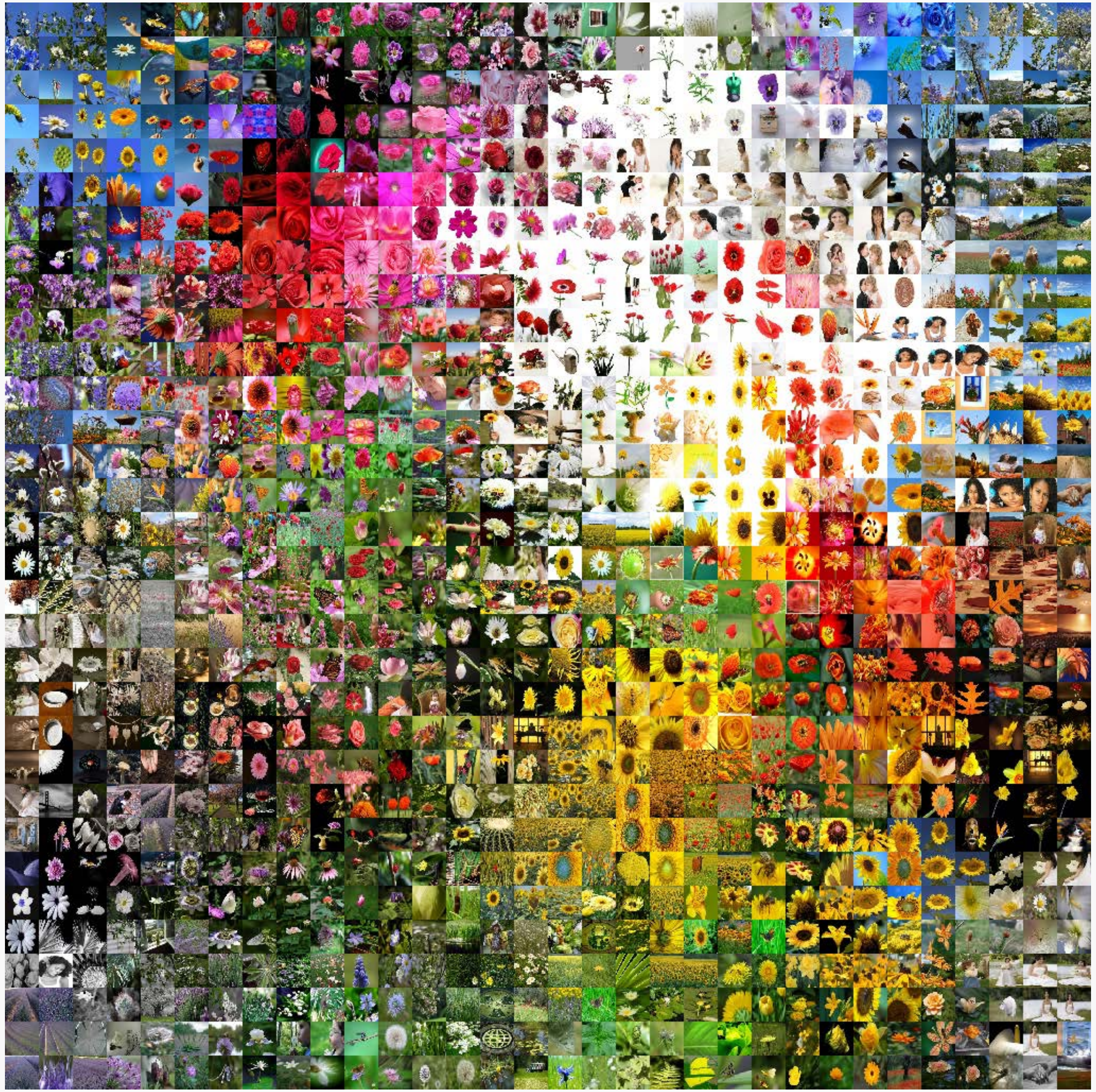
Only 10–20 images can be perceived at once.

Image Sorting

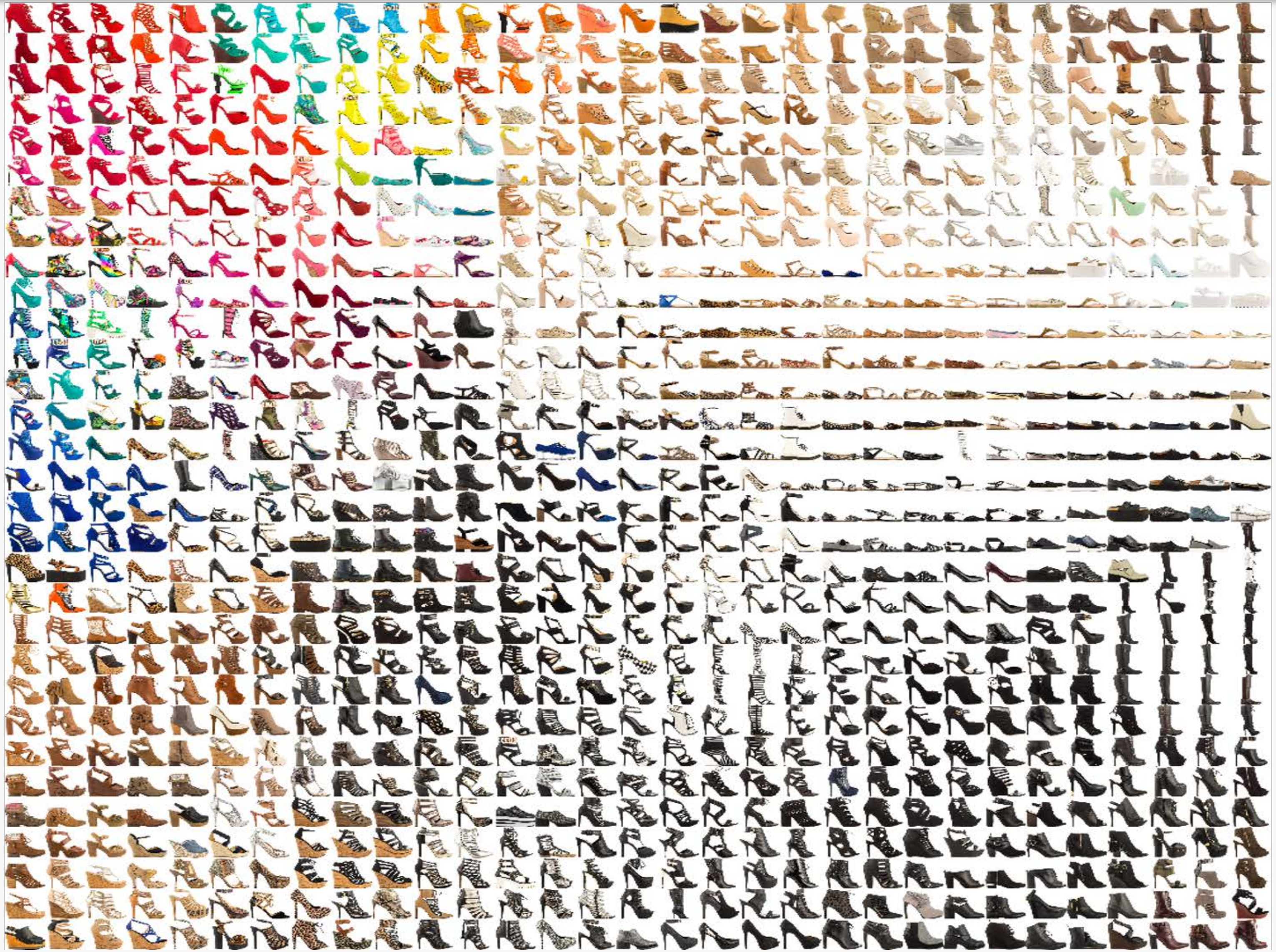
- Images sorted by similarity enables more images to be viewed simultaneously.
- Useful for stock photo agencies or e-commerce applications.
- Visually sorted grid layouts attempt to arrange images so that their proximity on the grid corresponds as closely as possible to their similarity.

256 IKEA kitchenware images

Visual sorting helps to view more images



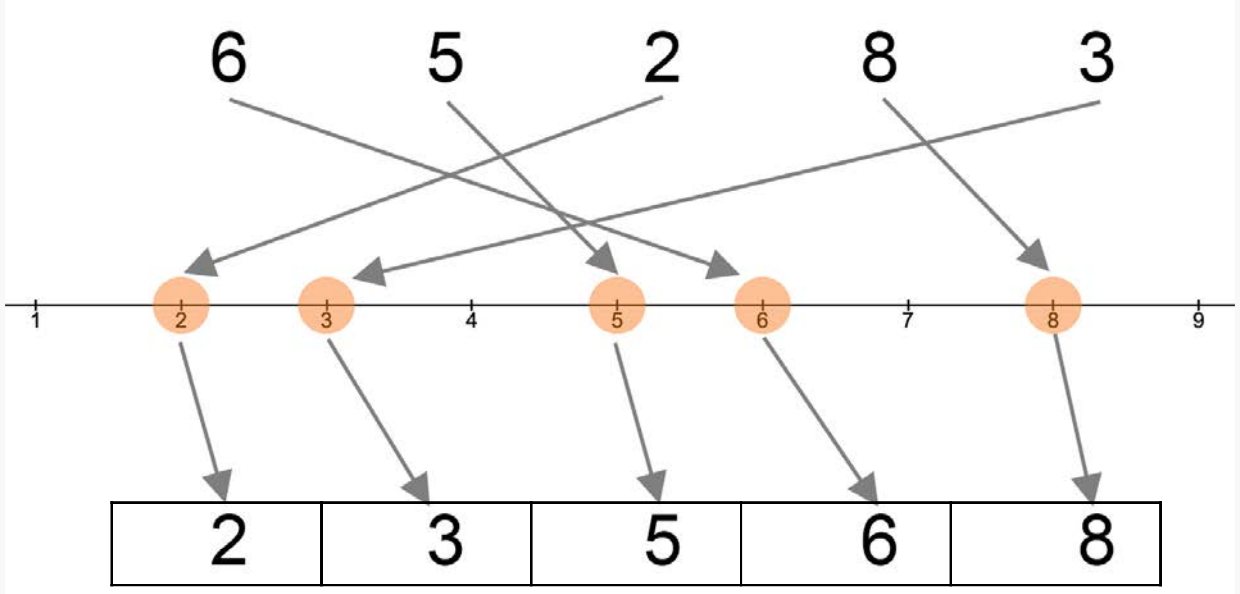
Visual sorting helps to view more images



Principle of Sorting

"Normal" Sorting

Sorting: Arranging scalars by their value =
Projecting 1D data optimally onto a line or 1D grid:



The number of possible arrangements
grows factorially with the number of data points!

"Extended" Sorting

Mapping / projecting data from

Source space → Target space

Source dimension ≥ **Target dimension**

Target space attributes:

wrapped layout (**torus**)?

no / yes

quantized positions (**grid**)?

no / yes

densely filled?

no / yes

additional constraints?

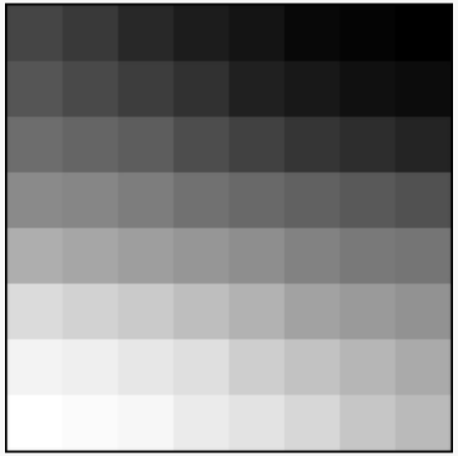
Sorting Types

- 1D → 1D
  Sorting numbers
  4, 1, 9, 3, 8

1	3	4	8	9

- 1D → 2D
  Sorting the numbers
  1-64 on a 8x8 grid
  no wrap

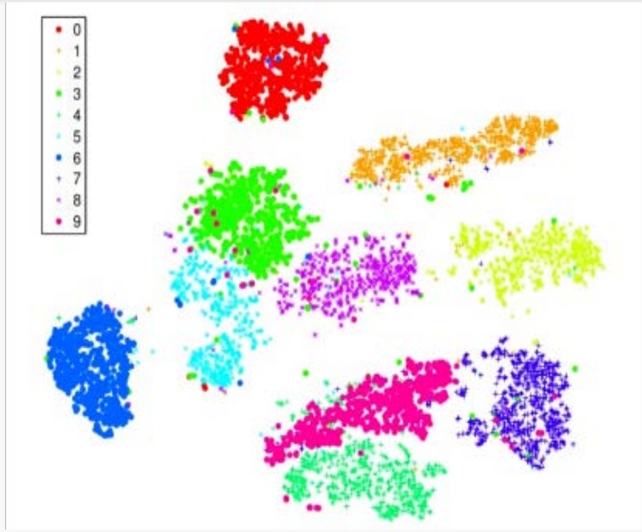
18	15	11	8	6	3	2	1
22	19	16	13	9	7	5	4
28	26	24	20	17	14	12	10
35	34	32	29	27	25	23	21
44	42	40	38	36	33	31	30
55	53	51	48	45	41	39	37
61	60	58	56	52	49	46	43
64	63	62	59	57	54	50	47



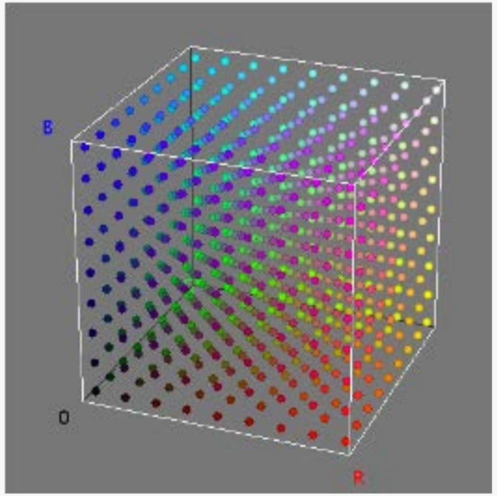
- 3D → 1D
  Sorting RGB colors
  on a line (a grid)

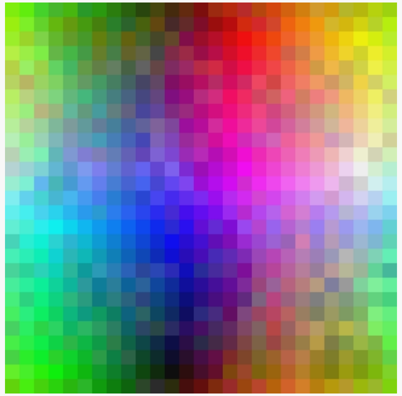
Sorting Types

- 784D → 2D
  Projecting
  MNIST images
  (28x28 pixels)
  on a 2D plane

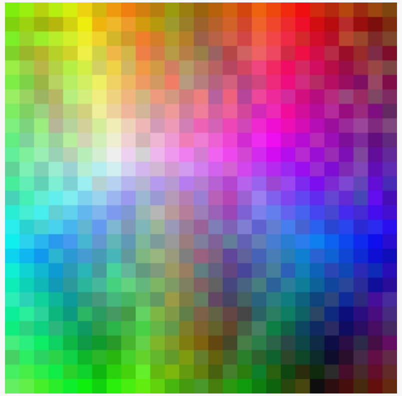


- 3D → 2D
  Sorting
  729 RGB colors
  on a 27x27 grid





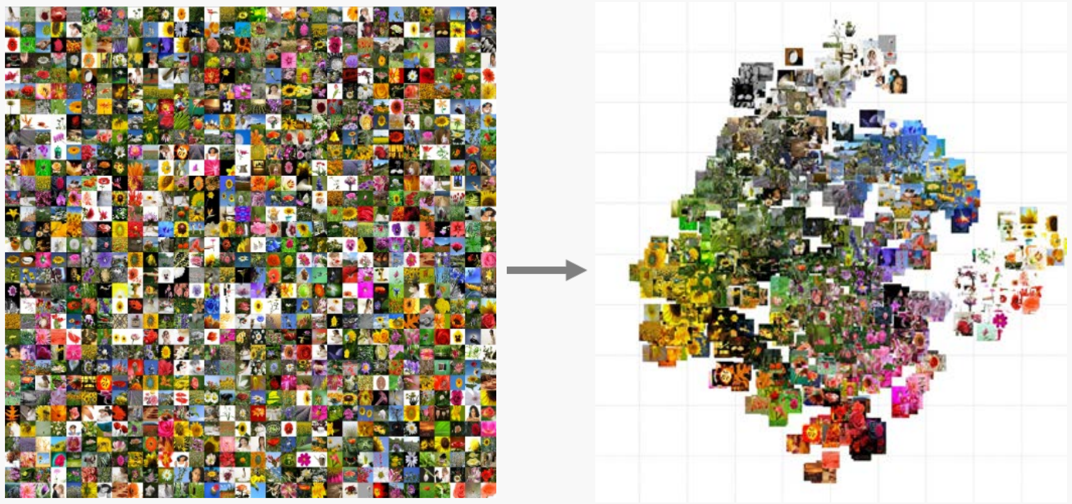
wrapped



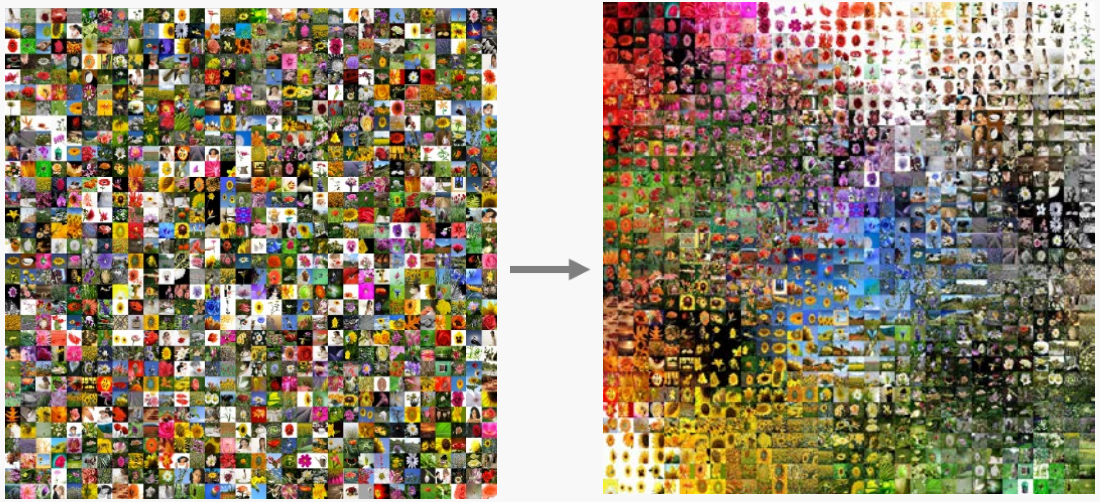
non wrapped

Sorting Types

- HD → 2D
  Projecting images onto a 2D plane



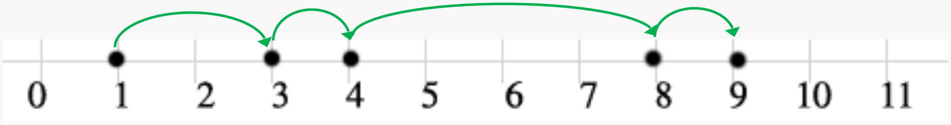
- HD → 2D
  Arranging images on a 2D grid



Evaluating Sortings I

1D Sorting:

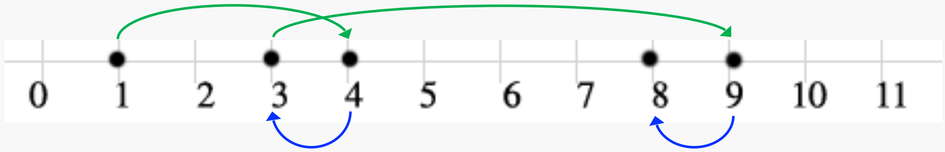
"correct" sorting: $1 \xleftrightarrow{2} 3 \xleftrightarrow{1} 4 \xleftrightarrow{4} 8 \xleftrightarrow{1} 9$



Path length
= 2+1+4+1 = 8

Incorrect sorting extends the path through the data

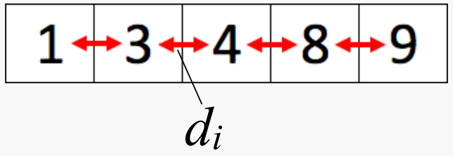
"wrong" sorting: 1, 4, 3, 9, 8



Path length
= 3+1+6+1 = 11

Evaluating Sortings II

Normal 1D Sorting (non wrapped):



Derived from the Root Mean Square (RMS)

D_p represents the average magnitude of the neighbor distances of the n data points

$$RMS = \left(\frac{1}{n}\sum_{i=0}^{n-1} e_i^2\right)^{\frac{1}{2}} \quad D_p = \left(\frac{1}{n-1}\sum_{i=0}^{n-2} \|d_i\|^p\right)^{\frac{1}{p}} \quad d_i = f_i - f_{i+1}$$

A 1D sorting is "optimal" if D_p is minimal.

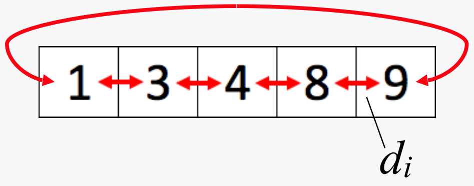
For any $p > 0$ the optimal order the same.

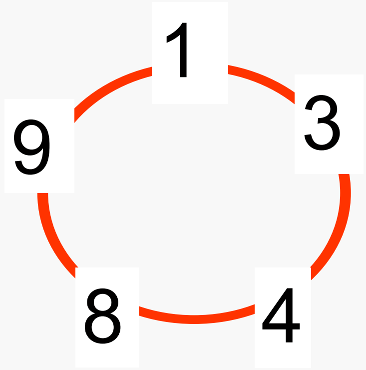
Questions:

- What happens for wrapped sortings?
  (Sorting on a torus)
- What is different for target dimensions ≥ 2?
- How to choose p?
- How to derive a general metric for
  evaluating sorted arrangements?

Sorting on a Torus

1D Sorting (wrapped):





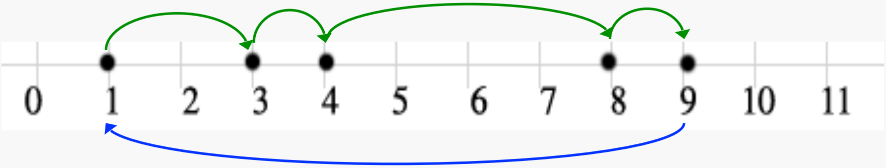
$$D_p = \left(\frac{1}{n} \sum_{i=0}^{n-1} \|d_i\|^p \right)^{\frac{1}{p}}$$

$$d_i = f_i - f_{(i+1)\%n}$$

Depending on p the "optimal" sorting differs!

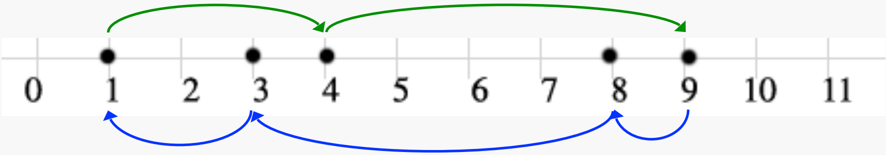
Sorting on a Torus p=1

Sorting: 1, 3, 4, 8, 9, 1



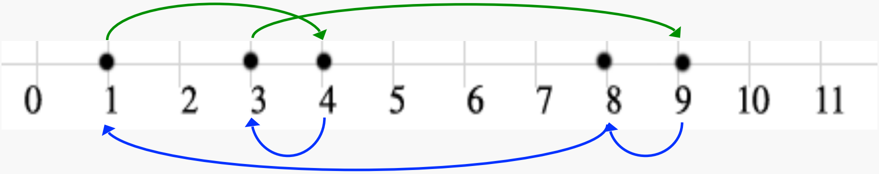
$$D_1 = (2+1+4+1+8)/5 = \boxed{3.2}$$

Sorting: 1, 4, 9, 8, 3, 1



$$D_1 = (3+5+1+5+2)/5 = \boxed{3.2}$$

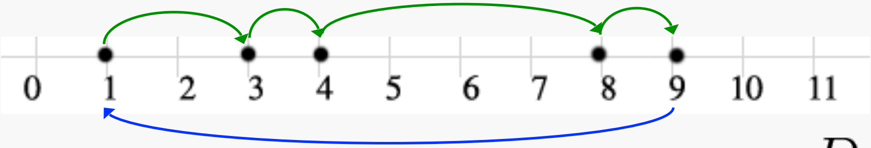
Sorting: 1, 4, 3, 9, 8, 1



$$D_1 = (3+1+6+1+7)/5 = \boxed{3.6}$$

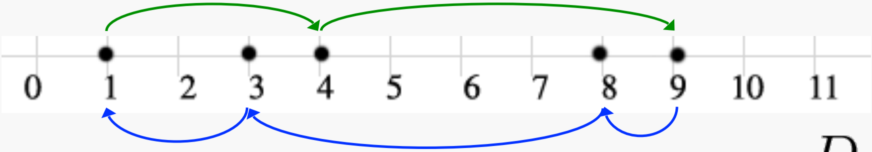
Sorting on a Torus p=2

Sorting: 1, 3, 4, 8, 9, 1



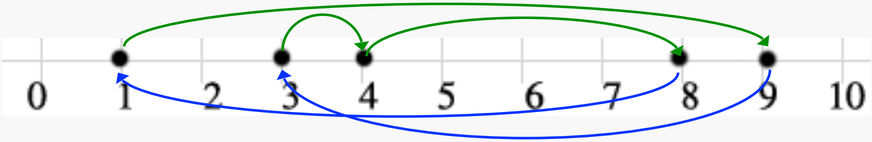
$$D_2 = (\frac{1}{5}(2^2 + 1^2 + 4^2 + 1^2 + 8^2))^{\frac{1}{2}} = 4.15$$

Sorting: 1, 4, 9, 8, 3, 1



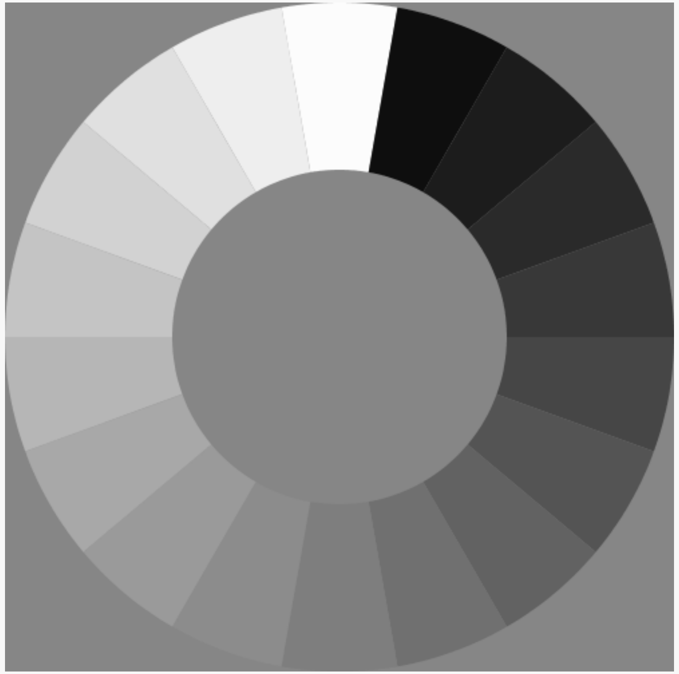
$$D_2 = (\frac{1}{5}(3^2 + 5^2 + 1^2 + 5^2 + 2^2))^{\frac{1}{2}} = \boxed{3.58}$$

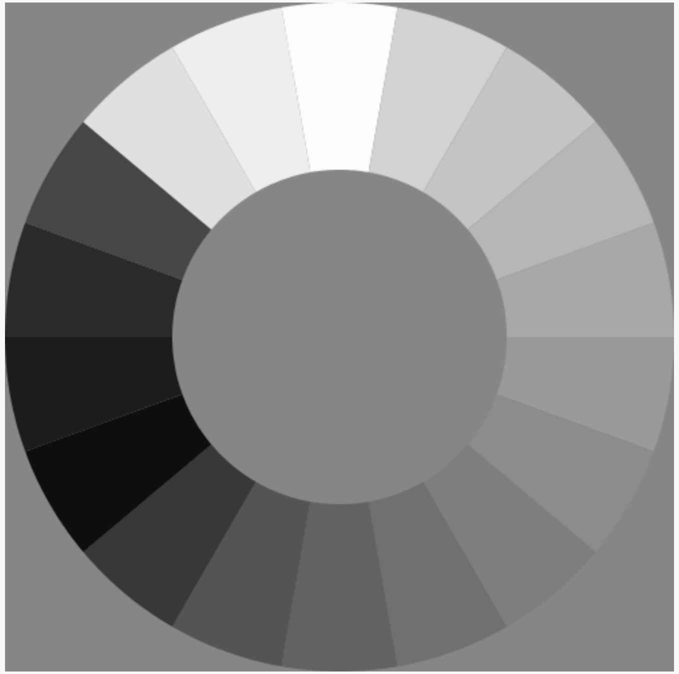
Sorting: 1, 9, 3, 4, 8, 1

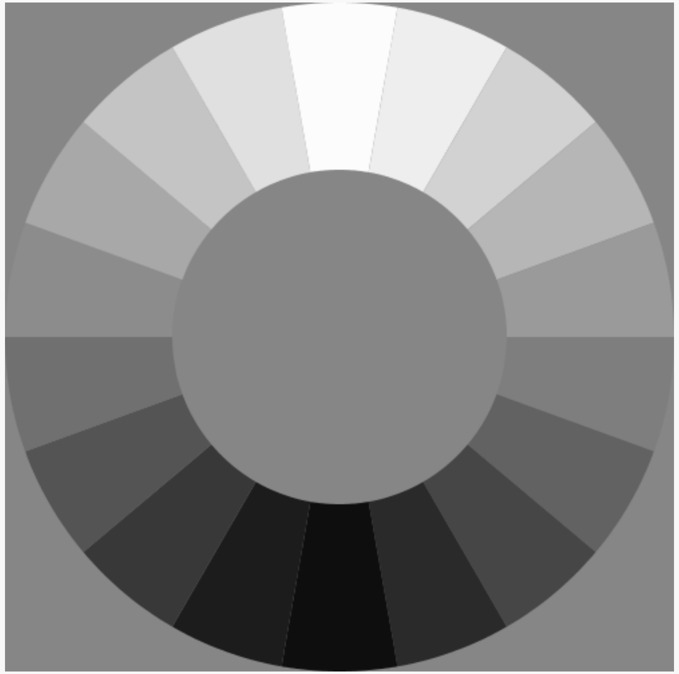


$$D_2 = (\frac{1}{5}(6^2 + 1^2 + 4^2 + 7^2 + 8^2))^{\frac{1}{2}} = \boxed{5.76}$$

Sorting on a Torus



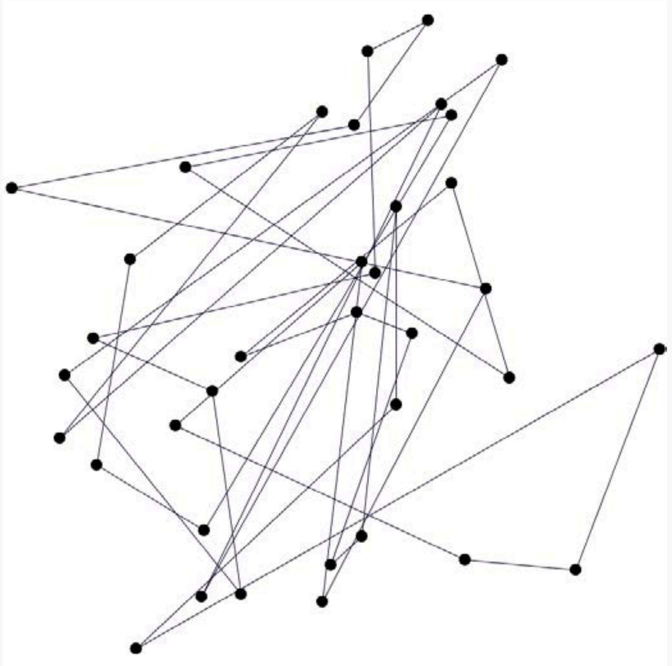


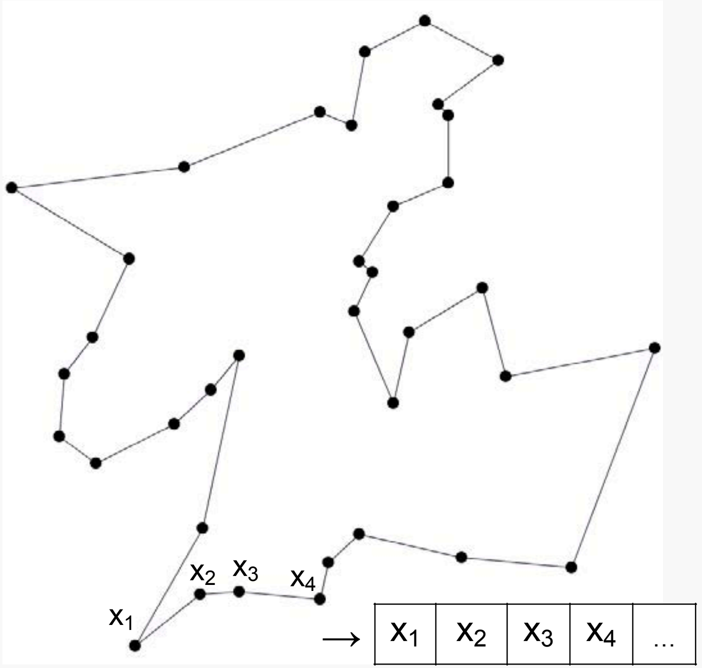


$D_{0.5} = 19.3$	$D_{0.5} = 22.2$	$D_{0.5} = 26.2$
$D_1 = 26.4$	$D_1 = 26.4$	$D_1 = 26.4$
$D_2 = 57.7$	$D_2 = 41.7$	$D_2 = 26.8$

Extension to higher source dimensions

- Wrapped sorting of high-dimensional (HD) data.
- Meaningful 1D sorting is produced by minimizing D_1 or D_2, i.e. the path along the points in HD.
- 2D example: Traveling salesman problem (closed-loop)



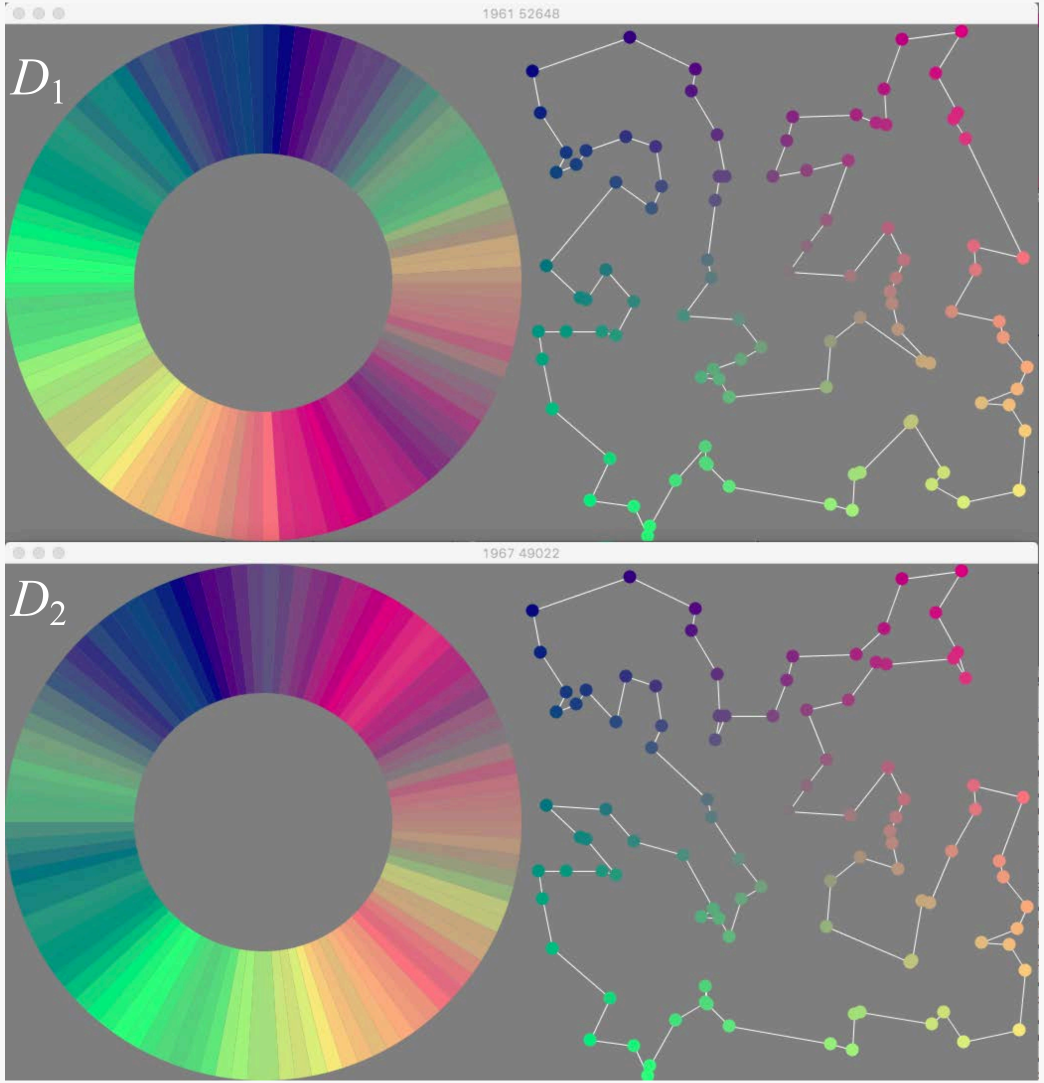


1D Torus sorting of 2D colors

Colors: R, G, B=128

Optimal paths in terms of D_1 and D_2
No fundamental difference of sorting.

D_2 has a preference for shorter local distances.



1D Sorting quality

- must represent how well the optimal (sorting) order is preserved.
- Sorting quality could be defined as

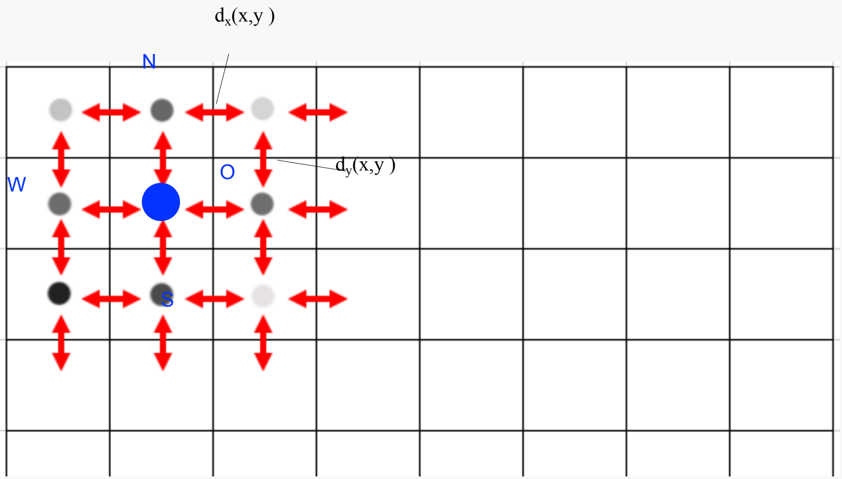
$$\frac{\bar{D} - D_{sort}}{\bar{D} - D_{opt}} \quad [-1..1]$$

$\bar{D}$ = mean distance of all data points (p omitted)

- Optimal sorting → 1
  Random sorting → 0
  (worse than random < 0)
- Problem:
  D_{opt} is difficult to determine for source dimensions > 1
  → Traveling Salesman Problem

2D Grid Sorting quality

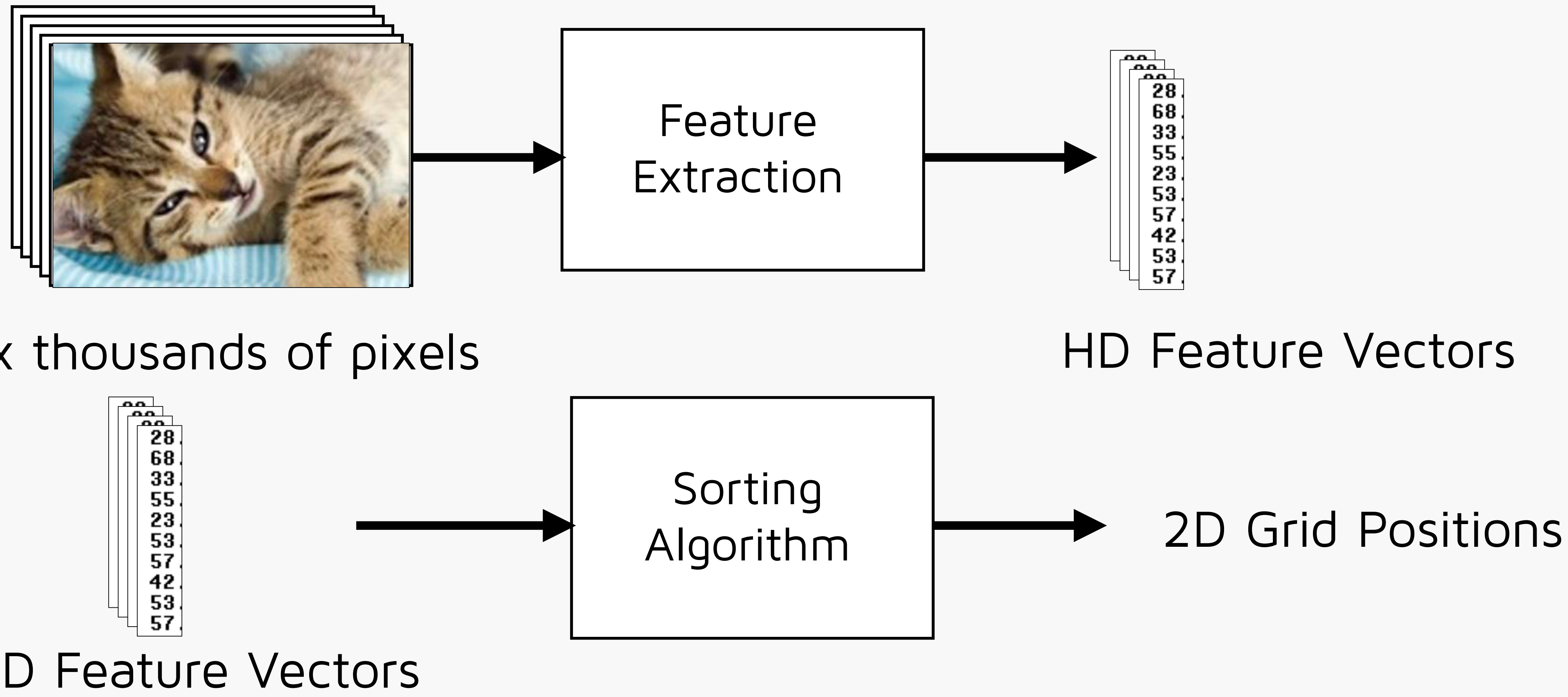
- For 2D target dimensions
  it gets worse ...



- If D_{sort} is minimal
  (average magnitude of the distances
  to all 4 neighbors), then the sorting is optimal.
- Again, since the optimal 2D sorting is not known, the sorting quality cannot be determined in the previously proposed way. :(
- We will present a solution later in this tutorial ...

2D Image Sorting

If images are to be sorted on a grid, feature vectors are needed.



Visual Feature Vectors

Visual Feature Vectors

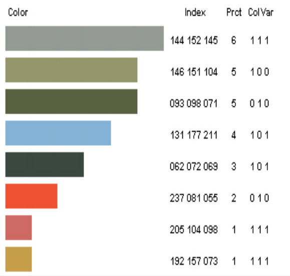
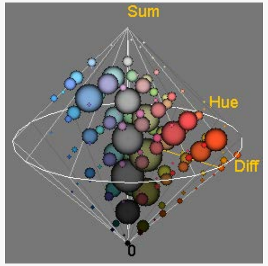
Representing images as vectors is essential for sorting them, but finding universally applicable "good" feature vectors is an ongoing research area.

Two types of feature vectors:

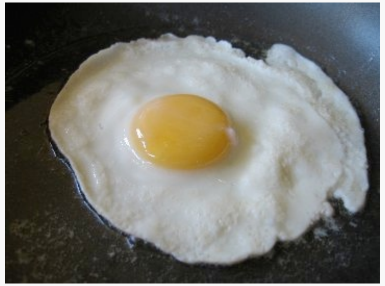
- **Low-level feature vectors**
  describe visual appearance and
  are effective for grouping images.
- **Deep learning feature vectors**
  describe image content and
  are useful for image retrieval.

Low-Level Feature Vectors

describe visual image features like colors, textures or edges.

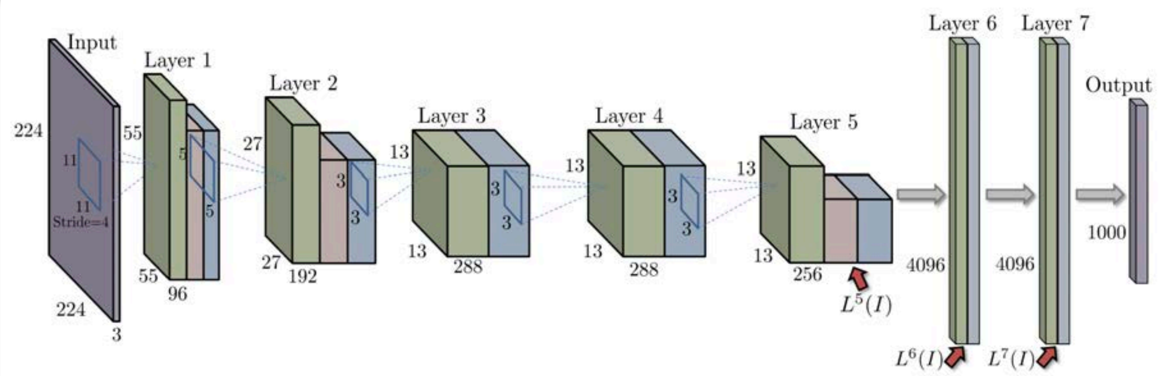
<table>
<tr><td>Descriptor</td><td>Color Layout</td><td>Dominant Color</td><td>Color Structure</td></tr>
<tr><td>Dimension of feature vectors</td><td>3 -192</td><td>9 – 24
(depending on image)</td><td>32-256</td></tr>
<tr><td>Example
</td><td></td><td>
</td><td>
</td></tr>
</table>

These "primitive" features often provide poor retrieval results due to the "semantic gap".



→

Deep Learning Feature Vectors



Deep learning feature vectors enable the extraction of semantically meaningful representations from images, for tasks such as image search, similarity comparisons, and image classification.

- 2014: Using Activations of Neural Networks: Babenko et al., "Neural Codes for Image Retrieval"
- 2016: End-to-End Fine-tuning for Retrieval: Gordo et al., "Deep image retrieval: Learning global representations for image search"
- 2018: GeM Pooling: Radenovic et al., "Fine-tuning CNN image retrieval with no human annotation"

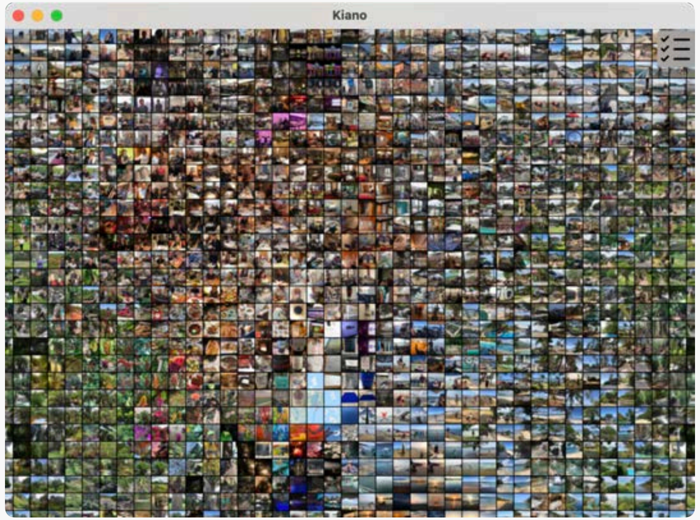
Deep Learning vs Low-Level Feature Vectors

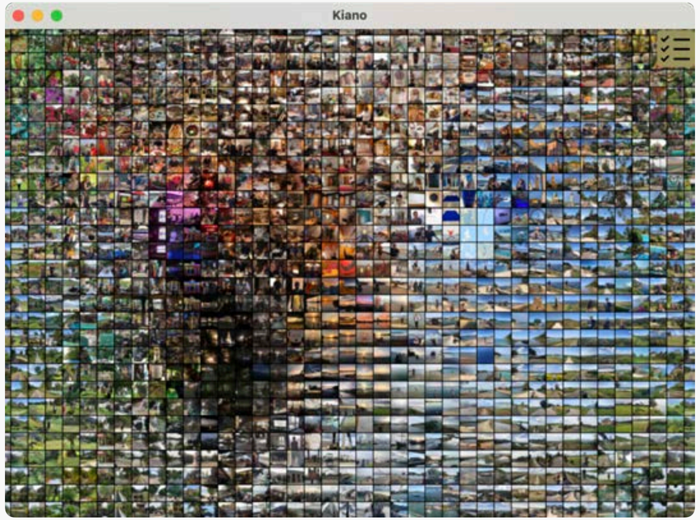
Images sorted using deep learning feature vectors

Images sorted using low-level feature vectors

Image Feature Vectors for Image Sorting

- User-friendly image overview is prioritized over semantic separation when sorting similar topic images or search results.
- Effective image sorting requires feature vectors that capture both semantic and visual aspects.
- Generating a well-structured overview becomes more crucial as the number of displayed images increases.





Dimensionality Reduction

Motivation for Dimensionality Reduction

- Data representation with fewer dimensions
  - Data compression
  - Feature extraction
- Insight into high-dimensional data
  - Visualization of HD data
  - Sorting/arrangement based on similarities of high-dimensional feature vectors or images …
  - Dimensionality reduction schemes for visualization purposes aim to capture and preserve the inherent relationships within the data by representing it in two or three dimensions.
  - None of these schemes are grid-based.

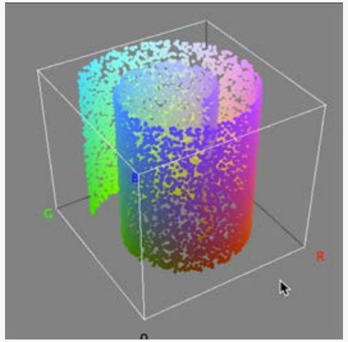
Linear and Nonlinear Techniques

- *(Scatterplot Matrix)*
- *Principal component analysis (PCA)*

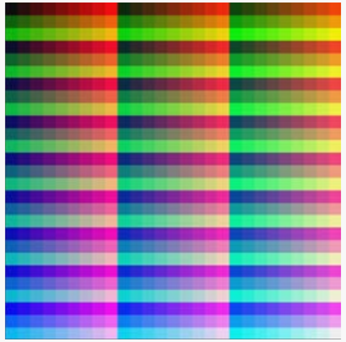
- Multidimensional scaling (MDS)
- Isomap
- Local-linear embedding (LLE)
- t-Distributed Stochastic Neighbor Embedding (t-SNE)

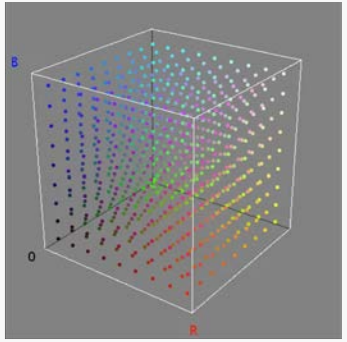
Demonstration Test Sets

- Swiss Roll data RGB colors
  3D



- Cube 9x9x9 RGB colors
  3D

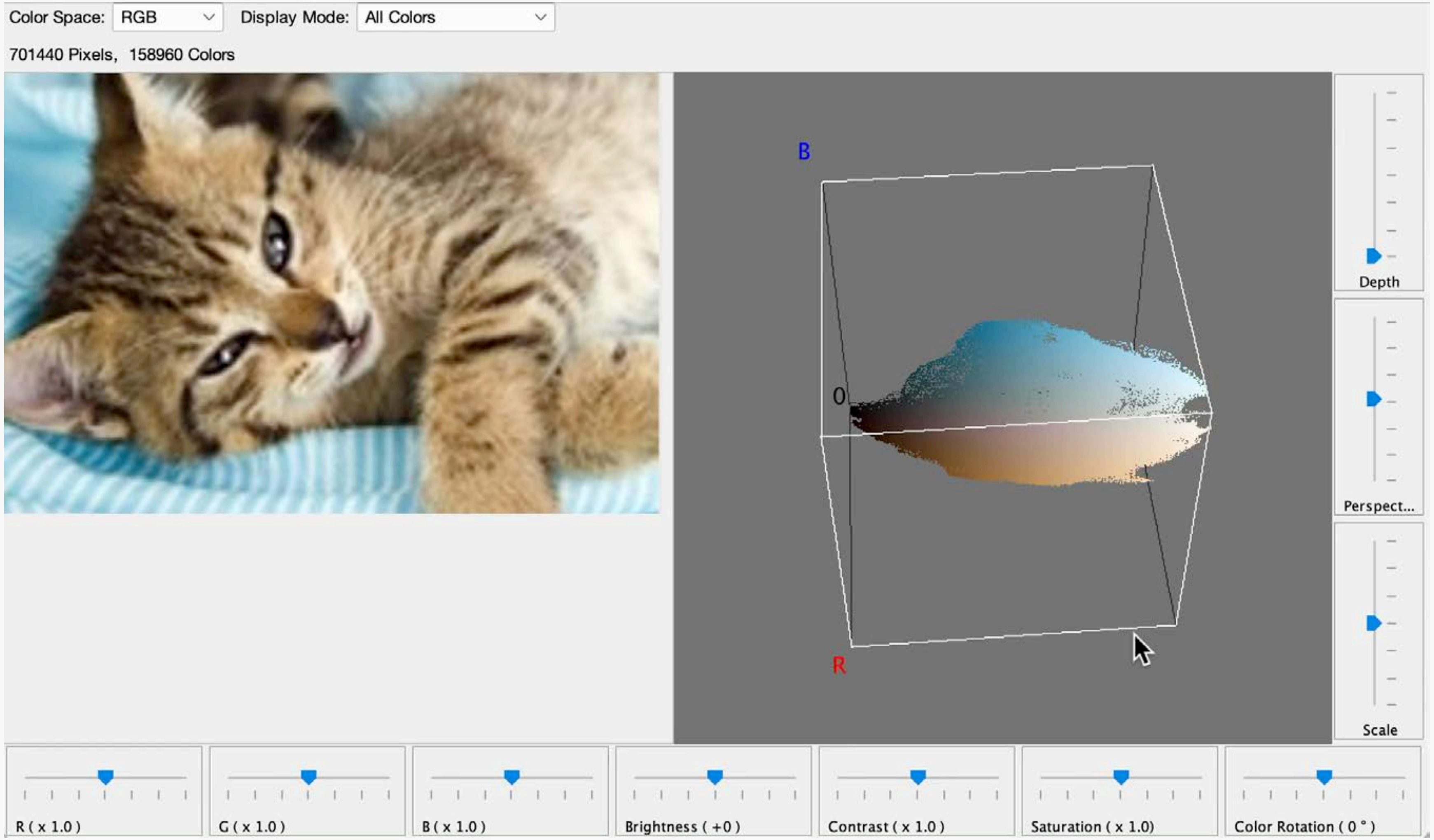




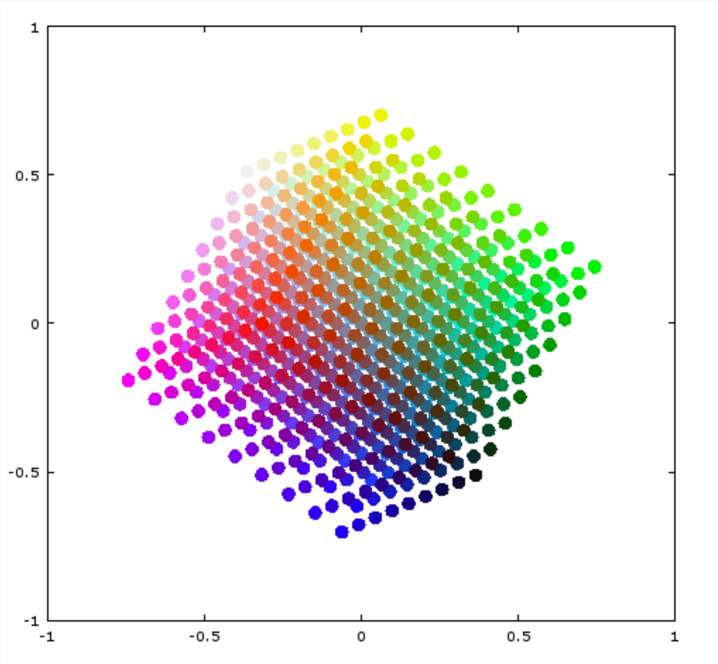
- MNIST data,
  images with 28x28 pixels
  784D

Principal component analysis (PCA)

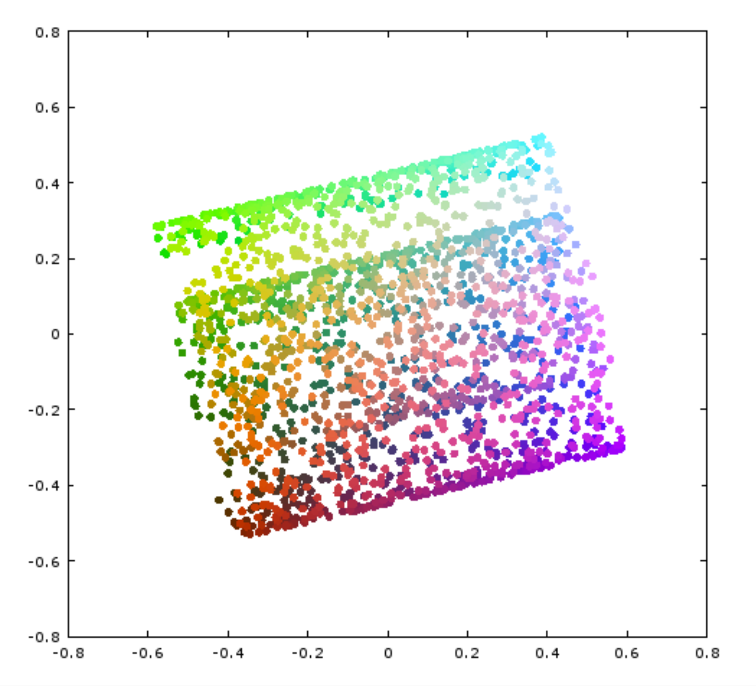
Optimal linear projection that maximizes the variance of the kept dimensions



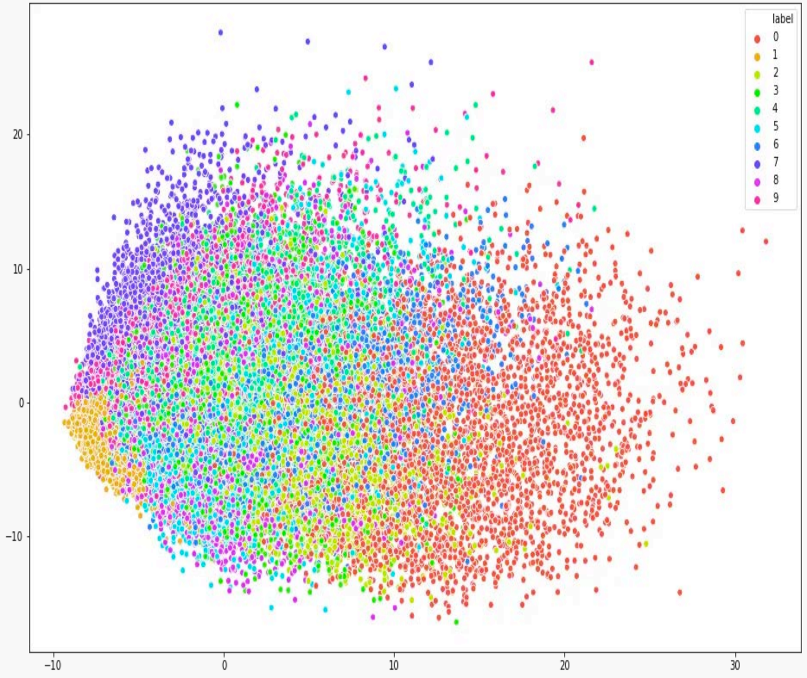
Principal Component Analysis (PCA)



RGB cube data ☹️



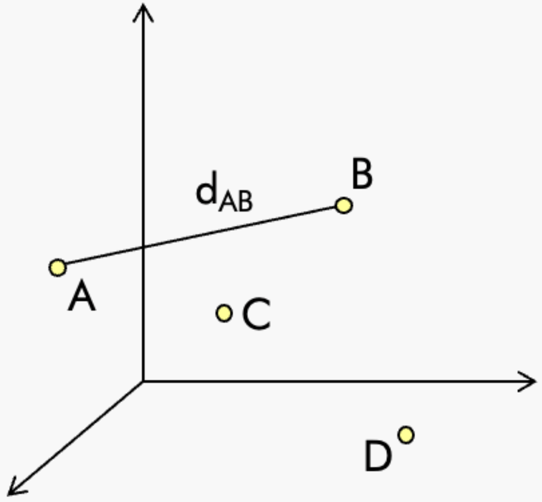
Swiss roll data ☹️

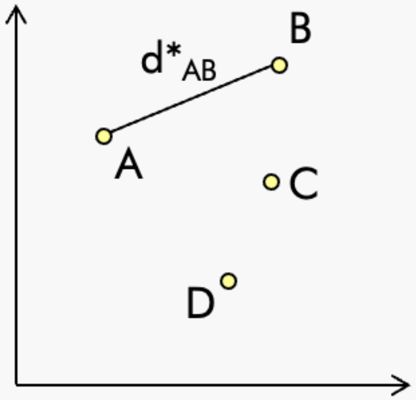


MNIST ☹️

Multidimensional Scaling (MDS)

- Idea: Projection of high-dimensional data while preserving the distances between the data points.
- Iterative comparison between the spatial distance of the projection (disparity) d^* and the actual distance d.
- "Stress" value describes the quality of the projection.





$$STRESS = \sqrt{\frac{\sum\left(d - d^*\right)^2}{\sum d^2}}$$

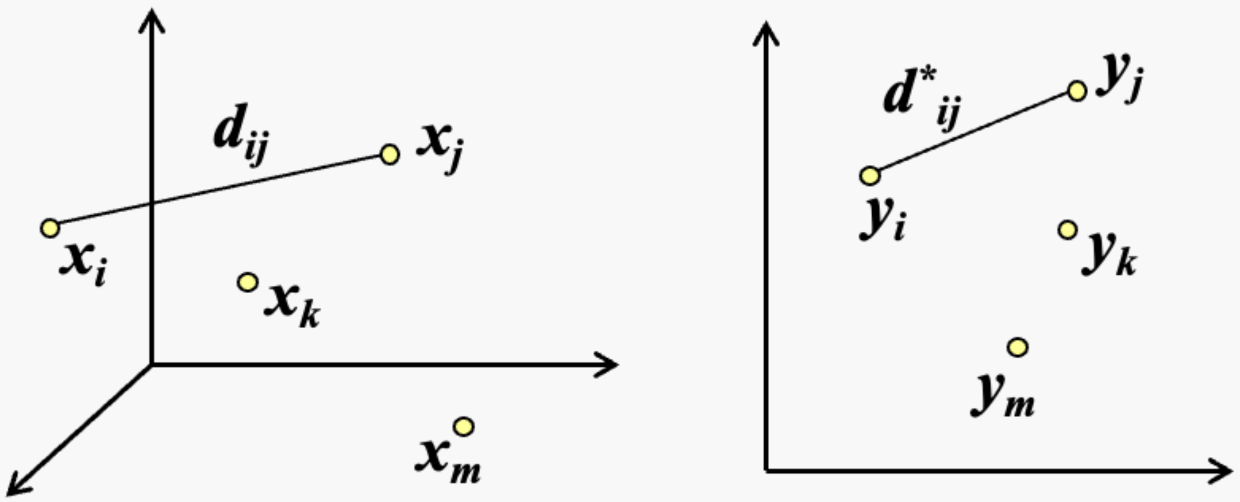
Multidimensional Scaling (MDS)

Loss Function:

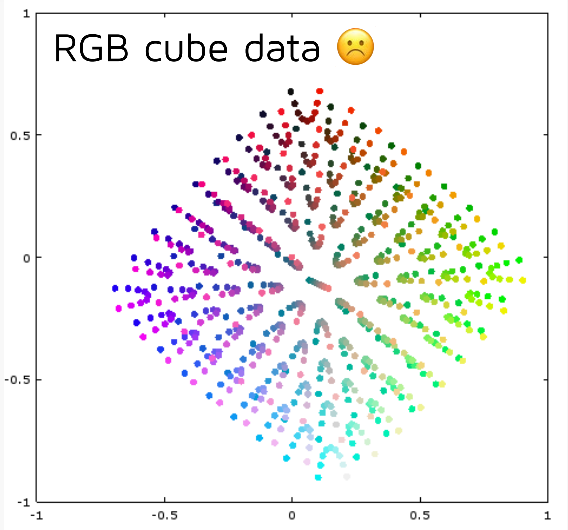
$$\phi(Y) = \sum_{ij} (\|x_i - x_j\| - \|y_i - y_j\|)^2$$

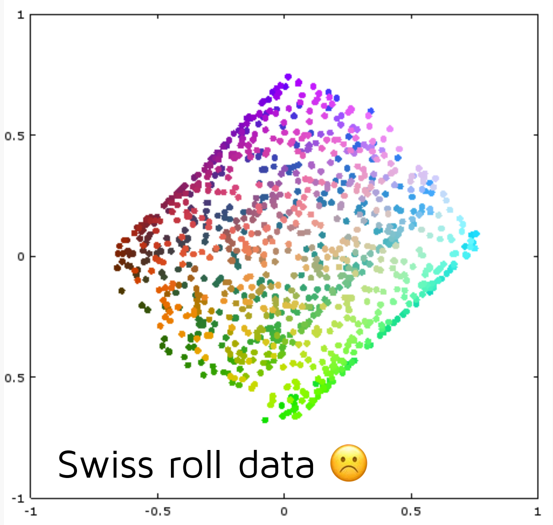
$$\phi(Y) = \frac{1}{\sum_{ij} \|x_i - x_j\|} \sum_{i \neq j} \frac{(\|x_i - x_j\| - \|y_i - y_j\|)^2}{\|x_i - x_j\|}$$

Sammon Mapping



Multidimensional Scaling / Sammon Mapping

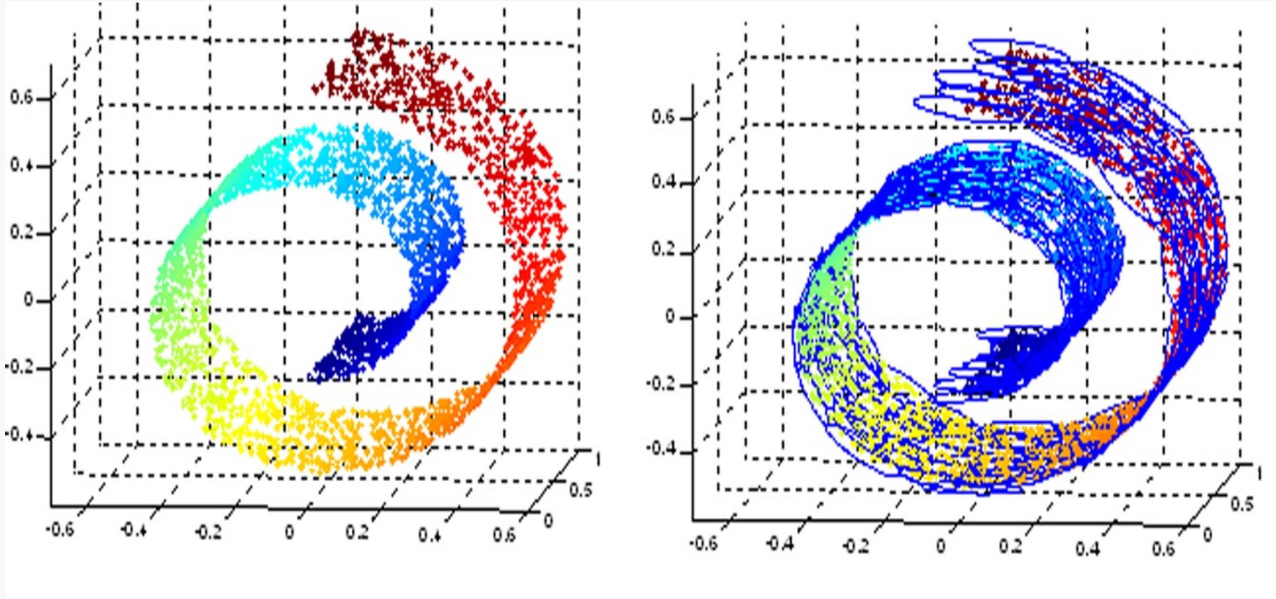


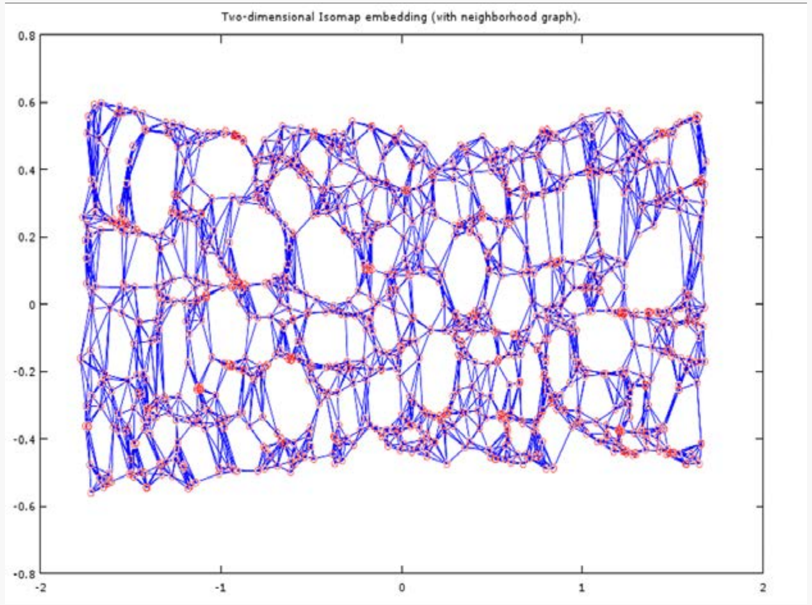


MNIST 😐

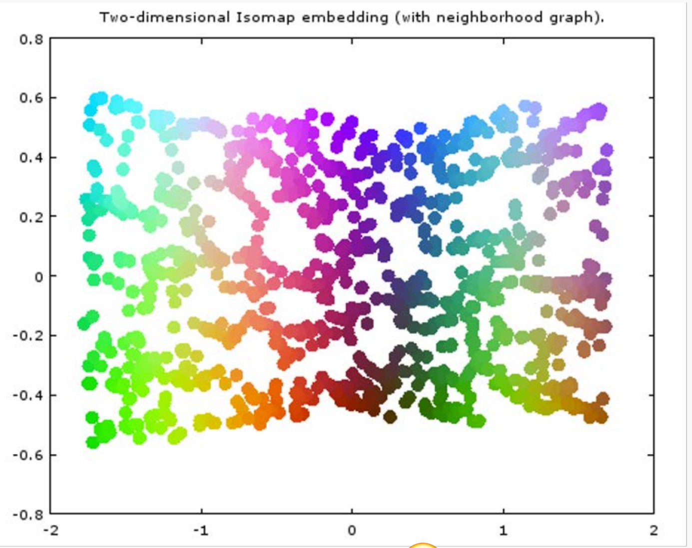
Isomap

- Constructs a graph representation based on the k nearest neighbors of each data point.
- Computes geodesic distances along the graph.
- Applies multidimensional scaling to find a lower-dimensional configuration.

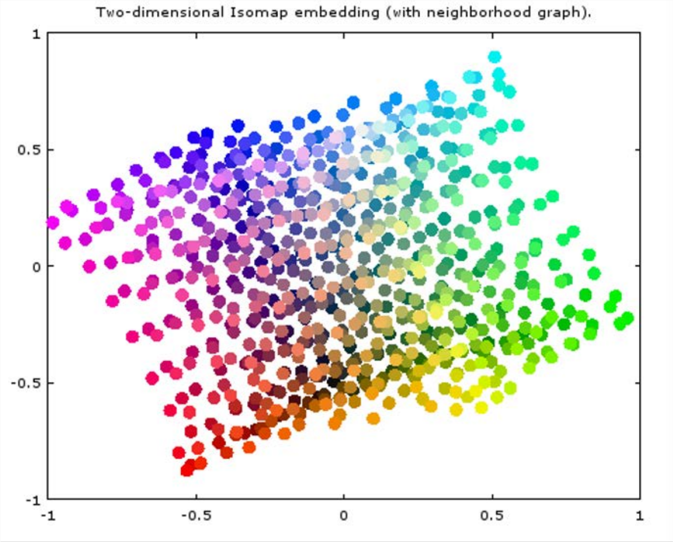




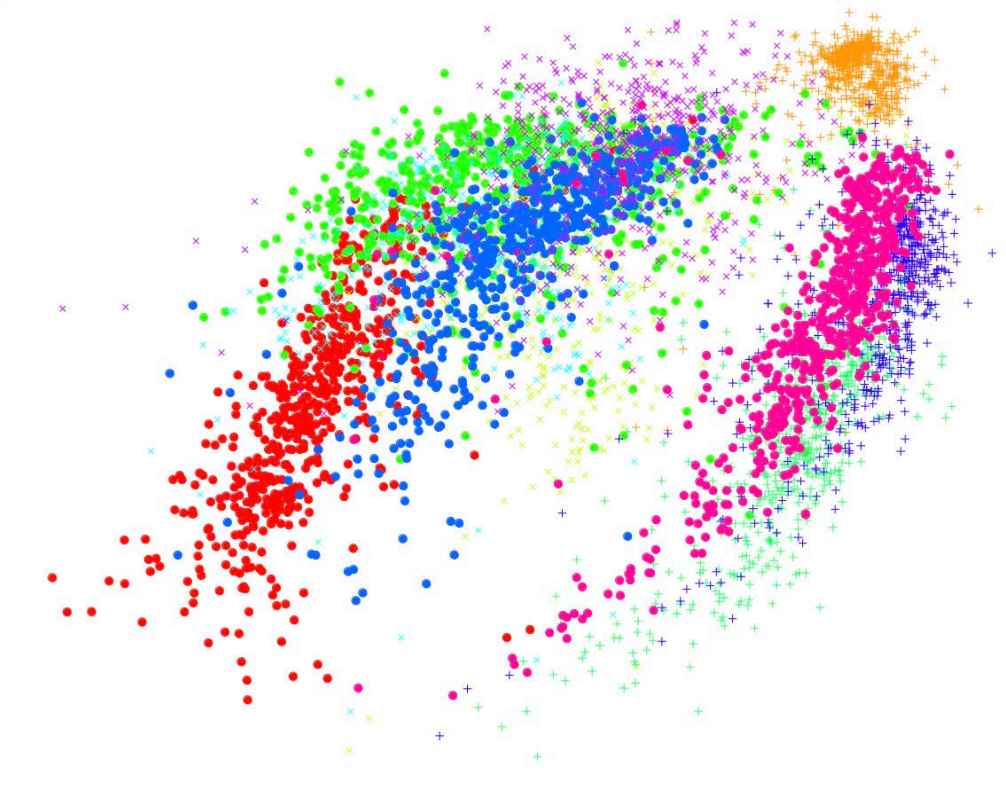
Isomap



swiss roll data 🙂



RGB cube data ☹️



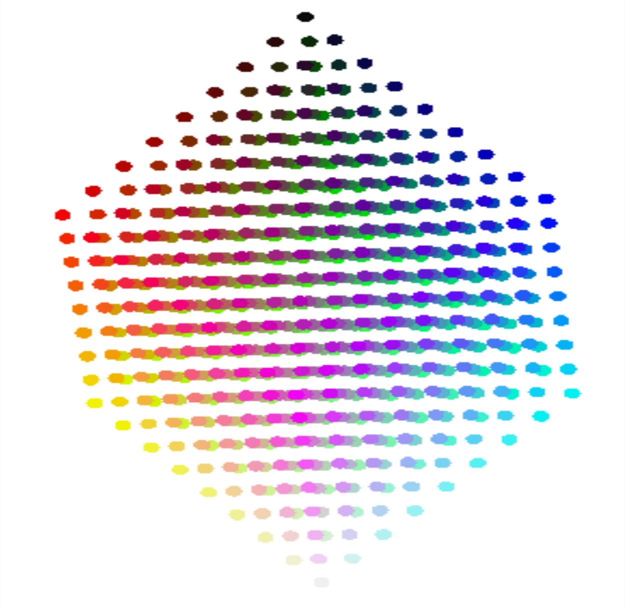
MNIST 😐

Local Linear Embedding (LLE)

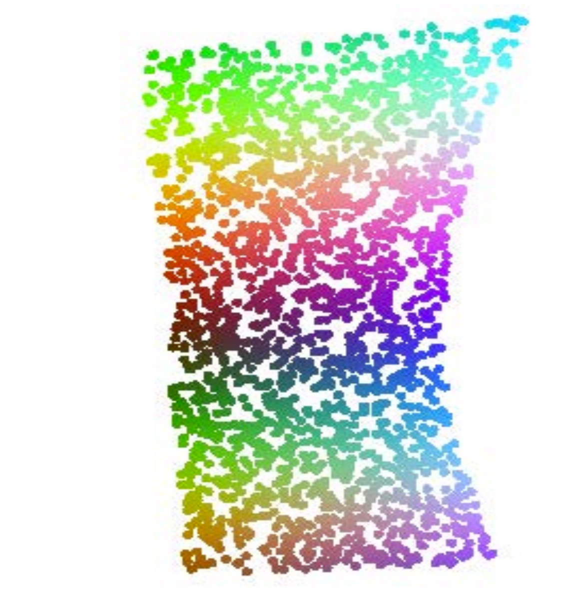
- Preserves local linear relationships in the data during dimensionality reduction.
- Reconstructs each data point as a linear combination of its neighboring points.
- Finds weights that minimize the reconstruction error.
- Constructs a lower-dimensional representation while preserving pairwise distances.

$$\phi(Y) = \sum_i (y_i - \sum_{j=1}^{k} w_{ij} y_{i_j})^2$$

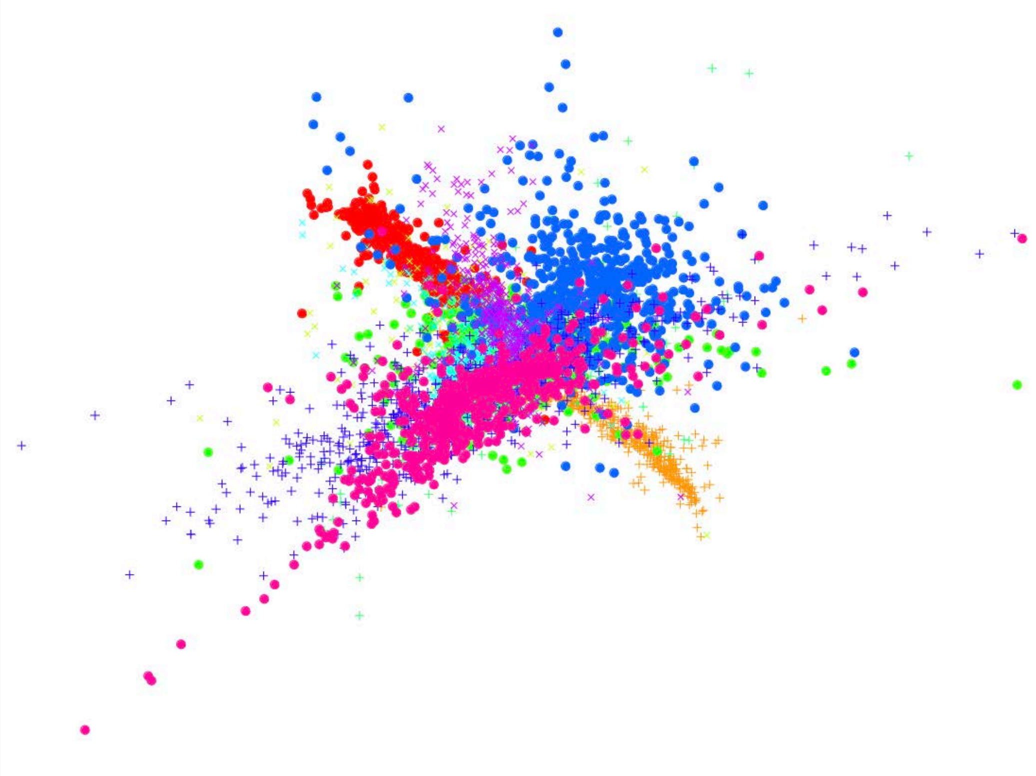
Local Linear Embedding (LLE)



RGB cube data ☹️



Swiss roll data 🙂



MNIST ☹️

t-Distributed Stochastic Neighbor Embedding

- Conversion of high-dimensional distances into conditional probabilities representing similarities.

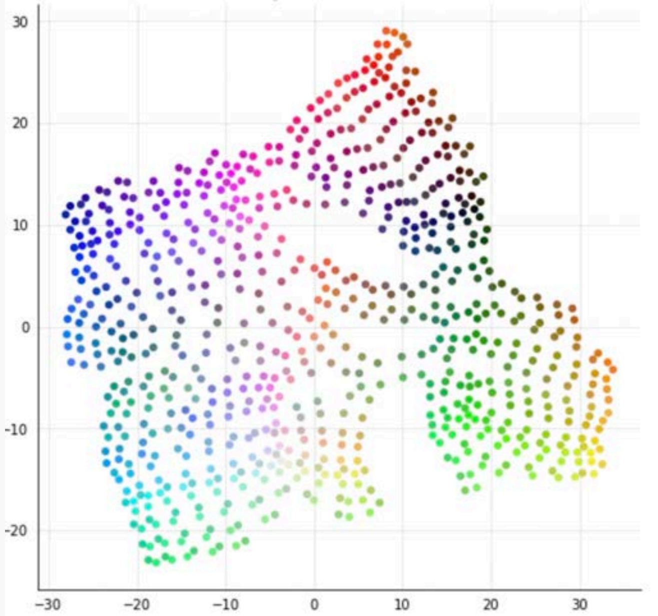
- Similarities of the data points:
  High Dimension: Low Dimension:

$$p_{ij} = \frac{e^{-\frac{\left\|x_i - x_j\right\|^2}{2\sigma^2}}}{\sum_{k\neq l} e^{-\frac{\left\|x_k - x_l\right\|^2}{2\sigma^2}}} \qquad q_{ij} = \frac{\left(1+\left\|y_i - y_j\right\|^2\right)^{-1}}{\sum_{k\neq l}(1+\|y_k - y_l\|^2)^{-1}}$$

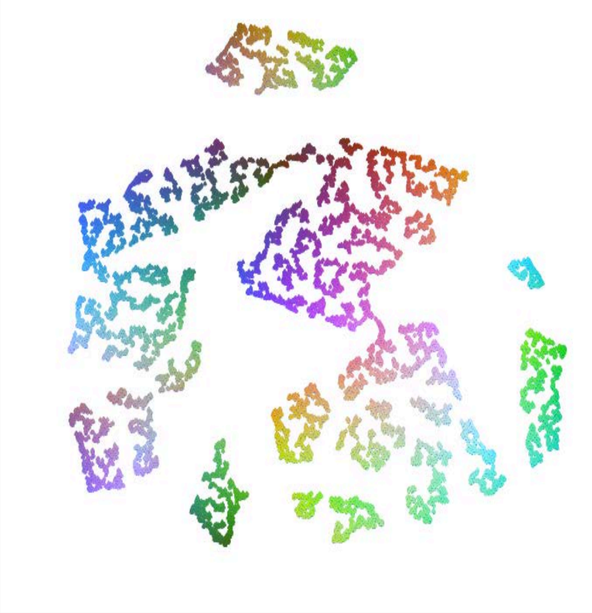
- Loss function $C = KL(P||Q) = \sum_i \sum_j p_{ij} log \frac{p_{ij}}{q_{ij}}$

- Optimization through Gradient Descent

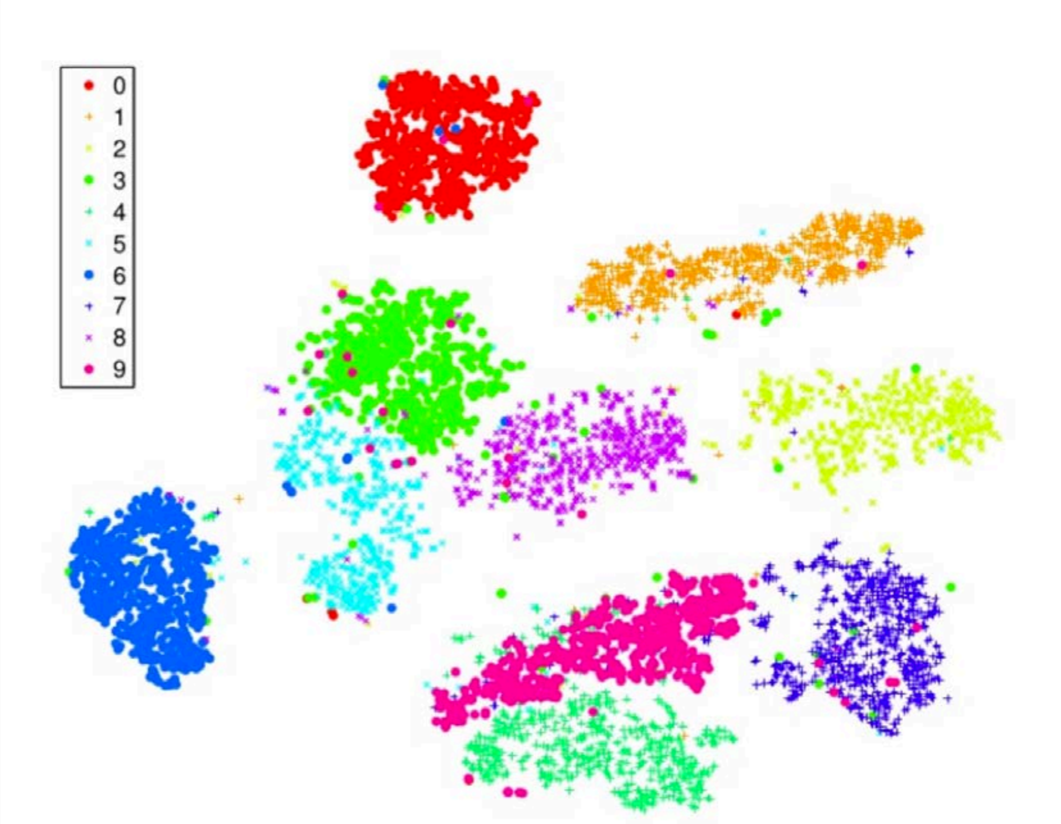
t-Distributed Stochastic Neighbor Embedding



RGB cube data 🙂



Swiss roll data 😐



MNIST 🙂

Limitations of dimensionality reduction techniques

- Most algorithms are rather slow.
- Not suited for arranging images. Due to the dense positioning of the projected images, some overlap and are partially obscured.
- Only a fraction of the display area is used.

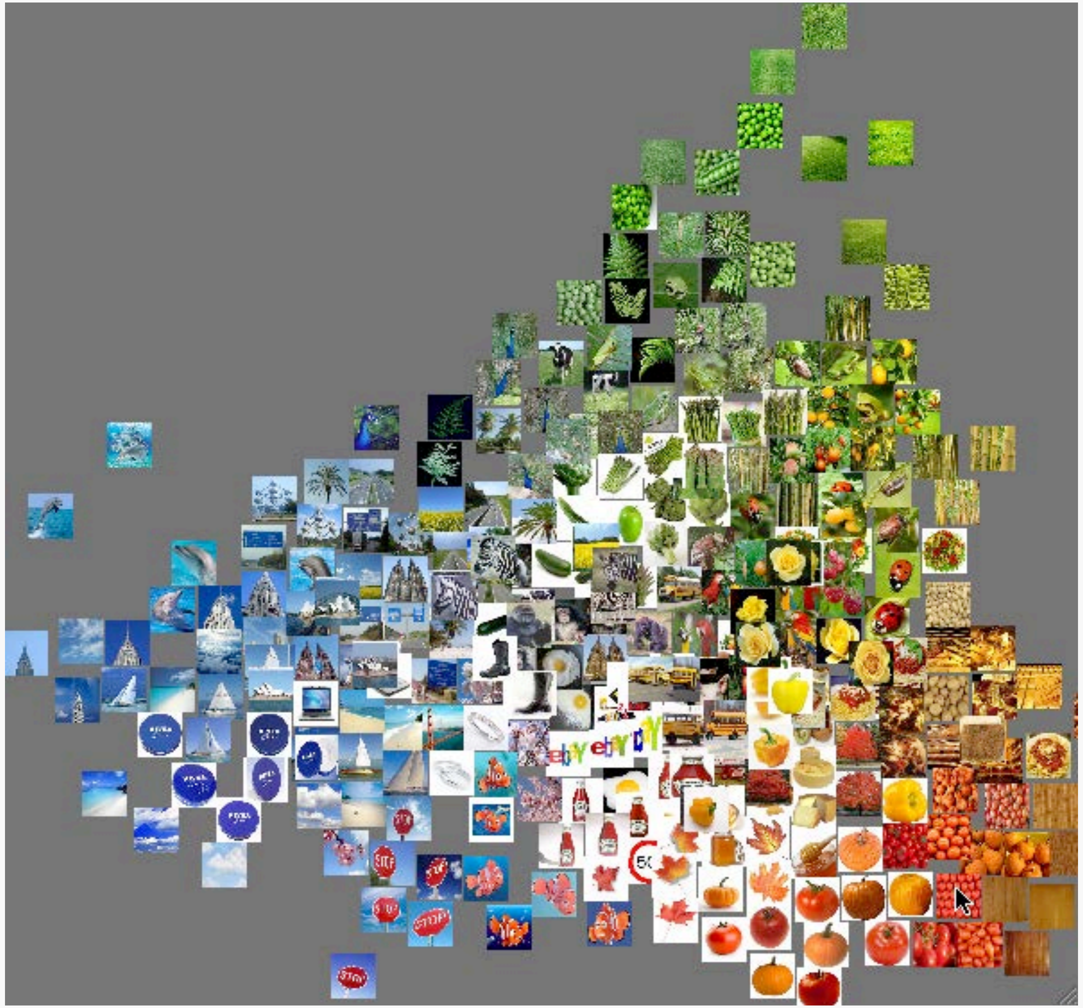


Image Sorting

-

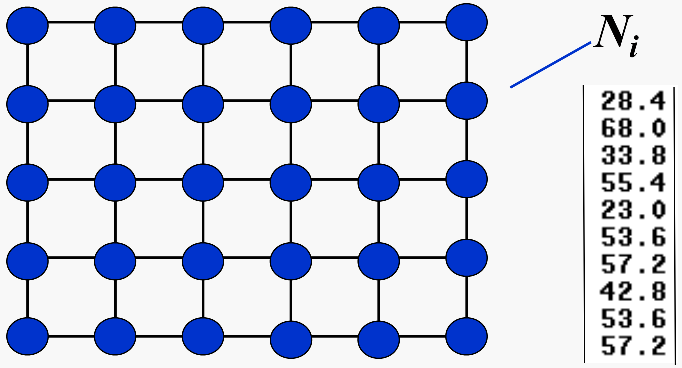
Visually Sorted Grid Layouts

Requirements & Main Algorithms

- In order to avoid overlapping images,
  a grid-based approach must be used.
- Each grid position may only be "occupied" by one image.
  → The number of grid positions size must be ≥ than the number of images.
- Main Algorithms
  - Self Organizing Maps (SOM)
  - Self-Sorting Maps (SSM)
  - IsoMatch
  - "Dimensionality Reduction to Grid"
  - Neural networks for learning permutations

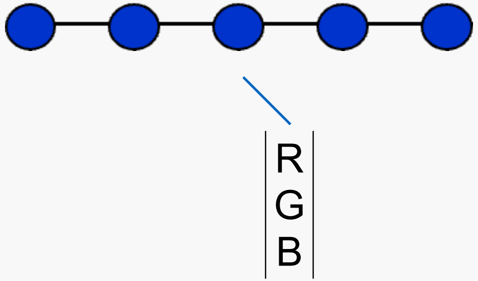
Self Organizing Map (SOM)

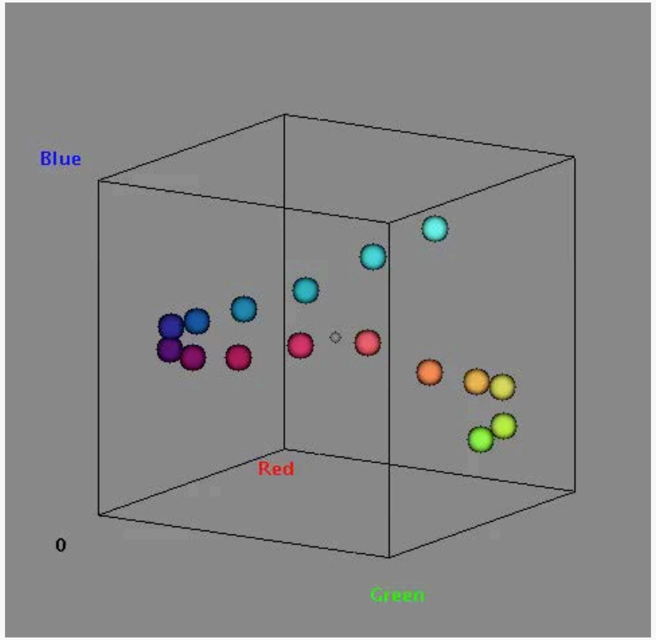
Kohonen's idea:
Use a low-dimensional network (grid):
a 1D, 2D or 3D map
of high-dimensional nodes



Adapt the nodes of the map to the high-dimensional data

Example:
Mapping of colors to a line
3D ➞ 1D

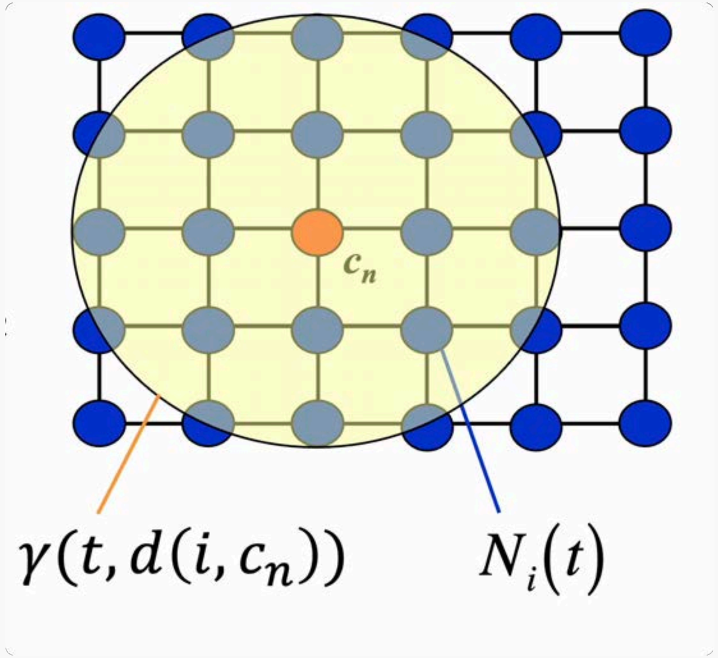




Self Organizing Map (SOM)

1. Map each feature vector X_n to the map with nodes $N_i(t)$: Search for the best representation c for X_n.

$$c_n(t) = \arg\min_i \left(\left\| \mathbf{X}_n - \boldsymbol{N}_i(t) \right\| \right)$$

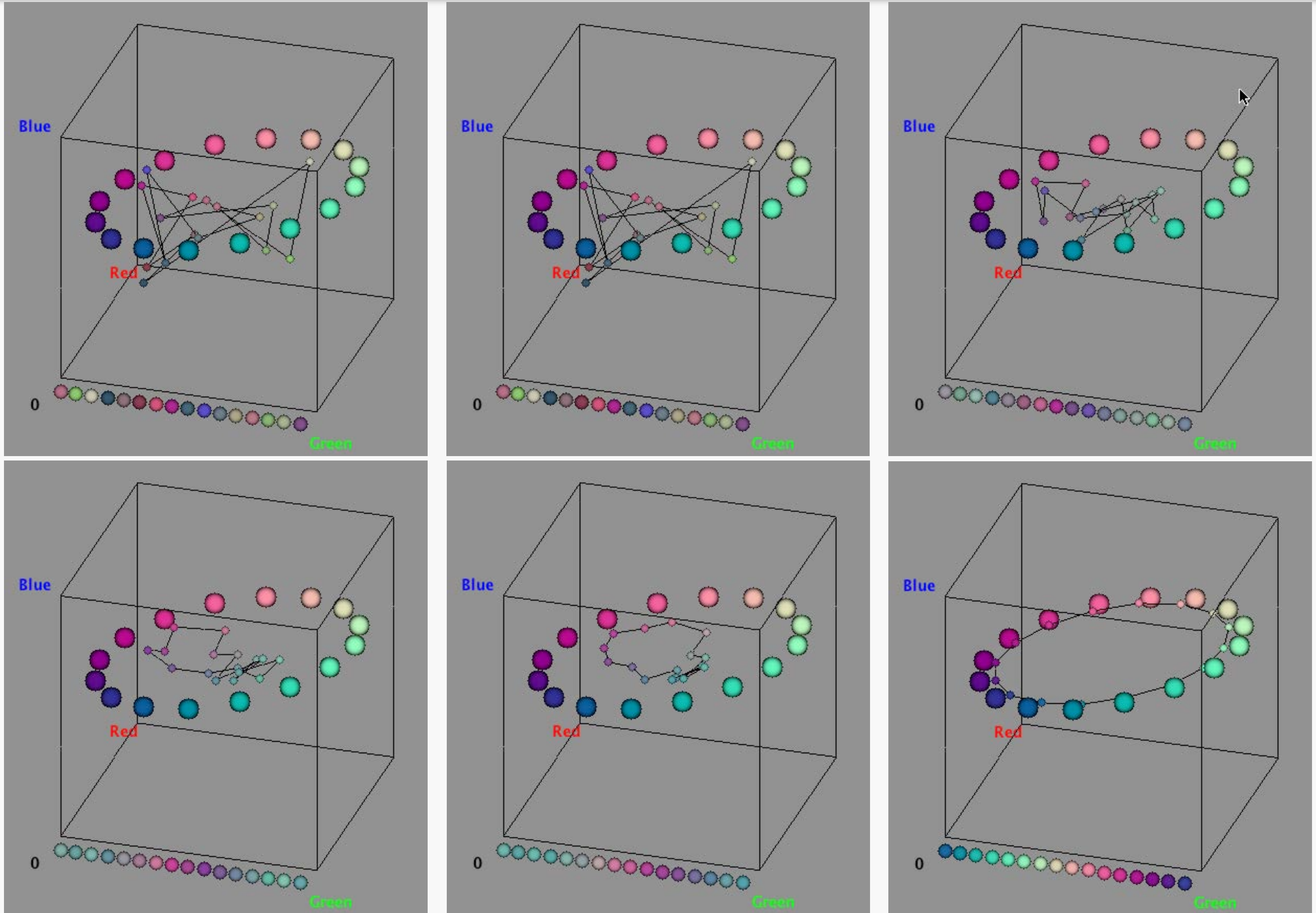


2. Update the neighborhood:

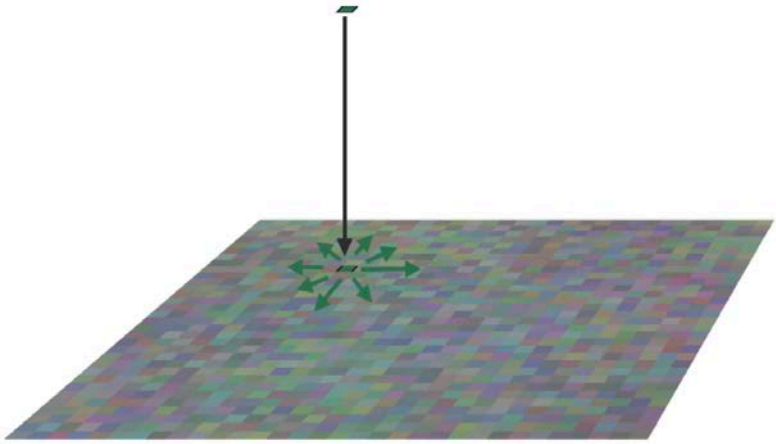
$$N_i(t+1) = N_i(t) + \alpha(t) \cdot \gamma\big(t, d(i, c_n)\big) \cdot (X_n - N_i(t))$$

Iterate with decreasing learning rate α and neighborhood function γ

Self Organizing Map (SOM)



Self Organizing Map



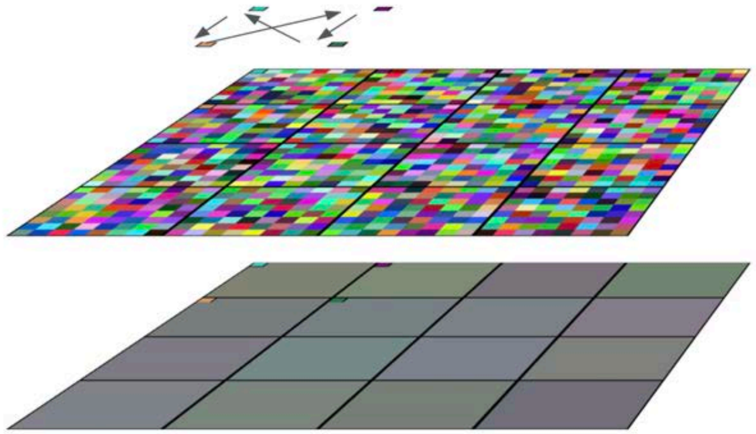
For images no node may be occupied by more than one feature vector (image).

Map size must be ≥ than the number of images.

Algorithm 1 SOM

1: Initialize all map vectors with random values,
   set learning rate α (< 1) and neighbor radius
2: **while** not convergence **do** // convergence by reducing α and radius
3: **for all** high-dimensional input vectors x_i **do**
4: Find the unassigned map position with most similar vector m_j
5: Assign the vector x_i to this position and
   update the neighbor map vectors: $m_{j'} = \alpha \cdot x_i + (1 - \alpha) \cdot m_{j'}$
6: Reduce α and the neighbor radius

Self Sorting Map

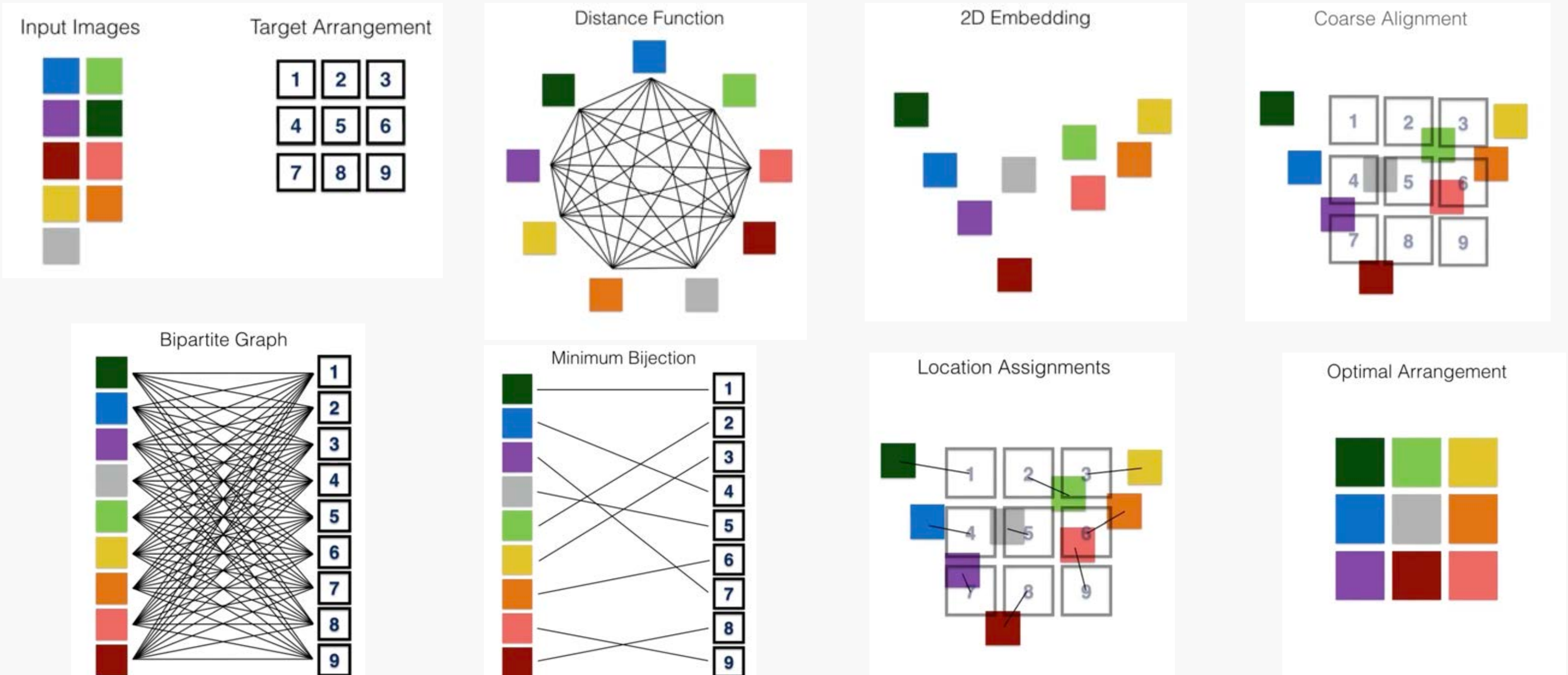


Algorithm 2 SSM

Copy all input vectors into random but unique cells of the grid
Divide the grid into 4x4 blocks
while size of the blocks ≥ 1 **do**
    Divide each block into 2x2 smaller blocks
    for iteration $= 1, 2, \ldots L$ **do** // L = maximum number of iterations
        For each block its target vector (the mean vector of its cells and adjacent blocks' cells) is calculated
        for all blocks **do**
            for all cells of the block **do**
                Find the best swapping permutation for the 4 cells from corresponding positions of the adjacent 2x2 blocks by minimizing the sum of squared differences between the cell vectors and the target vectors of the blocks

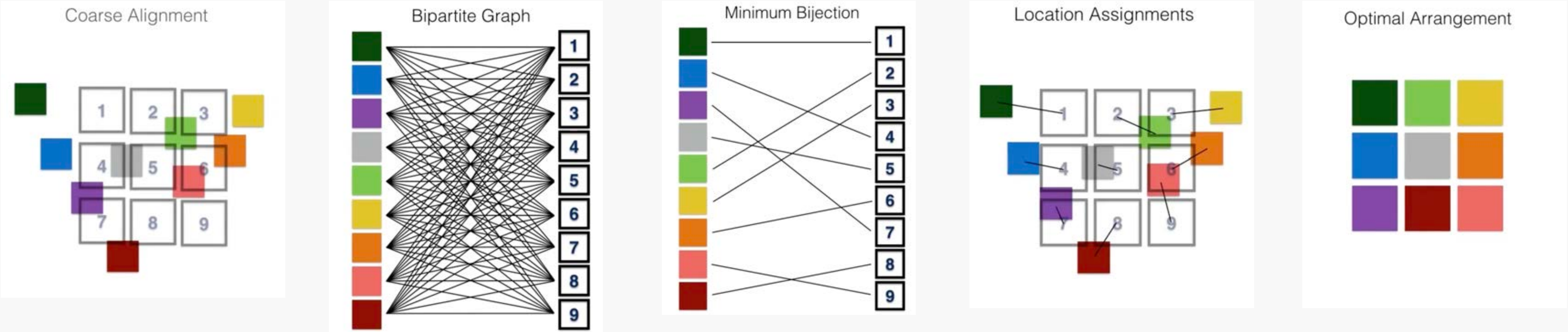
IsoMatch

- The data is first projected into a 2D plane using the Isomap technique.
- A complete bipartite graph is created between the projection and the grid positions. The Hungarian algorithm is applied to determine the optimal assignment for the projected 2D vectors to the grid positions.



"Dimensionality Reduction to Grid"

- Any (non-quantized) projection can be assigned to a grid.
- A Linear Assignment Solver can be used to determine the optimal assignment for the projected 2D vectors to the grid positions.
- "t-SNE to grid", … and others are possible



Neural networks for learning permutations

Sorting of numbers can be described by a matrix multiplication of the number vector with a permutation matrix:

$$\mathbf{n}^T = \begin{bmatrix} 5 \\ 1 \\ 3 \\ 4 \end{bmatrix} \qquad P = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{bmatrix} \qquad P\mathbf{n}^T = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 5 \\ 1 \\ 3 \\ 4 \end{bmatrix} = \begin{bmatrix} 1 \\ 3 \\ 4 \\ 5 \end{bmatrix}$$

Machine learning can be used to learn the permutation matrix.

Neural networks for learning permutations

Problems:

The permutation matrix is not differentiable.
The iterated Sinkhorn Operator can generate a differentiable permutation matrix.

Which loss function to use?
The loss function has to assure that the permutation matrix is a doubly stochastic matrix and that the distance of nearby grid elements is very small.

It does work and is very slow. But up to now, I did not manage to achieve better results than with other schemes. **:(**

Metrics for Evaluating Sorted Arrangements

User Evaluation

Metrics should reflect the sorting quality
as perceived by humans.

Please rate some sorted arrangements of images:

https://experiment.visual-computing.com/

Metrics for Evaluating Sorted Arrangements

- Mean average precision
- k-neighborhood preservation index
- Cross-correlation
- Normalized energy function

Mean Average Precision

The Mean Average Precision (mAP) is the commonly used metric to evaluate image retrieval systems.

$$\mathrm{AP}(q) = \frac{1}{m_q} \sum_{k=1}^{N} \mathrm{P}_q(k)\,\mathrm{rel}_q(k) \qquad \mathrm{mAP} = \frac{1}{N} \sum_{n=1}^{N} \mathrm{AP}(n)$$

- mAP defines "good" sorting when nearest neighbors share the same class.
- Often, mAP cannot be used due to lack of class information.
- mAP overlooks the order of other images, focusing only on same-class images.

k-Neighborhood Preservation Index

The k-neighborhood preservation index evaluates the preservation of the neighborhood of the high-dimensional data of the sorting S on the grid.

$$\mathrm{NP}_k(S) = \frac{1}{N} \sum_{i=1}^{N} \frac{\left| \mathcal{N}_{k,i}^{\mathrm{HD}} \cap \mathcal{N}_{k,i}^{2\mathrm{D}} \right|}{k}$$

Problems:

- The quality of an arrangement is described by individual values for each neighbor size k.
- High sensitivity to noisy HD data or similar distances on the grid.

Cross-Correlation

The cross-correlation is used to determine how well the distances of the projected grid positions λ correlate with the distances of the original vectors δ.

$$\mathrm{CC}(S) = \sum_{i=1}^{N} \sum_{j=1}^{N} \frac{(\lambda(y_i, y_j) - \bar{\lambda})(\delta(x_i, x_j) - \bar{\delta})}{\sigma_\lambda \sigma_\delta}$$

Problems:

- Cross-correlation in image arrangement prioritizes large distance differences over small differences.
- Preserving small and large distances is crucial to preserve similarity in image sorting.

Normalized Energy Function

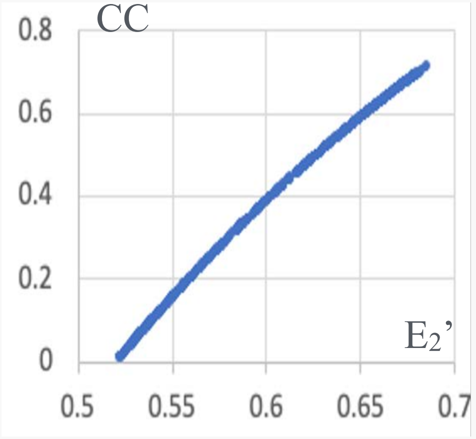
The normalized energy function measures how well distances between the data instances are preserved by the corresponding spatial distances on the grid.

$$\mathrm{E}_p(S) = \min_c \left(\sum_{i=1}^{N} \sum_{j=1}^{N} \frac{\left| c \cdot \delta(x_i, x_j) - \lambda(y_i, y_j) \right|^p}{\sum_{r=1}^{N} \sum_{s=1}^{N} (\lambda(y_r, y_s))^p} \right)^{\frac{1}{p}}$$

$$\mathrm{E}'_p(S) = 1 - \mathrm{E}_p(S)$$

Parameter p adjusts the balance between small and large distances, commonly values of 1 or 2 are used.

The normalized energy function shares the properties and issues with cross-correlation.

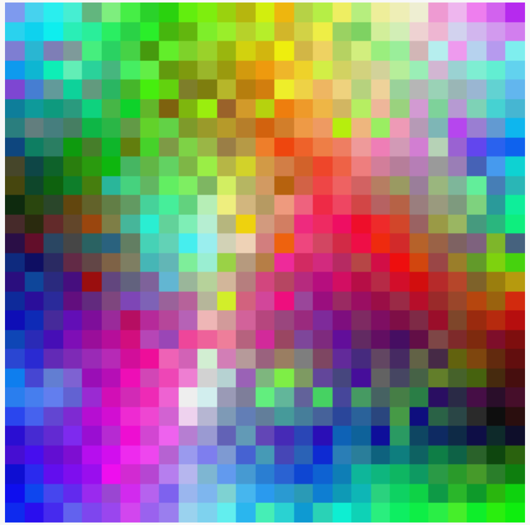


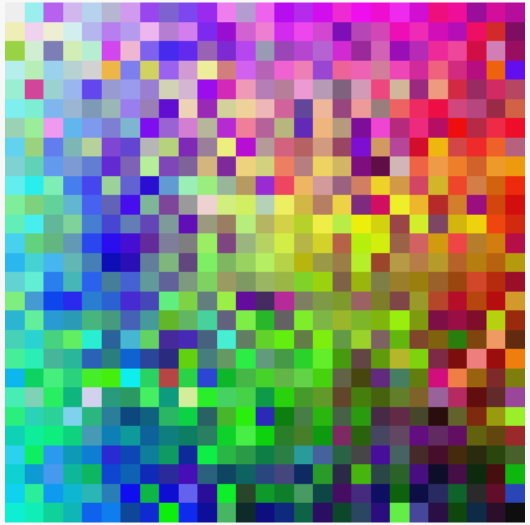
E'_2 rates arrangements the same way as CC.

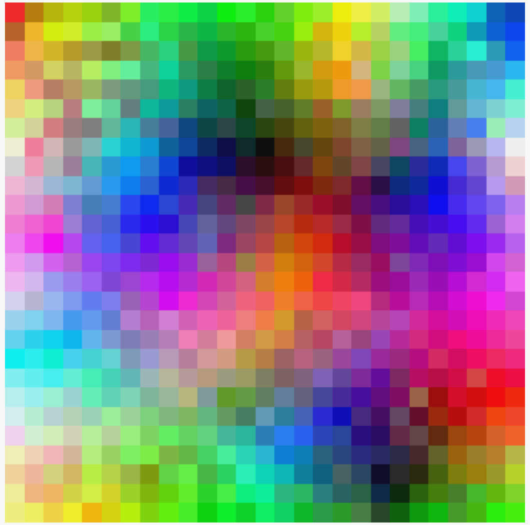
A New Quality Metric for Grid Layouts

Ranking different arrangements

Please rank the three arrangements
in the order of their visual sorting quality.







E'_2:*)

0.553
medium
arrangement

0.648
best
arrangement

0.524
worst
arrangement

*) higher is better

Rank the arrangements by their quality

E'_1: 0.577 | 0.579 | 0.612

worst arrangement | medium arrangement | best arrangement

Motivation

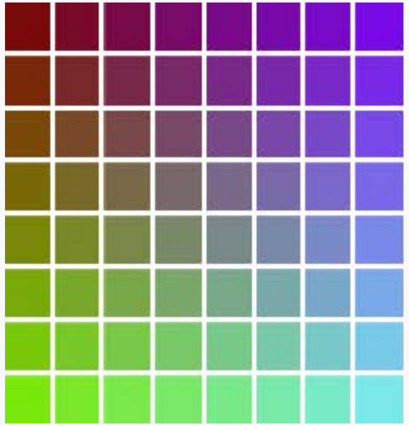
- The metrics currently in use do not reflect perceived sorting quality well.
- Our goal was to develop a metric that better correlates with human perceived quality. The quality should be expressed by a single value, where 0 represents random order and 1 represents perfectly sorted arrangement.

There are two approaches in developing a suitable quality function for grid layouts.

- The first option would be to refer to the best possible 2D sorting. However, this approach is not applicable because the best possible sorting is usually not known.

Motivation

- The only viable way is to refer to the distribution of the high-dimensional data.
- A perfect sorting here means that all 2D grid distances are proportional to the HD distances.
- Depending on the specific HD distribution, it is usually not possible to achieve this perfect order in a 2D arrangement.



possible

impossible

to preserve the HD order in a 2D arrangement

Neighborhood Preservation Quality

Our first Idea: Combination of the k-neighborhood preservation indices $\mathrm{NP}_k(S)$ into a single quality value.
The k-neighborhood preservation indices for an optimal and random arrangements are:

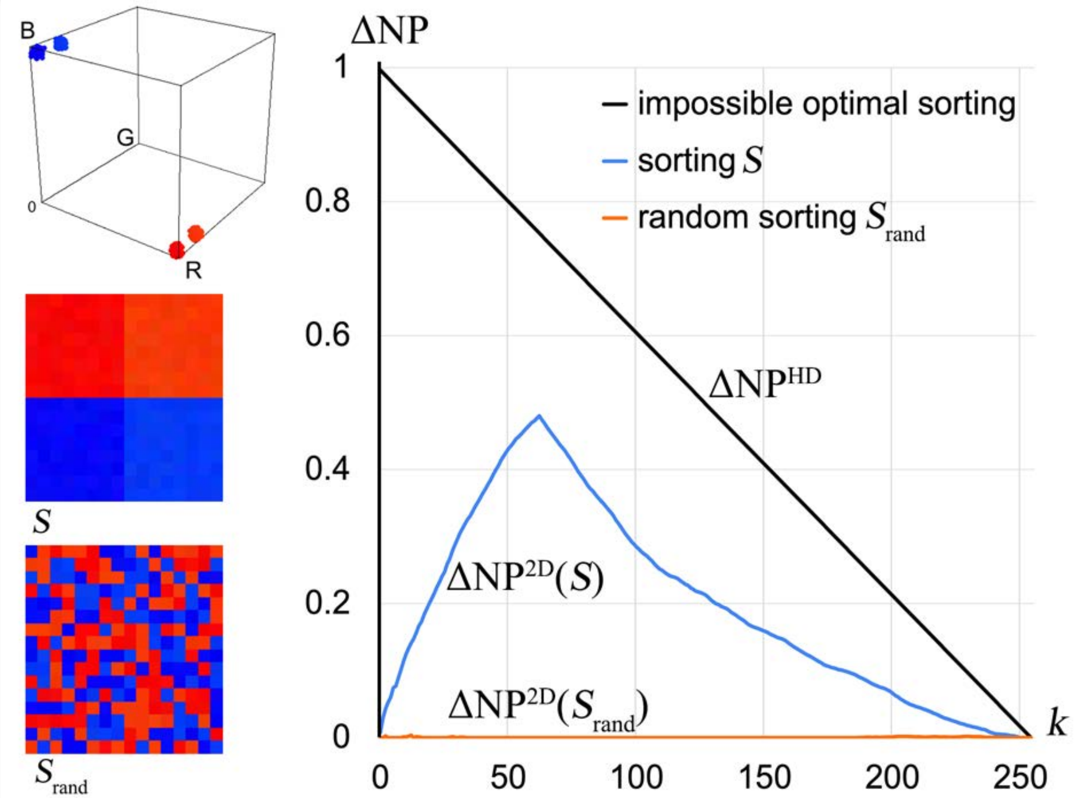
$$\mathrm{NP}_k(S_{\mathrm{opt}}) = 1 \qquad \mathbb{E}[\mathrm{NP}_k(S_{\mathrm{rand}})] = \frac{k}{K}$$

(K = number of neighbors)

For a 2D arrangement S the **Neighborhood Preservation Gain** $\Delta\mathrm{NP}_k^{\mathrm{2D}}(S)$ is the difference between the actual $\mathrm{NP}_k(S)$ values and the expected values for random arrangements.

$$\Delta\mathrm{NP}_k^{\mathrm{HD}} = \Delta\mathrm{NP}_k^{\mathrm{2D}}(S_{\mathrm{opt}}) = 1 - \frac{k}{K} \qquad \Delta\mathrm{NP}_k^{\mathrm{2D}}(S) = \max\left(\mathrm{NP}_k(S) - \frac{k}{K}, 0\right)$$

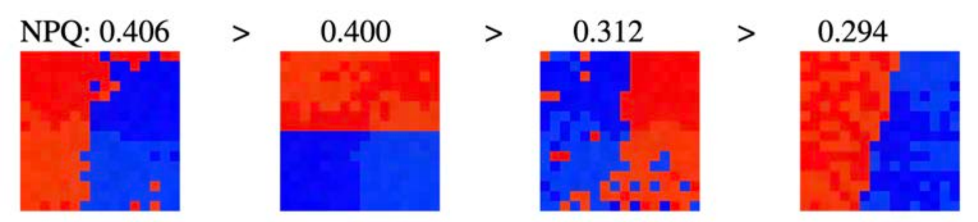
Neighborhood Preservation Gain & Quality



Neighborhood Preservation Quality:

$$\text{NPQ}_p(S) = \frac{\|\Delta \text{NP}^{\text{2D}}(S)\|_p}{\|\Delta \text{NP}^{\text{HD}}\|_p}$$

$$0 \leq \text{NPQ}_p(S) \leq 1$$



It can be seen that the order resulting from the NPQ does not correspond with the human perception of sorting quality. :(

Distance Preservation

- The neighborhood preservation only focuses on correct ranking of neighbors, neglecting the actual similarity of wrongly ranked neighbors.
- Our proposal involves comparing the averaged distances of the corresponding neighborhoods.

$$\mathrm{D}_k^{\mathrm{HD}} = \frac{1}{kN} \sum_{i=1}^{N} \sum_{j \in \mathcal{N}_{k,i}^{\mathrm{HD}}} \delta(x_i, x_j) \quad \mathrm{D}_k^{2\mathrm{D}}(S) = \frac{1}{kN} \sum_{i=1}^{N} \sum_{j \in \mathcal{N}_{k,i}^{2\mathrm{D}}} \delta(x_i, x_j)$$

Again we compare the average neighborhood distance with the expectation value of the average neighborhood distance of random arrangements, which is equal to the global average distance.

$$\mathbb{E}[\mathrm{D}_k^{2\mathrm{D}}(S_{\mathrm{rand}})] = \bar{\mathrm{D}} = \frac{1}{N(N-1)} \sum_{i=1}^{N} \sum_{j=1}^{N} \delta(x_i, x_j)$$

Distance Preservation Quality

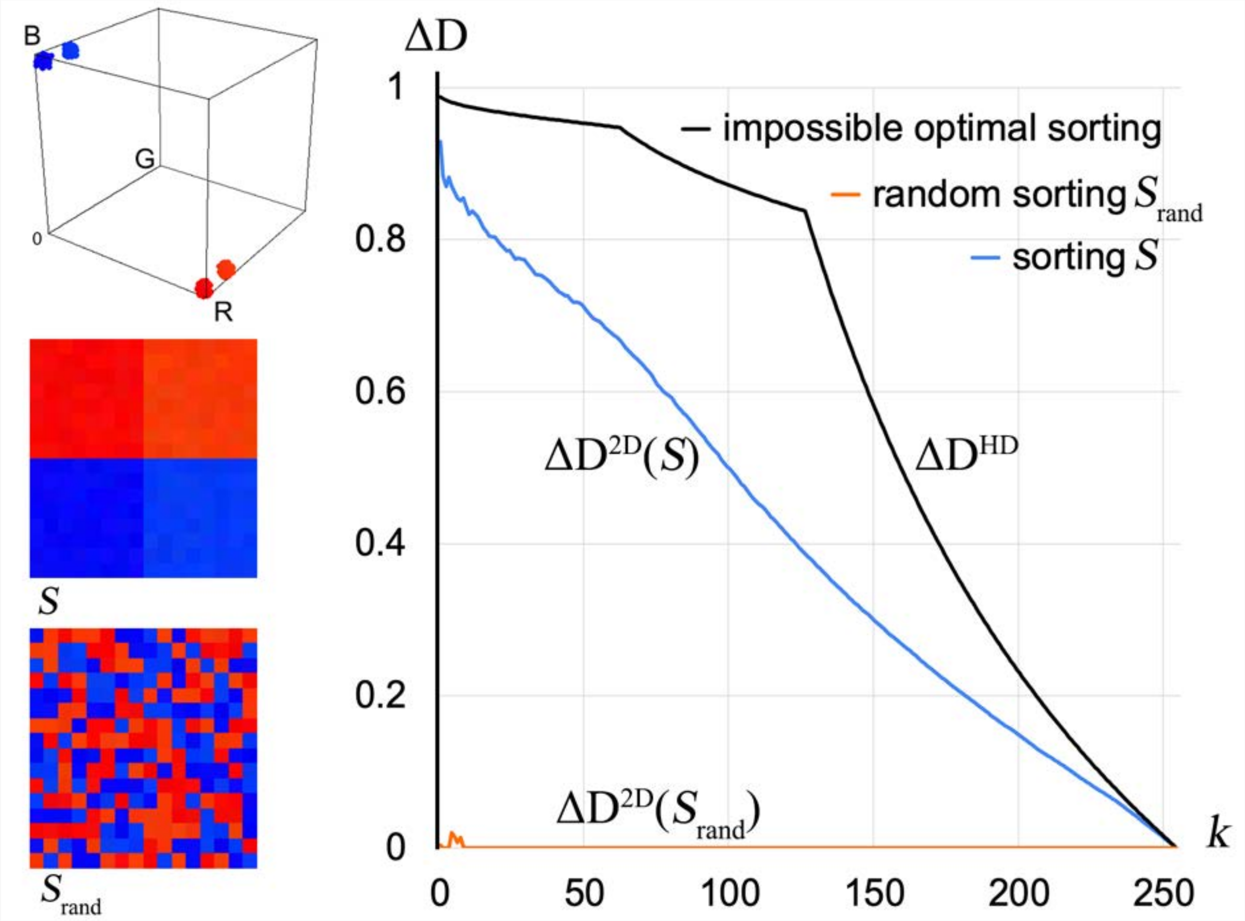
Analogous to $\Delta \mathrm{NP}_k^{2\mathrm{D}}(S)$ the **Distance Preservation Gain** $\Delta \mathrm{D}_k$ is defined as the difference between the average neighborhood distance of a random arrangement and that of the arrangement.

$$\Delta \mathrm{D}_k^{\mathrm{HD}} = \frac{1}{\overline{\mathrm{D}}}(\overline{\mathrm{D}} - \mathrm{D}_k^{\mathrm{HD}}) \qquad \Delta \mathrm{D}_k^{2\mathrm{D}}(S) = \max\left(\frac{1}{\overline{\mathrm{D}}}(\overline{\mathrm{D}} - \mathrm{D}_k^{2\mathrm{D}}(S)), 0\right)$$

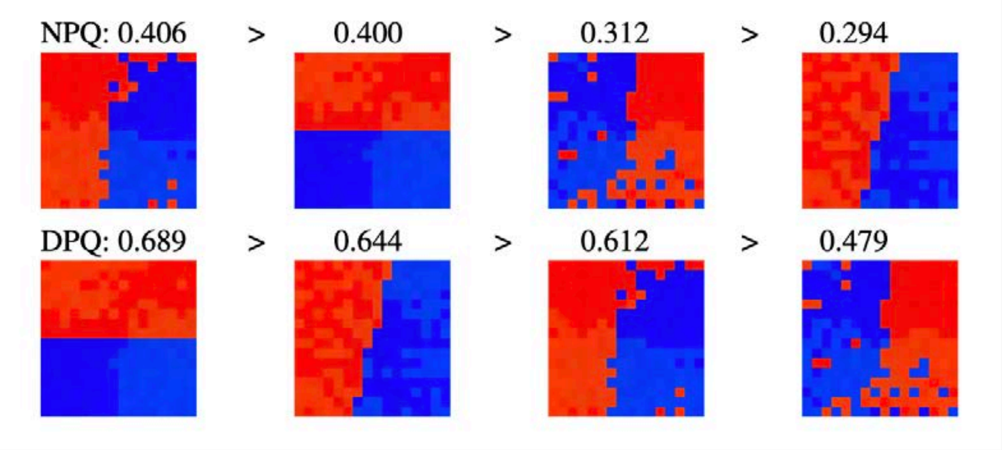
The **Distance Preservation Quality** $\mathrm{DPQ}_p(S)$ is defined as the ratio of the p-norms of the distance preservation gains of the actual arrangement to a perfect arrangement:

$$\mathrm{DPQ}_p(S) = \frac{\|\Delta \mathrm{D}^{2\mathrm{D}}(S)\|_p}{\|\Delta \mathrm{D}^{\mathrm{HD}}\|_p} \qquad 0 \leq \mathrm{DPQ}_p(S) \leq 1$$

Distance Preservation Gain & Quality



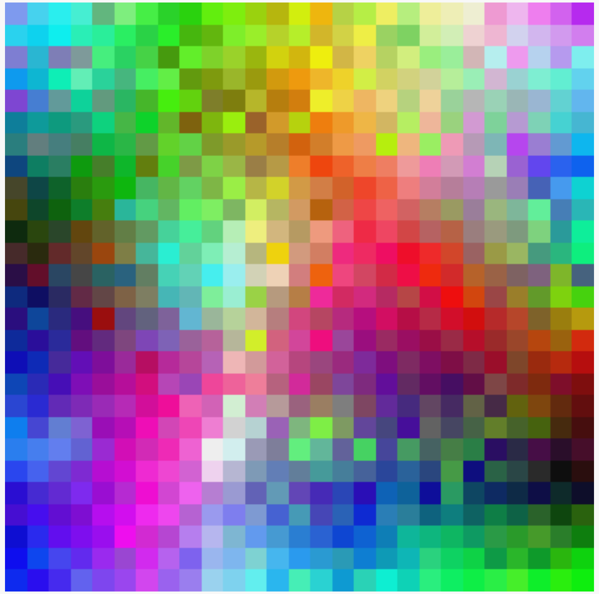
$$\mathrm{DPQ}_p(S) = \frac{\|\Delta \mathrm{D}^{2\mathrm{D}}(S)\|_p}{\|\Delta \mathrm{D}^{\mathrm{HD}}\|_p}$$

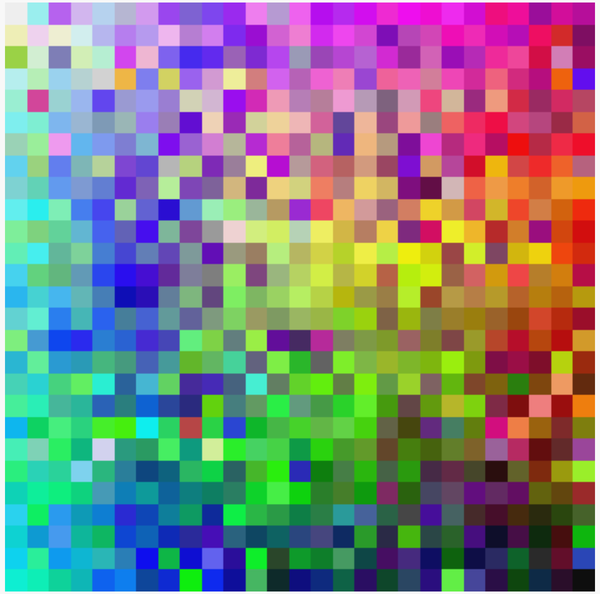


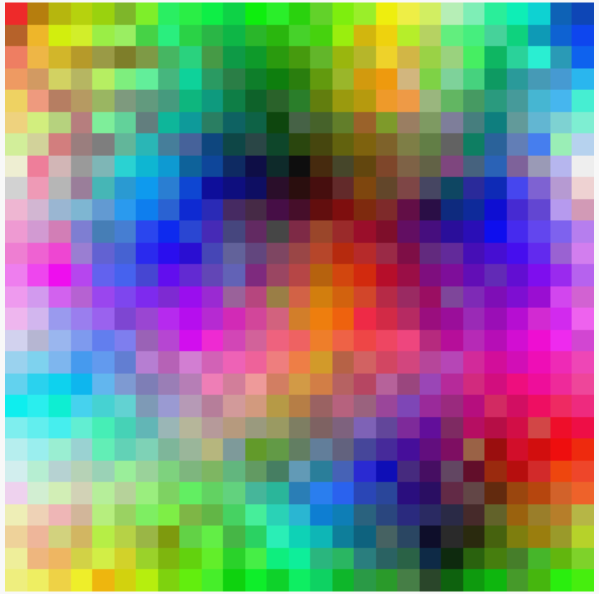
It can be seen that the order resulting from the DPQ metric is more consistent with the human perception of sorting quality than NPQ. :)

Ranking different arrangements

The same three arrangements in the order of their DPQ.







DPQ_{16}:*) 0.774 | 0.570 | 0.816

medium arrangement | worst arrangement | best arrangement

*) higher is better

Rank the arrangements by their quality

DPQ$_{16}$: 0.679

best arrangement

0.263

medium arrangement

0.194

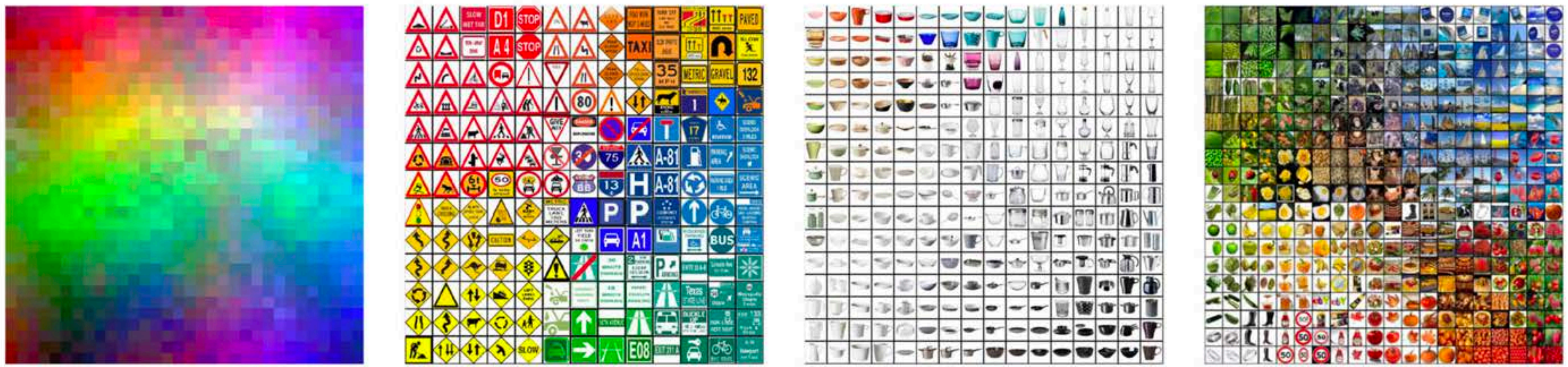
worst arrangement

Human Evaluation of Sorted Images

Motivation

- User tests are necessary to determine the suitability of the Distance Preservation Metric for describing sorting quality compared to other metrics.
- Two types of user tests:
  Preference Tests & Search Tests
- A better evaluation metric should demonstrate a higher correlation with user scores and a higher (negative) correlation with user search times for finding images in the arrangements.

Evaluation of Algorithms and Metrics



1024 RGB colors | 169 traffic sign images | 256 kitchenware images | 400 images from the web

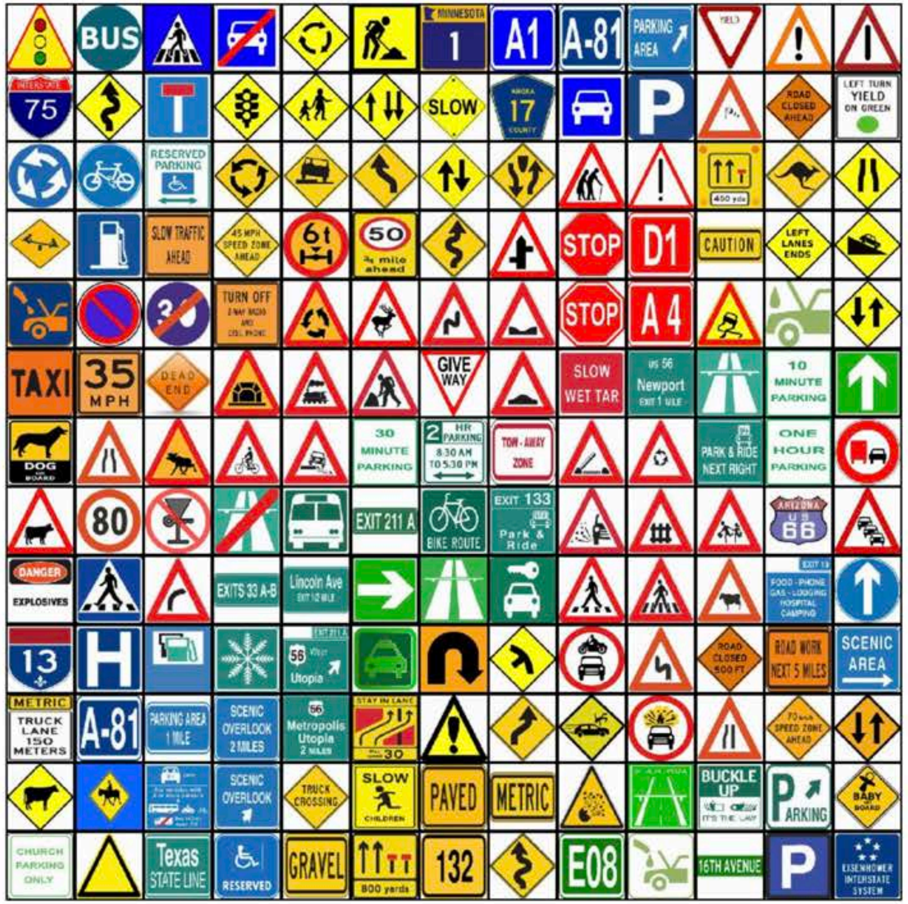
- The set of 1024 randomly generated RGB colors, used solely for assessing perceived quality of sorting methods.
- The other three image sets were also used to record the time taken to find searched images.
- These sets were chosen to represent different search scenarios and exhibit significant differences in search speed between sorted and random arrangements.

Evaluation of Algorithms and Metrics

- Over 2000 users participated in evaluating arrangements
- We used various sorting image algorithms with different parameter settings (if applicable).
- A 50 dimensional low-level feature vector was used to describe the images.
- The compared metrics included E'_1, E'_2, and DPQ_p with varying p values.

Preference Tests

Select the sorting that you find clearer (9/16)



NEXT

Preference Test Implementation

- All users had to evaluate 16 pairs and decide whether they preferred the left or the right arrangement. They could also state that they considered both to be equivalent.
- The number of different arrangements were 32 for the color set and 23 for each of the three image sets, (giving 496 pairs for the color set and 253 pairs for each image set).
- Each pair of arrangements was evaluated by at least 35 users.

Preference Test Evaluation

For each comparison of S_i with S_j, the preferred arrangement gets one point. In case of a tie, both get half a point each.

Let $v_r(S_i, S_j)$ be the points received by S_i in the r^{th} out of R comparisons between S_i and S_j.

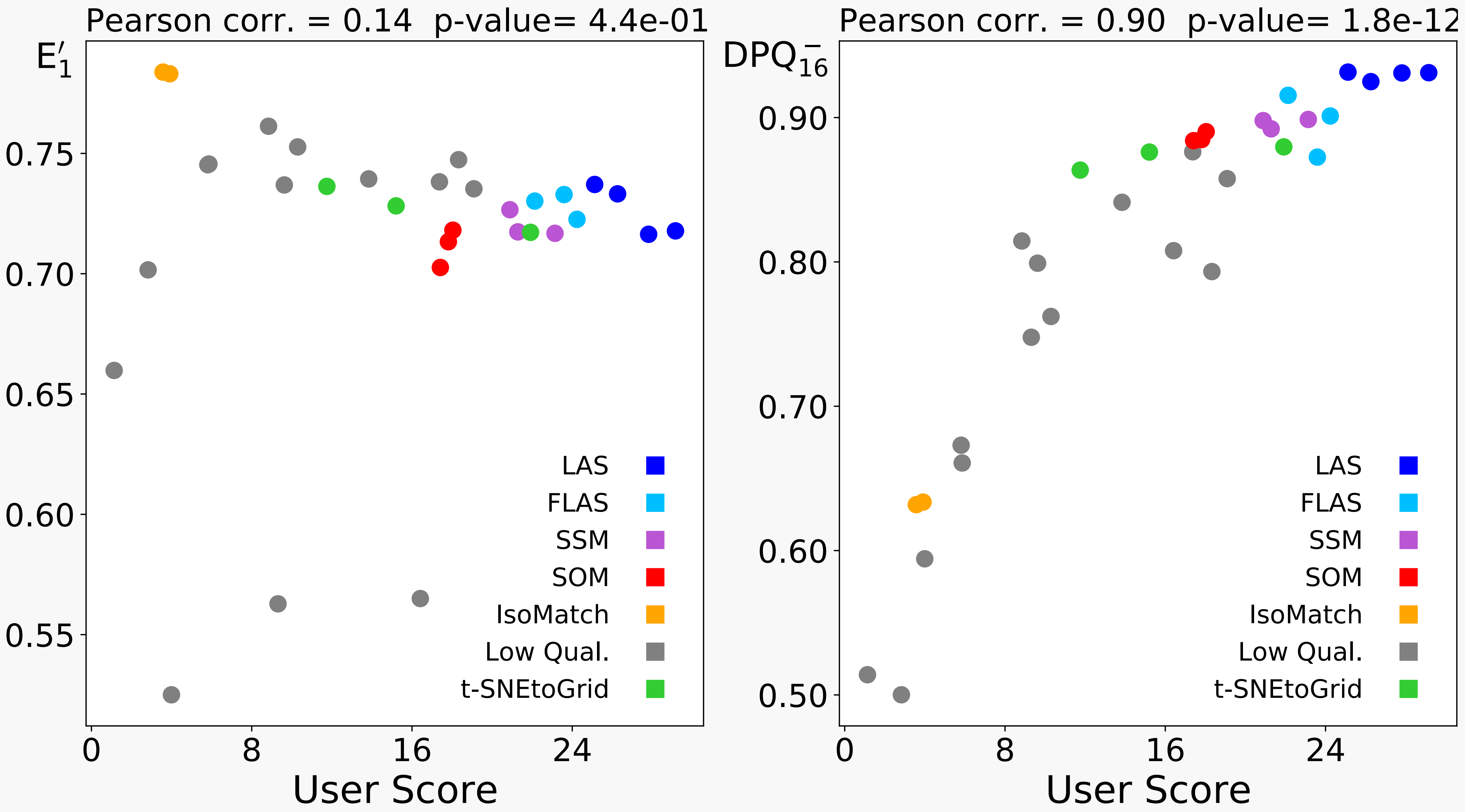
Let

$$P(S_i, S_j) = \frac{1}{R} \sum_{r=1}^{R} v_r(S_i, S_j)$$

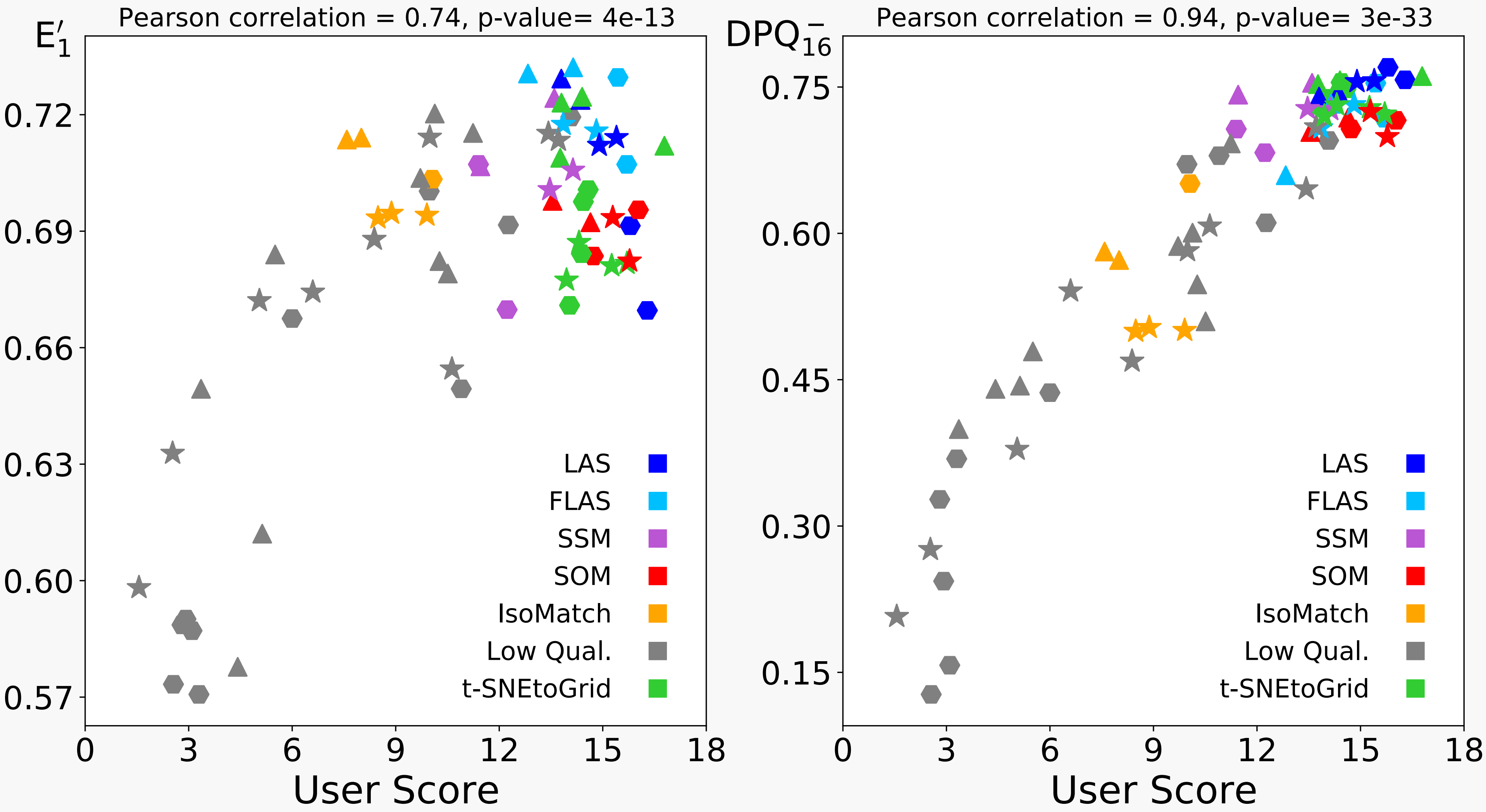
be the probability that S_i receives a higher quality assessment in comparison to S_j, ($\mathrm{P}(S_i, S_j) + \mathrm{P}(S_j, S_i) = 1$).

The final user score for S_i is defined by $\text{User Score}(S_i) = \sum_j P(S_i, S_j)$

Metrics vs User Scores (RGB Colors)



Metrics vs User Scores (Image Sets)

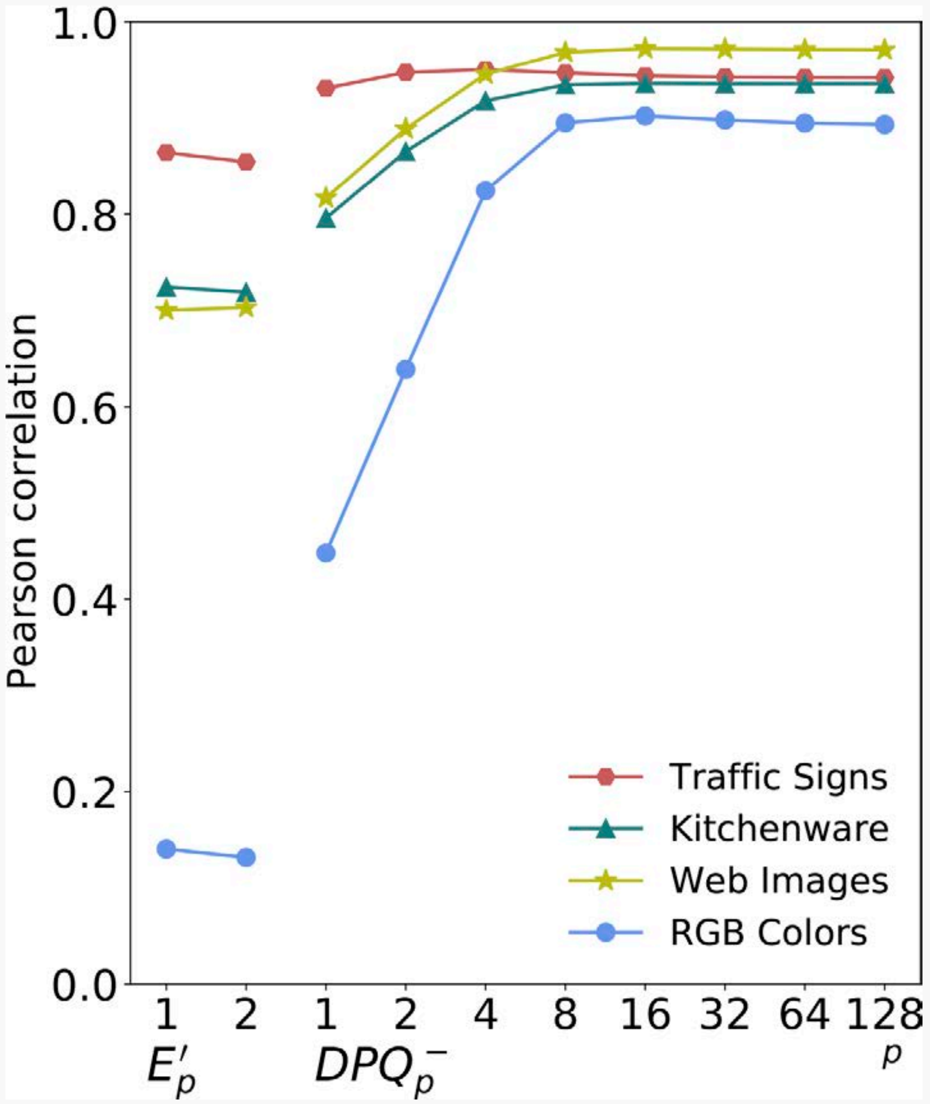


traffic signs (⬢), kitchenware (▲), web images (★)

Correlation of Metrics & User Scores

Correlation of the metrics E'_p and DPQ_p with user scores for the color and the three image sets with respect to the p values

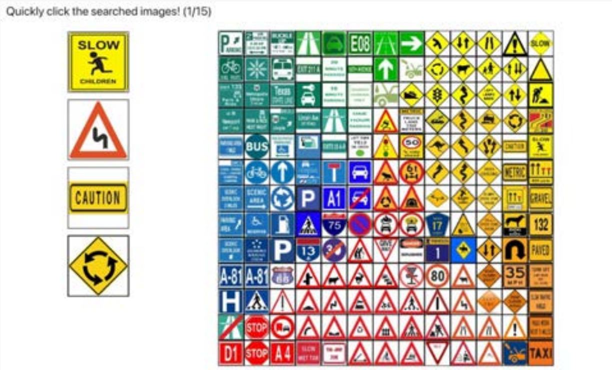
Correlation for DPQ_p is much higher.



Search Tests

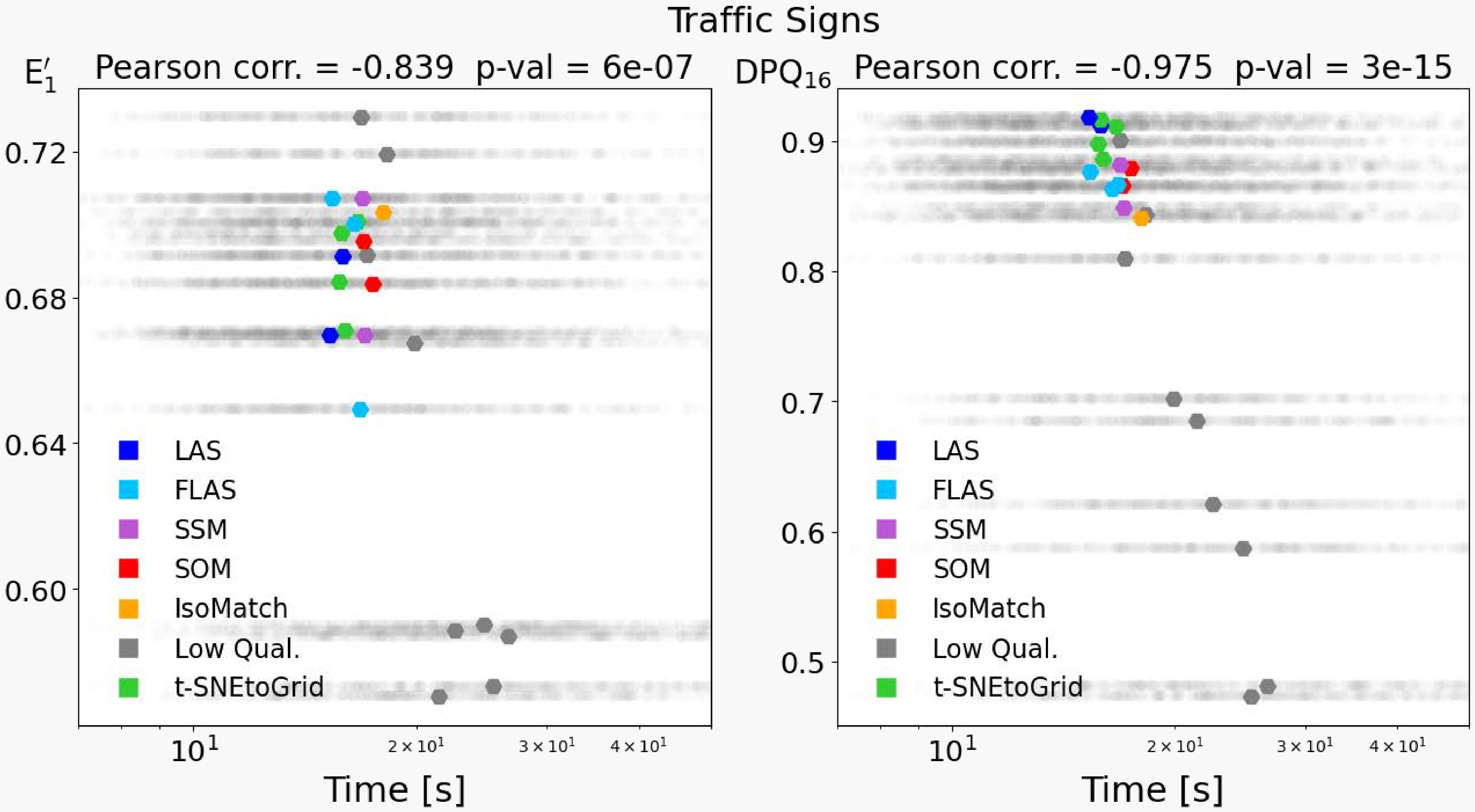
Quickly click the searched images! (1/15)

Search Test Implementation

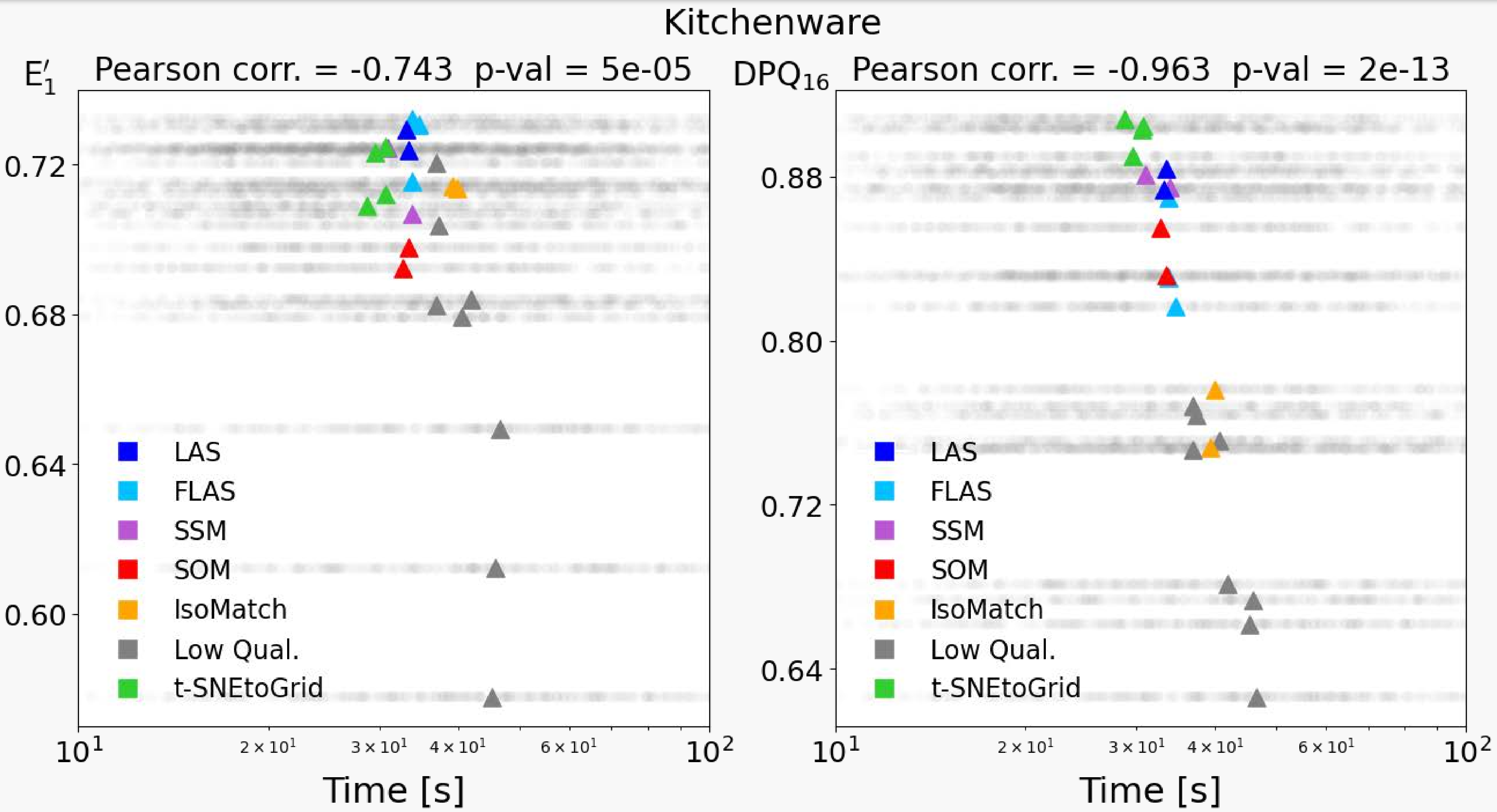


- In the second part of the user study, the users had find four images in different arrangements (the same as in the first experiment).
- The four images to be searched were randomly chosen and shown one after the other.
- The search times required for each of the 23 arrangements for the three image sets were recorded.
- Over 400 search tasks were conducted for four images in each arrangement, ensuring compensation for variations in search difficulty and participant abilities.

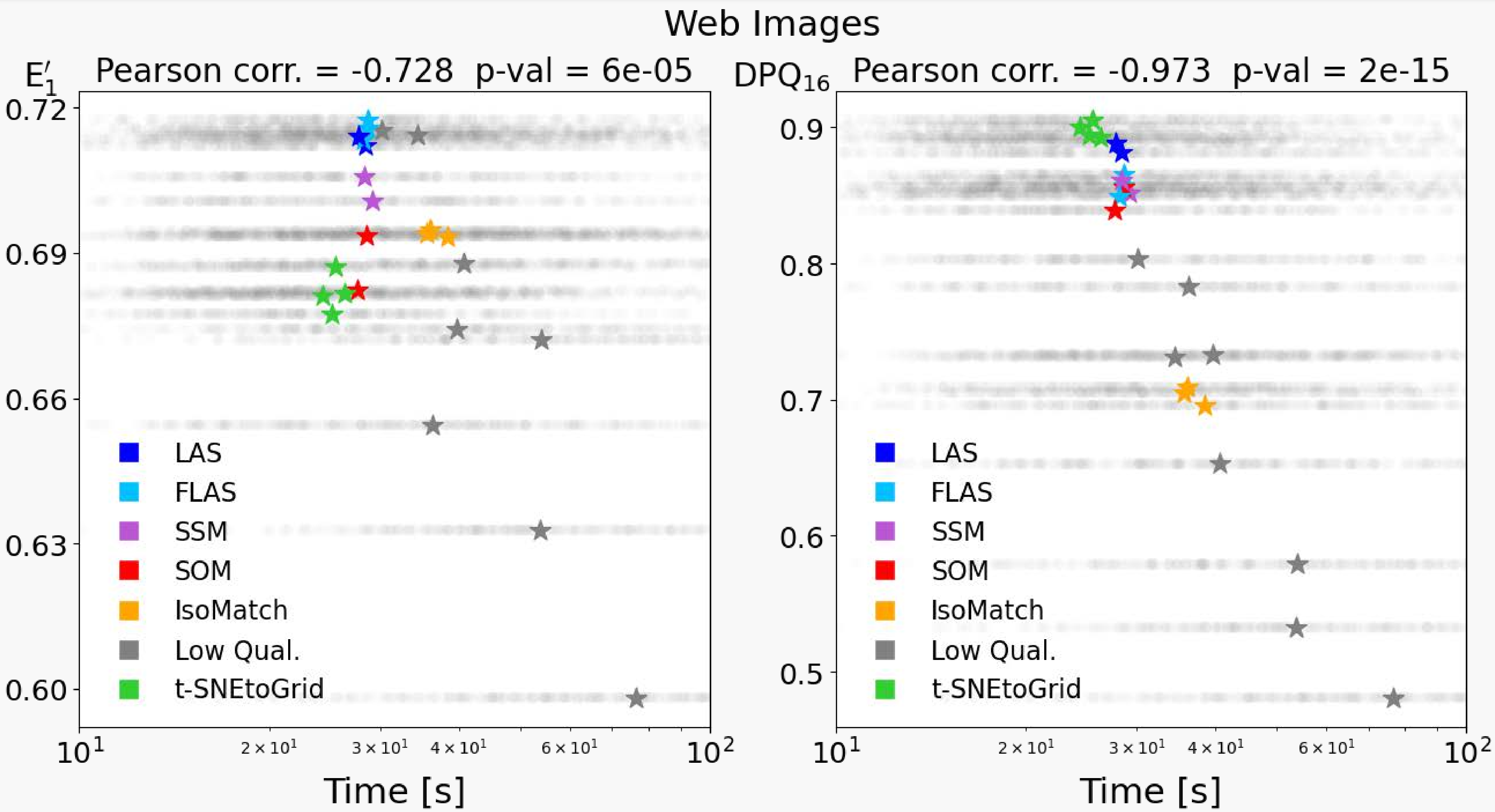
Search Times vs Metrics



Search Times vs Metrics

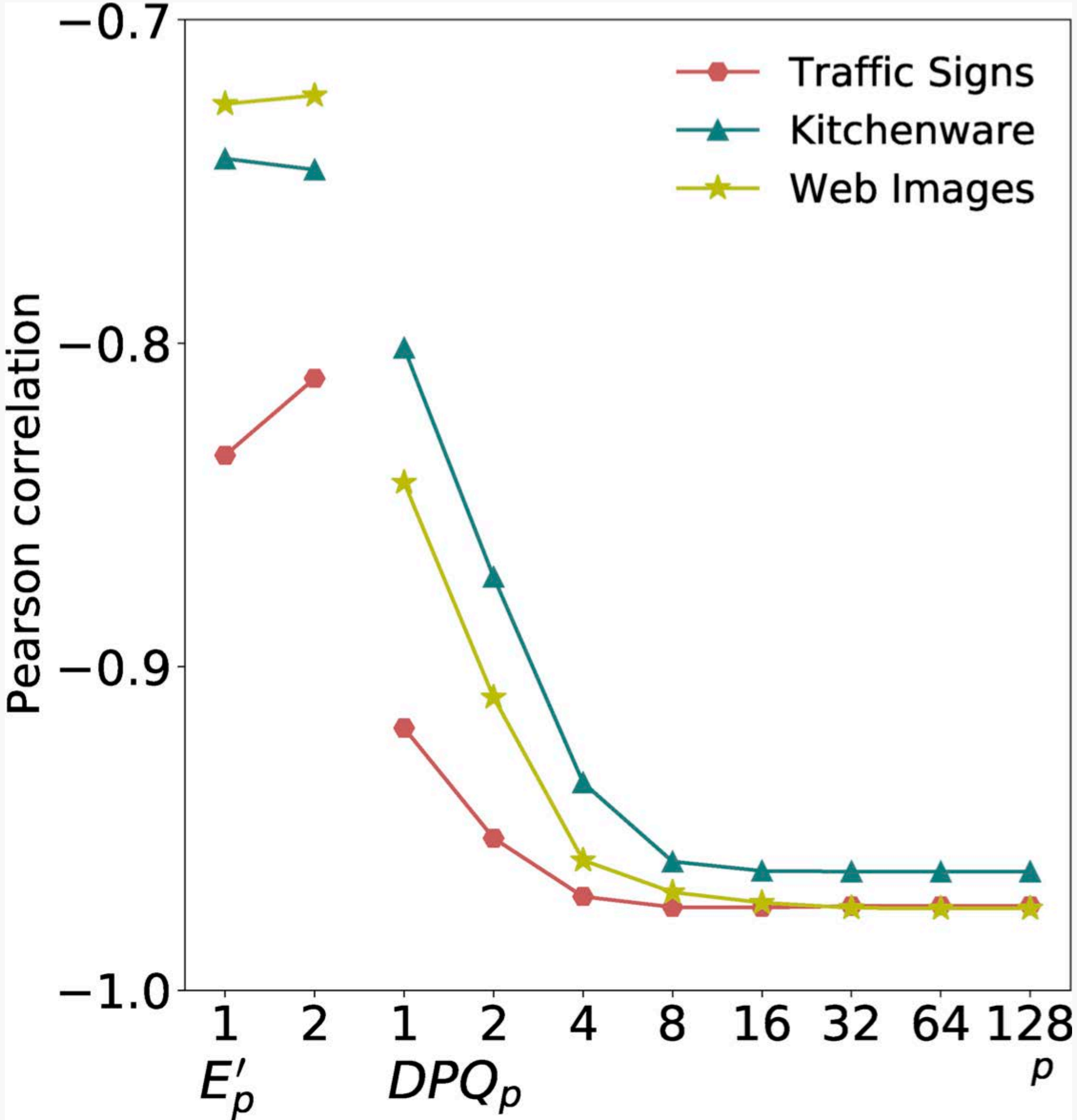


Search Times vs Metrics



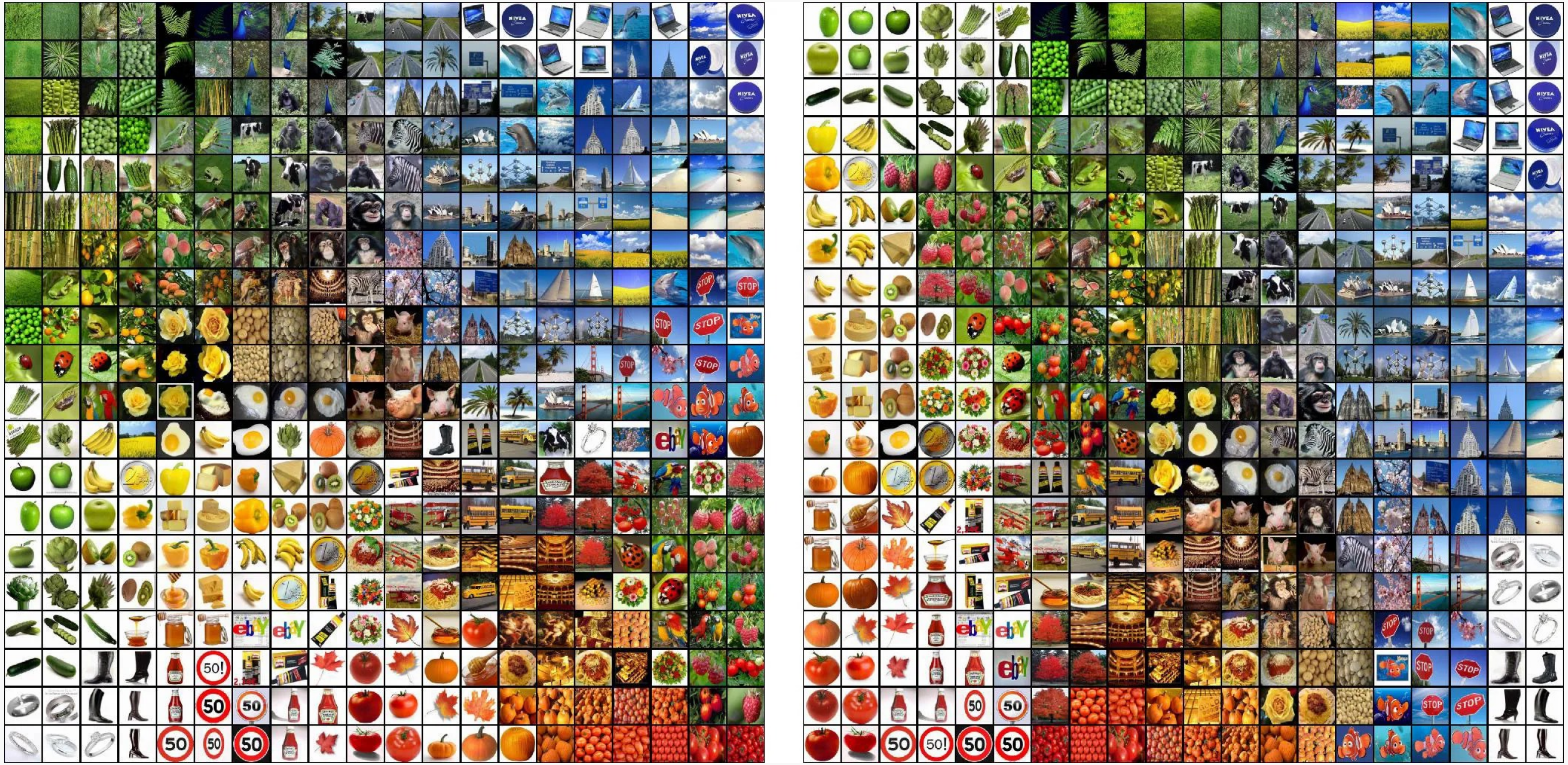
Correlation of Search Times & Metrics

Correlation of search times with the metrics E'_p and DPQ_p with respect to the p values for the three image sets.



Correlation of Search Speed & User Score

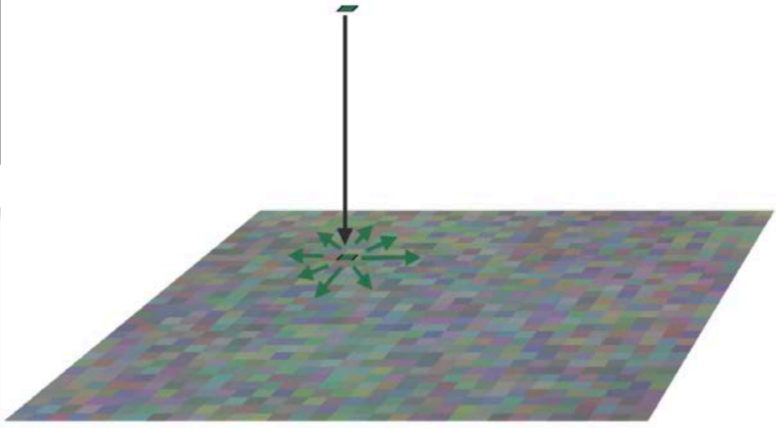
Search speed and user score are highly correlated.



Left: the sorting that was rated the best. Right: the sorting in which the searched images were found the fastest.

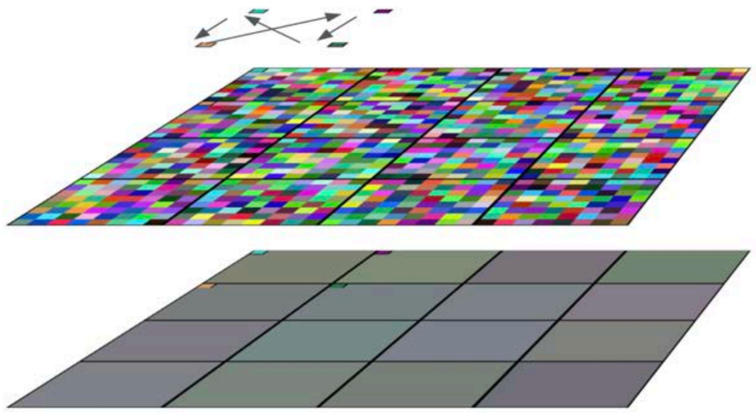
Linear Assignment Sorting

SOM Revisited



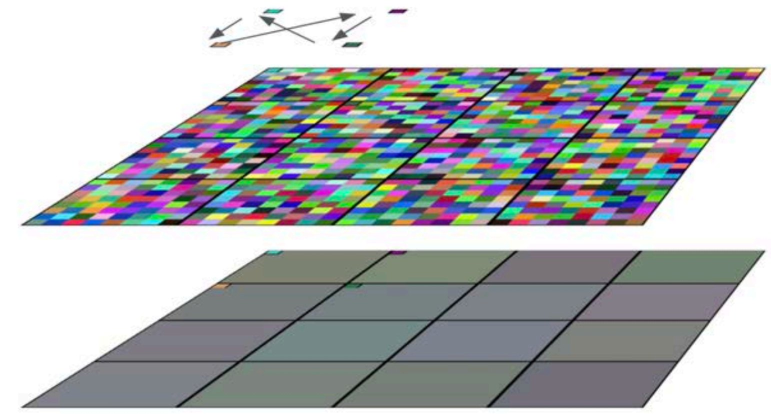
- The SOM assigns each input vector to the best map vector and updates its neighbors.
- This update can be seen as blending the map vectors with the spatially low-pass filtered assigned input vectors, where the filter radius = neighborhood radius.
- We propose a faster process: first copying input vectors to the most similar unassigned map vector and then spatially filtering all map vectors. Integral filters allow constant complexity independent of the radius.
- Due to the sequential process of the SOM, the last input vectors can only be assigned to the few remaining unassigned map positions. This results in isolated, poorly positioned vectors.

SSM Revisited I



- To address isolated, poor assignments, the SSM employs a four-input vector swapping approach.
- The best swap is determined through a brute force comparison with 4! = 24 possible swaps between the four input vectors and the mean vectors of the corresponding blocks.
- Due to the factorial number of permutations, using more swap candidates become computationally complex. To overcome this, we suggest using linear programming to search for the optimal permutation.

SSM Revisited II

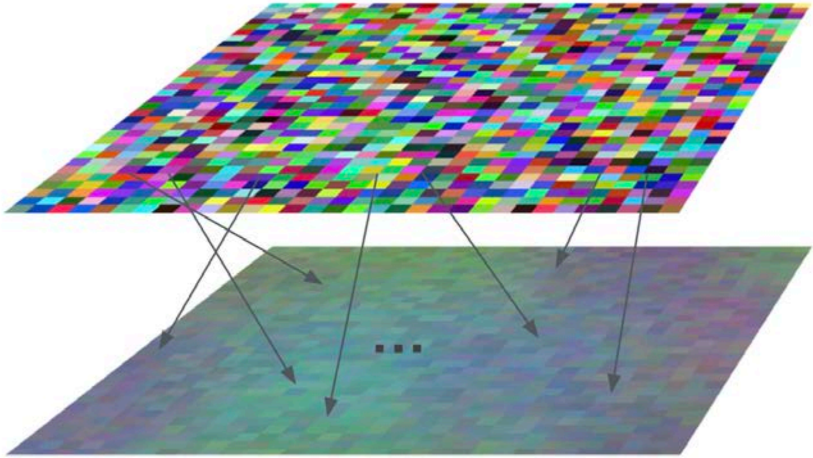


- Another issue with the SSM is its reliance on a single mean vector per block, incorrectly assuming equivalence among positions within a block when swapped.
- The usage of a single mean vector per block can be considered as a subsampled version of the continuously filtered map vectors.
- We propose using map filtering without subsampling, as this allows a better representation of the neighborhoods of the map.
- The block sizes of the SSM remain the same for multiple iterations, this can be seen as repeated use of the same filter radius. We propose continuously reducing the filter radius.

Linear Assignment Sorting (LAS)

- Our proposed image sorting scheme "Linear Assignment Sorting" combines the proposed improvements for the SOM and the SSM and extends this to optimally swapping all vectors simultaneously.
- Initially all map vectors are randomly filled with the input vectors. Then, the map vectors are spatially low-pass filtered to obtain a smoothed version of the map representing the neighborhoods. In the next step all input vectors are assigned to their best matching map positions.
- Since the number of mappings is factorial, we use the Jonker-Volgenant linear assignment solver to find the best swaps with reduced run time complexity of $O(N^3)$.

Linear Assignment Sorting



Algorithm 3 LAS

Set $r_f = \lfloor max(W,H) \cdot f_{r0} \rfloor$ // initial filter radius ($f_{r0} \leq 0.5$)
f_r // radius reduction factor ($f_r < 1$)
Assign and copy all input vectors to random but unique map vectors
while $r_f > 1$ **do**
    Filter the map vectors using the actual filter radius r_f
    Find the optimal assignment for all input vectors
    Copy all input vectors to the map vectors of their new positions
    Reduce the filter radius: $r_f = r_f \cdot f_r$

Fast Linear Assignment Sorting

- Linear Assignment Sorting is a simple algorithm with very good sorting quality. However, for larger sets in the range of thousands of images, the computational complexity of the LAS algorithm becomes too high.
- With a slight modification of the LAS algorithm, very large image sets can still be sorted.
- **Fast Linear Assignments Sorting** (FLAS) is able to handle larger quantities of images by replacing the global assignment with multiple local swaps.

Fast Linear Assignment Sorting

Algorithm 4 FLAS

1: Set $r_f = \lfloor max(W,H) \cdot f_{r0} \rfloor$ // initial filter radius ($f_{r0} \leq 0.5$)
f_r // radius reduction factor ($f_r < 1$)
n_c // number of swap candidates
$iterations = W \cdot H / n_c$

2: Assign and copy all input vectors to random but unique map vectors

3: **while** $r_f > 1$ **do**

4: Filter the map vectors using the actual filter radius

5: **for** $i = 1, 2, \ldots iterations$ **do**

6: Select a random position & select n_c random swap candidates (assigned input vectors) within a radius of $max(r_f, \frac{\sqrt{n_c} - 1}{2})$

7: Find the best swapping permutation

8: Assign the input vectors to their new map positions

9: Copy the input vectors to the map vectors of their assigned positions

10: Reduce the filter radius: $r_f = r_f \cdot f_r$

Coding Example

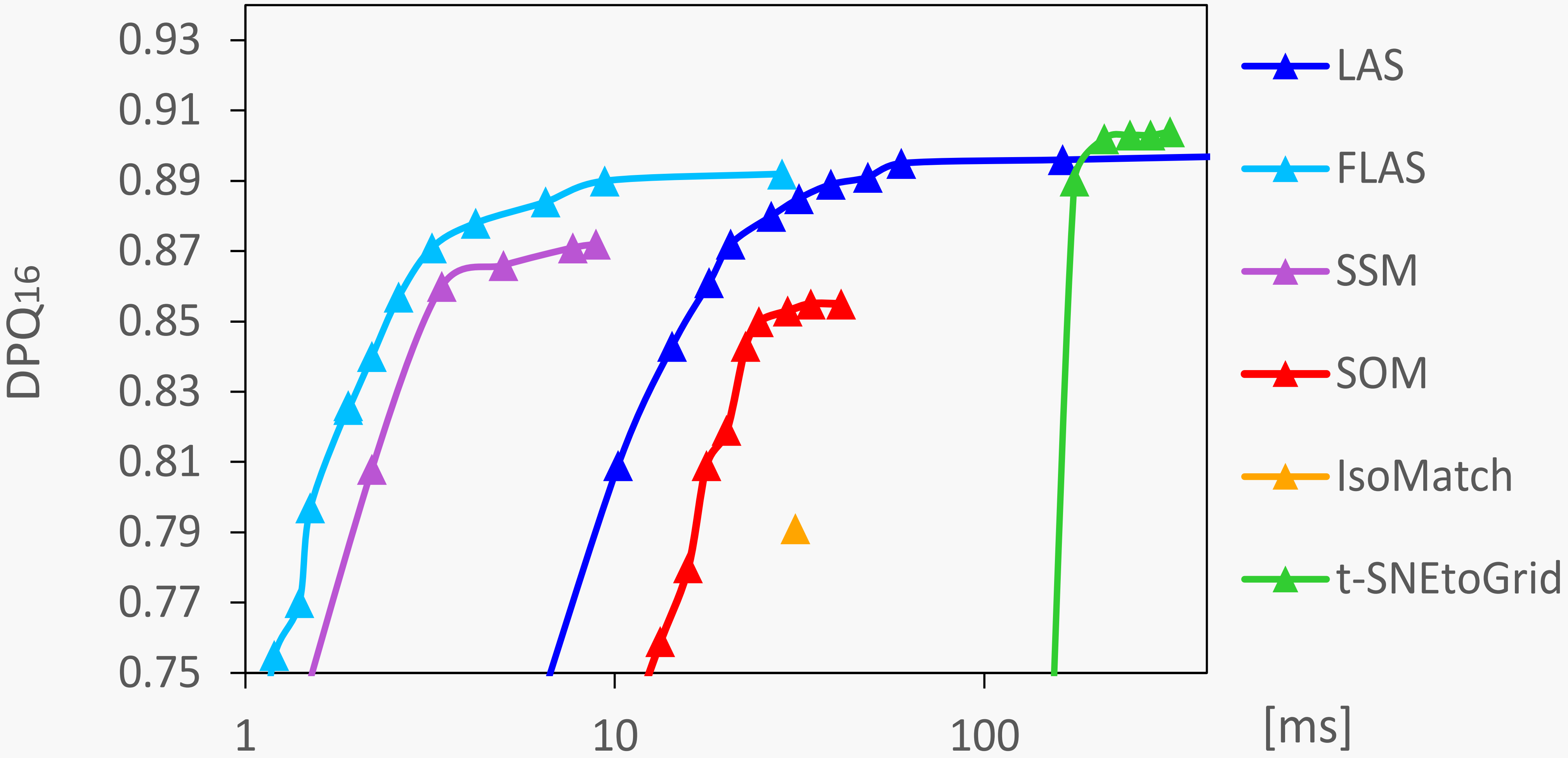
https://github.com/Visual-Computing/LAS_FLAS

Technical Evaluation of Image Sorting Algorithms

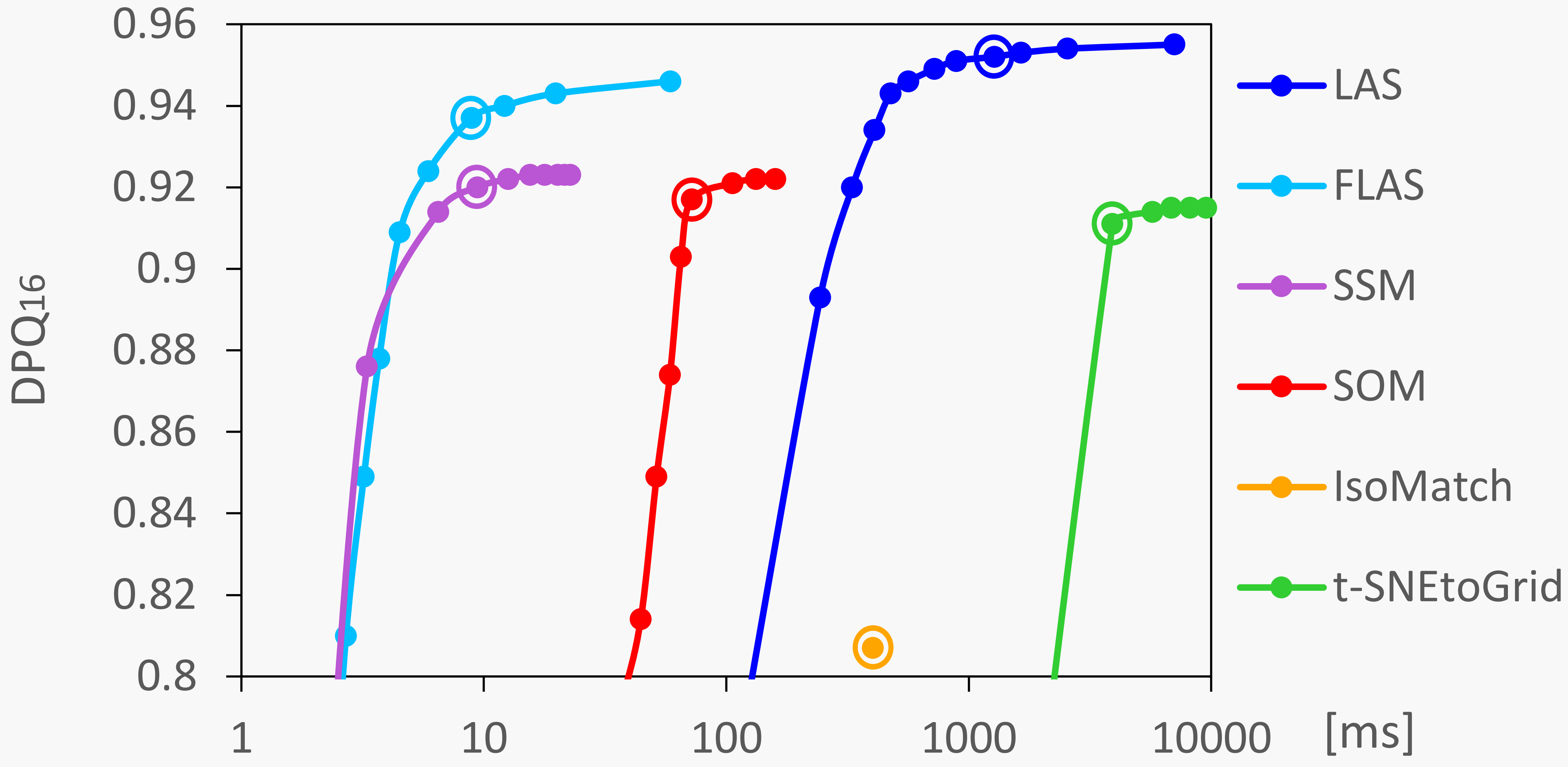
Sorting Quality vs Run-Time

- Since the Distance Preservation Quality (DPQ_{16}) has shown high correlation with user preferences, it was used to compare various algorithms in terms of their achieved "quality" and the run time required to generate the sorted arrangement.
- At startup, all data is loaded into memory. Then the averaged run time and DPQ_{16} value of 100 runs were recorded.
- We ensured the algorithms received the same initial order of images for all runs.

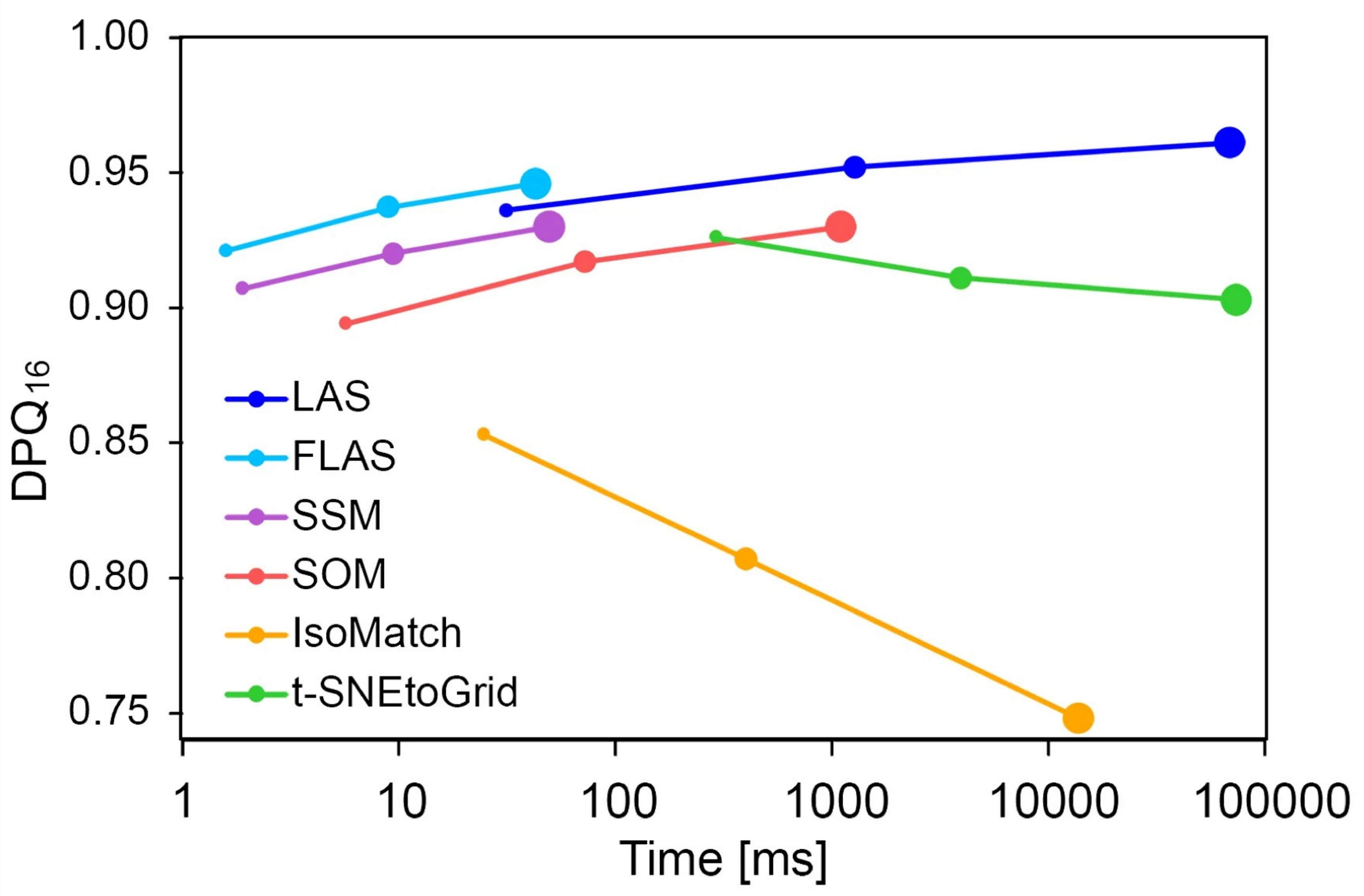
Sorting Quality vs Run-Time: Kitchenware



Sorting Quality vs Run-Time: Colors



Runtime Dependence on the Size of the Image Set



The mean achieved sorting quality as a function of the required computation time for 256 (•), 1024 (●), and 4096 (⬤) RGB random colors for the different sorting methods.

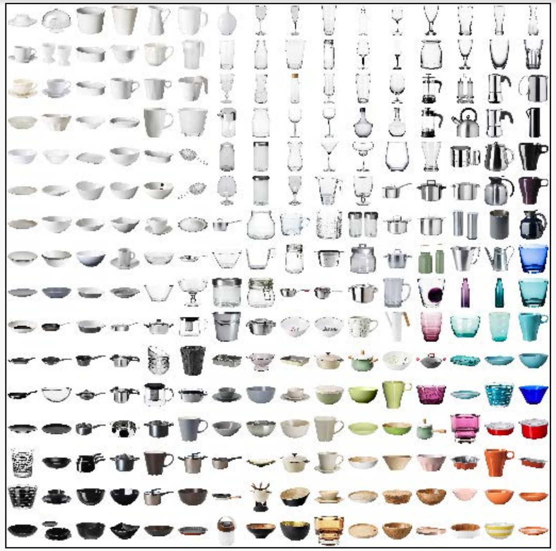
Sorting with Spatial Constraints

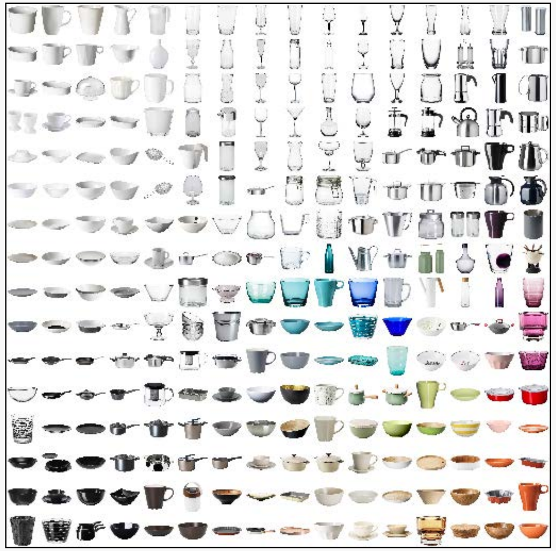
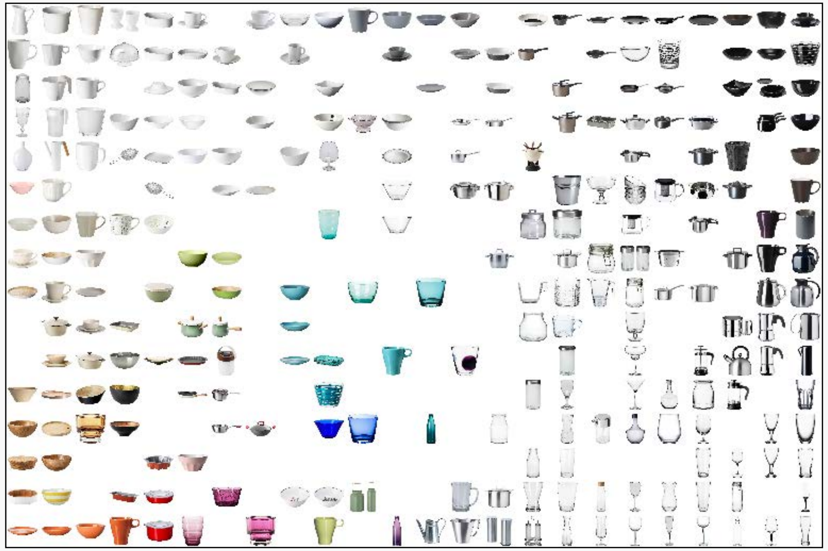
Sorting with special layout requirements

Sometimes there are special requirements for the layout of a sorted arrangement.

- From 2D to 1D and 3D arrangements
  (LAS and FLAS can easily be realized in 1D or 3D)
- Fixed positions of specific images
- Sorting on a larger map (map size > number of images)
- Non-rectangular grid shapes

Fixing the Positions of Specific Images

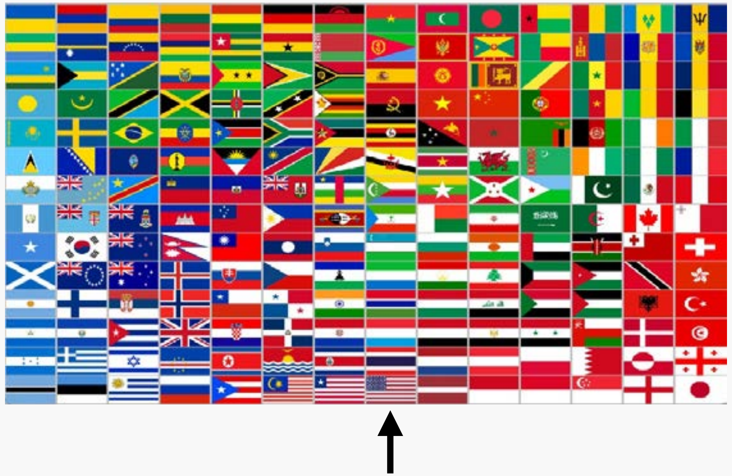


Fixing the Positions of Specific Images

Sometimes it is desirable to fix positions images on the map.
The approach depends on number of images and sorting type.

Sorting Layout	wrapped		non wrapped
Number of fixed images	1	> 1	≥1
Solution	move image to desired position	fix the image(s) at the desired position(s) and use weighted filters	

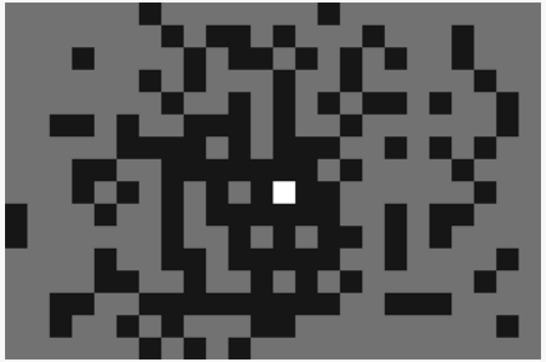


Weighted Filters

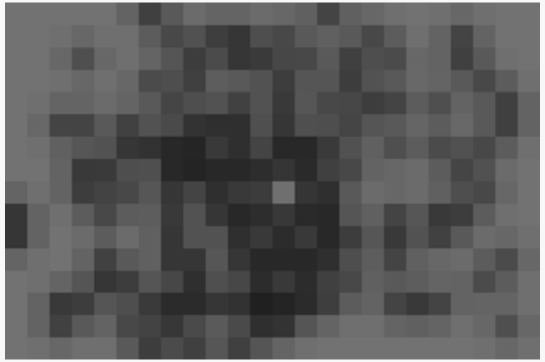
Weighted filters are needed for grids with more positions than images and for fixed image positions. Different weights are used for holes (0.01), normal images (1) and fixed images (10).

Instead of copying the feature vectors to the map and then filtering, feature vectors are scaled with the weights, the scaled feature vectors are filtered and then divided by the filtered weights.

arrangement



weights



filtered weights

Non-Rectangular Grid Shapes

- Non-rectangular grid shapes can be achieved by extending the shape to the bounding box.
- Map positions outside the desired shape may not be assigned and are treated as holes.

unsorted colors

map

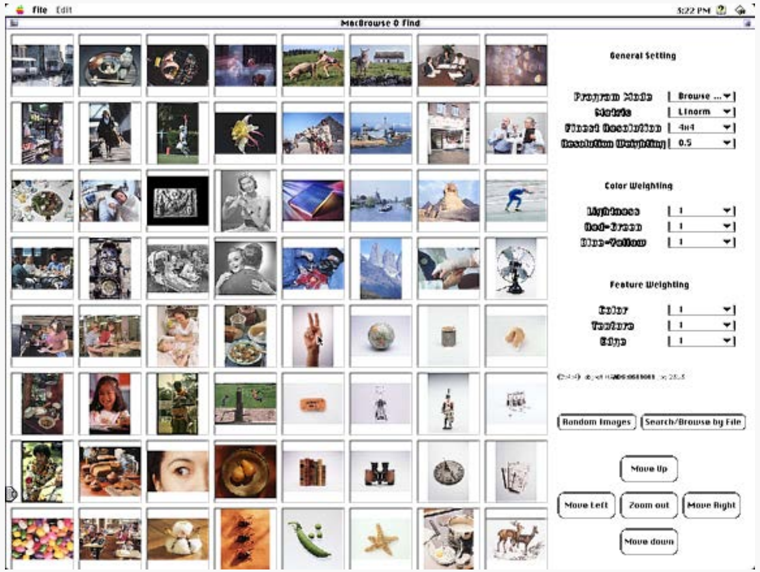
sorted colors in 🖤 shape

Visual Exploration & Navigation of Image Collections

Image Exploration

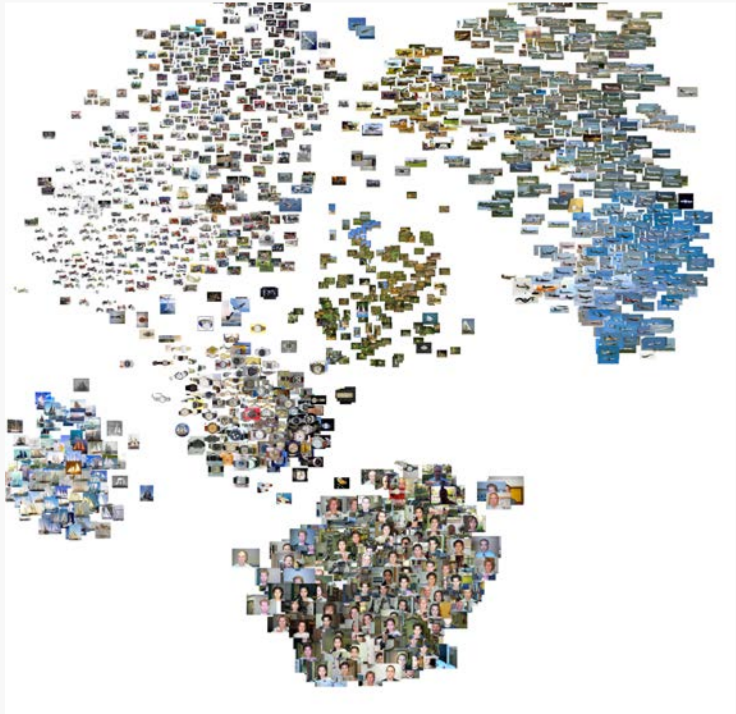
- While many efforts have been made to improve visual similarity search, there is little research for user-driven visual image exploration.
- The FLAS method (together with an image graph) is so fast that it becomes possible to visually explore millions of images.
- Navigu.net is an example of such a visual image exploration tool.

Image Exploration Examples



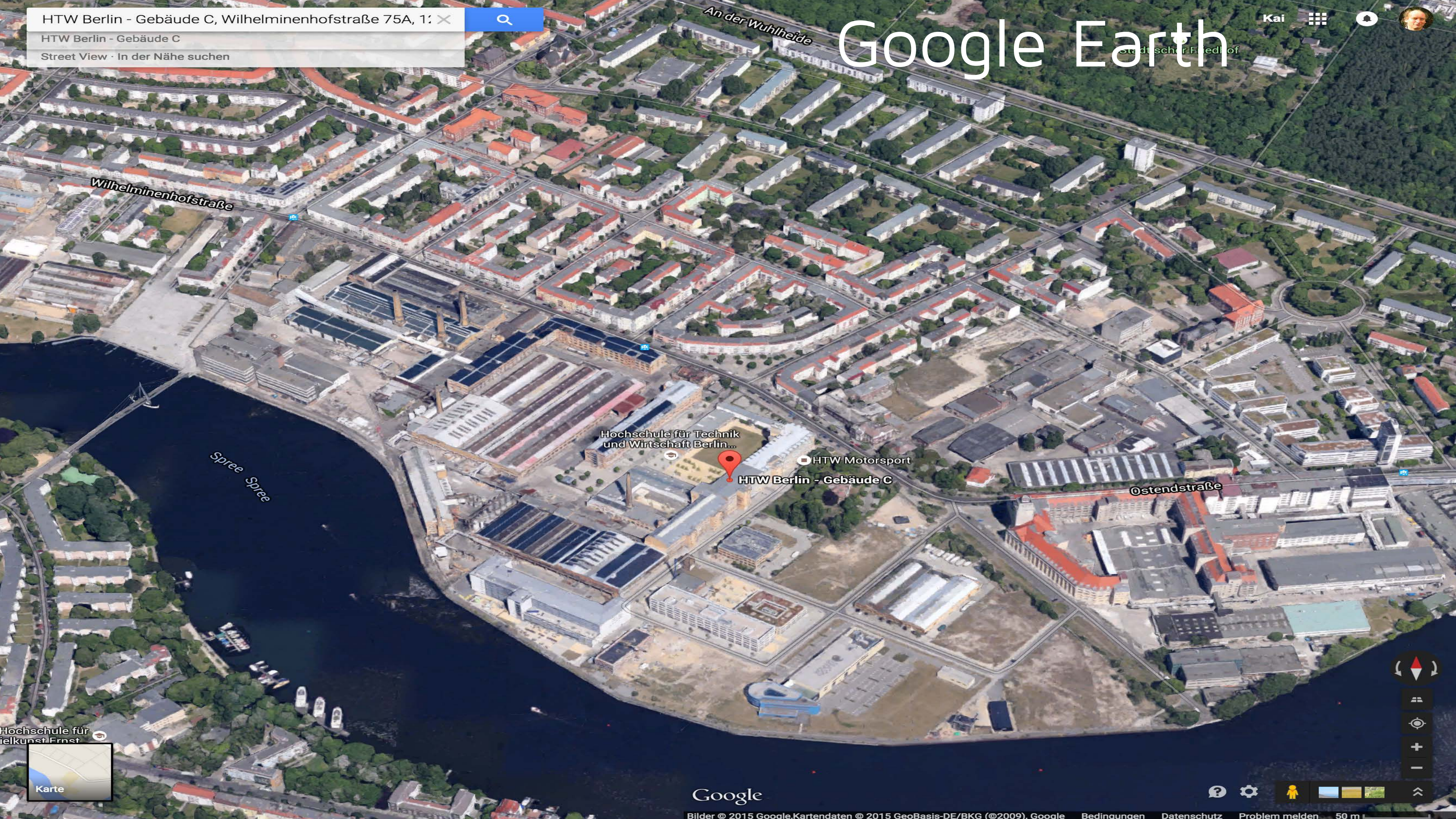
1998, Chen, "Similarity Pyramids for Browsing and Organization of Large Image Databases"

2009, Google Image Swirl "A Large-Scale Content-Based Image Visualization System"



2008, van der Maaten, "Visualizing Data using t-SNE"

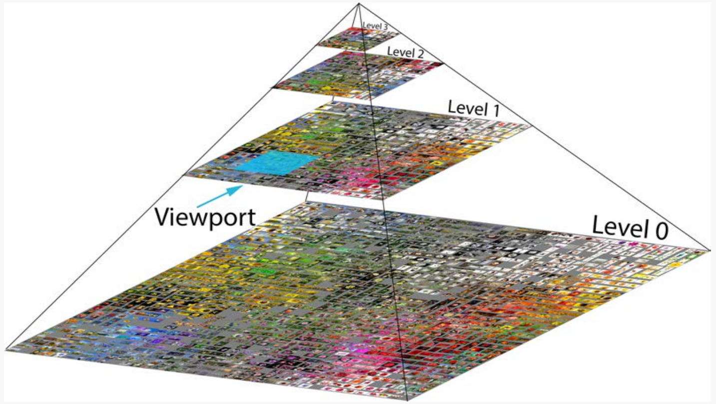
Google Earth

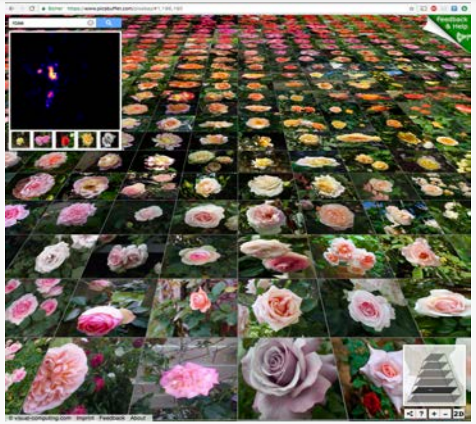



picsbuffet.com

Motivation

Previously we proposed
picsbuffet.com
a pyramid based image exploration system





- \+ Suited for very large image sets
- \+ Good visualization, fast & easy navigation
- – No support for dynamically changing image sets
- – Image relationships cannot be preserved with a static 2D map

Idea: Combine the idea of picsbuffet with graph-based browsing:

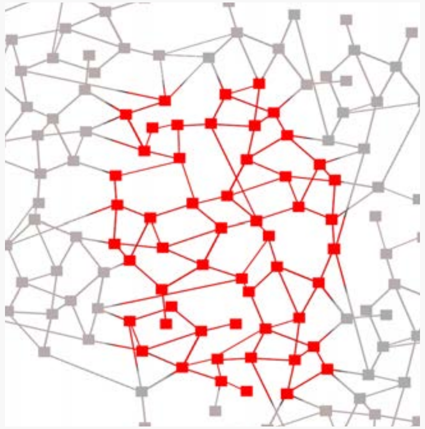
→ **Visual navigation using hierarchical image graphs**

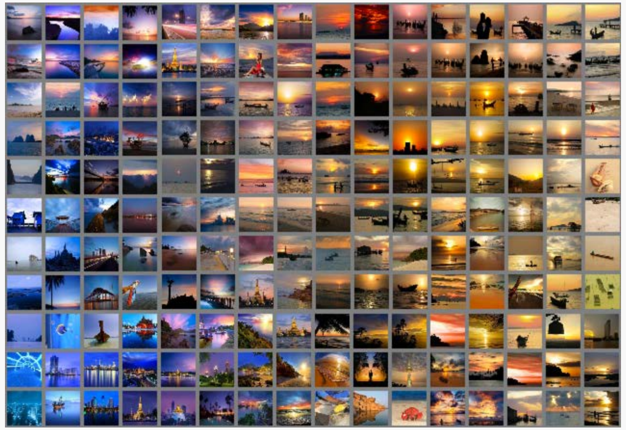
Three different feature vectors

- **Text-to-image retrieval**:
  CLIP feature vector
- **Image-to-image retrieval**:
  Visual search feature vector
- **Visual Sorting**:
  Low-level feature vector
  describing color and texture

Our graph visualization scheme

- Subsets of images are successively retrieved from the image similarity graph and displayed as a visually sorted 2D image map.

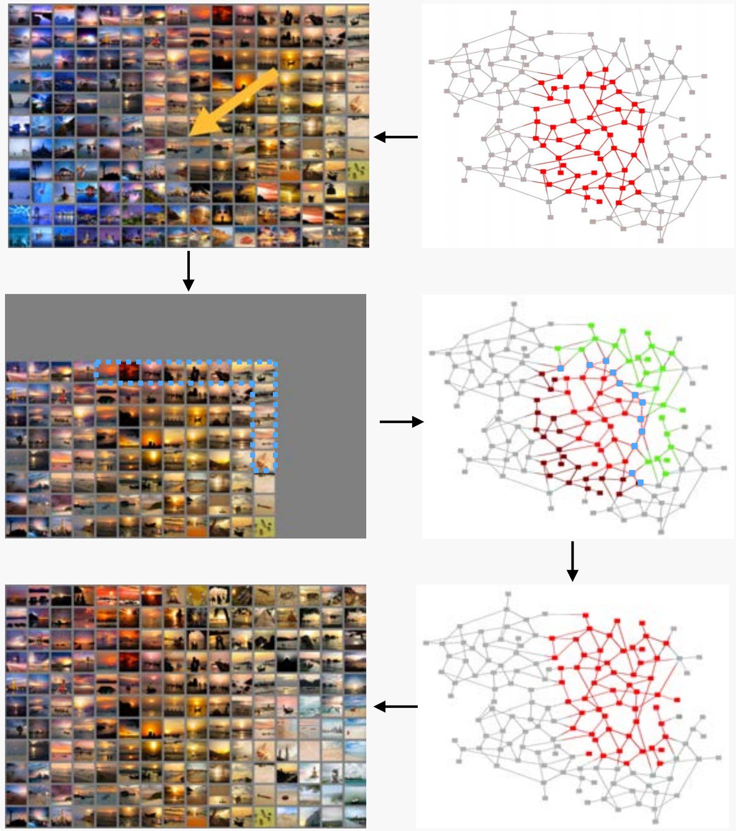




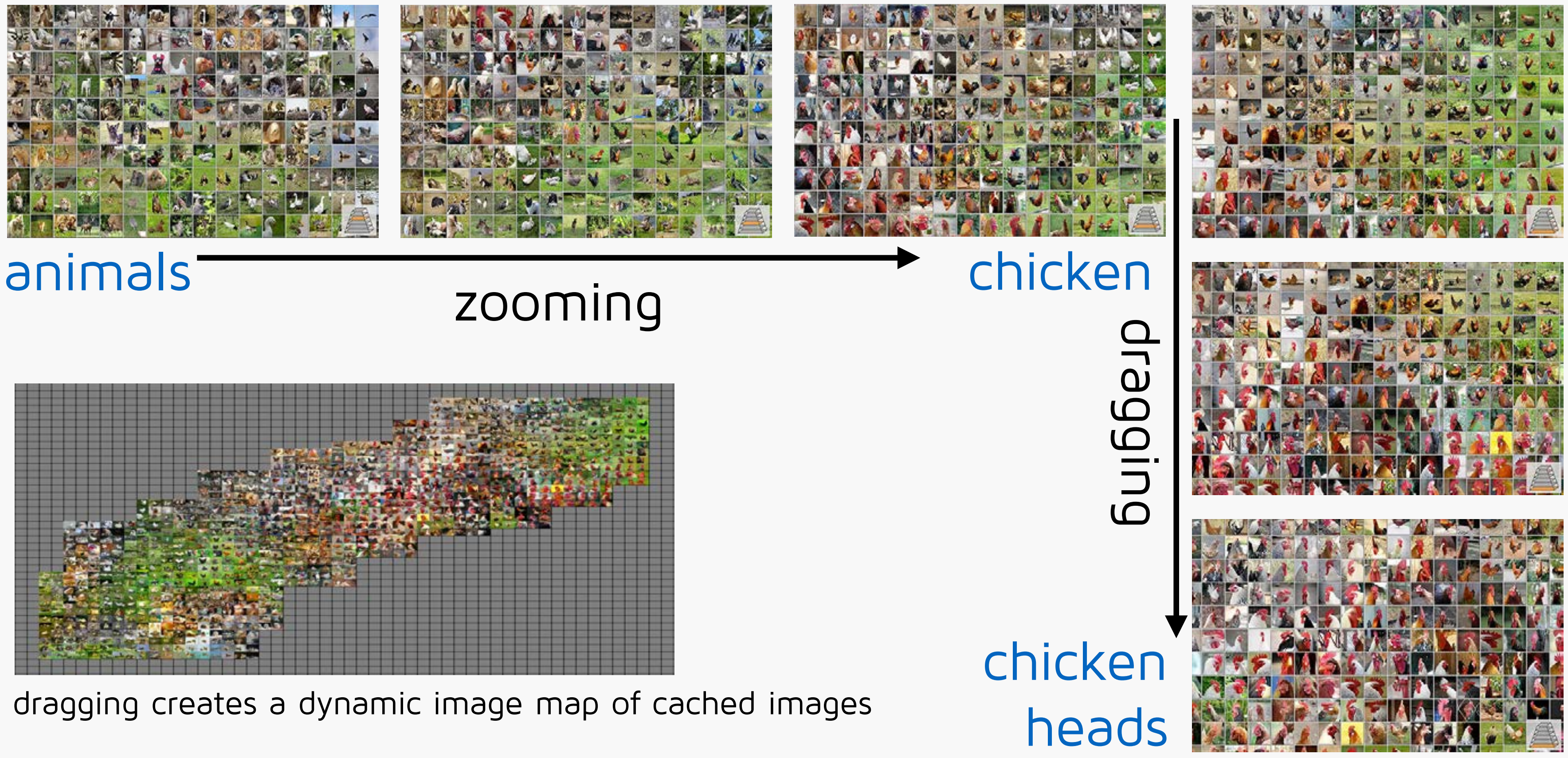
- The map can be zoomed and dragged to explore related image concepts.
- This approach allows an easy image-based navigation, while preserving the complex image relationships of the graph.

Graph navigation

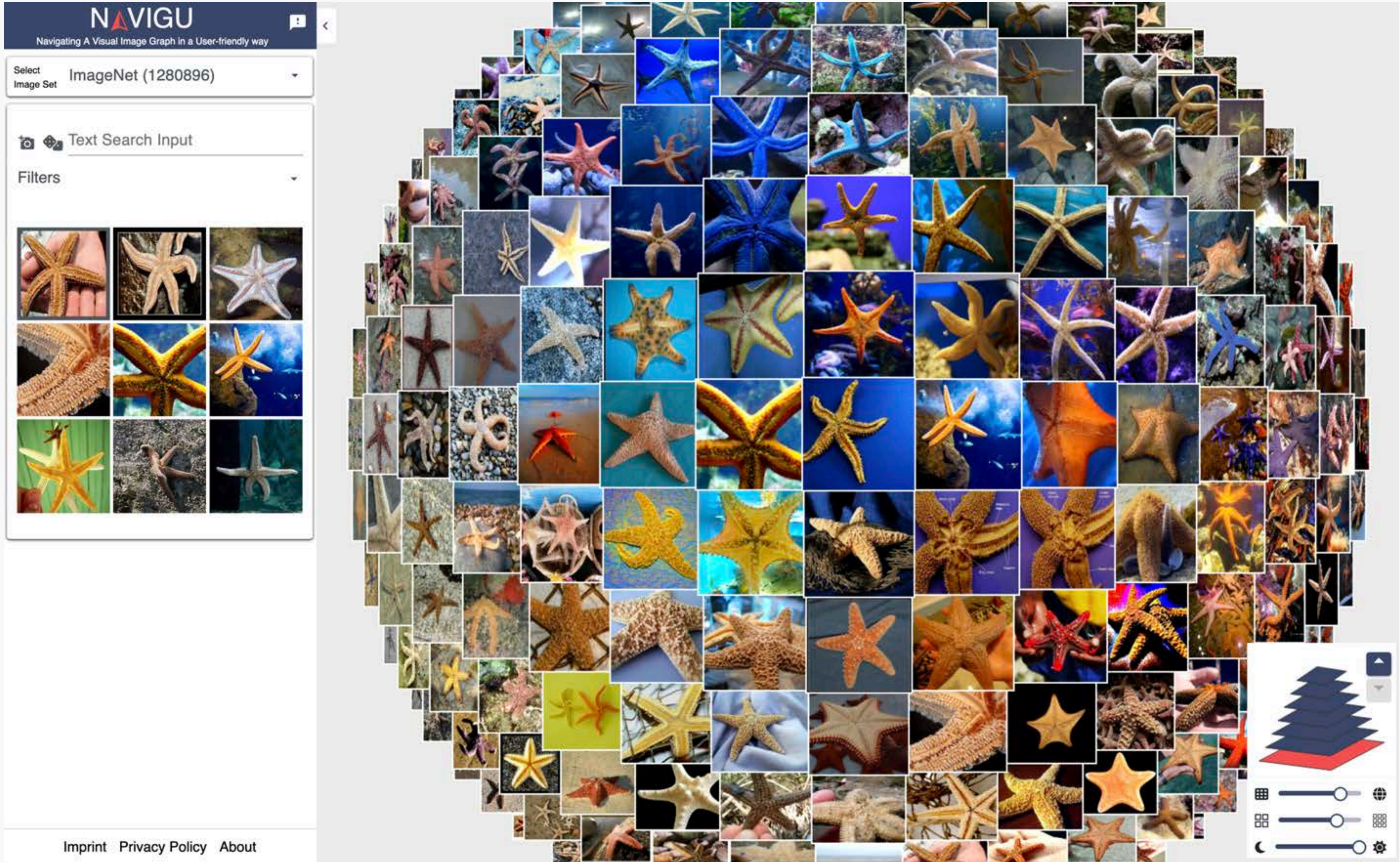
- Dragging images out of view leaves an empty space on the opposite side.
- The images next to this space indicate images of interest.
- Following the graph-edges of these images, new images are retrieved.
- New images are placed visually sorted into the empty map region. Positions of previously displayed images remain unchanged.



Visual navigation by zooming and dragging



Try our demo: navigu.net

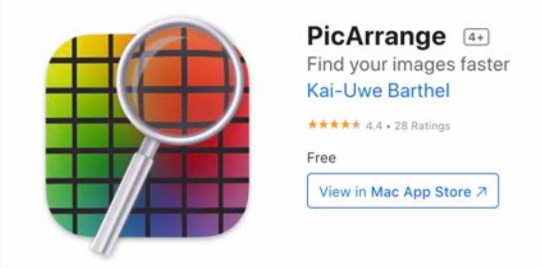


Thank you for your attention!

- An article about the image sorting experiment can be found here:
  https://uxdesign.cc/the-image-sorting-experiment-4ac425812ee6

- More details about DPQ, LAS and FLAS can be found in our paper:
  Improved Evaluation and Generation of Grid Layouts Using Distance Preservation Quality and Linear Assignment Sorting,
  2022, K. U. Barthel, N. Hezel, K. Jung, K. Schall
  https://onlinelibrary.wiley.com/doi/full/10.1111/cgf.14718

- https://github.com/Visual-Computing/LAS_FLAS



www.visual-computing.com

Our Apps: