Improved Evaluation and Generation Of Grid Layouts Using Distance Preservation Quality and Linear Assignment Sorting

K. U. Barthel, N. Hezel, K. Jung and K. Schall

Visual Computing Group, HTW Berlin, Berlin, Germany
{barthel, hezel, Klaus.Jung, Konstantin.Schall}@htw-berlin.de

Abstract
Images sorted by similarity enables more images to be viewed simultaneously, and can be very 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. Various metrics exist for evaluating such arrangements, but there is low experimental evidence on correlation between human perceived quality and metric value. We propose distance preservation quality (DPQ) as a new metric to evaluate the quality of an arrangement. Extensive user testing revealed stronger correlation of DPQ with user-perceived quality and performance in image retrieval tasks compared to other metrics. In addition, we introduce Fast linear assignment sorting (FLAS) as a new algorithm for creating visually sorted grid layouts. FLAS achieves very good sorting qualities while improving run time and computational resources.

Keywords: interaction, user studies, visualization, information visualization, high dimensional sorting, assistive interfaces

CCS Concepts: • Human-centred computing → Visualization design and evaluation methods; Empirical studies in visualization; • Information systems → Presentation of retrieval results; • Theory of computation → Sorting and searching

1. Introduction

It is difficult for humans to view large sets of images simultaneously while maintaining a cognitive overview of its content. As set sizes increase, the viewer quickly loses their perception of specific content contained in the set (Figure 1 left). For this reason, most applications and websites typically display no more than 20 images at a time, which in many cases is only a tiny fraction of the images available. However, if the images are sorted according to their similarity, up to several hundred can be perceived simultaneously. It has been shown that a sorted arrangement helps users to identify regions of interest more easily and thus find the images they are looking for more quickly [SA11, QKTB10, RRS13, HZL*15]. The simultaneous display of larger image sets is particularly interesting for e-commerce applications and stock photo agencies.

In order to be able to sort images according to their similarity, a suitable measure of this similarity must be specified. Image analysis methods can generate visual feature vectors and image similarity is then expressed by the similarity of their feature vectors. While low-level feature vectors generated by classical image anal-

ysis techniques represent the general visual appearance of images (such as colours, shapes and textures), vectors generated with deep neural networks can also describe the content of images [BSCL14, ZIE*18, RTC19, CAS20]. The dimensions of these vectors are on the order of a few tens for low-level features, while deep learning vectors generated with neural networks typically have up to thousands of dimensions.

If the images are represented as high-dimensional (HD) vectors, their similarities can be expressed by appropriate visualization techniques. A variety of dimensionality reduction techniques have been proposed to visualize HD data relationships in two dimensions. Often a distinction is made between methods that use vectors or pairwise distances. However, these methods can be converted from one another; pairwise distances can be calculated from the vectors, and the rows of a distance matrix can be used as vectors.

Numerous techniques (*principal component analysis* (PCA) [Pea01], *multi-dimensional scaling* (MDS) [Sam69], *locally linear embedding* (LLE) [RS00], *Isomap* [TdSL00] and others) are described in Sarveniazi [Sar14]. Other methods that work very well are *t-distributed stochastic neighbourhood embedding* (t-SNE)
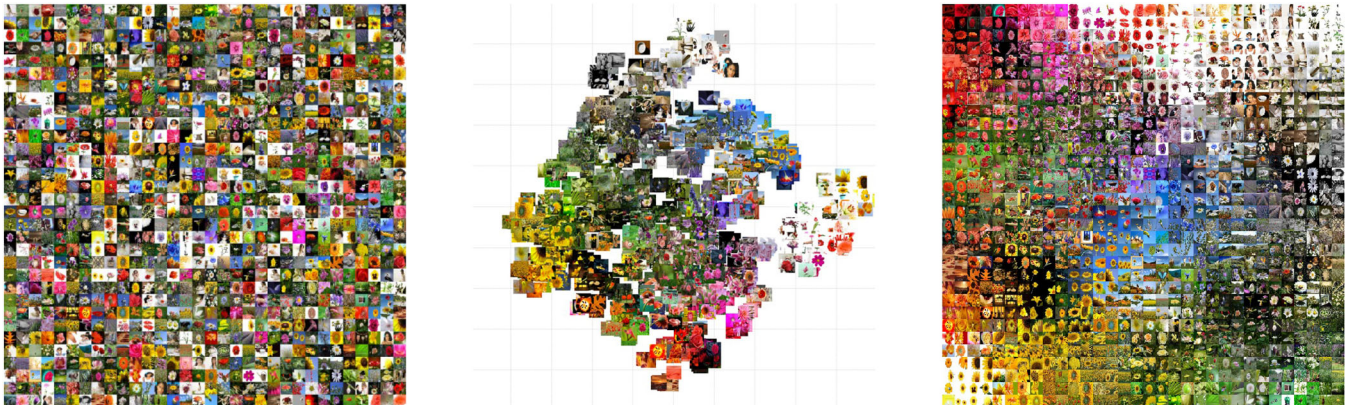
Figure 1: *1024 images tagged with "flower". Left: grid in random order. Centre: t-SNE projection. Right: arranged with LAS.*

[vdMH08], *uniform manifold approximation and projection* (UMAP) [MH18] and *subset embedding networks* [XTL*21].

Visualization is achieved by projecting the HD data onto a two-dimensional plane. However, all of the above techniques are of limited use if the images themselves are to be displayed. The centre of Figure 1 shows a t-SNE projection of the relative similarity of 1024 flower images. Due to the dense positioning of the projected images, some overlap and are partially obscured. Furthermore, only a fraction of the display area is used. Using techniques such as *DGrid* [HMJE*21] would solve the overlap problem, but would still not make best use of the available space.

To arrange or sort a set of images by similarity while maximizing the display area used, three requirements must be satisfied:

1. The images should not overlap.
2. The image arrangement should cover the entire display area.
3. The HD similarity relationships of the image feature vectors should be preserved by the 2D image positions.

Requirements 1 and 2 can only be met if the images are positioned on a rectangular grid. For the 3rd requirement, the images have to be positioned such that their spatial distance corresponds as closely as possible to the HD distance of their feature vectors, despite the given grid structure.

The *self-organizing map* is one of the oldest methods for organizing HD vectors on a grid [Koh82, Koh13]. *Self-sorting maps* [SG11, SG14] are a more recent technique that orders images using a hierarchical swapping method. Other approaches first project the HD vectors to two dimensions, which are then mapped to the grid positions. Various metrics exist for assessing the quality of such arrangements, but there is little experimental evidence of correlation between human-perceived quality and these metrics.

In our paper, we first describe other existing quality metrics for evaluating sorted grid layouts, then we give an overview of existing algorithms for generating sorted 2D grid layouts.

The key contributions of our work are: 1. Inspired by the *k-neighbourhood preservation index* [FFDP15], we propose *distance preservation quality (DPQ)* as a new metric for evaluating grid-

based layouts. 2. We then propose *linear assignment sorting (LAS)*, an algorithm that very efficiently produces high-quality 2D grid layouts. 3. We conducted an extensive user study examining different metrics and show that distance preservation quality better reflects the quality perceived by humans. We furthermore performed qualitative and quantitative comparisons with other sorting algorithms. 4. In the last section, we show how to generate arrangements with special layout constraints with our proposed sorting method.

A preprint of this paper has been published in Barthel *et al.* [BHJS22]. This paper is based on our previous work [BH19], in which we proposed a predecessor of the arrangement quality metric and a combination of SOM and SSM. The differences between this paper and these previous approaches are described in Sections 3.2 and 4.1.

2. Related Work

2.1. Quality evaluation of distance preserving grid layouts

A high-quality image arrangement is one that provides a good overview, places similar images close to each other and images being searched can be found quickly. An evaluation metric expresses the quality of a sorted arrangement with a single number. This value should highly correlate with the quality perceived by humans. We review commonly used evaluation metrics and examine their properties and problems.

Grid-based arrangement of HD data X consists of finding a mapping (a sorting function) $S : X \mapsto Y$ or $S : x_i \mapsto y_i$, where x_i is the ith HD vector, whereas y_i is the ith position vector on the grid in $\mathbb{R}^2$. The distance between HD vectors is denoted by $\delta(\cdot, \cdot)$ whereas $\lambda(\cdot, \cdot)$ denotes the corresponding spatial distance of positions of the 2D grid.

Mean average precision. The mean average precision (mAP) is the commonly used metric to evaluate image retrieval systems.

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

Figure 2: *An example of two arrangements of colours with the same mAP values, where people would rate the sorting quality unequally.*

AP is the average precision, N the number of total images, m_q the number of positive images per class. $P_q(k)$ represents the precision at rank k for the query q, $rel_q(k)$ is a binary indicator function (1 if q and the image at rank k have the same class and 0 otherwise).

The mAP metric defines a sorting as 'good' if the nearest neighbours belong to the same class. In most cases, the mAP cannot be used because typically images do not have class information. Another problem is that mAP only considers images of the same class and ignores the order of the other images (see Figure 2).

***k*-Neighbourhood preservation index**. The k-neighbourhood preservation index $NP_k(S)$ is similar to the mAP in that it evaluates the extent to which the neighbourhood of the HD data set X is preserved in the projected grid Y. It is defined as

$$NP_k(S) = \frac{1}{N} \sum_{i=1}^{N} \frac{|\mathcal{N}_{k,i}^{HD} \cap \mathcal{N}_{k,i}^{2D}|}{k} \tag{2}$$

where k is the number of considered neighbours, $\mathcal{N}_{k,i}^{HD}$ is the set of the k nearest neighbours of x_i in the HD space, whereas $\mathcal{N}_{k,i}^{2D}$ is the set of the k nearest neighbours to y_i on the 2D grid.

The k-neighbourhood preservation index has several problems: The quality of an arrangement is not described by a single value, but by individual values for each neighbour size k. Because of the discrete 2D grid, many spatial distances λ are equal, which means that there is no unique ranking of the grid elements. However, the biggest problem is a high sensitivity to noisy or similar distances.

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

$$CC(S) = \sum_{i=1}^{N} \sum_{j=1}^{N} \frac{(\lambda(y_i, y_j) - \bar{\lambda})(\delta(x_i, x_j) - \bar{\delta})}{\sigma_\lambda \sigma_\delta} \tag{3}$$

The main problem of cross-correlation is that differences of large distances have a higher impact than differences of small distances. It may be problematic to assess the quality of an image arrangement with cross-correlation as it is equally important to maintain both small and large distances to keep similar images together and prevent dissimilar images from being arranged next to each other.

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

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

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

$$E'_p(S) = 1 - E_p(S) \tag{4}$$

The normalized energy function has essentially the same properties and problems as cross-correlation. The parameter p can be used to tune the balance between small and large distances. Usually, p values of 1 or 2 are used. Throughout this paper, we use E'_p with a range of $[0,1]$ with larger values representing better results.

2.2. Algorithms for Sorted Grid Layouts

2.2.1. *Grid arrangements*

Since our new sorting method is based on both self-organizing map and self-sorting map, we present them here in more detail.

A **self-organized map** (SOM) uses unsupervised learning to produce a lower dimensional, discrete representation of the input space. A SOM consists of a rectangular grid of map vectors M having the same dimensionality as the input vectors X. To adapt a SOM for image sorting, the input vectors must all be assigned to different map positions, since multiple assignments would result in overlapping images. Algorithm 1 describes the SOM sorting process.

A **self-sorting map** (SSM) arranges images by initially filling cells (grid positions) with the input vectors. Then for sets of four cells, a hierarchical swapping procedure is used by selecting the best permutation from $4! = 24$ swap possibilities. Algorithm 2 describes the sorting process with a SSM. In [SJGE13], an alternative to SSMs is described that uses more sophisticated swapping strategies to achieve better global correlation, but at a much higher computational cost.

Level-of-detail grid (LDG) is a recent method for creating hierarchical grid layouts [Fre22]. A progressive optimization method based on local search generates hierarchical grids. The method is based solely on pairwise distances and jointly optimizes homogeneity within interior nodes and between grid neighbours.

Algorithm 2. SSM

1: Copy all input vectors into random but unique cells of the grid
2: Divide the grid into 4x4 blocks
3: **while** size of the blocks ≥ 1 **do**
4:     Divide each block into 2x2 smaller blocks
5:     **for** iteration $= 1, 2, \ldots L$ **do** // L = maximum number of iterations
6:         For each block its target vector (the mean vector of its cells and adjacent blocks' cells) is calculated
7:         **for** all blocks **do**
8:             **for** all cells of the block **do**
9:                 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

---

2.2.2. *Graph matching*

Kernelized sorting (KS) [QSS08] and *Convex KS* [DGV12] generate distance-preserving grids and find a locally optimal solution to a quadratic assignment problem [BK57]. KS creates a matrix of pairwise distances between HD data instances and a matrix of pairwise distances between grid positions. A permutation procedure on the second matrix modifies it to approximate the first matrix as well as possible, resulting in a one-to-one mapping between instances and grid cells.

IsoMatch also uses an assignment strategy to construct distance preserving grids [FDH*15]. First, it projects the data into the 2D plane using the Isomap technique [TdSL00] and creates a complete bipartite graph between the projection and the grid positions. Then, the Hungarian algorithm [Kuh55] is used to find the optimal assignment for the projected 2D vectors to the grid positions. IsoMatch uses the normalized energy function E_1 trying to maximize the overall distance preservation.

Similarly, DS++ presents a convex quadratic programming relaxation to solve this matching problem [DML17]. KS, IsoMatch and DS++ are not limited to rectangular grids. They can create layouts of any shape. As with IsoMatch, any other dimensionality reduction methods (such as t-SNE or UMAP) can be used to first project the HD input vectors onto the 2D plane, and then re-arrange them on the 2D grid. A fast placing approach can be found in Hilasaca *et al.* [HMJE*21]. Any linear assignment scheme like the Jonker–Volgenant Algorithm [JV87] can be used to map the projected 2D positions to the best grid positions. Many non-linear dimensionality reduction methods have been recently proposed, but the question of their assessment and comparison remains open. Methods comparing HD and 2D ranks are reviewed in Refs. [LV08, LMBH11].

3. A New Quality Metric for Grid Layouts

Our goal is to develop a metric that better reflects perceived quality. The quality is to be expressed with a single value, where 0 stands for a random and 1 for a perfect arrangement. There are two approaches when designing a suitable quality function for grid layouts. The first option would be to refer to the best possible 2D sorting that can theoretically be achieved. However, this approach is not applicable because the best possible sorting is usually not known. The

Figure 3: *Two examples of best possible 2D arrangements for 64 colour samples. Left: The perfect HD (3D) order is preserved. Right: Four different colours with 16 samples each, here the HD order cannot be preserved. Colours adjacent to a different colour on the 2D grid are not near their nearest HD neighbours.*

only viable way is to refer to the distribution of the HD data. A perfect sorting here means that all 2D grid distances are proportional to the HD distances. However, depending on the specific HD distribution, it is usually not possible to achieve this perfect order in a 2D arrangement (see Figure 3).

3.1. Neighbourhood preservation quality

Our initial approach towards a new evaluation metric was to combine the k-neighbourhood preservation index values $\mathrm{NP}_k(S)$ to a single quality value. The $\mathrm{NP}_k(S)$ values for a perfect arrangement S_{opt} and the expected value for random arrangements S_{rand} are

$$\mathrm{NP}_k(S_{\mathrm{opt}}) = 1 \qquad \mathbb{E}[\mathrm{NP}_k(S_{\mathrm{rand}})] = \frac{k}{K} \qquad (5)$$

where k is the evaluated neighbourhood size, K is the maximum number of neighbours which is the number of HD data elements-1. The expected NP_k value for a random arrangement is $\frac{k}{K}$, since more and more correct nearest neighbours are found as k increases.

For a given 2D arrangement S, we define the *neighbourhood preservation gain* $\Delta\mathrm{NP}_k^{\mathrm{2D}}(S)$ as the difference between the actual $\mathrm{NP}_k(S)$ value and the expected value for random arrangements.

$$\Delta\mathrm{NP}_k^{\mathrm{2D}}(S) = \max\left(\mathrm{NP}_k(S) - \frac{k}{K}, 0\right) \qquad (6)$$

The maximum is taken because theoretically an arrangement can be worse than a random arrangement. This happens very rarely, but if it does, the negative values are very small. Since an optimal arrangement preserves all HD neighbourhoods perfectly, we define

$$\Delta\mathrm{NP}_k^{\mathrm{HD}} = \Delta\mathrm{NP}_k^{\mathrm{2D}}(S_{\mathrm{opt}}) = 1 - \frac{k}{K} \qquad (7)$$

Figure 4 shows an example of four different primary colours, each used 64 times. All colours were slightly changed by some noise, resulting in 256 different colours. On the left side, two arrangements and the colour histogram are shown. The $\Delta\mathrm{NP}$ curves are shown on the right. Here the optimal HD order cannot be preserved in 2D.

We determine the vectors of neighbourhood preservation gains of the actual 2D arrangement and of the perfect HD arrangement. We define the *neighbourhood preservation quality* $\mathrm{NPQ}_p(S)$ as the ratio of the norms of these vectors. $\mathrm{NPQ}_p(S)$ is close to 0 for a random and 1 for a perfect sorting.

$$\mathrm{NPQ}_p(S) = \frac{\|\Delta\mathrm{NP}^{\mathrm{2D}}(S)\|_p}{\|\Delta\mathrm{NP}^{\mathrm{HD}}\|_p} \qquad 0 \leq \mathrm{NPQ}_p(S) \leq 1 \qquad (8)$$
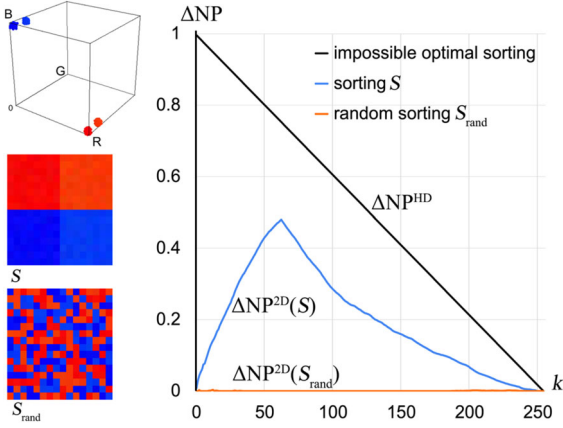
Figure 4: *Left: A 3D RGB histogram of a set of 64 × 4 colours, all slightly modified by noise. Below two arrangements of this set with sorted (S) and random positions (S_{rand}). Right: The corresponding curves of the neighbourhood preservation gain ΔNP. Although the sorting S is rather good, $\Delta NP^{2D}(S)$ is very low for small k values.*

Each 2D grid position has several other positions with the same spatial distance λ (e.g. the four nearest neighbours with a distance of 1). To determine $NP_k(S)$ for all k-values, the mean k-neighbourhood preservation index values must be determined for equal spatial distances. This ensures that isometries (rotated or mirrored arrangements) result in equal NPQ values.

To determine the neighbourhood preservation quality $NPQ_p(S)$, the value for the p-norm must be chosen. Higher values give more weight to NP_k values with smaller k, so the preservation of nearer neighbours becomes more important. In the case of very large p values, only the four adjacent positions are taken into account.

One problem with the proposed neighbourhood preservation quality is its sensitivity to noisy distances in the HD data. This often occurs when using visual feature vectors. As image analysis is not perfect, a feature vector can be considered as a 'perfect' vector that is disturbed by some noise. This effect can be seen in Figure 4. Although sorting S is rather good, the $\Delta NP_k(S)$ values are very low, especially for near neighbours. The top row of Figure 5 shows the resulting orde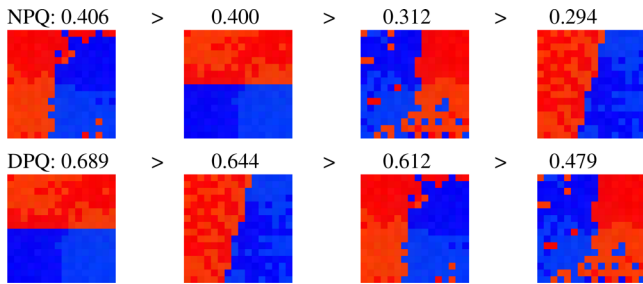r when ranking different arrangements of this data set according to their NPQ values for $p = 2$. It can be seen that NPQ does not reflect the perceived sorting quality well.

3.2. Distance preservation quality

The problem of the proposed neighbourhood preservation quality consists of the fact that only the correct ranking of the neighbours is taken into account. The actual similarity of wrongly ranked neighbours is not considered. To address the noise-induced degradation of the neighbourhood preservation quality, we propose not to compare the correspondence of the closest neighbours, but to compare the averaged distances of the corresponding neighbourhoods $\mathcal{N}_{k,i}^{HD}$ and $\mathcal{N}_{k,i}^{2D}$. For this, the *average neighbour distances* for the k closest neighbours are determined in HD and 2D:

$$D_k^{HD} = \frac{1}{kN} \sum_{i=1}^{N} \sum_{j \in \mathcal{N}_{k,i}^{HD}} \delta(x_i, x_j) \qquad D_k^{2D}(S) = \frac{1}{kN} \sum_{i=1}^{N} \sum_{j \in \mathcal{N}_{k,i}^{2D}} \delta(x_i, x_j) \quad (9)$$

It should be noted that the distances δ of the HD vectors are used for both the HD and the 2D neighbourhoods. The only difference is that the sets of the actual k nearest neighbours in HD and 2D are not the same if the 2D arrangement is not optimal.

Similar to the neighbourhood preservation quality, we compare the average neighbourhood distance with the expectation value of the average neighbourhood distance of random arrangements, which is equal to the global average distance $\overline{D}$ of all HD vectors x_i.

$$\mathbb{E}[D_k^{2D}(S_{rand})] = \overline{D} = \frac{1}{N(N-1)} \sum_{i=1}^{N} \sum_{j=1}^{N} \delta(x_i, x_j) \qquad (10)$$

Analogous to $\Delta NP_k^{2D}(S)$, we define the *distance preservation gain* ΔD_k as the difference between the average neighbourhood distance of a random arrangement and the sorted arrangement.

$$\Delta D_k^{HD} = \frac{1}{\overline{D}}(\overline{D} - D_k^{HD}) \qquad \Delta D_k^{2D}(S) = \max(\frac{1}{\overline{D}}(\overline{D} - D_k^{2D}(S)), 0) \quad (11)$$

Compared to ΔNP, the order of subtraction is reversed for ΔD, since a higher distance is considered instead of a lower neighbour preservation. Taking the difference between $\overline{D}$ and the average neighbour distance ensures $\Delta D_k^{2D}(S_{rand})$ is approximately 0 for random arrangements. In theory, the division by $\overline{D}$ is not necessary, but limiting the values to a range from 0 to 1 improves the numerical stability when calculating the norm of the distance preservation gain for larger p values. Figure 6 shows the ΔD curves of the previous example. It shows that for the sorted arrangement S, the $\Delta D_k^{2D}(S)$ values are much higher for small neighbourhoods k, indicating that close neighbours on the grid are similar. For neighbours with equal 2D distances, the mean of the corresponding HD distances was used.

The *distance preservation quality* $DPQ_p(S)$ is defined as the ratio of the p-norms of the distance preservation gains of the actual arrangement to a perfect arrangement:

$$DPQ_p(S) = \frac{\|\Delta D^{2D}(S)\|_p}{\|\Delta D^{HD}\|_p} \qquad 0 \le DPQ_p(S) \le 1 \qquad (12)$$

NPQ: 0.406 > 0.400 > 0.312 > 0.294

DPQ: 0.689 > 0.644 > 0.612 > 0.479

Figure 5: *Arrangements in the order of NPQ_2 (above) and DPQ_2 (below). It can be seen that the NPQ order does not match the human perception of sorting quality.*
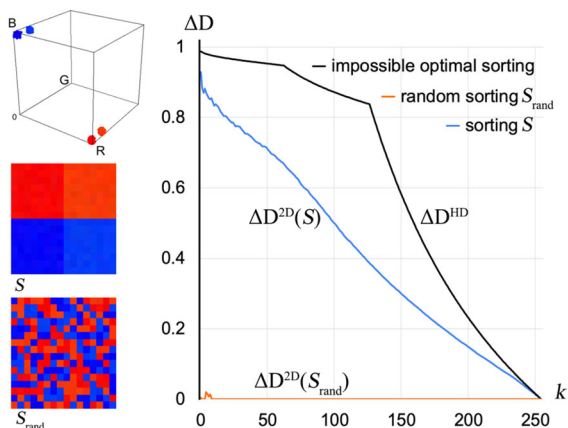
Figure 6: *The same arrangements as in Figure 4. On the right side, the curves show the distance preservation gain ΔD.*

For a random arrangement, $DPQ_p(S_{rand})$ will be approximately 0, for a perfect arrangement, $DPQ_p(S_{opt})$ will be 1. The influence of p is evaluated in the user study in Section 5.

In Barthel and Hezel [BH19], we have proposed a predecessor of this approach. For all vectors, the differences between the global average distance and the weighted distances to their neighbourhood vectors was determined in HD and in 2D. A *projection quality* was obtained as the ratio of the means of these differences in HD and 2D. A Gaussian $\mathcal{N}(0, \sigma^2)$ based on the normalized neighbour ranks was used as the weighting function. The weighting of distances for higher ranks could be controlled by σ. Both, the choice of the weighting function and its parametrization were in some sense arbitrary.

Again, the problem of equal spatial distances must be considered when determining the average distances for the k nearest neighbours on the grid. There are two ways to approach this: One is to use the mean HD distance for neighbours with equal 2D distance. The other possibility is to sort these neighbours by their HD distance. The former would be a pessimistic estimate of D_k^{2D}, whereas the latter would be an optimistic estimate (see Figure 7). The use of mean
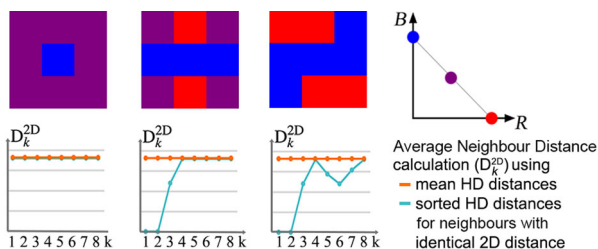

Figure 7: *Three neighbour constellations for the blue colour in the centre. The distance from blue to purple is equal to the mean of the distances to red and blue. Here D_k^{2D} for $k = 1 \dots 8$ is the same when using the mean HD distances for neighbours with equal 2D distances. Sorting these neighbours by their HD distance results in lower D_k^{2D} values. This would better describe human perception if they preferred the right constellation to the left.*

HD distances for equal 2D distances is denoted as DPQ_p^-. Whereas DPQ_p denotes the use of sorted HD distances. Figure 5 shows a better ranking of arrangements when evaluated with DPQ quality than with NPQ (for $p = 2$).

4. Our new Sorting Algorithm: Linear Assignment Sorting

First, we show how SOM and SSM can be optimized for speed and quality, which in combination leads to our new sorting scheme.

4.1. Speed and quality optimizations of SOM and SSM

The SOM described in Section 2.2 assigns each input vector to the best map vector and updates its neighbourhood. The map update can be thought of as a blending of the map vectors with the spatially low-pass filtered assigned input vectors, where the filter radius corresponds to the current neighbourhood radius. We propose to replace this time-consuming updating process: First, all input vectors are copied to the most similar unassigned map vector. Then, all map vectors are spatially filtered using a box filter. It is possible to achieve constant complexity independent of the kernel size by using uniform or integral filters [Lew94, VJ01]. 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.

The SSM avoids the problem of isolated, bad assignments by swapping the assigned positions of four input vectors at a time. To find the best swap, the SSM uses a brute force approach that compares the four input vectors with the four mean vectors of the blocks to which each swap candidate belongs. Due to the factorial number of permutations, adding more candidates would be computationally too complex. In order to still be able to use more swap candidates, we propose optimizing the search for the best permutation by linear programming. Another problem of the SSM is the use of a single mean vector per block, which incorrectly implies that all positions in the block are equivalent when they are swapped. The usage of a single mean vector per block can be considered as a sub-sampled version of the continuously filtered map vectors. Therefore, in Barthel and Hezel [BH19], we proposed using map filtering without sub-sampling, as this allows a better representation of the neighbourhoods of the map vectors. 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.

4.2. Linear assignment sorting

Our proposed new (image) sorting scheme called *linear assignment sorting* (LAS) combines ideas from the SOM (using a continuously filtered map) with the SSM (swapping of cells) and extends this to optimally swapping all vectors simultaneously. The principle of the LAS algorithm can be described as follows: 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 neighbourhoods. In the next step, all input vectors are assigned to their best matching map positions. This is done by

Algorithm 3. LAS

```
1: Set  r_f = ⌊max(W,H) · f_r0⌋     // initial filter radius (f_r0 ≤ 0.5)
        f_r                         // radius reduction factor (f_r < 1)
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 r_f
5:      Find the optimal assignment for all input vectors (acc. to Eq. 13)
6:      Copy all input vectors to the map vectors of their new positions
7:      Reduce the filter radius: r_f = r_f · f_r
```

finding the optimal solution by minimizing the cost C:

$$C = \sum_i^N \sum_j^N a_{ij} c_{ij} \quad \text{with} \quad a_{ij} \in \{0, 1\}, \quad c_{ij} = \|x_i - m_j\|^q$$

$$\text{subject to} \quad \sum_j^N a_{ij} = 1, \quad \sum_i^N a_{ij} = 1 \tag{13}$$

a_{ij} is a binary assignment value, whereas c_{ij} is the distance between the input vectors x_i and the map vectors m_j. The power q allows the distances to be transformed in order to balance the importance of large *versus* small distances. Since the number of possible mappings is factorial, we use the Jonker–Volgenant linear assignment solver [JV87] to find the best swaps with reduced run time complexity $\mathcal{O}(N^3)$ and memory complexity $\mathcal{O}(N^2)$.

The actual sorting is achieved by repeatedly assigning the input vectors and filtering the map vectors with a successively reduced filter radius. The principle of the LAS sorting scheme for a grid of size $N = W \cdot H$ is summarized in Algorithm 3.

The only parameters of the LAS algorithm are the initial filter radius and the radius reduction factor, which controls the exponential decay of the filter radius and thus the quality and/or the speed of the sorting. Examining different q values for transforming the distances between the input and map vectors did not reveal much difference; in the interest of faster computations, we use $q = 2$.

LAS is a simple algorithm with very good sorting quality (see next section for results). However, for larger sets in the range of thousands of images, the computational complexity of the LAS algorithm becomes too high. However, 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, as described in Algorithm 4. This approach allows much faster sorting while having little impact on the quality of the arrangement. Comparisons between LAS and FLAS are given in the next section.

The selection of FLAS parameters allows the control of the quality and speed of the sorting process. In this way, we generated many sorted arrangements of different quality, which were then used in the user study in Section 5. For an example implementation of the LAS and FLAS algorithms, the distance preserving quality DPQ$_p$ together with the images, and the feature vectors used in this paper, see https://github.com/Visual-Computing/LAS_FLAS.

Algorithm 4. FLAS

```
1: Set  r_f = ⌊max(W,H) · f_r0⌋     // initial filter radius (f_r0 ≤ 0.5)
        f_r                         // radius reduction factor (f_r < 1)
        n_c                         // number of swap candidates
        iterations = W · 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, ... iterations do
6:          Select a random position & select n_c random swap candidates
               (assigned input vectors) within a radius of max(r_f, (√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 · f_r
```

5. User Study

5.1. Experiment design

To evaluate the proposed DPQ$_p$ metric and the new sorting schemes (LAS and FLAS), an extensive user study was conducted. In a first experiment, we determined the correlation between user preferences and the quality metrics described in Sections 2.1 and 3.2. In a second experiment, we examined the relationship between the time required to find images in arrangements and the metrics' quality scores and the users' ratings, respectively.

5.1.1. *Image sets*

Figure 8 shows the four image sets used in the experiments. The first set consists of 1024 random RGB colours. The random selection implies that there is no specific low-dimensional embedding that can be exploited to project the data to 2D. While the RGB colour set is a somewhat artificial set, we also used image sets.

In advance, we conducted tests with various image sets of different sizes and colour distributions. It became apparent that some image sets, regardless of the arrangement, were too difficult for users to search. With smaller image sets, on the other hand, the arrangement of the images had little effect on the speed of the search, as the searched images could often be spotted immediately. The three chosen image sets covered different scenarios, where a significant difference in search performance between sorted and random arrangements can be observed. The first set consists of 169 images of traffic signs taken from Pixabay [BS10] in 2017, excluding photos of real traffic signs and nearly identical images. This set contains several groups of visually similar images as might be found when searching for signs or logos. The second set consists of 256 images of kitchen items crawled from the IKEA website in 2016. From a total of 10,262 images, all images with kitchen items were selected. Images with multiple or small objects, duplicates and many very colourful images that are potentially easy to find were removed. This set is an example of what one might find on e-commerce websites. Some of these images are very similar, which makes it difficult to find them. The last set consists of 400 images for 70 unrelated concepts crawled from the Internet. This set was chosen because the low-level feature vectors used in the experiment are capable of
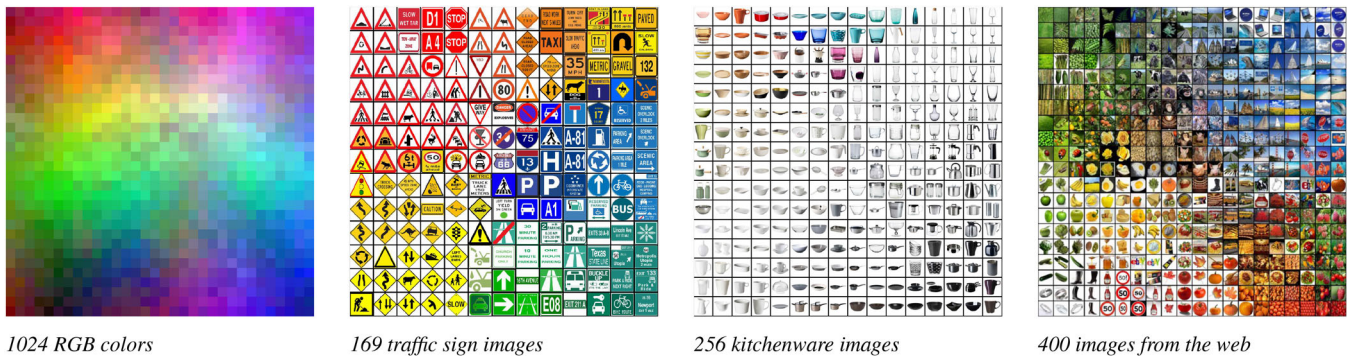
| 1024 RGB colors | 169 traffic sign images | 256 kitchenware images | 400 images from the web |

Figure 8: *The four image sets used in the experiments. The arrangements shown are the ones that received the highest user scores.*
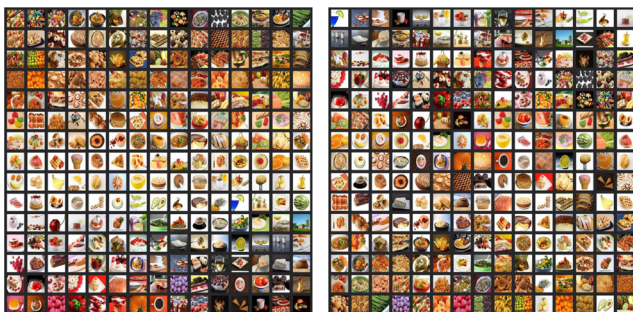

Figure 9: *Left: 256 images tagged with 'food' sorted with low-level feature vectors. Right: the same images sorted with feature vectors from a neural network (MobileNetV3 small [HPA*19]).*

describing images of the same class with similar feature vectors. This ensures that a supposedly bad arrangement is not due to a poor description by the feature vectors used. This set has an uncharacteristically high proportion of coloured images, but there are many images that belong to different object categories despite similar appearance, potentially complicating visual retrieval.

5.1.2. *Feature vectors*

For the RGB colour set, the R, G and B values were taken directly as vectors. Theoretically, the Lab colour space would be better suited for human colour perception, but even the Lab colour space is not perceptually uniform for larger colour differences. To ensure easy reproducibility of the results, we kept the RGB values.

For images, one might expect that feature vectors from neural networks would be best suited to describe them, which is definitely true for retrieval tasks. However, when neural feature vectors are used to visually sort larger sets of images, the arrangements often look somewhat confusing because images can have very different appearances even though they represent a similar concept (see Figure 9). Since people pay strong attention to colours and visually group similar-looking images when viewing larger sets of images, feature vectors describing visual appearance are usually more suitable for arrangements that are perceived as 'well-organized'. For this rea-

son, in our experiment, we used 50 dimensional low-level feature vectors describing the colour layout, the fuzzy YCoCg colour histogram and the MPEG-7 edge histogram of the images. However, the choice of feature vectors has limited impact on the experiments performed, since all sorting methods use the same feature vectors and the metrics indicate how well their similarities are preserved.

5.1.3. *Implementation*

We organized the experiment as online user tests, where participants could take part in a raffle after completing the experiments. It was possible to perform the experiment more than once, but it was ensured that participants would not see the same arrangements twice. In total, more than 2000 people participated in the experiment. About half of them were employees and students of our university, coming from different fields of study, and only a small proportion was from the field of computer science. Little is known about the other participants, as the experiment was advertised as a raffle on various websites.

5.1.4. *Investigated sorting methods and metrics*

In our experiments, we used sorted arrangements generated with the following methods: SOM, SSM, IsoMatch, LAS, FLAS and the t-SNE 2D projection that was mapped to the best 2D grid positions (indicated as t-SNEtoGrid). Several of these generated arrangements were then selected based on the range of variation in sorting results per method. The UMAP method was not investigated because in many cases, its KNN graph broke into multiple components, which made an arrangement onto the 2D grid impossible. In order to also have examples of low quality for comparison, some sorted arrangements were generated with FLAS using poor parameter settings (indicated as Low Qual.).

The evaluated quality metrics were the Energy function E'_1 and E'_2 (Equation 4) and the distance preservation quality DPQ_p (Equation 12) with different p values. As the normalized energy function E'_2 and cross-correlation provide an almost identical quality ranking for different arrangements, we did not evaluate the cross-correlation metric.
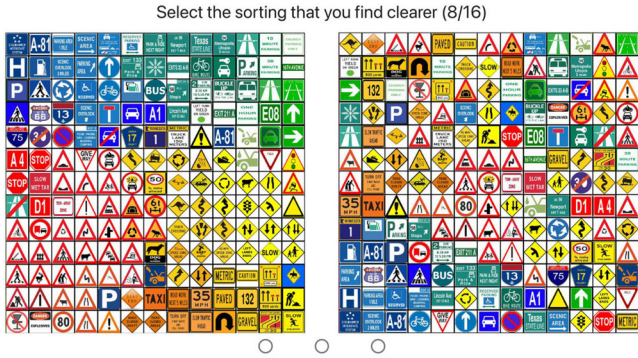
Figure 10: *User preferences. In this study, we ask users to choose the arrangement they preferred.*

5.2. Evaluation of user preferences

In the first experiment, pairs of sorted image arrangements were shown. Users were asked to decide which of the two arrangements they preferred in the sense that 'the images are arranged more clearly, provide a better overview and make it easier to find images they are looking for'.

Figure 10 shows a screenshot of this experiment. 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. To detect misuse, the experiment contained one pair of a very good and a very bad sorting. The decisions of users who preferred the bad sorting here were discarded. The number of different arrangements were 32 for the colour set and 23 each for the three image sets, (giving 496 pairs for the colour set and 253 pairs for each image set). Each pair was evaluated by at least 35 users.

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 rth out of R comparisons between S_i and S_j. Let

$$P(S_i, S_j) = \frac{1}{R} \sum_{r=1}^{R} v_r(S_i, S_j) \qquad (14)$$

be the probability that S_i receives a higher quality assessment in comparison to S_j, $(P(S_i, S_j) + P(S_j, S_i) = 1)$. The final user score for S_i is defined by

$$\text{User Score}(S_i) = \sum_j P(S_i, S_j) \qquad (15)$$

Because the number of comparisons per pair was quite high and nearly constant, sophisticated methods for unbalanced pairwise comparison such as the Bradley–Terry model [BT52, Hun04] were not necessary since they provided an equal ranking.

The overall result of the user evaluation of the arrangements is shown in Figure 11. Figures 12 and 13 show the relationship between user ratings and the values of the E_1' and DPQ_{16}^- metrics for the colour set and the three image sets. It can be seen that the Pearson correlation is significantly higher for DPQ_{16}^- compared to E_1'. In the case of RGB colours, users liked the LAS arrangements the best. For
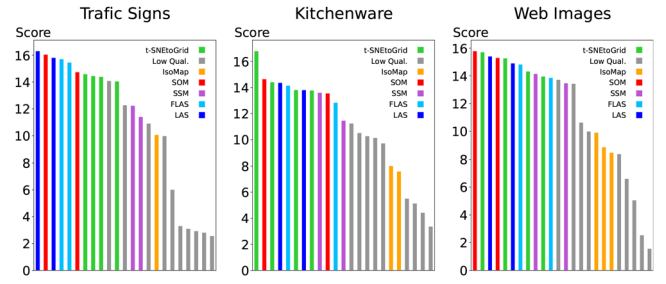

Figure 11: *User scores for the image sets. The user scores for the colour set can be seen in Figure 12.*
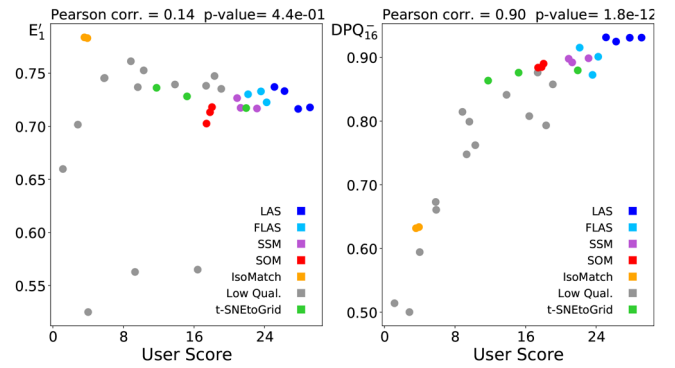

Figure 12: *User scores correlated with energy function E_1' (left) and distance preservation quality DPQ_{16}^- (right) for the colour set.*
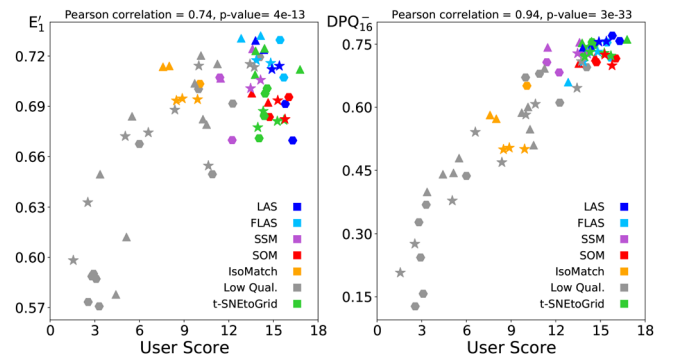

Figure 13: *User scores correlated with the energy function E_1' (left) and DPQ_{16}^- (right) for the three image sets: traffic signs (●), kitchenware (▲) and web images (★). The correlation between users scores and DPQ_{16}^- is higher than that of E_1'.*

the image sets, there is no clear winning method. The t-SNEtoGrid method obtained rather low scores for the RGB colours, but much higher ones for the image sets.

Figure 14 shows the degree of correlation between user scores and quality metrics for different p values of the E_p' and the DPQ_p metrics. For all four sets, the correlation of DPQ with the user scores is higher than that of E_1' and E_2' for all p values. For predicting user scores, DPQ_p^- values (using mean HD distances for equal 2D distances) with higher p values give the best results (left of Figure 14).
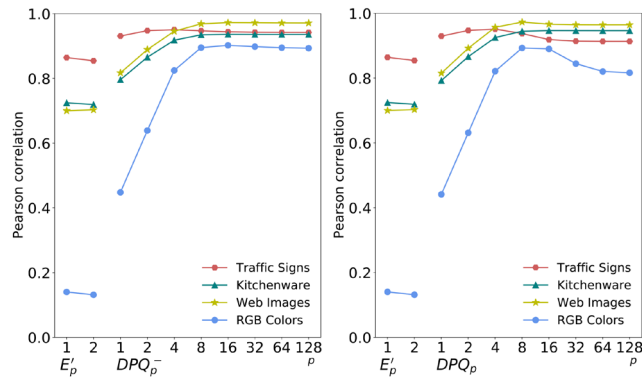
Figure 14: *The correlation of user scores with the metrics E'_p and DPQ_p with respect to the p values for the colour and image sets. For equal 2D distances, the DPQ metric was computed using mean HD distances (left) and sorted HD distances (right) (see Figure 7).*

The high correlation for larger p values with the user scores could indicate that users essentially pay attention to how well the immediate nearest neighbours have been preserved.

5.3. Evaluation of user search time

In the second part of the user study, the users were shown different arrangements in which they were asked to find four images in each case. The four images to be searched were randomly chosen and shown one after the other. As soon as one image was found, the next one was displayed. Participants were asked to pause only when they had found a group of four images, but not during a search. At the beginning, users were given a trial run to familiarize themselves with the task. Here, the time was not recorded. Figure 15 shows a screenshot of the search experiment. The overall set of arrangements was identical to those from the first part of the experiment, in which the pairs had to be evaluated.

Obviously, the task of finding specific images varies in difficulty depending on the image to be found. In addition, the participants are characterized by their varying search abilities. However, a total of more than 28,000 search tasks were performed, each with four
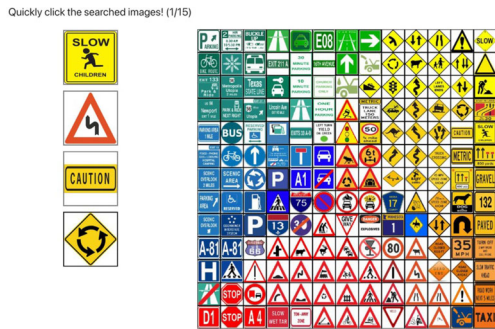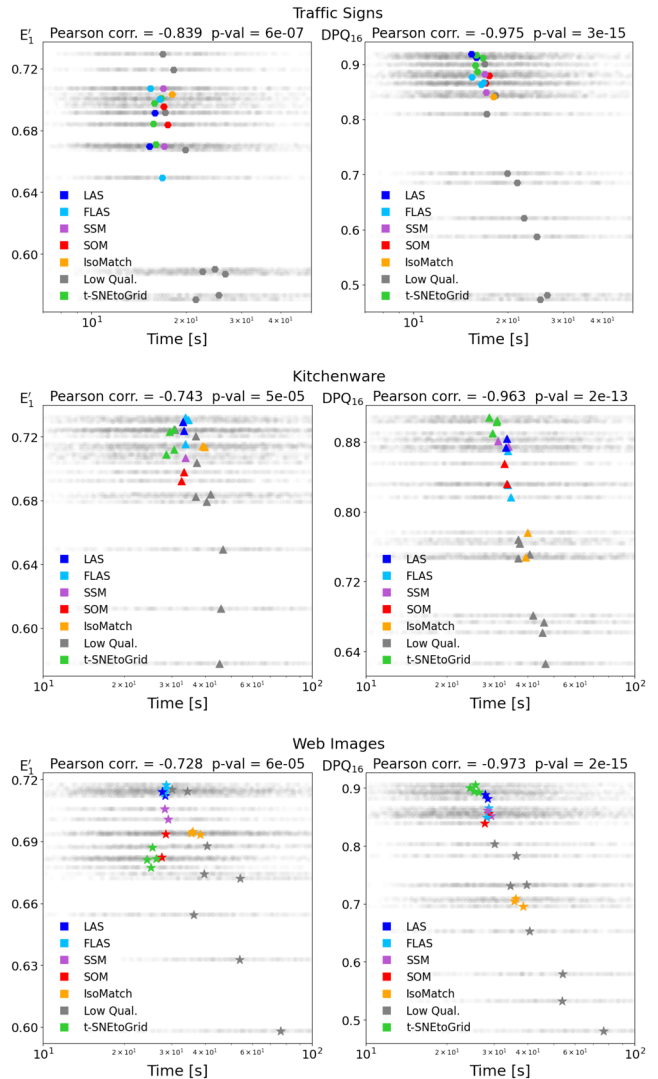

Figure 16: *Distribution of search times versus metrics for the three image sets. The median search time (shown as coloured markers) correlated with the energy function E'_1 (left) and the distance preservation quality DPQ_{16} (right).*

images to be found. This means that for each arrangement, more than 400 search tasks were performed for four images each. This compensated for differences in both the difficulty of the search and the abilities of the participants.

The search times required for each of the 23 arrangements per image set were recorded. It was found that the time distribution of the searches is approximately log-normal. Search times that fell outside the upper three standard deviations were discarded to filter out experiments that were likely to have been interrupted. Figure 16 shows the search time distribution of different arrangements for the three image sets. The median values of the search times of the different arrangements are shown as coloured markers. Again it can be seen that the correlation of the median search times is higher with the DPQ_{16} metric than with the E'_1 metric.

Figure 15: *Image search. In this study, we ask users to quickly find the images shown on the left.*
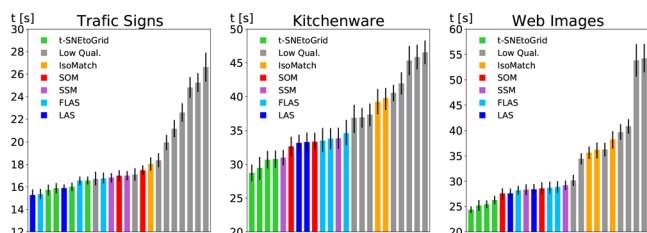
Figure 17: *The median search times for the three image sets. The standard errors of the medians are indicated. The time axis is clipped to better visualize differences of the median search times.*
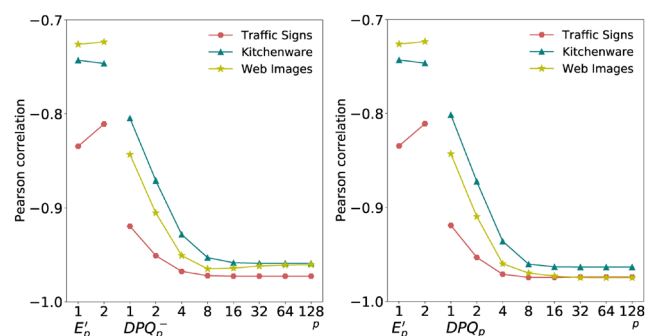

Figure 18: *The correlation of median image search time with the metrics E'_p and DPQ^-_p / DPQ_p with respect to the p values.*

Figure 17 shows the median values of the search times for all arrangements, sorted from the arrangement where the images were found the fastest to the one where the image search took the longest. The standard error of the median search times was determined by bootstrapping with 10,000 runs. While the ranking order of the algorithms is similar to the order of the user preferences, it occasionally differs, suggesting that an apparently well-sorted arrangement is only conditionally indicative of finding images quickly.

To evaluate the degree of correlation between search time and the quality metrics, Figure 18 compares the normalized energy function (E'_1 and E'_2) with the distance preservation quality (DPQ_q and DPQ^-_q). Again it can be seen that DPQ_p outperforms E'_p. Contrary to the user preference evaluation, for image retrieval tasks, DPQ_q performs slightly better than DPQ^-_q. Both show maximum correlation for high p values. This in turn indicates that it seems to be most important for people to locate similar images very close to each other in order to find them quickly, as hypothesized in Figure 7.

We also investigated the use of the squared $L2$ distance instead of the $L2$ distance to calculate the DPQ, the correlations remained similar, but were slightly lower. Other distance transformations could be investigated, but this is beyond the scope of this paper.

6. Qualitative and Quantitative Comparisons

6.1. Quality and run time comparison

To get a better understanding about the behaviour of FLAS and other 2D grid-arranging algorithms using different hyperparameter settings, we conducted a series of experiments. Since the run time strongly depends on the hardware and implementation quality, the numbers given in this section only serve as comparative values. In the previous section, DPQ_{16} has shown high correlation with user preferences and performance, we, therefore, use it when comparing algorithms in terms of their achieved 'quality' and the run time required to generate the sorted arrangement.

At this point, it is important to emphasize that LAS and FLAS, and likewise SOM and SSM, do not optimize an objective quality function or metric, unlike many other dimensionality reduction methods. The only used methods that perform an optimization in terms of a quality function or a metric are IsoMatch and t-SNE. IsoMatch attempts to maximize the normalized energy function E'_1, while the t-SNE objective is to make HD and 2D similarity distributions as similar as possible by minimizing their Kullback–Leibler divergence.

Our test machine is a Ryzen 2700x CPU with a fixed core clock of 4.0 GHz and 64GB of DDR4 RAM running at 2133 MHz. The tested algorithms were all implemented in Java and executed with the JRE 1.8.0_321 on Windows 10. Only the single-threaded sorting time was measured. As much code as possible was re-used (e.g. the solver of LAS, FLAS, IsoMatch and t-SNEtoGrid) to make the comparison as consistent as possible.

The Isomap and t-SNE projection implementation is from the popular library SMILE [Li14] (version 2.6). The SSM code is an implementation adapted from Strong and Gong [SG14] to match the characteristics of our implementation of SOM, LAS and FLAS. 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.

There are different hyperparameters that can be tuned. Some of them affect the run time and/or the quality of the arrangement, while others result in only minor changes. Figure 19 shows the relationship between speed and quality when varying the hyperparameters. For t-SNE, SSM and SOM, the number of iterations were changed, the t-SNE learning rate (eta) was set to 200. For LAS and FLAS, the radius reduction factor was gradually reduced from 0.99 to 0, while the initial radius factor was 0.35 and 0.5, respectively. FLAS used nine swap candidates per iteration. IsoMatch has only the k-neighbour setting which does not influence the quality nor the run time and therefore produces only a single data point in the plots. For small data sets like the 256 kitchenware images, FLAS offers the best trade-off between DPQ and computation time. LAS and t-SNE can produce higher DPQ_{16} values but are 10–100 times slower. There is no reason to use a SSM or SOM, since both are either slower or generate inferior arrangements. For the 1024 random RGB colours, LAS and FLAS yielded the highest DPQ.

In order to compare the scalability of the analysed algorithms, three data sets of different sizes were analysed, containing 256, 1024 and 4096 random RGB colours. The hyperparameters of the points marked with a ⊙ in Figure 19 were used for all the tests shown in Figure 20. It can be seen that FLAS and SSM have the same scaling properties, while FLAS exhibits better qualities. Even higher DPQ values can be achieved by LAS at the expense of run time. As the number of colours increases, arrangements with smoother gradients
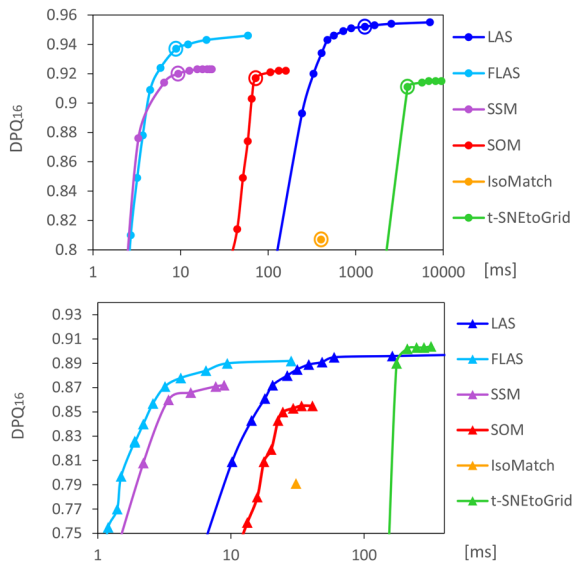
Figure 19: *Mean DPQ versus mean run time using different parameter settings (see the text for further information) for the 1024 RGB colours (top) and the 256 kitchenware images (bottom). The ⊙ markers indicate the parameters used in the scalability experiments shown in Figure 20.*
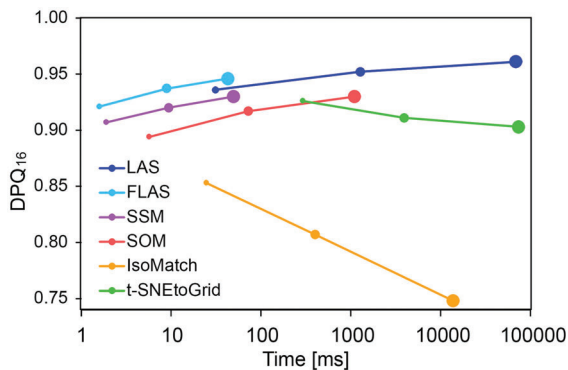

Figure 20: *Scalability of different algorithms. The mean* DPQ_{16} *and mean computation time for data sets of 256 (•), 1024 (•) and 4096 (•) random colours using fixed hyperparameters.*

become possible, resulting in better quality. Most approaches can exploit this property, except for IsoMatch and t-SNEtoGrid. Both initially project to 2D and rely on a solver to map the overlapping and cluttered data points to the grid layout. Since the number of grid cells is equal to the number of data points, it is difficult for the solver to find a good mapping. This often results in hard edges, as can be seen in Figure 21.

To summarize, LAS can be used for high quality arrangements, while FLAS should be used if the number of images is very high (several thousand) or if fast execution is important.

6.2. Visual comparison

While the quantitative qualities of the various algorithms are quite similar in some cases, visual inspection reveals specific differences.

For the 1024 RGB colour data set, the run closest to the DPQ_{16} mean was selected from the 100 test runs used for Figure 20. The corresponding arrangements are shown in the order of their distance preservation quality DPQ_{16} in the top of Figure 21. In addition, the normalized energy function value (E'_1) is given. LAS has the smoothest overall arrangement, followed by FLAS and SSM. The SOM arrangement is disturbed by isolated, poorly positioned colours, while the t-SNEtoGrid approach shows boundaries between regions. This is due to previously separate groups of projected vectors being re-distributed over the grid, resulting in visible boundaries where these regions touch. The noisy looking arrangement of IsoMatch is caused by the normalized energy function trying to equally preserve all distances. This leads to a kind of dithering of the vectors respectively the colours.

Most of these effects are less visible when real images are used instead of colours. The lower row of Figure 21 shows the arrangements from our user study 5.3, which required searching images in the web images set. The DPQ_{16} values and the E'_1 values are given. The arrangements are ordered by the median time it took users to find the images they were looking for (fastest on the left to slowest on the right). t-SNEtoGrid again shows some boundaries between regions, but this time the boundaries apparently help to better identify the individual groups of images, reducing the time needed to find them. The LAS, FLAS, SSM and SOM arrangements have a similar appearance for this data set. The dithered appearance of the IsoMatch arrangement apparently makes it difficult for users to quickly find the images they are looking for.

7. Applications

In this section, we present a variety of applications that use our algorithms introduced in Section 4.2 to efficiently manage or search larger sets of images.

7.1. Image management systems

For browsing local images on a computer, a visually sorted display of the images can help to view more images at once. Given userscrolling of a view tends to be in a vertical direction, it is important that the images on one horizontal line are similar to each other, and one perceives obvious changes in the vertical direction. This can be achieved by using a larger filter radius for horizontal filtering. See Figure 22 showing the PicArrange app [Jun21] as an example.

7.2. Image exploration

For very large sorted sets with millions of images, it may be useful to use a torus-shaped map which gives the impression of an endless plane. If one can navigate this plane in all directions, it is possible to bring regions of interest into view. Such an arrangement can be achieved by using a wrapped filter operation. This means part of the low-pass filter kernel uses vectors from the opposite edge of the
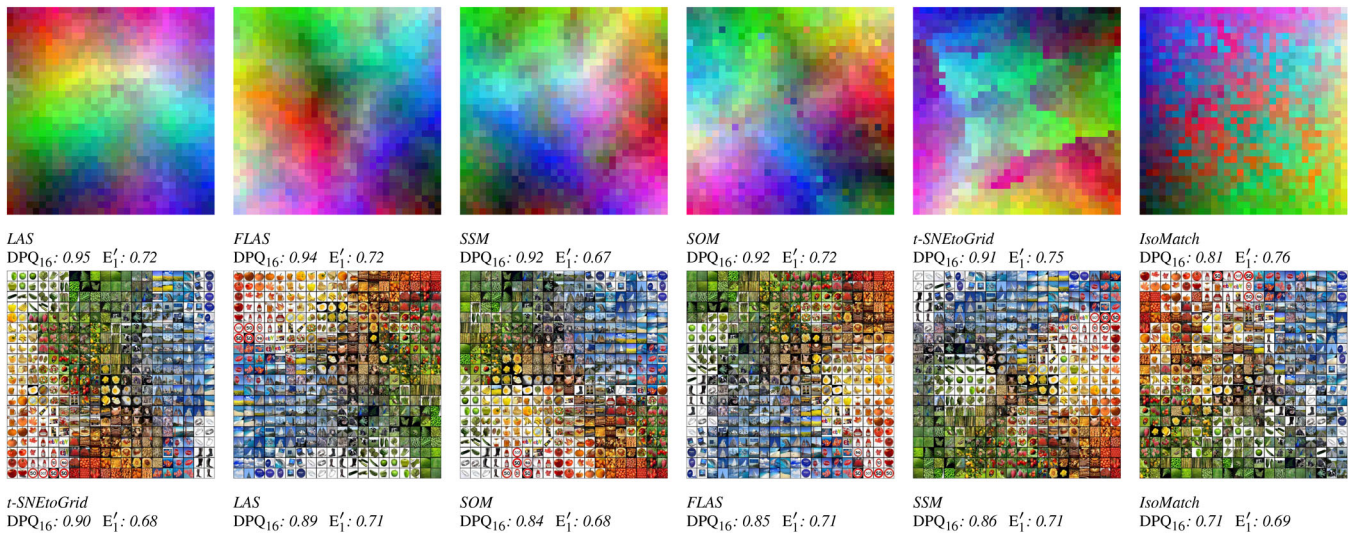
LAS	*FLAS*	*SSM*	*SOM*	*t-SNEtoGrid*	*IsoMatch*
DPQ$_{16}$: 0.95 E$'_1$: 0.72	DPQ$_{16}$: 0.94 E$'_1$: 0.72	DPQ$_{16}$: 0.92 E$'_1$: 0.67	DPQ$_{16}$: 0.92 E$'_1$: 0.72	DPQ$_{16}$: 0.91 E$'_1$: 0.75	DPQ$_{16}$: 0.81 E$'_1$: 0.76

t-SNEtoGrid	*LAS*	*SOM*	*FLAS*	*SSM*	*IsoMatch*
DPQ$_{16}$: 0.90 E$'_1$: 0.68	DPQ$_{16}$: 0.89 E$'_1$: 0.71	DPQ$_{16}$: 0.84 E$'_1$: 0.68	DPQ$_{16}$: 0.85 E$'_1$: 0.71	DPQ$_{16}$: 0.86 E$'_1$: 0.71	DPQ$_{16}$: 0.71 E$'_1$: 0.69

Figure 21: *Comparison of different image sorting schemes. Top: Arrangements of the 1024 random RGB colours ordered by distance preservation quality. Bottom: Arrangements of the web image set ordered by the median user search time (fastest on the left). For each algorithm, the arrangement that provided the fastest search is shown (see the leftmost points in Figure 16).*

Figure 22: *For vertical scrolling in an image management system, the images on horizontal lines should be similar to each other.*

map. Figure 23 shows a (small) example of such a torus-shaped arrangement. If zooming is possible, images of interest can be found and inspected very easily. This idea can be combined with a hierarchical pyramid of sorted maps that allows visual exploration for huge image sets. The user can explore the image pyramid with an interface similar to the mapping service (e.g. Google Maps). By dragging or zooming the map, other parts of the pyramid can be explored. The online tool *wikiview.net* [Bar19] allows the exploration of millions of Wikimedia images using this approach.

7.3. Layouts with special constraints

Although our algorithms work with rectangular grids, other shapes can also be sorted. The map has the size of the rectangular bound-

wrapped arrangement	*zoomed & centered on orange pots*

Figure 23: *An example of a wrapped (torus-shaped) grid arrangement. Left: the entire collection, right: zoomed and dragged to centre on orange pots.*
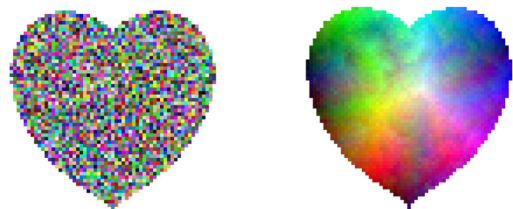

Figure 24: *2403 random RGB colours in a heart shape. Unsorted on the left and sorted with linear assignment sorting on the right.*

ing box of the desired shape, however, only the map positions within the shape can be assigned. Figure 24 shows an example where the colours to be sorted were only allowed within the shape of a heart. The corresponding algorithm remains the same, the only difference is that the vectors of the assigned map positions are used to fill
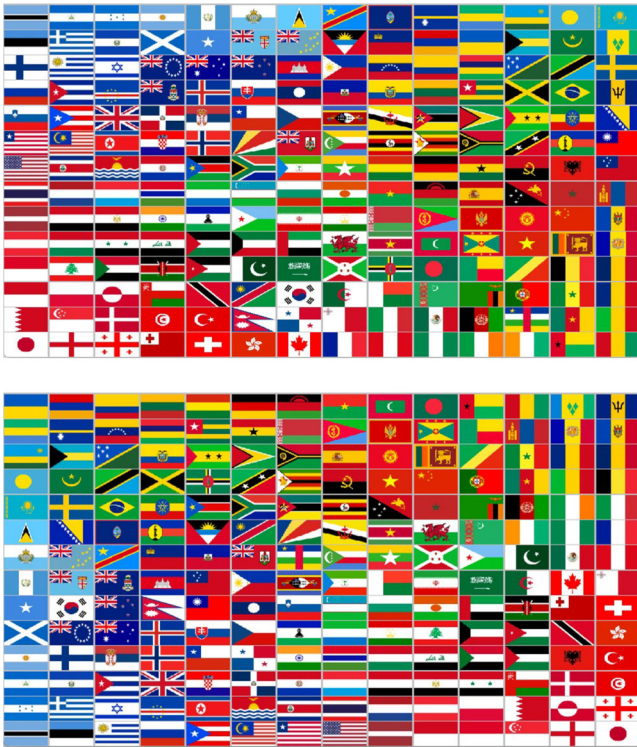
Figure 25: *Flags arranged by their similarity. Above, the American flag was pinned to the middle of the left side. Below, it was pinned at the centre of the bottom line.*

the rest of the the map positions. Each unassigned map position is filled with the nearest assigned vector of the map's constrained positions.

Sometimes it is desirable to keep some images fixed at certain positions (see Figure 25). This is possible with two minor changes to the algorithm: In a first step, the images or the corresponding vectors are assigned to the desired positions. These positions are then never changed again. Also, an additional weighting factor is introduced for filtering, where the fixed positions are weighted more. This results in neighbouring map vectors becoming similar to these fixed vectors, which in turn results in similar images also being placed nearby. The rest of the algorithm remains the same.

8. Conclusions

We presented a new evaluation metric to assess the quality of grid-based image arrangements. The basic idea is not to evaluate the preservation of the HD neighbour ranks of an arrangement, but the preservation of the average distances of the neighbourhood. Furthermore, we do not weight all distances equally either, because for humans, the preservation of small distances seems to be more important than that of larger distances. User experiments have shown that DPQ better represents human-perceived qualities of sorted arrangements than other existing metrics. If the overall impression of an arrangement is to be evaluated, the DPQ_p^- metric had a small advantage. For predicting how fast images can be found for an ar-

rangement, DPQ_p was better. In general, however, these differences are small and we recommend always using DPQ_p.

Large p values lead to the highest correlations with user perception. This implies that for a 'good' arrangement, it is essentially important that the sum of the distances to the immediate neighbours on the 2D grid is as small as possible. The same is true for one-dimensional sorting, which is 'optimal' only if the sum of the differences to the direct neighbours is minimal. It remains to be investigated whether DPQ is a useful metric for evaluating the quality of other, non-grid-based dimensionality reduction methods.

Furthermore, we have presented LAS which is a simple but at the same time very effective sorting method. It achieves very good arrangements according to the new metric as well as for other metrics. The FLAS variant can achieve better arrangements than existing methods with reduced complexity. In single-threaded execution, a CPU can sort tens of thousands of data vectors in a fraction of a second and over a million in less than 30 s. The FLAS algorithm is fully parallel because different swappings do not interfere with each other. Therefore, the algorithm can run efficiently on parallel hardware.

The ideas presented in this paper can be developed in numerous directions. Since the new DPQ metric allows a better prediction of the quality of an arrangement, it also better predicts the expected search time. If a sorting scheme were optimized in terms of DPQ, searched images would also be found faster. In this context, it remains to be investigated how a sorting algorithm can be optimized directly in the sense of a high DPQ value.

The filtering approach used determines mean HD distances for equal 2D distances. We will investigate whether further improvement is possible by exploiting the fact that for equal 2D distances on the grid, the sorted HD distances better represent the perceived quality.

Currently, our sorting method only supports regular grids. We want to investigate how this approach can be extended to densely packed rectangles of various sizes.

References

[Bar19] BARTHEL K. U.: Wikiview. https://wikiview.net (2019).

[BH19] BARTHEL K. U., HEZEL N.: *Visually Exploring Millions of Images using Image Maps and Graphs* (pp. 289–315). John Wiley & Sons, Ltd., New Jersey, USA, 2019. https://doi.org/10.1002/9781119376996.ch11.

[BHJS22] BARTHEL K. U., HEZEL N., JUNG K., SCHALL K.: Improved evaluation and generation of grid layouts using distance preservation quality and linear assignment sorting, 2022. https://arxiv.org/abs/2205.04255

[BK57] BECKMAN M., KOOPMANS T.: Assignment problems and the location of economic activities. *Econometrica 25* (1957), 53–76.

[BS10] BRAXMEIER H., STEINBERGER S.: Pixabay. https://www.pixabay.com (2010).

[BSCL14] BABENKO A., SLESAREV A., CHIGORIN A., LEMPITSKY V. S.: Neural codes for image retrieval. In *ECCV, Lecture Notes in Computer Science (1)* (2014), D. J. Fleet, T. Pajdla, B. Schiele and T. Tuytelaars (Eds.), vol. 8689, Springer, pp. 584–599.

[BT52] BRADLEY R. A., TERRY M. E.: Rank analysis of incomplete block designs: I. the method of paired comparisons. *Biometrika 39, 3/4* (1952), 324–345.

[CAS20] CAO B., ARAUJO A., SIM J.: Unifying deep local and global features for image search. In *ECCV, Lecture Notes in Computer Science (20)* (2020), A. Vedaldi, H. Bischof, T. Brox and J.-M. Frahm (Eds.), vol. 12365, Springer, pp. 726–743.

[DGV12] DJURIC N., GRBOVIC M., VUCETIC S.: Convex kernelized sorting. In *AAAI* (2012), J. Hoffmann and B. Selman (Eds.), AAAI Press.

[DML17] DYM N., MARON H., LIPMAN Y.: Ds++: A flexible, scalable and provably tight relaxation for matching problems. *ACM Transactions on Graphics 36*, 6 (Nov. 2017). https://doi.org/10.1145/3130800.3130826.

[FDH*15] FRIED O., DIVERDI S., HALBER M., SIZIKOVA E., FINKELSTEIN A.: Isomatch: Creating informative grid layouts. *Computer Graphics Forum 34, 2* (2015), 155–166.

[FFDP15] FADEL S. G., FATORE F. M., DUARTE F. S. L. G., PAULOVICH F. V.: Loch: A neighborhood-based multidimensional projection technique for high-dimensional sparse spaces. *Neurocomputing 150* (2015), 546–556.

[Fre22] FREY S.: Optimizing grid layouts for level-of-detail exploration of large data collections. *Computer Graphics Forum 41*, 3 (2022), 247–258.

[HMJE*21] HILASACA G. M., MARCÍLIO-JR W. E., ELER D. M., MARTINS R. M., PAULOVICH F. V.: Overlap removal of dimensionality reduction scatterplot layouts. http://arxiv.org/abs/1903.06262 (2021).

[HPA*19] HOWARD A., PANG R., ADAM H., LE Q. V., SANDLER M., CHEN B., WANG W., CHEN L.-C., TAN M., CHU G., VASUDEVAN V., ZHU Y.: Searching for mobilenetv3. In *ICCV* (2019), IEEE, pp. 1314–1324. https://doi.org/10.1109/ICCV.2019.00140.

[Hun04] HUNTER D. R.: MM Algorithms for Generalized Bradley–Terry Models. *The Annals of Statistics 32, 1* (2004), 384–406.

[HZL*15] HAN X., ZHANG C., LIN W., XU M., SHENG B., MEI T.: Tree-based visualization and optimization for image collection. *CoRR abs/1507.04913* (2015).

[Jun21] JUNG K.: Picarrange. https://visual-computing.com/project/picarrange (2021).

[JV87] JONKER R., VOLGENANT A.: A shortest augmenting path algorithm for dense and sparse linear assignment problems. *Computing 38, 4* (1987), 325–340.

[Koh82] KOHONEN T.: Self-organized formation of topologically correct feature maps. *Biological Cybernetics 43* (1982), 59–69.

[Koh13] KOHONEN T.: Essentials of the self-organizing map. *Neural Networks 37* (2013), 52–65.

[Kuh55] KUHN H.: The Hungarian method for the assignment problem. *Naval Research Logistics Quarterly 2, 1-2* (1955), 83–97.

[Lew94] LEWIS J.: Fast template matching. *Vision Interface 95* (Nov. 1994), 6.

[Li14] LI H.: Smile. https://haifengl.github.io (2014).

[LMBH11] LUEKS W., MOKBEL B., BIEHL M., HAMMER B.: How to evaluate dimensionality reduction? - Improving the co-ranking matrix. *CoRR abs/1110.3917* (2011).

[LV08] LEE J. A., VERLEYSEN M.: Quality assessment of nonlinear dimensionality reduction based on k-ary neighborhoods. In *FSDM, JMLR Proceedings* (2008), Y. Saeys, H. Liu, I. Inza, L. Wehenkel and Y. V. de PEER (Eds.), vol. 4, JMLR.org, pp. 21–35.

[MH18] MCINNES L., HEALY J.: Umap: Uniform manifold approximation and projection for dimension reduction. *CoRR abs/1802.03426* (2018).

[Pea01] PEARSON K.: On lines and planes of closest fit to points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science 2, 11* (1901), 559–572.

[QKTB10] QUADRIANTO N., KERSTING K., TUYTELAARS T., BUNTINE W. L.: Beyond 2D-grids: A dependence maximization view on image browsing. In *Proceedings of the Multimedia Information Retrieval* (2010), J. Z. Wang, N. Boujemaa, N. O. Ramirez and A. Natsev (Eds.), ACM, pp. 339–348.

[QSS08] QUADRIANTO N., SONG L., SMOLA A. J.: Kernelized sorting. In *Proceedings of the NIPS* (2008), D. Koller, D. Schuurmans, Y. Bengio and L. Bottou (Eds.), Curran Associates, Inc., pp. 1289–1296.

[RRS13] REINERT B., RITSCHEL T., SEIDEL H.-P.: Interactive by-example design of artistic packing layouts. *ACM Transactions on Graphics 32, 6* (2013), 218:1–218:7.

[RS00] ROWEIS S. T., SAUL L. K.: Nonlinear dimensionality reduction by locally linear embedding. *Science 290, 5500* (2000), 2323–2326.

[RTC19] RADENOVIC F., TOLIAS G., CHUM O.: Fine-tuning cnn image retrieval with no human annotation. *IEEE Trans. Pattern Anal. Mach. Intell. 41, 7* (2019), 1655–1668.

[SA11] SCHOEFFMANN K., AHLSTROM D.: Similarity-based visualization for image browsing revisited. In *Proceedings of the 2011 IEEE International Symposium on Multimedia* (2011), pp. 422–427. https://doi.org/10.1109/ISM.2011.76.

[Sam69] SAMMON J. W.: A nonlinear mapping for data structure analysis. *IEEE Transactions on Computers 18*, *5* (1969), 401–409.

[Sar14] SARVENIAZI A.: An actual survey of dimensionality reduction. *American Journal of Computational Mathematics 04* (Jan. 2014), 55–72.

[SG11] STRONG G., GONG M.: Data organization and visualization using self-sorting map. In *Proceedings of the Graphics Interface* (2011), S. Brooks and P. Irani (Eds.), Canadian Human-Computer Communications Society, pp. 199–206.

[SG14] STRONG G., GONG M.: Self-sorting map: An efficient algorithm for presenting multimedia data in structured layouts. *IEEE Transactions on Multimedia 16*, *4* (2014), 1045–1058.

[SJGE13] STRONG G., JENSEN R., GONG M., ELSTER A. C.: Organizing visual data in structured layout by maximizing similarity-proximity correlation. In *ISVC, Lecture Notes in Computer Science (2)* (2013), G. Bebis, R. Boyle, B. Parvin, D. Koracin, B.

Li, F. Porikli, V. B. Zordan, J. T. Klosowski, S. Coquillart, X. Luo, M. Chen and D. Gotz (Eds.), vol. 8034, Springer, pp. 703–713.

[TdSL00] TENENBAUM J. B., DE SILVA V., LANGFORD J. C.: A global geometric framework for nonlinear dimensionality reduction. *Science 290* (2000), 2319–2323.

[vdMH08] VAN DER MAATEN L., HINTON G.: Visualizing data using t-SNE. *Journal of Machine Learning Research 9* (2008), 2579–2605.

[VJ01] VIOLA P., JONES M.: Robust real-time object detection. In *Proceedings of the Second International Workshop on Statistical and Computational Theories of Vision – Modeling, Learning, Computing, and Sampling* (2001).

[XTL*21] XIE P., TAO W., LI J., HUANG W., CHEN S.: Exploring Multi-dimensional data via subset embedding. *Computer Graphics Forum 40*, 3 (2021), 75–86. https://doi.org/10.1111/cgf.14290.

[ZIE*18] ZHANG R., ISOLA P., EFROS A., SHECHTMAN E., WANG O.: The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the 31st Meeting of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2018* (Dec. 2018), IEEE Computer Society, pp. 586–595. https://doi.org/10.1109/CVPR.2018.00068.