EUROGRAPHICS 2015 / O. Sorkine-Hornung and M. Wimmer
(Guest Editors)

Volume 34 (2015), Number 2

IsoMatch: Creating Informative Grid Layouts

O. Fried[1], S. DiVerdi[2], M. Halber[1], E. Sizikova[1], and A. Finkelstein[1]

[1]Princeton University [2]Google Inc.

Abstract

Collections of objects such as images are often presented visually in a grid because it is a compact representation that lends itself well for search and exploration. Most grid layouts are sorted using very basic criteria, such as date or filename. In this work we present a method to arrange collections of objects respecting an arbitrary distance measure. Pairwise distances are preserved as much as possible, while still producing the specific target arrangement which may be a 2D grid, the surface of a sphere, a hierarchy, or any other shape. We show that our method can be used for infographics, collection exploration, summarization, data visualization, and even for solving problems such as where to seat family members at a wedding. We present a fast algorithm that can work on large collections and quantitatively evaluate how well distances are preserved.

Categories and Subject Descriptors (according to ACM CCS):
I.3.3 [Computer Graphics]: Picture/Image Generation—Display algorithms I.3.6 [Computer Graphics]: Methodology and Techniques—Interaction techniques

1. Introduction

Web services like Flickr, Facebook, and Google+, as well as desktop applications like Lightroom, iPhoto and Aperture arrange images in grids. One natural benefit of tiling images into a grid is that it makes efficient use of screen real estate. In addition, it provides natural orderings, such as left-to-right, top-to-bottom. Often the images themselves can be sorted trivially by some numeric quality, e.g. date or search rank. However, for some applications a good ordering is not so obvious. For example, it may be desirable to arrange photos of one's Facebook friends, organized such that people who are friends with each other are near each other. Or one might like to arrange a group of natural images such that pictures with similar colors are near each other. Research like that of Schoeffmann and Ahlstrom [SA11] shows that similarity-based placement of images is useful, and improves retrieval tasks. Their study indicates that even a simple similarity metric based on color can improve a retrieval task by 20%.

In the design world, such as page layouts for magazines, webpages, and photo albums, it is important to support arbitrary arrangements of content on the page. Currently, the designer must assign images to locations by hand, balancing a complex set of goals that may be difficult to formalize. However, the process can be time consuming and difficult to fully optimize. Moreover, this approach is not amenable to automation, which would be necessary for online contexts like Facebook. A standard automatic way to organize objects by similarity is to use dimensionality reduction algorithms such as principle component analysis or multidimensional scaling [LV07] to arrange the objects continuously in two dimensions. While arbitrary placement can optimally reproduce pairwise distances, many applications have hard requirements about explicit, discrete positions for the final arrangement. For example, creating a regular 2D grid of images or placing photos in a predefined page layout—these applications specify the exact target positions as an input and therefore a continuous arrangement algorithm is insufficient. Likewise, the layout method of Reinert et al. [RRS13] which beautifully packs items tightly in the plane while arranging according to two major dimensions is inappropriate for such applications. In contrast, recent methods such as the *kernelized sorting* approach of Quadrianto et al. [QKTB10] provide automatic layout of designs where a specific set of target locations is known in advance.

Taking inspiration from these methods, this paper presents *IsoMatch*, a fully-automatic method for arrangement of images (or other items) on a grid (or other pre-established arrangement). The input is a set of items to be arranged, to-

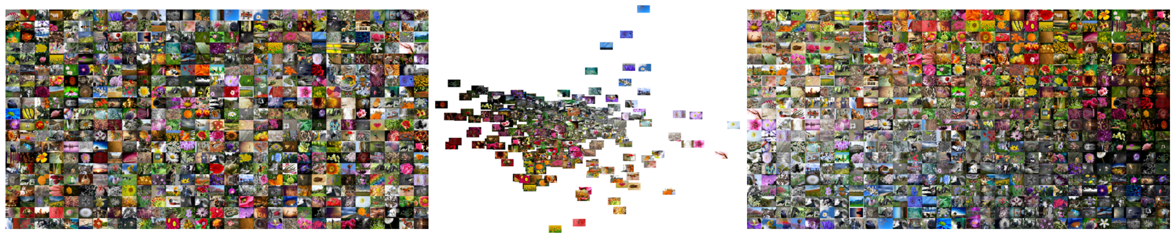


Figure 1: Images of flowers from Flickr. Left: Random ordering makes image set hard to understand. Center: Isomap preserves structure but is poor for page layouts. Right: Arranged by our algorithm using the L2 distance between mean RGB values. Note that good results are obtained even with such a simplistic feature (three values per image).

gether with pairwise distances between them (e.g., mean color difference, or distance in a social network), and an arbitrary set of target locations. Our goal is to arrange the items while preserving their pairwise distances as much as possible in the output locations.

The key contributions of our work are: (1) a general objective function that can be used to evaluate solutions to the distance preserving layout problem, with validation via a user study, (2) the IsoMatch algorithm which efficiently arrives at an approximate optimization of the objective function, with quantitative evaluation, and (3) a number of layout and visualization applications for images, audio, and 3D models, demonstrating that IsoMatch unifies and extends the major features of previous approaches.

2. Related Work

Dimensionality Reduction. One line of research addresses the task of mapping high dimensional data to lower dimension while preserving the original data as much as possible. This goal is the same as ours, with one major distinction—we add the constraint that the result should lie only on predefined, discrete locations. The seminal methods *Isomap* [TdSL00] and *Locally Linear Embedding* (LLE) [RS00] reconstruct low dimensional manifolds that reside in a higher dimensional space. We use Isomap as the first step of our algorithm, though its output cannot be directly used for layout (Figure 1). Our algorithm, IsoMatch, is more similar to Dwyer's constrained graph layout [Dwy09], which fits into the clutter reduction taxonomy of Ellis and Dix [ED07] as a "space-filling" approach. However, these approaches generally only allow relative constraints between points in the output arrangement, as opposed to IsoMatch, which restricts to assignment on discrete locations.

Grid Arrangements. Graph visualizations sometimes place vertices in discrete locations, for readability and aesthetics. Illustrating biological networks, the work of Kojima et al. [KNJ*07] uses a force-based method that places each vertex in the closest empty grid location. Likewise, Li and Kurata [LK05] visualize biochemical networks using a sparse placement of labels on a grid. Inoue et al. [ISYK12] propose a two-phase approach to the graph layout problem. The first step performs standard (fast) graph layout without the grid constraint, and the second phase aligns vertices onto grid points. The planar graph layout problem differs from ours as edge crossings are avoided and because a sparse layout is desired. Fried et al. [FJOF14] arrange sound samples on a grid to improve search performance. They used kernelized sorting [QSST10] as the underlying algorithm, against which we show favorable results (Section 7).

Self organizing maps [Koh82] are a common method for arranging high dimensional data (e.g. [SG11b]). However, the input to SOMs are descriptors and not pairwise distances, which make them unsuitable when only distances are available. Also, SOMs produce results with gaps and overlaps, which are inadequate if we want exactly one image to appear in each grid cell.

The Self-Sorting-Maps (SSM) of Strong et al. [SG11a, SG14] arrange images on a grid, using a hierarchical swapping process that seeks to maximize normalized cross correlation between the input images and the grid locations. This approach only works on a regular lattice structure and does not have an obvious extension to an arbitrary set of target locations or densities (required, e.g., in Figures 9 and 10). SSMs also need to be adjusted for each change in dimensionality or grid structure (i.e. the algorithm changes slightly from 2D to 3D and from rectangular to hexagonal grids). Moreover, they appear to suffer the curse of dimensionality, so it is unclear how they will behave for embeddings in higher dimensions. Strong et al. [SJGE13] also describe an alternative to SSMs that seeks a better global optimum for the correlation, at the expense of decreased performance.

Treemaps (e.g. [BSW02, WD08]) require a tree structure as input. They sort according to one attribute and typically show one or two more attributes (i.e. area and color) decoupled from the ordering. Similarly, work based on "jigsaw maps" [BHL05, Wat05, WFM*12] use space filling curves to partially fill discrete grids, but they cannot exactly match arbitrary arrangements. Liu et al. [LHN*13] use a two step approach to arrange small sets. However, they cannot enforce a bijection between sets and grid cells and they do not support arbitrary layouts nor layouts in 3D.

Reinert et al. [RRS13] show impressive results of placing objects on a continuous domain. However, as they try to pack objects together they cannot support specific target locations, which makes their approach inappropriate for applications which must constrain objects to discrete positions, such as a photo album layout or graphic design template. They also do not attempt to preserve pairwise distances in an objective and quantifiable manner, focusing only on aesthetics. Frishman and Tal's physically based uncluttering [FT09] has similar limitations.

Graph Matching. The work closest to ours is the *Kernelized Sorting* (KS) method of Quadrianto et al. [QSST10] and its derivatives, Convex Kernelized Sorting [DGV12] and CDOM with Model Selection [YS11]). They aim to solve a more general problem, matching two sets of objects while intra-set distances are known but inter-set distances are not. Our formulation is a special case of this problem where we can take advantage of the metric nature of the target arrangement's distances, as well as the (potentially) metric distances of the set of objects to be arranged. Kernelized Sorting (KS) involves maximization (not minimization) over a convex function of the permutation matrix, thus is unstable and highly sensitive to initialization values. Convex Kernelized Sorting yields better results, but at the cost of greater computational demands, with running times that exceed ours by an order of magnitude (Section 7.2). Another limitation of the above methods is that they require an equal number of elements in the two sets (i.e. equal number of images and target locations), while we can do a one to many matching to produce a digest of a photo collection (Section 6.6). Also, we believe our algorithm is simpler than KS, which enables future work to more easily extend the technique. In Section 7 we show quantitative results of comparisons to these methods. The further work of Quadrianto et al. [QKTB10] uses KS as its underlying method and produces several applications that overlap with ours. We discuss the drawbacks of the 3D image layout in Section 6.2 and of their hierarchical organization in Section 6.7.

Quality Metrics. The objective function we propose in Section 3 is similar to those surveyed by Bertini [Ber11]. Those metrics are more concerned with interactive refinement and measuring higher level properties such as clustering and separability, while our function is more similar to the precision measure of Schreck et al. [SvLB10], which estimates how well distances are preserved by a projection. However, they do not show a normalized result that is consistent across different contexts (different distance scales, numbers of samples, dimensions, etc.), and it is not shown how to use their local function to automatically refine an arrangement.

3. Problem Formulation

Our general problem takes as input a set of items $\mathscr{I}$ together with a distance matrix D, where each item is to be assigned to a set of spatial locations $\mathscr{L}$. Most of the applications we consider involve placing images into a 2D grid, so for the remainder of this section we will refer to $\mathscr{I}$ as *images* and $\mathscr{L}$ as the *grid*. Our objective is to assign the images into the grid such that, as much as possible, the pairwise target distances d_{ij} in D are preserved by the corresponding Euclidean distances $d'_{k\ell}$ in the grid, up to some arbitrary scaling factor c. That is, we seek the permutation $\pi(i) = k$ indicating that image i is assigned to location k, that minimizes the error under norm p:

$$\mathscr{E}_p(\pi) = \min_c \left(\sum_\omega \left| c\, d_\omega - d'_{\pi(\omega)} \right|^p \right)^{\frac{1}{p}} \tag{1}$$

where ω ranges over all pairs i, j of images and $\pi(\omega) = \{\pi(i), \pi(j)\}$. This problem is similar to *multidimensional scaling* [LV07] with the constraint that images must be one-to-one assigned to discrete locations.

3.1. Normalized Energy Function

The formulation in Equation (1) implies a simple way to evaluate different solutions (assignments) for any given input. We can treat the error as an "energy" that can be used to compare different solutions, with lower energy corresponding to better solutions, which leads to some straightforward optimization techniques described below. In addition, we normalize the energy by dividing by the sum of all pairwise distances in the grid:

$$E_p(\pi) = \min_c \left(\sum_\omega \left| c\, d_\omega - d'_{\pi(\omega)} \right|^p \Big/ \sum_\omega {d'_{\pi(\omega)}}^p \right)^{\frac{1}{p}} \tag{2}$$

Of course E_p is minimized under the same permutation and scale as in Equation (1), and for a "perfect" assignment like assigning the output grid to itself, it yields energy $E_p = 0$. For a given permutation π we are left with the task of determining the scaling constant c. In the case of the 2-norm, Equation (2) can be minimized with respect to c by setting $\frac{\partial}{\partial c} E_2(\pi) = 0$. The case of $E_1(\pi)$ is more subtle, because the energy incorporates a sum of absolute values that have derivative discontinuities in c. Nevertheless, examination of this derivative reveals it to be piecewise-constant with discontinuities at the $\binom{n}{2}$ locations where $c = d'_{\pi(\omega)}/d_\omega$. Thus, it suffices to evaluate Equation (2) at these locations and choose the minimum. It is possible to consider only a subset, achieving a final complexity of $O(n^2 \log n)$; the proof is outside the scope of this paper.

In our experiments we found the general behavior of $E_1(\pi)$ and $E_2(\pi)$ to be similar, but we prefer $E_1(\pi)$ because it favors solutions which preserve smaller distances more than larger distances. Intuitively, relative arrangements of nearby elements (e.g. in the 1-neighborhood) are more salient than more distant relationships. Therefore, for the rest of the paper we use the energy $E(\pi) = E_1(\pi)$. Figure 2 shows typical E_1 values for several arrangements.

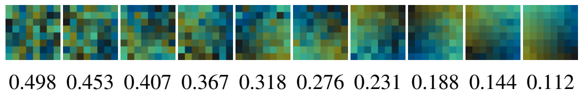


Figure 2: Energy values for various arrangements. We show grids of colors arranged by RGB distance, along with the corresponding $E_1(\pi)$ values (Equation 2). Notice how low energy values correlate with a more ordered color grid.

3.2. Nonlinear Integer Programming

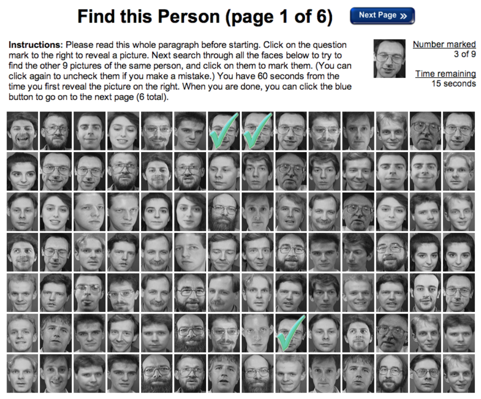
Having defined an energy function, a naive way to solve our optimization problem is to generate random arrangements and then randomly swap pairs until convergence. Such an approach is prone to get stuck in local minima, and converges too slowly for practical applications.

A more principled approach is to set up the minimization of Equation (2) as a nonlinear integer programming problem. This formulation would normally be a quadratic integer program but for the inclusion of the c term, which makes it more complex. Even ignoring the optimization over c, solving such a problem is generally NP-hard [PV91], and therefore we are forced to seek heuristic solutions for any reasonably large input. Note that this problem is similar to the *quadratic assignment problem* which is also known to be NP-hard, as discussed by Quadrianto et al. [QSST10]. While general-purpose functions exist to find approximate solutions, our algorithm (Section 5) can take advantage of known structure in the problem, by splitting it into two easily solvable steps. Our approximation cannot guarantee a globally minimal solution, but it does efficiently generate reasonable approximations (Section 7).

4. Study Correlating Energy with User Performance

Schoeffmann and Ahlstrom [SA11] showed that similarity-based placement improves image retrieval tasks, while Fried et al. [FJOF14] had similar findings for audio retrieval. Thus, we know that organization makes a difference in at least some tasks. The normalized energy function Equation (2) provides a measure of the degree of organization of any particular grid assignment. Even though this measure was designed to match our intuition about what it means for a grid to be organized, we would like to show that it contains meaningful information relative to some human task. Therefore, in this section we describe a study that shows a correlation between the energy in a grid of images and the ability to search for pictures of a particular person. We chose this task as a surrogate for a variety of image searching tasks, but it bears strong resemblance to activities like labeling individuals in photo collections in commercial software like Picasa and iPhoto.

We ran this study on the Amazon Mechanical Turk, and the interface for the task (called a "HIT") is shown in Figure 3. For each of 6 pages in the HIT, the subject (a "worker") was shown a photo of a *target* person as well as a 7×14 grid of photos containing 9 other photos of the same individual, mixed in among 89 photos of other people selected randomly from the 40 individuals in the face database of Samaria and Harter [SH94]. The pages had a range of Equation (2) energies from about 0.28 to 0.45, based on RMS distance in pixel luminance. Workers were given 40 seconds per page to find as many of the 9 target images as possible. In roughly half of the pages shown, the worker found all 9 target images in 40 seconds, and in the vast majority they found at least 5. Workers were allowed to perform the task more than once and while most did it just once or a few times, some worked on as many as 20 HITs. Nevertheless, they were never shown the same page more than once. Any page where the worker failed to find 3 of the 9 target images was omitted from the dataset, under the assumption that the worker was not trying or did not understand the task well. In total 617 HITs were performed by 236 unique workers, 3054 worker-pages were retained, and each of 160 pages was viewed by 15 - 25 workers.


Find this Person (page 1 of 6)

Instructions: Please read this whole paragraph before starting. Click on the question mark to the right to reveal a picture. Next search through all the faces below to try to find the other 9 pictures of the same person, and click on them to mark them. (You can click again to uncheck them if you make a mistake.) You have 60 seconds from the time you first reveal the picture on the right. When you are done, you can click the blue button to go on to the next page (6 total).

Number marked
3 of 9

Time remaining
15 seconds

Figure 3: Face search. In this study we ask subjects to find pictures of a person in a grid of photos.

We measured the performance of each worker on each page as follows. We found the integral over the 40 second period of the number correctly identified, normalized to the range [0,1] by dividing by 9×40. Next, because we found that some target individuals were easier to find than others (for example wearing glasses or a beard) we found the "relative performance" as the difference between the mean performance for any particular page and the mean performance for that individual target. These values are plotted in Figure 4. They exhibit a strong correlation between energy and relative performance – a Pearson's product-moment correlation of -0.52 with p-value $< 10^{-10}$.

Thus we conclude that Equation (2) does correlate with human performance on at least one search task, and this provides evidence that it is suitable as a measure of the

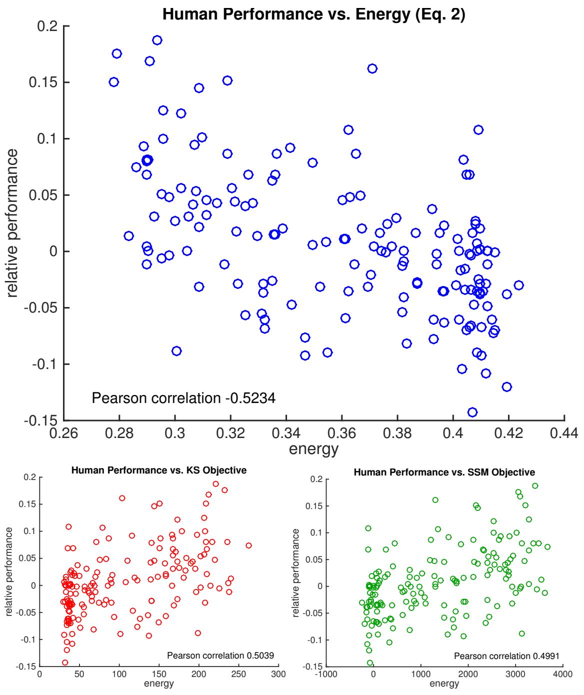


Figure 4: Human performance correlated with grid energy. Above: Our study shows better search performance for low energy grids (more organized, as measured by equation 2). Below: Our study also shows strong correlation with (left) the goal function used in kernelized sorting and (right) the normalized cross-correlation used in SSMs. For these goal functions, larger values correspond to more organized grids.

degree of organization of items in a grid. We use this measure in the remainder of the paper as a mechanism to evaluate various organization strategies.

Figure 4 also shows that the user performance from our study is correlated with the objective functions used in two previous methods: kernelized sorting (KS) [QSST10] and self-sorting maps (SSM) [SG14]. Each of these shows a Pearson's correlation of 0.50, similar to that of Equation (2) albeit slightly weaker. Note that for these objective functions, larger values are better, and that the scores are not fit to a standard range but vary with the experiment. SSM uses a normalized cross-correlation related to the vector inner product, and is of the form $\sum Add' + Bd + Cd'$; KS uses the trace of the product of the kernel matrices which has a similar final form. Conversely, our energy function is related to a scaled Minkowski distance and is of the form $\sum Dd + Ed'$. The impact of the product term dd' is that SSM may be overly sensitive to outlier distances. KS avoids this problem using the kernel transform, whereas we address it by selecting the 1-norm for Equation (2).

5. Our Algorithm: IsoMatch

In this section we describe the algorithm used to align images in a specific formation, according to a given distance matrix. The inputs to IsoMatch are a set of images $\mathscr{I}$ such that $|\mathscr{I}| = n$, a symmetric pairwise distance matrix D of size $n \times n$ such that d_{ij} denotes the distance between the images $\mathscr{I}_i$ and $\mathscr{I}_j$, and a desired output spatial arrangement which is usually a 2D grid but can take an arbitrary form. Note that the distance matrix can be built using an arbitrary distance measure between the images, such as 2-norm in $\mathbb{R}^m$, some m-dimensional feature space, or from a non-metric set of graph edge weights, such as affinity in a social network. This flexibility allows our visualization to convey many different types of relationships through arrangement.

Our core algorithmic contribution is, rather than attempting to solve the distance-preserving arrangement problem as a single optimization, we approximate the solution in two main steps. First, we use nonlinear dimensionality reduction to project the set of images onto 2D (we will limit our discussion to 2D for now, and revise it later on). Once we have a 2D arrangement, we transform it so that it will roughly fit our desired output pattern (Section 5.2). After the coarse alignment has been established, we construct a bipartite graph and find a bipartite matching in order to solve a minimal movement goal (Section 5.3). From the bipartite matching we infer the output arrangement.

5.1. Step I: Dimensionality Reduction

There are many options for algorithms in nonlinear dimensionality reduction, including Multidimensional Scaling [LV07], Isomap [TdSL00], and Locally-Linear Embedding [RS00]. Succinctly stated, the goal is to represent an n-dimensional set of points in m-dimensions, where $m \ll n$, by discovering underlying structure to the data. Each algorithm has its strengths, but we use Isomap specifically because of its ability to handle input datasets that are defined by a distance matrix. Conversely, techniques such as Locally-Linear Embedding require the input dataset to have a metric distance function defined. This requirement is acceptable when arranging, e.g., images by RGB histograms (or any other descriptor) because the feature transform converts the images into points in a high dimensional space which can then be characterized by euclidean distance. However, a non-metric dataset such as affinity in a social network may not actually define a valid distance metric in some dimension – for instance, the triangle inequality may not hold for graph distances. These datasets are still interesting to arrange though, so we need an algorithm that can support them. Ultimately, we do not depend specifically on Isomap, and other dimensionality reduction algorithms can be trivially swapped in if they can process the input dataset type.

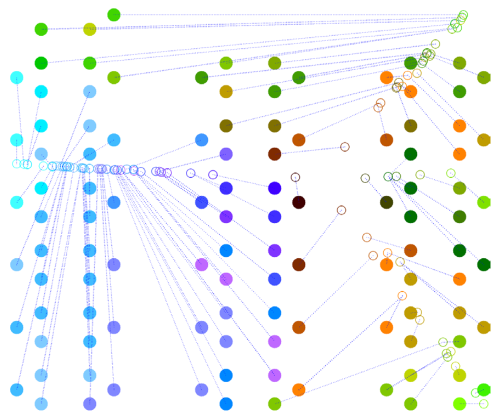


Figure 5: Bipartite matching example. Isomap output points are hollow circles, while target positions are filled circles, and dotted lines indicate the minimum distance-moved assignment determined by bipartite matching.

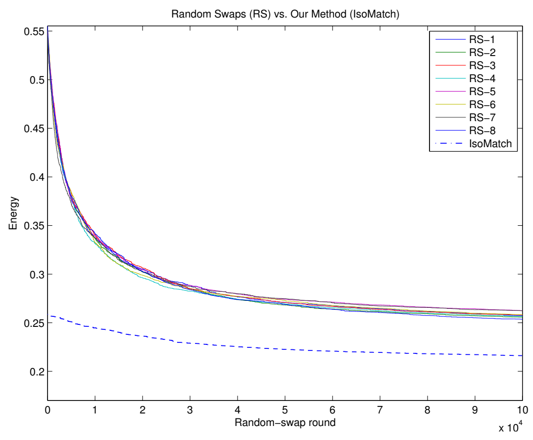


Figure 6: Performance of random swaps. The x-axis is the number of random swap iterations, and the y-axis is the energy (Eq. 2). Eight runs of random swaps (each starting from a random permutation) show the aggregate behavior, which is outperformed by our algorithm. Adding random swaps to refine our output improves it further (dash-dot line).

5.2. Step II: Coarse Alignment

We use the output of Section 5.1 to generate a bounding box for the target arrangement, while reducing the impact of outliers. We convolve a Gaussian with the locations given by Isomap and use a threshold to find connected components. The largest component of the result is considered the important area of our point distribution. Its bounding box is set to be the bounding box for the target arrangement.

5.3. Step III: Bipartite Matching

The inputs to the bipartite matching step are some object arrangement given by the dimensionality reduction step (Section 5.1) and a target arrangement scaled and translated to match the calculated bounding box (Section 5.2). We would like to assign images at starting locations $s_i = (x_i, y_i)$ to the target locations $t_i = (x'_i, y'_i)$ while minimizing the total distance moved. Formally, we are looking for a permutation π of $[n]$ such that $\sum_{i=1}^{n} d(s_i, t_{\pi(i)})$ is minimized, where $d()$ denotes Euclidean distance.

In order to solve the problem and find a *globally* minimal solution, we construct a bipartite graph. On one side of the graph we have a vertex for each starting location, on the other side we have a vertex for each target location (a total of n vertices on each side). We construct a complete bipartite graph such that edge e_{ij} is given a weight $d(s_i, t_j)$. By construction, finding a minimal bipartite matching on this graph will minimize our objective function. We use the Hungarian algorithm [Kuh55]) to solve the minimal bipartite matching problem.

5.4. Optional: Random Refinement

In Section 3.2, we discussed that random swapping is not an effective overall strategy. However, it is often able to improve arrangements at least a small amount in a reasonable amount of time. We leave it as an optional refinement step to run some number of random swaps, as time allows (where swaps are only kept if they improve the objective function). Figure 6 compares the performance of random swaps as the sole solution strategy, vs. as a refinement on IsoMatch. Notice that IsoMatch's initial result (left part of dashed line) is comparable to 10^5 swaps starting from a random permutation (right part of solid lines) while being much faster. Starting from an IsoMatch arrangement, further improvement is possible via swapping (right part of dashed line).

5.5. Optional: Location Constraints

For an additional level of user control, specifying fixed locations for some images is an easy extension to our current formulation. Assume we want to fix image i in location j. In the bipartite graph construction step, we assign $e_{ij} = 0$. This defines that placing i in location j incurs zero cost, but it does not yet guarantee the placement. We also multiply all other edges with some large constant α, which ensures that the assignment of i to location j will indeed take place. The softness of these constraints can be varied with α, allowing the assignments to be more preference than requirement. This variation could be the result of some other fitness function, for example if a single image in a target arrangement was to be presented larger than the other images, the edge weights in the bipartite graph could be modulated by the quality (sharpness, resolution, clarity, etc.) of the images, to preferentially place a more suitable image in the special position.

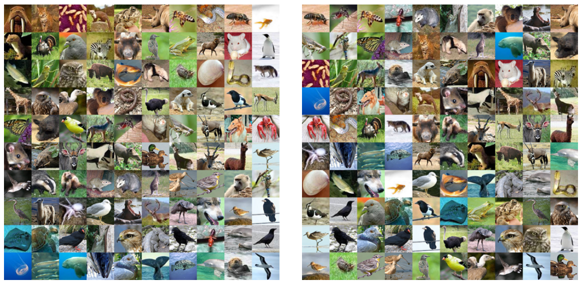


Figure 7: Grid layouts can convey additional information, complementing a photo album. Here we arrange 100 images of animals using either average colors (left) or word similarity using WordNet (right). Note how animals on the right are clustered according to taxonomy: insects (top-left), horned animals (middle-right), small birds (lower-left).

6. Applications

In this section we consider a variety of applications of the algorithmic framework introduced in Section 5.

6.1. Photo Grids

Finding an image within a large collection can be like searching for a needle in a haystack. Sorting the images according to some intuitive measurement like overall colors or semantic content can assist in these tasks. Studies such as that of Schoeffmann and Ahlstrom [SA11] and the one described in Section 4 demonstrate that such arrangements can improve image retrieval tasks. A simple way to arrange images is by color similarity; we found that mean RGB value differences yield the best arrangements (Figure 1).

Some datasets have an inherent structure in them, such as taxonomy information regarding objects in the images. Figure 7 presents a collection of animals. We show two grid layouts: one by color and one according to taxonomy. The taxonomy layout was created using WordNet [Fel98]. For each pair of animals we calculate the inverse of the number of edges in the path between these two words in the WordNet hierarchy. Feeding these distances to our system produces a 10x10 grid in which similar animals are clustered together. Such a grid can be useful, for example, as an index page for a website with information about the different animals. Another example for this type of arrangement would be an online shopping catalog, where items are sorted according to their similarity or according to peoples' purchasing habits (a seamless integration of the ubiquitous feature "people who bought x also bought y" into the main layout of the website).

6.2. Spherical Grids

As mentioned previously, our algorithm is not limited to 2D arrangements. We demonstrate this by placing the images on a 3D sphere (we can actually support a manifold of arbitrary

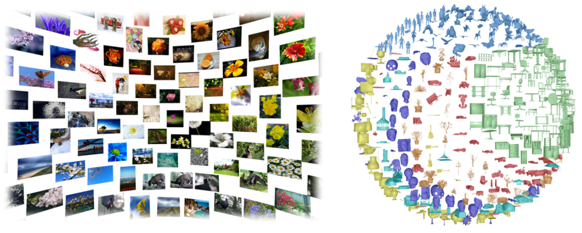


Figure 8: 3D layouts. Left: A view from inside a sphere of images arranged by color. Right: An external view of a sphere of 3D models from the Princeton Shape Benchmark, arranged by shape.

topology). The user is placed within the sphere, such that she sees an apparent 2D grid of images that extends infinitely in both the horizontal and vertical directions (Figure 8). The user can scroll in any direction and see a continuous, smoothly varying arrangement of images (where neighbors are similar). Our pipeline stays the same for 3D layouts. We use Isomap to project into 3D and the bipartite matching weights are euclidean distances between Isomap's output and 3D points on the surface of the sphere. The locations on the sphere were obtained by iteratively trying to minimize the energy, given repulsion forces between the points. The uniform arrangement prevents "squeezing" artifacts at the poles (as opposed to Quadrianto et al. [QSST10]). For this application, the descriptors are as follows: for each image we calculate its average HSL value and project that value to the outside of the HSL cone. The distance between a pair of images is defined to be the arc-length between these projections. This type of distance encourages a spherical output from the Isomap step.

While our main goal was to arrange images, the system is general purpose and can arrange any type of data for which we can supply pairwise distances. We demonstrate this by arranging a grid of 3D models. Models from the Princeton Shape Benchmark [SMKF04] are placed on a sphere such that similar models are next to each other. The pairwise distance is defined to be the reciprocal of the number of edges in the path that connects the models, using the Princeton Shape Benchmark hierarchy as our graph (Figure 8).

6.3. Infographics

Infographics (short for *information graphics*) are graphical illustrations representing some underlying data. These illustrations are extremely useful in conveying information and explaining data in an instantaneous way, especially for hard to grasp numerical or quantitative differences. There are many tools that support the creation of infographics, and these usually come in two flavors: manual and task-specific. The manual tools are more traditional vector art manipulation tools, and require the artist to create the infographic

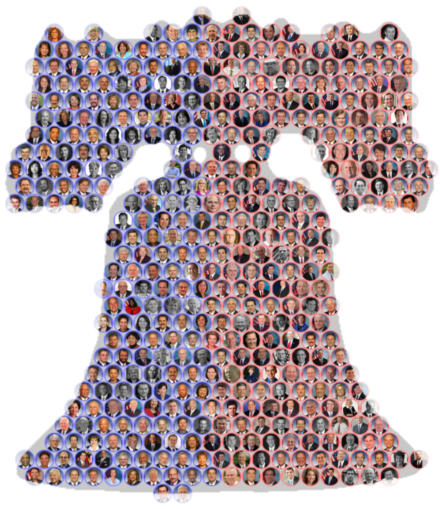


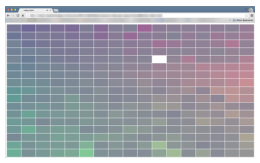
Figure 9: Infographics. US congress members arranged atop the liberty bell by voting record similarity and colored by party affiliation to illustrate partisanship. Notice that party affiliation was introduced after the fact, and was not part of the data given to our algorithm. The two parties were automatically detected from voting records.

from basic primitives such as shapes and lines. The task specific tools automatically create very rich infographics from data, but the data has to be of specific type (for example: a LinkedIn profile or some measured quantity for each of the 50 United States). The work of Reinert et al. [RRS13] makes pleasing infographics, but cannot support use cases where specific target locations are required, or where the arrangement may have holes. We handle the more general case.

Our system can automate placement aspects of the creation process for infographics based on pairwise distance data. For example, consider a visualization that aims to show US members of congress. The artist may have a specific layout in mind, such as a rectangular grid, a circle, or the shape of the liberty bell. On top of such a layout, the artist wishes to convey an extra layer (or layers) of data. Figure 9 shows US congresspeople, placed by their voting records, onto an arrangement in the shape of the liberty bell. The pairwise distance between two individuals is defined as the number of instances in which the pair voted in the same way for the 113th congress. The frame around each portrait is either red or blue, according to party affiliation (republican or democrat). Note that the two distinct groups emerge automatically, visualizing US partisanship.

6.4. Sound Layout

Musicians with many audio samples usually manage their library alphabetically by filename or in folders according to some manual classification, which is difficult to maintain and navigate. We followed the work of Fried et al. [FJOF14] and created an interface for browsing a collection of snare drum audio samples (shown inset). Our interface is a grid of colored rectangles each of which plays a different sample upon mouse hover. The grid is arranged in a meaningful way that aids exploration and retrieval, and the colors are a visual cue as to the similarities between samples. To compute the sound descriptors we first apply a threshold onset detector on a normalized time domain signal. Then, we take the first two 2048-sample (46ms) windows and compute their Mel-frequency Cepstral Coefficient (MFCC) [Log00]. We further refine the feature space through metric learning [XNJR03] by labeling 61 (different) sound samples and using them to learn a good mapping from initial descriptors to refined descriptors that respect the labeling. We refer the reader to the accompanying video to see (and hear) this application.



6.5. Data Layout

A difficult part of planning formal events such as weddings is seating assignments, in which family and friends are arranged by social affinity. A similar problem is room assignment in a college dormitory setting. These problems can be viewed as arrangements of people on arbitrary grids (seating charts / floor plans), using social affinity as a distance function. Social affinity could be computed using information from a social network such as Facebook or Google+, though quantifying these relationships is an open area of research.

We demonstrate a simplified form of the problem: arranging family members onto wedding seat charts. We define our distance function based on the combined family tree of the bride and groom for 111 people total, where edge weights are based on the nature of the relationship (e.g. parent-child has a weight of 1.0 while marriage has a weight of 1.3 and sibling has a weight of 1.6). Distances between people are the reciprocal of the path length between them on this graph, while distances between available seats are euclidean. To visualize the seating assignments, family members are assigned colors based on their position in the family tree relative to the bride (blue to purple) and groom (green to red), though these colors do not impact the arrangement.

Figure 10 shows three possible seating arrangements, around rectangular tables, circular tables, or in concentric arcs. It demonstrates that our algorithm is able to ensure that close relatives sit near one another across a variety of scenarios. Additionally, the final energy computed for each arrangement can be used to select the best layout of seats among the available options.

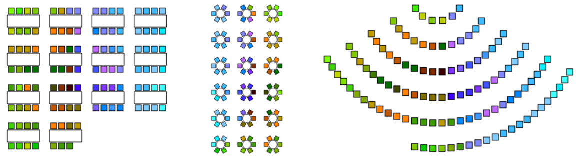


Figure 10: Seating assignments. Guests are colored according to their family tree graph distance from the bride (blue-purple) and groom (green-red). We show three different venue layouts. Note that we can generate assignments for many such layouts and automatically select the one with the best score (Eq. 2).

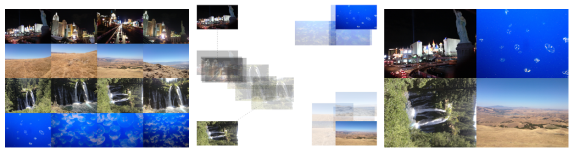


Figure 11: A photo album digest. The input collection of 16 images (left) is reduced to 4 representative images (right). We extend our bipartite matching algorithm to allow n-to-k matching, such that $n > k$. Middle: the input images, as placed by Isomap (transparent images) and the four selected images in their target grid locations (opaque images). Note our method keeps one image from each group of four similar images.

6.6. Photo Album Digests

We extended our system to create photo album “digests,” where a representative subset of the album’s photos are selected. Given a collection of photos of size n, we would like to extract a subset of size k ($k < n$), such that the subset is comprised of representative images. While representative images are hard to define, we want to achieve the intuitive goal of having the smaller collection “tell the same story” as the larger superset.

In its simplest form, we can create a digest with no modification to our algorithm. Instead of matching n target locations to n images, we match k target locations ($k < n$) to n images. We create a digest by selecting the closest point to each grid cell as the representative image for that cell. We rely on the fact the similar images will be close, thus the probability that two similar images get picked as representatives is small. This simple implementation is demonstrated in Figure 11.

6.7. Hierarchical Visualization

The downside of the digest method is that it relies on Isomap’s output being spread (roughly) evenly in the 2D space. Recognizing that, we enhance the method with an extra clustering step and add support for hierarchical visualizations. Given a set of images of size n, and a target arrangement of k images in each level of the hierarchy, the enhanced algorithm is as follows:

1. Use Isomap to transform distances between image descriptors to n 2D coordinates.
2. Use bipartite matching to match k of those images to grid locations.
3. Use the k images as an initialization to a clustering algorithm that produces equally sized clusters $\{c_i\}_{i=1}^{k}$.
4. In each cluster, pick the closest point to the centroid as its representative (for a total of k representatives).
5. Match the k representative images to the k grid points using bipartite matching.
6. Repeat recursively for each cluster $\{c_i\}_{i=1}^{k}$.

We have introduced a new step of clustering images into equally sized sets (± 1 if $n \not\equiv 0 \pmod{k}$). That step can be implemented with our current infrastructure. Given n points and k centroids we duplicate the centroid so that we will have n centroids, and now we calculate a minimal bipartite match. Since every centroid was duplicated $\lceil n/k \rceil$ or $\lfloor n/k \rfloor$ times, we will match that number of points to each centroid, thus creating clusters of equal size. This approach finds the global minimum in terms of total distance from centroid. It is, however, impractical for very large datasets, and for those we use an iterative approach similar to K-means, while selecting equal number of points for each cluster per-iteration (assigning greedily to centroids in increasing order of distances).

Note that an alternative approach to creating a hierarchy is to directly assign images to a pyramidal 3D grid, as done by [QKTB10]. However, this does not create a strict hierarchy – e.g. an image in layer l_i does not necessarily represent the children images below it in layer l_{i+1}. This approach is more akin to arranging images in a 3D volume that happens to be pyramid-shaped. Our algorithm on the other hand, strictly enforces parent-child relationships between images, thereby creating a true hierarchical visualization in which parent images represent child images in aggregate.

We tested our hierarchical visualization using a subset of the SUN database [XHE*10]: 4096 images from 256 different categories. The distance between images was calculated using the Extended Gloss Overlaps measure (lesk) [BP03] between the image categories, using the WordNet database [Fel98] (the lesk distance calculates overlaps in the glosses of words). In lower levels of the hierarchy (i.e. when $n = k$ and we no longer create a digest) we used differences in mean RGB values. Some results can be seen in Figure 12. The interactive browser is demonstrated in the accompanying video.

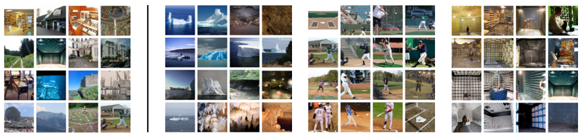


Figure 12: Hierarchical image collection visualization. Left: the upper (root) level of the hierarchy. Note how images differ and none of them belong to the same SUN category, providing an overview of the database. Rest: three examples from the lowest level of the hierarchy. Notice how the depicted scenes are very similar within a cluster and very different across clusters.

Method	Dataset			
	Animals (100)	KS-DB (320)	Flowers (580)	3D Models (907)
RandPerm	0.442 (σ=0.006)	0.453 (σ=0.005)	0.566 (σ=0.004)	0.354 (σ<0.001)
Ours	0.349	0.317	0.259	0.309
Ours-RS	**0.264**	**0.290**	**0.243**	**0.300**

Table 1: Scored results, as computed by Eq. 2. We test four datasets with the size of each set indicated. We arrange each set using three algorithms: random permutations, our algorithm, and our algorithm plus 10,000 random swaps. For random permutations, we show the mean and standard deviation of 1,000 trials. The best result for each dataset is marked in bold, lower is better.

7. Evaluation

Ideally applications like those in Section 6 would all be evaluated by user studies like that of Sections 4. However, such studies are task-specific and do not necessarily generalize to other applications. Furthermore, just because a particular method (such as IsoMatch) achieves a particular performance on one or a few grids does not mean that other input data would yield similar results; studies need to involve many data sets in order to provide confidence in the outcomes. Therefore, because the studies themselves cost time and effort, evaluating a set of methods across a range of applications with high confidence is prohibitive. Fortunately, our energy function (Equation (2)) provides an objective measure of success that is consistent across different distance functions, different problem sets, different algorithms, and different applications, which makes it an ideal means of evaluation. Figure 2 suggests that our energy function is intuitively correlated with an aesthetic sense of arrangements for aesthetic distance functions such as mean RGB difference. Moreover, in section 4 we validated the function via a user study with respect to a fairly generic image searching task. Thus, in this section we use our energy function as a proxy for evaluating the performance of the IsoMatch algorithm with respect to naive and previous solutions.

7.1. Distance Preservation

Table 1 shows distance preservation results as evaluted by Equation (2). We compare a baseline measurement to our method and to our method with random refinement (Section 5.4), on four datasets. The "Animals" database consists of 100 animal names and images arranged by word distance. "KS-DB" are the 320 images used in kernelized sorting [QSST10], arranged using the descriptors given in their paper (down-sampled 40x40 L*a*b* images). "Flowers" is a set of 580 photographs of flowers, arranged using L2 distance between mean RGB values. "3D Models" are 907 polygon models from the Princeton Shape Benchmark arranged by tree distance in the shape hierarchy. For more information regarding these datasets and our results, see Sections 6.1 and 6.2. The baseline results in Table 1 establish a lower bound on performance that shows the difficulty of the problem (via the standard deviation σ). Our results significantly out perform the baseline in all datasets, sometimes by more than 50x standard deviations. Additional refinement is able to further improve the results, mostly limited by the runtime allotted to it.

We found that the variance in results can be quite large, depending on the specific dataset (e.g. not all collections of 100 animals will yield similar results). For this reason we performed an experiment with 500 data sets and report aggregate behavior. For each experiment, we randomly select 100 images from the SUN database [XHE*10], which contains 108,754 images. We arrange them on a 10x10 grid using five methods: kernelized sorting (KS) [QSST10], kernelized sorting with random swaps (RS, using the same random refinement step as in our algorithm), self-sorting maps (SSM) [SG14], IsoMatch, and IsoMatch with random swaps. For each arrangement, we compute the energy in Eq. 2, shown in aggregate in figure 13 using Matlab's "boxplot" function. In the plot, two medians are significantly different ($p < 0.05$) if their notches do not overlap. This shows with statistical significance in aggregate for this task that: (1) adding random swaps after KS outperform KS alone; (2) IsoMatch without swaps outperforms KS improved by swaps; (3) random swaps improves the raw output of IsoMatch; and (4) IsoMatch without swaps also outperforms SSM, which we do not refine with random swaps because it is already a swap-based algorithm.

In Figure 14 we evaluate the same data sets from Figure 13 using the objective functions of KS and SSM. We find that KS out performs both IsoMatch and SSM with respect to its own objective function, but that IsoMatch and SSM both outperform KS as measured by normalized cross-correlation (the objective used in SSM) with IsoMatch slightly ahead of SSM (with statistical significance). Finally we note that in these plots as well as those of 13 the variance of energies for KS is substantially higher than those of the other methods, which may be a drawback in some application areas.

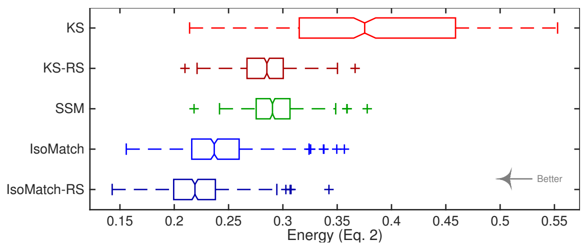


Figure 13: Algorithm comparison. We sampled 500 random subsets of 100 images each from the SUN database [XHE*10]. We show results for 5 algorithms, from top to bottom: kernelized sorting [QSST10], kernelized sorting with random swaps, self-sorting maps [SG14], IsoMatch and IsoMatch with random swaps. Lower values are better. Notice that IsoMatch outperforms kernelized sorting, even when adding random swaps to the latter (which is not part of the original implementation).

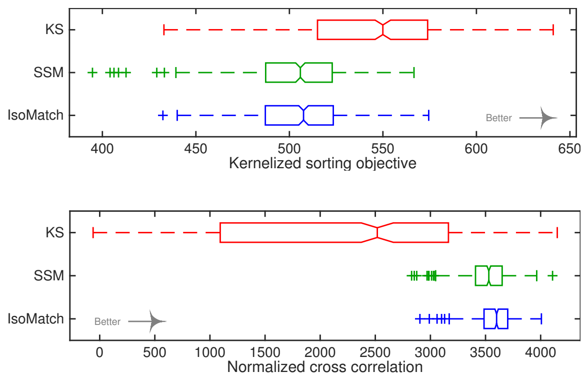


Figure 14: Objective function comparison. Using the same methodology as Figure 13, we compare the results as measured by the objective functions for kernelized sorting (above) and normalized cross-correlation used by self-sorting maps (below). Higher values are better.

7.2. Runtime

Our algorithm's time complexity can be calculated by looking at its two major parts: Isomap and bipartite matching. Assuming a constant number of nearest neighbors and constant input and output dimensions, Isomap's runtime is bounded by $O(n^2 \log n)$ due to the Dijkstra phase, where n is the number of elements. The Hungarian algorithm [Kuh55] used for bipartite matching takes $O(n^3)$. Other parts of the algorithm are negligible. Putting it all together we get $O(n^3)$ runtime.

For comparison, we note that Kernelized Sorting [QSST10] also solves an assignment problem, thus incur $O(n^3)$. The improved method of Convex Kernelized Sorting [DGV12] tries to solve a non-linear system, and is the slowest algorithm. As an example, when running the respective algorithms on the 320 KS-DB images, the runtimes of convex kernelized sorting, kernelized sorting and our algorithm are 10721, 5, and 3 seconds, respectively. On the same dataset, 10000 random swap iterations take 76 seconds. Self sorting maps [SG14] are faster than both kernelized sorting and IsoMatch.

All the above experiments were run on a MacBook Pro, 2.3 GHz Intel Core i7 with 8 GB of RAM and an SSD. We used the KS Python implementation, CKS Matlab implementation and SSM GPU accelerated Java implementation given by their respective authors, and our own Matlab implementation.

8. Conclusion

There are limitations worth considering as potential future work. It would be helpful to support additional types of constraints in the arrangements, such as requiring two items to be adjacent in the output arrangement. Our algorithm does not provide an obvious mechanism to support this type of constraint. In another direction, our performance on large sets is currently limited by computing the minimum bijection. It is interesting to consider if there are effective approximations that may be applied to improve the runtime and enable collections orders of magnitude larger. Until then, we have demonstrated a novel method to place a collection of objects onto a dense grid, or any other arrangement, while respecting pairwise distances between objects. By utilizing various distance functions, we have shown the usefulness of our method for infographics, photo and model exploration, beautification and photo summarization. Our method has a simple formulation and compares well against other state of the art methods.

9. Acknowledgements

We thank the reviewers for their helpful suggestions. This work was funded in part by generous support from Google Inc., Adobe Corp., and the National Science Foundation (IIS-1421435).

References

[Ber11] BERTINI E.: Quality metrics in high-dimensional data visualization: An overview and systematization. *IEEE Transactions on Visualization and Computer Graphics 17*, 12 (2011), 2203–2212. 3

[BHL05] BLANCHARD F., HERBIN M., LUCAS L.: A new pixel-oriented visualization technique through color image. *Information Visualization 4*, 4 (Oct. 2005), 257–265. 2

[BP03] BANERJEE S., PEDERSEN T.: Extended gloss overlaps as a measure of semantic relatedness. In *International Joint Conference on Artificial Intelligence* (2003), pp. 805–810. 9

[BSW02] BEDERSON B. B., SHNEIDERMAN B., WATTENBERG M.: Ordered and quantum treemaps: Making effective use of 2D space to display hierarchies. *ACM Transactions on Graphics 21*, 4 (Oct. 2002), 833–854. 2

[DGV12] DJURIC N., GRBOVIC M., VUCETIC S.: Convex Kernelized Sorting. In *AAAI Conference on Artificial Intelligence* (2012), pp. 893–899. 3, 11

[Dwy09] DWYER T.: Scalable, versatile and simple constrained graph layout. In *Proceedings of the Eurographics Conference on Visualization* (2009), pp. 991–1006. 2

[ED07] ELLIS G., DIX A. J.: A taxonomy of clutter reduction for information visualisation. *IEEE Transactions on Visualization and Computer Graphics 13*, 6 (2007), 1216–1223. 2

[Fel98] FELLBAUM C.: *WordNet: An Electronic Lexical Database*. Bradford Books, 1998. 7, 9

[FJOF14] FRIED O., JIN Z., ODA R., FINKELSTEIN A.: AudioQuilt: 2D Arrangements of Audio Samples using Metric Learning and Kernelized Sorting . *International Conference on New Interfaces for Musical Expression (NIME)* (2014), 281–6. 2, 4, 8

[FT09] FRISHMAN Y., TAL A.: Uncluttering graph layouts using anisotropic diffusion and mass transport. *IEEE Transactions on Visualization and Computer Graphics 15*, 5 (2009), 777–788. 3

[ISYK12] INOUE K., SHIMOZONO S., YOSHIDA H., KURATA H.: Application of approximate pattern matching in two dimensional spaces to grid layout for biochemical network maps. *PloS one 7*, 6 (Jan. 2012), e37739. 2

[KNJ*07] KOJIMA K., NAGASAKI M., JEONG E., KATO M., MIYANO S.: An efficient grid layout algorithm for biological networks utilizing various biological attributes. *BMC bioinformatics 8* (Jan. 2007), 76. 2

[Koh82] KOHONEN T.: Self-Organized Formation of Topologically Correct Feature Maps. *Biological cybernetics 69* (1982), 59–69. 2

[Kuh55] KUHN H.: The Hungarian Method for The Assignment Problem. *Naval research logistics quarterly 2* (1955), 83–87. 6, 11

[LHN*13] LIU X., HU Y., NORTH S., LEE T., SHEN H.: *Correlatedmultiples: Spatially coherent small multiples with constrained multidimensional scaling*. Tech. rep., Oregon State University, 2013. 2

[LK05] LI W., KURATA H.: A grid layout algorithm for automatic drawing of biochemical networks. *Bioinformatics (Oxford, England) 21*, 9 (May 2005), 2036–42. 2

[Log00] LOGAN B.: Mel frequency cepstral coefficients for music modeling. In *International Symposium on Music Information Retrieval* (2000), pp. 1–11. 8

[LV07] LEE J. A., VERLEYSEN M.: *Nonlinear dimensionality reduction*. Springer, 2007. 1, 3, 5

[PV91] PARDALOS P., VAVASIS S.: Quadratic programming with one negative eigenvalue is NP-hard. *Journal of Global Optimization 1*, 1 (1991), 15–22. 4

[QKTB10] QUADRIANTO N., KERSTING K., TUYTELAARS T., BUNTINE W. L.: Beyond 2D-grids: a dependence maximization view on image browsing. In *Multimedia Information Retrieval* (2010), pp. 339–348. 1, 3, 9

[QSST10] QUADRIANTO N., SMOLA A. J., SONG L., TUYTELAARS T.: Kernelized sorting. *Transactions on Pattern Analysis and Machine Intelligence 32*, 10 (Oct. 2010), 1809–21. 2, 3, 4, 5, 7, 10, 11

[RRS13] REINERT B., RITSCHEL T., SEIDEL H.: Interactive by-example design of artistic packing layouts. *ACM Transactions on Graphics (TOG)* (2013). 1, 3, 8

[RS00] ROWEIS S. T., SAUL L. K.: Nonlinear dimensionality reduction by locally linear embedding. *Science 290*, 5500 (Dec. 2000), 2323–6. 2, 5

[SA11] SCHOEFFMANN K., AHLSTROM D.: Similarity-Based Visualization for Image Browsing Revisited. *International Symposium on Multimedia* (Dec. 2011), 422–427. 1, 4, 7

[SG11a] STRONG G., GONG M.: Data organization and visualization using self-sorting map. In *Proceedings of Graphics Interface 2011* (School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada, 2011), GI '11, Canadian Human-Computer Communications Society, pp. 199–206. 2

[SG11b] STRONG G., GONG M.: Similarity-based image organization and browsing using multi-resolution self-organizing map. *Image Vision Comput. 29*, 11 (Oct. 2011), 774–786. 2

[SG14] STRONG G., GONG M.: Self-sorting map: An efficient algorithm for presenting multimedia data in structured layouts. *Multimedia, IEEE Transactions on 16*, 4 (June 2014), 1045–1058. 2, 5, 10, 11

[SH94] SAMARIA F. S., HARTER A. C.: Parameterisation of a stochastic model for human face identification. In *Applications of Computer Vision, 1994., Proceedings of the Second IEEE Workshop on* (1994), IEEE, pp. 138–142. 4

[SJGE13] STRONG G., JENSEN R., GONG M., ELSTER A.: Organizing visual data in structured layout by maximizing similarity-proximity correlation. In *Advances in Visual Computing*, Bebis G., Boyle R., Parvin B., Koracin D., Li B., Porikli F., Zordan V., Klosowski J., Coquillart S., Luo X., Chen M., Gotz D., (Eds.), vol. 8034 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2013, pp. 703–713. 2

[SMKF04] SHILANE P., MIN P., KAZHDAN M., FUNKHOUSER T.: The Princeton shape benchmark. In *Shape Modeling International* (June 2004). 7

[SvLB10] SCHRECK T., VON LANDESBERGER T., BREMM S.: Techniques for precision-based visual analysis of projected data. *Information Visualization 9*, 3 (2010), 181–193. 3

[TdSL00] TENENBAUM J. B., DE SILVA V., LANGFORD J. C.: A global geometric framework for nonlinear dimensionality reduction. *Science (New York, N.Y.) 290*, 5500 (Dec. 2000), 2319–23. 2, 5

[Wat05] WATTENBERG M.: A note on space-filling visualizations and space-filling curves. In *IEEE Symposium on Information Visualization* (2005), pp. 181–186. 2

[WD08] WOOD J., DYKES J.: Spatially ordered treemaps. *IEEE transactions on visualization and computer graphics 14*, 6 (2008), 1348–55. 2

[WFM*12] WONG P. C., FOOTE H., MACKEY P., JR. G. C., HUANG Z., THOMAS J. J.: A space-filling visualization technique for multivariate small-world graphs. *IEEE Transactions on Visualization and Computer Graphics 18*, 5 (2012), 797–809. 2

[XHE*10] XIAO J., HAYS J., EHINGER K. A., OLIVA A., TORRALBA A.: SUN database: Large-scale scene recognition from abbey to zoo. In *Computer vision and pattern recognition* (2010), pp. 3485–3492. 9, 10, 11

[XNJR03] XING E. P., NG A. Y., JORDAN M. I., RUSSELL S.: Distance metric learning, with application to clustering with side-information. *Advances in Neural Information Processing Systems 15* (2003), 505–512. 8

[YS11] YAMADA M., SUGIYAMA M.: Cross-Domain Object Matching with Model Selection. In *International Conference on Artificial Intelligence and Statistics* (2011), pp. 807–815. 3