Beyond 2D-Grids: A Dependence Maximization View on Image Browsing

Novi Quadrianto*
SML, NICTA & RSISE, ANU
Canberra ACT, Australia
first.last@nicta.com.au

Kristian Kersting*
Fraunhofer IAIS
Sankt Augustin, Germany
first.last@iais.fraunhofer.de

Tinne Tuytelaars
ESAT-PSI, KU Leuven
Leuven, Belgium
first.last@esat.kuleuven.be

Wray L. Buntine
SML, NICTA & RSISE, ANU
Canberra ACT, Australia
first.last@nicta.com.au

ABSTRACT

Ideally, one would like to perform image search using an intuitive and friendly approach. Many existing image search engines, however, present users with sets of images arranged in some default order on the screen, typically the relevance to a query, only. While this certainly has its advantages, arguably, a more flexible and intuitive way would be to sort images into arbitrary structures such as grids, hierarchies, or spheres so that images that are visually or semantically alike are placed together. This paper focuses on designing such a navigation system for image browsers. This is a challenging task because arbitrary layout structure makes it difficult – if not impossible – to compute cross-similarities between images and structure coordinates, the main ingredient of traditional layouting approaches. For this reason, we resort to a recently developed machine learning technique: kernelized sorting. It is a general technique for matching pairs of objects from different domains without requiring cross-domain similarity measures and hence elegantly allows sorting images into arbitrary structures. Moreover, we extend it so that some images can be preselected for instance forming the tip of the hierarchy allowing to subsequently navigate through the search results in the lower levels in an intuitive way.

Categories and Subject Descriptors

H.5 [**Information Interfaces and Presentation**]: User Interfaces; I.4 [**Image Processing and Computer Vision**]: Applications

General Terms

Algorithms, Design, Theory

*Authors contributed equally.

MIR'10, March 29–31, 2010, Philadelphia, Pennsylvania, USA.

Keywords

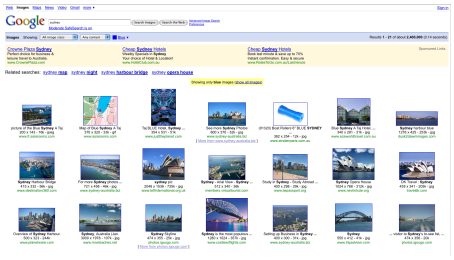
Multimedia visualization and browsing, Image layout, Image browsing

1. INTRODUCTION

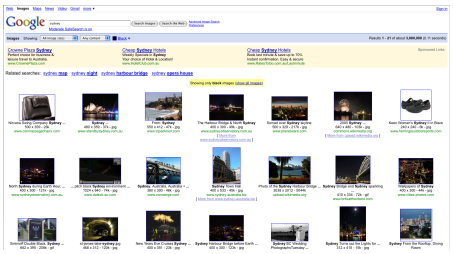
Consider a typical interaction between a user and a web image search engine. A user executes a query and considers hundreds, sometimes thousands of images returned by the search engine, looking for this one ideal picture or just browsing the results for fun. Most existing search engines, such as Google, Yahoo!, and Bing, among others, return a set of pages, where each page contains several sorted according to how relevant they are to the query, cf. Fig. 1). However, even though this type of interface has proven its value in common web search engines, it may not be the most appropriate format for presenting image search results. André *et al.* [1] have explored the differences in user behavior between image search and web search through informal interviews and query log analysis. They found image search is often more exploratory than web search. Also, it is often used purely for entertainment, to play or explore in visual space with no clear end goal. On average, image searchers view more pages than web searchers, spend more time looking at those pages, and click on more results. They also get sidetracked more often, when something else of interest catches their eye. Hence, a good user interface should support easy exploration and browsing.

Typically, however, not all the returned images are even related to the query. This situation is caused by the usage of keywords – the filename of the image and text near the image on a web page [5] – to represent visual characteristics of an image, rather than the actual image content. It can happen that visually different images have the same keywords while visually similar images have totally different keywords. Thus, it is not uncommon that in the returned results, the relevant images are evenly mixed with irrelevant images. The user then has to scan randomly layouted images on each page and devotes considerable efforts in navigating across the page to discover the images of his/her interest.

Content-based image retrieval (CBIR) [24, 22] was introduced to overcome the difficulty of keyword-based image search. In CBIR, images are indexed by their visual content, such as color or texture. However, the performance of CBIR is still far from being ready to be deployed as real-



(a) Blue images only for 'Sydney' query



(b) Black images only for 'Sydney' query

Figure 1: Google ranking based list interface for 'Sydney' query. Returned images are displayed on a set of pages and ranked based on how relevant they are to the query.

(a) Image layouting on 2D grid

(b) Image layouting on 3D sphere

Figure 2: Kernelized sorting layouting interface for 'Sydney' query. Returned images are organized onto a structure (2d grid and 3D sphere) such that similar color images are located close by.

world commercial image retrieval engines. One of the identified problems with the current CBIR approaches is the reliance on visual similarity for judging semantic similarity, which may be problematic due to the semantic gap [24] between low-level content and higher-level concepts. More semantic features should be utilized to bridge this gap. In the context of a web image search scenario, keywords can be used to enrich the visual content features [23]. A keywords-based search can then be refined, re-ranked or reorganized based on visual similarity. This is exactly what several commercial search engines have recently started to do. They support queries related to visual content, such as color of the returned images or whether the returned images must contain a face.

To summarize, a good user interface should also put visually and/or semantically alike images together in order to support easy exploration and browsing. Therefore, it is not surprising that significant research efforts have been devoted to exploring ways in effectively presenting search results to users. Arguably, a meaningful layout of the search results is as crucial as the search accuracy itself. Layouting of image search results will allow users to have a good global overview of a large amount of returned images rather than the traditional page-by-page ranked list. Rodden *et al.* [19] have shown that layouting a set of images according to their similarity is indeed beneficial.

Existing layout approaches, however, have consider rather rigid layout structures only. A general approach that features general structures such as hierarchies or spheres and in turn opens new dimensions for guiding the user's search process has not been proposed yet. This is exactly what we do in this paper. We present a layouting approach that sorts images

1. into *arbitrary layout structures* such as grids, hierarchies, or sphere so that
2. *visually or semantically alike images are placed together*,

cf. Fig. 2. Note that this is a challenging task. Arbitrary layout structures make it difficult – if not impossible – to compute cross-similarities between images and structure elements, the basic ingredient of traditional layouting approaches. Therefore, we have to resort to a recently developed machine learning technique called *kernelized sorting* [18]. Kernelized sorting allows one to perform matching without cross-domain similarity measure by maximizing the dependency between sets of objects, in our case images and layout coordinates. This naturally allows one to organize images into arbitrary structures. We demonstrate this by presenting layouts of images into several arbitrary structures: 2D grids, spheres, hierarchies of 2D grids, and hierarchies of spheres.

As a second contribution, we extend kernelized sorting so that it can take a user's preferences into account. Specifically, the user can pre-place a few images at particular lo-

cations on the 2D grid, sphere, etc. This is useful to guide the user's search in a similar way as CBIR systems do. For instance, using a hierarchical structure, the user can provide few query images at the top level of the hierarchy and subsequently navigate through the search results in the lower levels.

The remainder of the paper is organized as follows. After reviewing related work, we describe the techniques behind our image browsing approach, i.e. kernelized sorting and a novel informed variant in Section 3. We then explain the two essential ingredients for kernelized image sorting in Section 4: similarity measures for images and on the layout structure. Before concluding, we demonstrate kernelized image sorting in the context of a web image browsing application in Section 5.

2. RELATED WORK

Other researchers have looked into the problem of better visualizations of search results as well. Among them, methods based on multi-dimensional scaling (MDS) are the most widespread [19, 21]. Other methods for low dimensional object layout are self organizing maps [13], maximum variance unfolding [29], locally-linear embedding [20], generative topographic map [4], or stochastic neighbour embedding [9]. [17] experiments with several of these in an image collection visualization context and concludes that stochastic neighbour embedding yields the best results. However, most of these (non)linear dimensionality reduction based approaches suffer from the problem that the low dimensional presentation is nonuniform. This has the advantage of revealing the cluster structure but given a limited screen size the presentation is often undesirable, with some very cluttered regions where images highly overlap combined with empty spaces elsewhere. It has been studied that such aggressive overlapping in the visualization will affect users' understanding of the content of an image and thus will prevent them from finding images of interest [11]. To reduce this effect, [17] adds a visibility cost function. We overcome this problem by organizing the images based on a predefined grid or other embedding structure, based on kernelized sorting. Kernelized sorting allows to layout the images based on some similarity criterion in arbitrary structures (be it 2D grids, spirals, spheres, or even hierarchical structures). Moreover, this approach could also be adapted to do maximum variance unfolding, i.e., to compute low-dimensional embedding similar to MDS.

Apart from the degree of image overlap, the choice of the embedding structure for image presentation also has an important effect on a user's ability to have a good global overview of large amount of images. Many existing works focus on organizing images into a specific structure (e.g. 2D grids [14, 27], spirals [27], or 3D walls [1]) without being able to generalize easily to other potentially more effective structures. Therefore, what we would like to have is a method which is able to produce a non-overlapping embedding while also providing us flexibility in choosing the underlying structure. We will show that organization of images based on kernelized sorting provides exactly the two properties specified above. We will briefly review the kernelized sorting technique in the next section.

Finally, it is worth pointing out that beside the similarity-based layout and grid layout, there exist many other image layout approaches, such as thread layout [8, 7] for revealing linked sequence of shots based upon an aspect of their content, hierarchical layout [3] for revealing hierarchy structures of the images, graph layout [12] for revealing links among images, map layout [28] to reveal geographic location tags of images, and time quilt and time line layout [10] for time series images. All of them are subsumed by kernelized sorting. In this way, we propose a unifying framework that allows to focus on finding the right structure and not developing algorithms to sort images into the selected structures.

[1] `http://www.cooliris.com`

3. KERNELIZED SORTING

We will first review kernelized sorting and then show how to extend it so that a user's matching preferences are taken into account.

3.1 The Basic Setting

Kernelized sorting (KS) [18] is a general technique to perform matching between pairs of objects from different domains which only requires a similarity measure within each of the two domains. Indeed, the virtues of performing matching without the need of a cross-domain similarity measure allow us to organize images into arbitrary structures. We will begin reviewing the technique with some notations and definitions.

Denote by $X = \{x_1, \ldots, x_m\} \subseteq \mathcal{X}$ and $Y = \{y_1, \ldots, y_m\} \subseteq \mathcal{Y}$ two sets of objects between which we would like to find correspondences. Here m denotes the size of each set. KS tries to find a permutation π that maps $x_i \to y_{\pi(i)}$ by maximizing the dependence between the two sets of objects under a certain dependency measure. This also explains the title of the paper as we will apply KS for image layouting. In KS, the Hilbert Schmidt Independence Criterion (HSIC) [25] dependency measure is used and is maximized over the permutation group to find good matches. Hence, the resulting optimization problem of KS is given as

$$\pi^* = \operatorname{argmax}_\pi \operatorname{tr} \bar{K} \pi^\top \bar{L} \pi. \tag{1}$$

In the above equation, π denotes a permutation matrix whose entries are all 0 except that in row i, the entry $\pi(i)$ equals 1 and $K, L \in \mathbb{R}^{m \times m}$ are the kernel matrices for the set X and the set Y respectively, i.e. $K_{ij} = k(x_i, x_j)$ and $L_{ij} = l(y_i, y_j)$. Kernel $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ measures the pairwise similarity between objects in set X and likewise kernel l for set Y. Centering matrix H with $H_{ij} = \delta_{ij} - m^{-1}$ centers the observations of set X and set Y in feature space. The centered versions of K and L are then denoted as $\bar{K} := HKH$ and $\bar{L} := HLH$ respectively.

The KS optimization problem in (1) is NP hard [18]. Therefore, the first order lower bound of (1) is successively maximized instead. We refer to Algorithm 1 for a summarization of the KS algorithm.

3.2 Informed Sorting

Kernelized sorting – as employed in this paper – aims to organize a collection X of images such that images that are "similar" to each other are placed nearby in the layout structure Y. Next to the similarities among images $\bar{K}$ and layout positions $\bar{L}$, no information is provided to steer the layout of the images. In other words, kernelized sorting can be viewed as an unsupervised technique.

Algorithm 1 Kernelized Sorting

Input Two sets of objects $X = \{x_1, \ldots, x_m\}$ and $Y = \{y_1, \ldots, y_m\}$
Compute kernel similarity matrix K on set X
Compute kernel similarity matrix L on set Y
Center the kernel matrices:
$\bar{K} := HKH$ and $\bar{L} := HLH$ with $H_{ij} = \delta_{ij} - m^{-1}$
while not converge **do**
Solve linear assignment problem

$$\pi_{i+1} \leftarrow \operatorname*{argmax}_{\pi \in P_m} \left[\operatorname{tr} \bar{K} \pi^\top \bar{L} \pi_i \right]$$

$$\text{with } P_m := \left\{ \begin{array}{l} \pi \in \mathbb{R}^{m \times m} \text{ where } \pi_{ij} \geq 0 \text{ and} \\ \sum_i \pi_{ij} = 1 \text{ and } \sum_j \pi_{ij} = 1 \end{array} \right\}$$

end while
Return Locally optimum permutation matrix π^*

For image layouting, however, it might be beneficial to make use of a small amount of supervision to "guide" or "adjust" the layouting. Consider a typical interaction of a user with a web-based image search system. When the user submits a textual query to the system, the search engine returns a list of images often along with snippets extracted from the web documents the images appear in. The user looks at the list and, based on images and snippets, decides whether an image in the list is relevant to the query or not. Using a purely rank-based layout, this process can be quite costly as the user has to scan all images. In turn, important entities may never be considered by the user. To avoid this presentation effect, we instead want the user to express a small amount preference such as blue images are placed at "north pole" of the globe whereas black images are placed the "south pole".

More formally, we assume there is a small number of layout preferences $\mathcal{P} = \{(i, j)\}$ saying that the user prefers image i to be placed at position j. A first attempt to formalize the corresponding optimization problem could be to extend (1) as follows:

$$\pi^* = \operatorname{argmax}_\pi \operatorname{tr} \bar{K} \pi^\top \bar{L} \pi$$
$$\text{s.t.} \quad \pi_{ij} = 1 \quad \forall (i, j) \in \mathcal{P}$$

In turn, instead of solving a sequence of linear assignment problems to compute π^*, for instance using linear programming, we solve a sequence of *integer linear programs* that respect the $\pi_{ij} = 1$ preferences.

This approach indeed places i at position j but, unfortunately, tends to place images similar to i elsewhere in the layout structure. The reason is that image similarities and assignment preferences are unbalanced. To overcome this, we rebalance the cost matrix in each iteration. That is, in each iteration of KS, we are solving a linear assignment problem with a balanced version c' of the original cost matrix c. The cost matrix c' is computed as follows. Assume that we have l preferences $\{(i_l, j_l)\}$, $l = 1, 2, \ldots, k$. We relax the preference constraints as follows:

$$\forall i' : c_{ij} >= c_{i'j} \tag{2}$$
$$\forall j' : c_{ij} >= c_{ij'} \tag{3}$$

where $c_{ij} = \sum_{u,v} k_{iu} \pi_{uv} l_{vj}$ is the cost of assigning image i to position j. Both constraints can be written more compactly in matrix form. For instance, (2) can be written as

$$A\tilde{\pi} + b >= A'\tilde{\pi} + b' \tag{4}$$
$$\Leftrightarrow (A - A')\tilde{\pi} >= b' - b \tag{5}$$

where A is the matrix consisting of $a_{uv} = k_{i_u i_u} \cdot l_{j_v j_v}$, the matrix $\tilde{\pi}$ with $\tilde{\pi}_{uv} = \pi_{i_u j_v}$, i.e., the entries of π of the preference pairs, and the bs are the assignment costs without π_{ij} contributions such as $b = \sum_{u \neq i, v \neq j} k_{iu} \pi_{uv} l_{vj}$ and $b' = \sum_{u \neq i, v \neq j} k_{i'u} \pi_{uv} l_{vj}$. Clearly, we only have to focus on the largest difference $b' - b$ for each preference in $\mathcal{P}$. Let m_u be this maximum for the u-th preference and let m be the column-vector of m_{ij}s. To rebalance the costs, the idea now is to turn the inequalities into equalities and to solve the corresponding system of linear equalities,

$$\tilde{\pi} = \operatorname{inv}(A - A') \cdot m \tag{6}$$

Let π' be the resulting assignment matrix, i.e., π with the corresponding entries set to $\tilde{\pi}$. The cost matrix $c' = \hat{K} \pi' \hat{L}$ using π' balances the assignment preferences and the image similarities better than the original π.

4. SIMILARITIES FOR IMAGE SORTING

The last section has shown that kernelized sorting does not require a similarity measure *across* the sets of objects to be matched. In contrast to most existing matching algorithms, we only need to establish similarity measures *within* each set of objects independently. This is beneficial for image layouting – as we will show in the next Section – since embedding images onto a different structure now only involves changing the measure of similarity on the structures but not on the images. So, let us establish similarity measures for images and for coordinate positions in the layouting structures.

4.1 Image Similarity

Similarity of images is typically defined in terms of low-level or semantic features. We will discuss both of them in turn.

4.1.1 Color

Low level features, such as color and texture are often used to represent visual content of images. Color correlograms, color moments, and color histograms are among the most widely used color features while Gabor wavelet features [16] are one example of texture features. Image representation by combining color and texture features has also been explored [30]. Considering the trade-off between the richness of the features and the computation needed in extracting the features, we choose to represent color content of an image by its Lab color model. It is one of the most basic color features yet able to capture perceptual similarity. Given two images I_i and I_j, we use the following (kernel) measure to assess similarity between them

$$k(I_i, I_j) = \exp(-\gamma \left\| I_i^F - I_j^F \right\|_{\ell_2}^2) \tag{7}$$

where I_i^F and I_j^F denote the Lab color feature of image I_i and I_j, respectively. The kernel width γ is adjusted to the inverse median of $\left\| I_i^F - I_j^F \right\|_{\ell_2}^2$ such that the argument of the exponential is $O(1)$.

4.1.2 Semantic

The above color features describe an image in a global sense. In certain cases, we might desire to put more attention on the local semantic details of an image. Scale-invariant feature transform (SIFT) [15], and its variants and extensions such as speeded up robust features (SURF) [2] and Daisy [26], captures the appearance of semantic structures at local patches densely sampled over the image at different scales. We can then represent an image as a bag-of-visual-words [6], i.e. a histogram of vector quantized local image descriptors. A combination of densely sampled, overlapping patches with the SIFT descriptor is used in our study. Now, given two images I_i and I_j, we use the inverse of the exponentiated χ^2 distance on the bag-of-visual-words to assess the similarity between them

$$k(I_i, I_j) = \exp(-\gamma \left\| I_i^F - I_j^F \right\|_\chi^2) \tag{8}$$

where I_i^F and I_j^F denote bag-of-visual-words feature of image I_i and I_j, respectively. The kernel width γ is adjusted to the inverse median of $\left\| I_i^F - I_j^F \right\|_\chi^2$ such that the argument of the exponential is $O(1)$.

4.2 Layout Structure Similarity

We will now develop similarity measures between positions on several types of structures. Specifically, we will provide measures for non-hierarchical structures such as a 2D grid and a 3D sphere, and measures for their hierarchical extensions.

4.2.1 Non-Hierarchical Structure

Let us first focus on non-hierarchical structures, namely 2D grids and 3D sphere.

2D Grid: To produce a similarity-based image layouting on a 2D grid using kernelized sorting, we need to define a similarity measure between two points on a grid. Consider two points G_i and G_j on a grid located at abscissa x and ordinate y of (x_i, y_i) and (x_j, y_j), respectively. The straight line distance (dis-similarity measure) between those two points is expressed as

$$d_{G_i,G_j} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \tag{9}$$

Then to measure the similarity between the two points on a grid, the inverse of the exponentiated straight line distance, denoted as

$$k(G_i, G_j) = \exp(-\gamma (d_{G_i,G_j})^2) \tag{10}$$

is used. As we did for the similarity measure between images, the kernel width γ is again adjusted to the inverse median of $(d_{G_i,G_j})^2$ such that the argument of the exponential is $O(1)$.

3D Sphere: Here we need to establish a measure for (shortest) distance between two points on a surface of a sphere. On a sphere, the concept of straight lines is replaced with the concept of great circles, i.e. circles on a sphere whose centers are coincident with the center of the sphere. Let two points S_i and S_j located at latitude δ[2] and longitude λ of (δ_i, λ_i) and (δ_j, λ_j), respectively, on a unit sphere. The great circle distance (special case of geodesic distance on a sphere manifold) between these two points is expressed as

$$d_{S_i,S_j} = \cos^{-1}[\cos(\delta_i)\cos(\delta_j)\cos(\lambda_i - \lambda_j) + \sin(\delta_i)\sin(\delta_j)] \tag{11}$$

The above distance formula is simply the angular distance between two vectors on a spherical coordinate axes frame computed via the well-known law of cosine. However, this distance formula can have large rounding errors when the distance is small. Thus, we use the following Vincenty's formula which is accurate for all cases

$$\begin{aligned} d^v_{S_i,S_j} \;=\; & \tan^{-1}(x/y) \text{ with} \\ & x = \sqrt{x_1 + x_2} \\ & x_1 = (\cos(\delta_j)\sin(\Delta\lambda))^2 \\ & x_2 = (\cos(\delta_i)\sin(\delta_j) - \sin(\delta_i)\cos(\delta_j)\cos(\Delta\lambda))^2 \\ & \text{and } y = \sin(\delta_i)\sin(\delta_j) + \cos(\delta_i)\cos(\delta_j)\cos(\Delta\lambda) \end{aligned} \tag{12}$$

In these equations, $\Delta\lambda := \lambda_i - \lambda_j$. Then the inverse of the exponentiated great circle distance, denoted as

$$k(S_i, S_j) = \exp(-\gamma (d^v_{i,j})^2) \tag{13}$$

is used to measure the similarity between the two points on a sphere. Similarly, the kernel width γ is adjusted to the inverse median of $(d^v_{S_i,S_j})^2$ such that the argument of the exponential is $O(1)$.

4.2.2 Hierarchical Structure

Now, we are ready to turn towards hierarchical variants of 2D grids and 3D spheres.

2D Grid: It is quite straightforward to extend image organization on a 2D grid to a hierarchy of 2D grids. Here one additional axis can be used to specify the hierarchy level. Instead of (x, y) position, now a point is identified by its (x, y, z) coordinates in a three dimensional coordinate system. The similarity measure on the structure will then be either the similarity measure between points within the same hierarchy level or between points across different hierarchy levels (see Figure 3(a)). The z axis plays an important role on how spatial coherence in one hierarchy level is propagated to subsequent hierarchy levels. The higher its value, the more *independently* the organization of images on one level is done with respect to other levels. Equivalent to a 2D grid, the inverse of the exponentiated straight line distance is used to measure the similarity between two points on the hierarchy where the distance is now defined on the three coordinate axis.

3D Sphere: While the extension from a 2D grid to a hierarchy of 2D grids is trivial, unfortunately this is not the case for extending from a 3D sphere to a hierarchy of 3D spheres. The great circle distance is only defined to measure dis-similarity between two points on the surface of the same sphere. For the purpose of producing a hierarchical image organization on a 3D sphere, we develop the following between-sphere distance approximation trick (c.f. Figure 3(b)):

- set all the spheres in different hierarchy levels to have a unit radius while having different discretization levels
- compute the distance between two points on the same sphere using the standard great circle distance
- compute the distance between two points (i and j) on different spheres as follows

[2]The latitude δ is related to colatitude ϕ of spherical coordinate by $\delta = 90^\circ - \phi$.

– perform orthogonal projection of one point, i' such that it will now lie on the same sphere as the other point j
– compute the great circle distance between the two points, i' and j
– form a right triangle with vertices at i, i' and j
– the distance between the two points, i and j, is now the hypotenuse of the right triangle which can be computed via standard Pythagorean theorem

Finally, we again use the inverse of the exponentiated great circle distance to measure the similarity between two points on a hierarchy where the great circle distance between two points on different spheres is an approximation as described above. Similar to the z axis in the hierarchy of 2D grids, the projection length can be used to adjust how spatial coherence in one sphere is propagated to the subsequent spheres at different hierarchy levels.

5. WEB IMAGE BROWSING

Putting everything together, we can now easily realize flexible image browsing systems[3]. First, we consider the classical Google Image Search interface. Figure 1 shows the first page of Google's result for the query 'Sydney'. To guide the result, the user can specify the dominant color of the images, e.g. 'blue' or 'black'. Instead of using Google's interface – i.e. ranking the images based on relevance to the query and spreading them over several pages, we can improve the presentation of the returned images by layouting them on a 2D grid where similar images are found at proximal locations, as in Figure 2(a). Blue and black images are automatically grouped together, as are all other dominant colors present. As such, the user can immediately focus his attention (e.g. by zooming in) on what he considers to be the most relevant part of the retrieved data set, while still keeping a global overview of the entire search result.

Now suppose our user would like to find a particular image belonging to the class of, say, white images. The white images are located at the top left part of the 2D grid, and get affected by boundary effects. In contrast, by layouting the images onto a sphere (Figure 2(b)), such boundary effects can be avoided. Moreover, the user can turn the sphere so as to bring the most relevant part of the result set to the center of the display. Images with lower similarity gradually get smaller as they are 'further away' and closer to the horizon. Yet in principle the whole sphere still has to be explored or browsed. If many images are returned by the search engine, displaying all of them simultaneously becomes infeasible. The images become too small, and the user looses the global overview. The hierarchical version of the 2D grid and the 3D sphere shown in Figure 4 and Figure 5, respectively, overcome this problem as they allow the user to find quickly images of a particular type by ignoring large parts of the search space. Indeed, the user first scans the highest level only, and based on this determines a region of interest (zooms in on that part, or turns the sphere accordingly). Only then does he descend to the lower levels, where he can explore more images and finer dissimilarities. All of these different layouting results are produced using the same algorithm while changing only the structures.

Recently, Google Image Search lets you filter images by the predominant color. Using a textual interface, you can choose one of the 12 available colors and even select two predominant colors. It even has previously added filters for photos, images that contain faces, clip arts, and line drawings. However, again only a textual interface is provided. In contrast, the semi-supervised extension of kernelized sorting introduced in this paper allows the user to implement a visual interface which is more intuitive. For instance, in Figure 6, a user supplies four color content queries, i.e. blue, black, red, and white colors by providing four sample images (colors). Automatically, the images in the hierarchy are reorganized with a gradual transition between these four colors. Moreover, this approach is not limited to a limited number of predefined filters - the user is free to select any sample images he likes to guide the presentation of the search results.

6. DISCUSSION AND CONCLUSION

We have applied a recent machine learning technique, kernelized sorting, to the problem of image layouting where the dependency between a set of images and a set of positions on an embedding structure is maximized. This only requires a similarity measure within each of the two sets independently, making the approach very flexible and generic.

In the previous section, we applied this new approach to image layouting in the context of web based image browsing. We demonstrated several layouts for the query 'Sydney'. To change to a different embedding structure we only needed to provide a new similarity measure for the new structure.

Similarly, we can also easily change the similarity measure used by the layouting algorithm. This is illustrated by the following example. We collected 1000 keyframes from a 1.5 hour life broadcast of the recent summer olympic games in Beijing. As this is a life reportage, it switches back and forth between different simultaneous events. Yet our user might be interested in only one aspect (one sport, or one athlete, or the general atmosphere). Using our interface, he wants to get an overview of the broadcast as a whole or select a few interesting shots that he considers worth watching. Similarity based on color information is not likely to yield good layouting in this case as most of the images have similar colors. However, thanks to the strict separation of image and structure similarities underlying kernelized sorting it is easy to deal with this situation. We only have to replace the color based kernels with semantic (SIFT) based kernels. The similarity measure on the structures remains untouched. Figure 7 shows a layouting result, using a hierarchy of 2D grids. In this case, the user provided his own image preferences by entering four semantic content queries in the form of four sample images used at the top of the hierarchy.

In summary, we have provided a general approach of image layouting for the purpose of exploration and browsing. The benefit of kernelized sorting is that we only have to provide a similarity measure for images and structures independently. Our showcases have demonstrated the benefits of this: changing the structure similarity easily allows to layout into a wide variety of structures whereas changing the similarity measure on images allows us to adapt the layouting according to the color contents or semantic contents of the images. However we have only started to explore the

[3]A hierarchical 2D browser prototype on news images can be found at `http://homes.esat.kuleuven.be/~jhbecker/projects/CLASS/D7.3/`

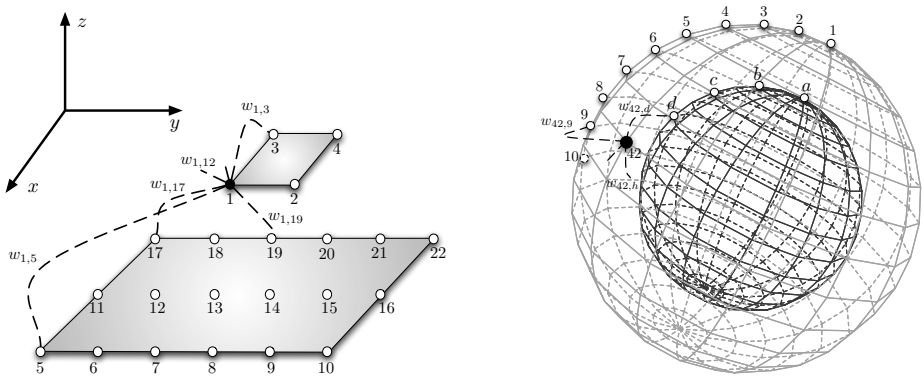


Figure 3: An illustration of similarity measure for hierarchical structures. ○ visualizes some of the positions on the structure. • denotes position under consideration and similarity measure is computed between these points and any other points on the structure. The procedure is repeated for all points on the structure to produce similarity (or dis-similarity) measure between positions on the structure required in kernelized sorting algorithm.

benefits of dependency maximization for image browsing. Today's web pages are usually media-rich, containing both images and texts. Layouting web images should be context-dependent, that is, it should take into account the textual context the image appears in. Using kernelized sorting, we can either simply combine textual and visual kernels or we can try to layout or sort both types of media jointly. That is, we layout both media into separate structures. This joint layouting will improve the layouting on both medias as information can freely flow. This is the topic of future work.

7. REFERENCES

[1] P. André, E. Cutrell, D. S. Tan, and G. Smith. Designing novel image search interfaces by understanding unique characteristics and usage. In *INTERACT*, 2009.

[2] H. Bay, T. Tuytelaars, and L. V. Gool. Surf: Speeded up robust features. In *ECCV*, 2006.

[3] B. Bederson. Photomesa: a zoomable image browser using quantum treemaps and bubblemaps. In *UIST*, pages 71–80, 2001.

[4] C. M. Bishop, M. Svensén, and C. K. I. Williams. GTM: The generative topographic mapping. *Neural Computation*, 10(1):215–234, 1998.

[5] S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. In *WWW*, 1998.

[6] G. Csurka, C. R. Dance, L. Fan, J. Willamowski, and C. Bray. Visual categorization with bags of keypoints. In *Workshop on Statistical Learning in Computer Vision*, pages 1–22, 2004.

[7] O. de Rooij, C. G. Snoek, and M. Worring. Balancing thread based navigation for targeted video search. In *CIVR*, pages 485–494, 2008.

[8] O. de Rooij, C. G. M. Snoek, and M. Worring. Query on demand video browsing. In *ACM Multimedia*, pages 811–814, 2007.

[9] G. Hinton and S. Roweis. Stochastic neighbor embedding. In *NIPS 15*, pages 833–840, 2002.

[10] D. Huynh, S. Drucker, P. Baudisch, and C. Wong. Time quilt: scaling up zoomable photo browsers for large, unstructured photo collections. In *CHI Extended Abstracts*, pages 1937–1940, 2005.

[11] H. Intraub, C. Gottesman, E. Willey, and I. Zuk. Boundary extension briefly glimpsed photographs: do common perceptual processes result in unexpected memory distortions? *Journal of Memory and Language*, 35(2):118–134, 1996.

[12] T. Jankun-Kelly and K. M. Moiregraphs. Radial focus+context visualization and interaction for graphs with visual nodes. In *Proc. IEEE Symposium on Information Visualization*, pages 59–66, 2003.

[13] T. Kohonen. Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, 43:59–69, 1982.

[14] H. Liu, X. Xie, X. Tang, Z.-W. Li, and W.-Y. Ma. Effective browsing of web image search results. In *ACM SIGMM international workshop on Multimedia information retrieval*, pages 84–90, 2004.

[15] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 60:91–110, 2004.

[16] B. Manjunath and W.-Y. Ma. Texture features for browsing and retrieval of image data. *IEEE Trans. PAMI*, 18(8):837–842, 1996.

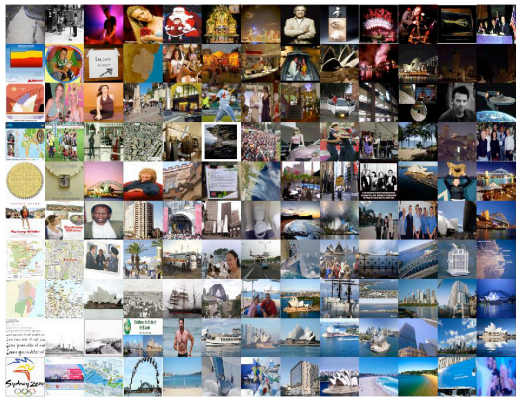
[17] G. P. Nguyen and M. Worring. Interactive access to large image collections using similarity-based visualization. *Journal of Visual Languages and Computing*, 19(2):203–224, 2008.

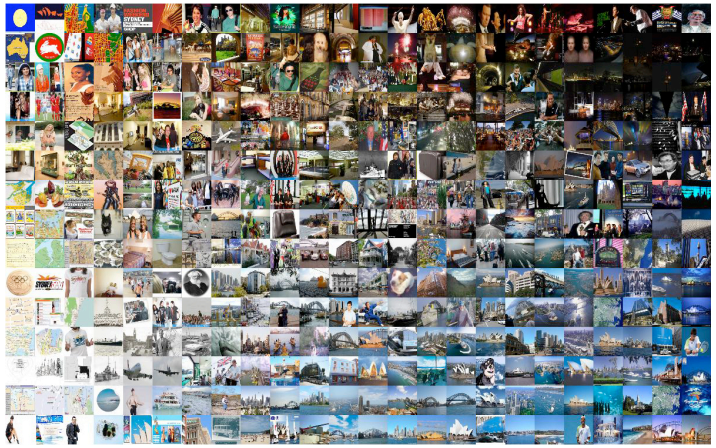
[18] N. Quadrianto, A. J. Smola, L. Song, and T. Tuytelaars. Kernelized sorting. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, (in press), 2009.

[19] K. Rodden, W. Basalaj, D. Sinclair, and K. Wood. Does organization by similarity assist image browsing? In *Proc. ACM SIGCHI Conf. on Human Factors in Computing Systems*, pages 190–197, 2001.

[20] S. Roweis and L. K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290:2323–2326, December 2000.

[21] Y. Rubner, C. Tomasi, and L. Guibas. Data





$^{\text{Top}}$Level $^{\text{Middle}}$Level $^{\text{Bottom}}$Level

Figure 4: Kernelized sorting layouting interface for 'Sydney' query. Hierarchical structure of 2D grid is used. User is given a visual summarization on the $^{\text{Top}}$Level of hierarchy and subsequently the user is able to browse by clicking on an image which will give a zoom-in to certain area of $^{\text{Middle}}$Level, and the procedure repeated for next subsequent levels.

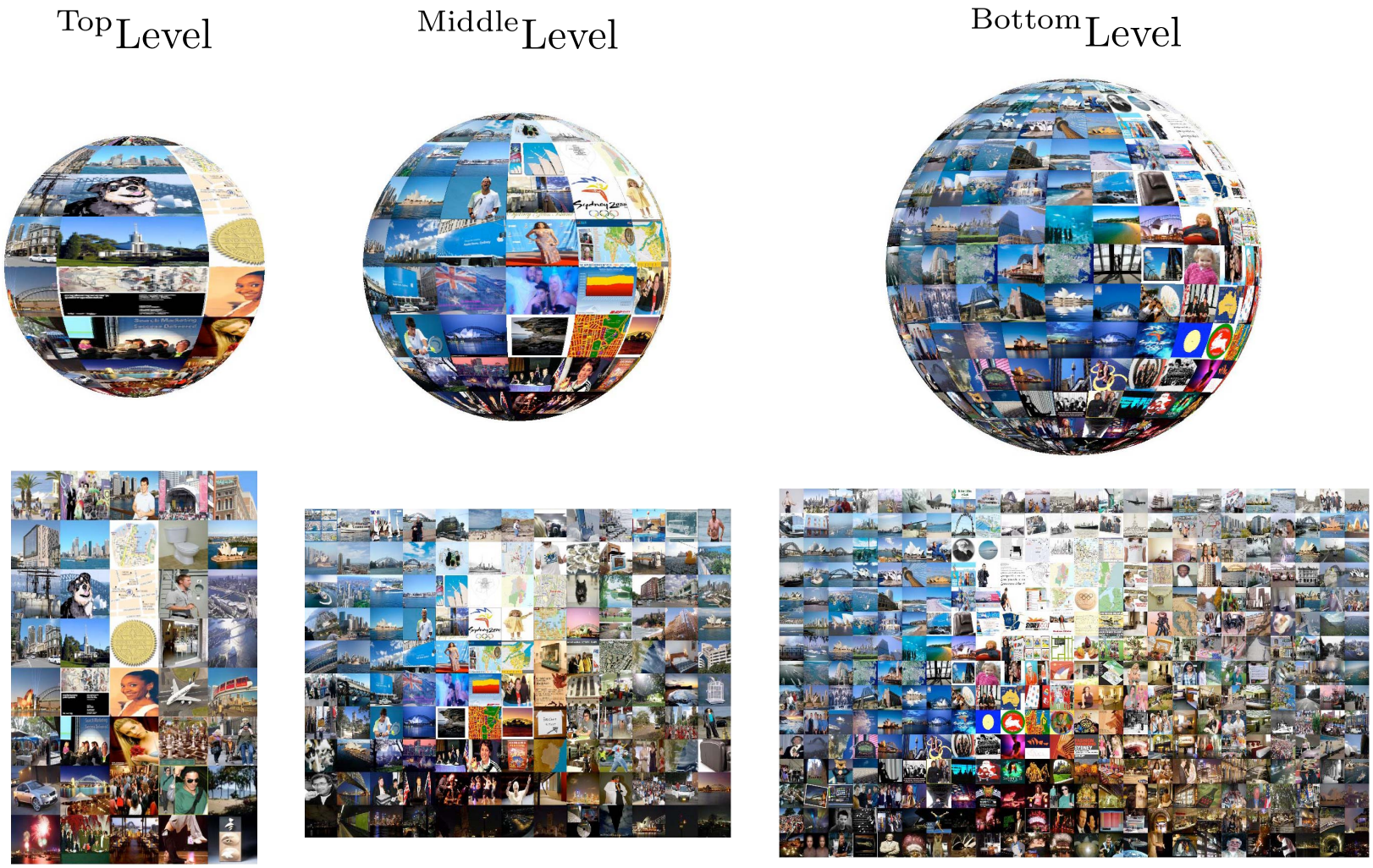


Figure 5: Kernelized sorting layouting interface for 'Sydney' query. Hierarchical structure of 3D sphere is used. User is given a visual summarization on the $^{\text{Top}}$Level of sphere hierarchy on which only a few images are shown. The user can turn the sphere to the part (class of images) he/she is interested in, subsequently he/she can proceed to the next hierarchy level by clicking on an image which will give a zoom-in, and the procedure repeated for next subsequent levels.

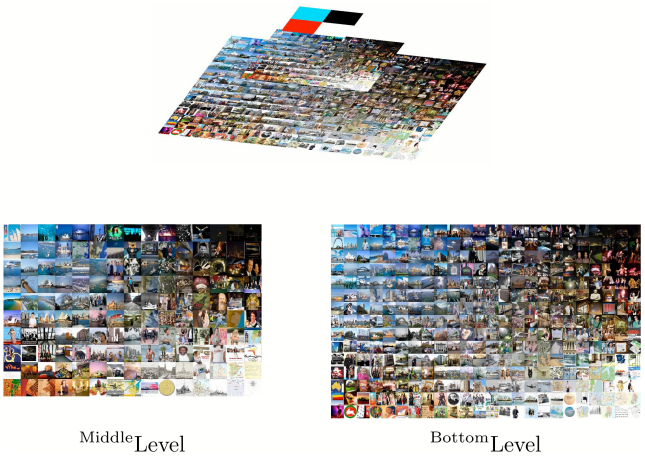


Figure 6: Kernelized sorting layouting interface for 'Sydney' query. The interface allows user to specify his/her preferences (semi supervised extension of kernelized sorting, Section 3.2). User supplies four *color* content queries at Top Level. The four queries are `blue` , `black` , `red` , and `white` images and the returned images associated with the color content of each query will be placed at top left corner, top right corner, bottom left corner, and bottom right corner, respectively.

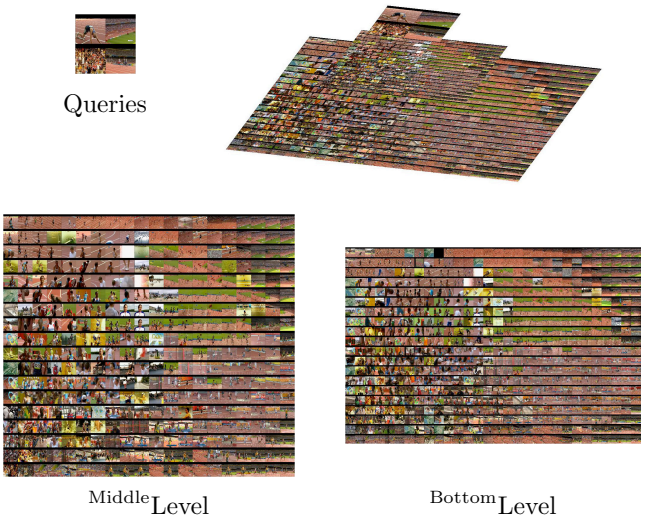


Figure 7: Kernelized sorting layouting interface for images of recent Beijing olympic games. The interface allows user to specify his/her preferences (semi supervised extension of kernelized sorting, Section 3.2). User supplies four *semantic* content queries by providing sample images at Top Level.

characterization for intelligent graphics presentation. In *ICCV*, pages 59–66, 1998.

[22] Y. Rui, T. Huang, and S. Chang. Image retrieval: current techniques, promising directions and open issues. *Journal of Visual Communication and Image Representation*, 10:39–62, 1999.

[23] H. Shen, B. Ooi, and K.-L. Tan. Giving meanings to www images. In *ACM Multimedia*, pages 39–47, 2000.

[24] A. W. Smeulders, M. Worring, S. Santini, A. Gupta, and R. Jain. Content-based image retrieval: the end of the early years. *IEEE Trans. PAMI*, 22(12):1349–1380, 2000.

[25] A. Smola, A. Gretton, L. Song, and B. Schölkopf. A hilbert space embedding for distributions. In *ALT*, LNCS. Springer, 2007.

[26] E. Tola, V. Lepetit, and P. Fua. A fast local descriptor for dense matching. In *CVPR*, pages 1–8, 2008.

[27] R. Torres, C. Silva, C. Medeiros, and H. Rocha. Visual structures for image browsing. In *Conference on Information and Knowledge Management*, pages 49–55, 2003.

[28] K. Toyama, R. Logan, and A. Roseway. Geographic location tags on digital images. In *ACM Multimedia*, pages 156–166, 2003.

[29] K. Q. Weinberger and L. K. Saul. An introduction to nonlinear dimensionality reduction by maximum variance unfolding. In *AAAI*, 2006.

[30] H. Yu, M. Li, H.-J. Zhang, and J. Feng. Color texture moments for content-based image retrieval. In *ICIP*, pages 24–28, 2002.