IDC HERZLIYA | Efi Arazi School of Computer Science

THE INTERDISCIPLINARY CENTER, HERZLIA

EFI ARAZI SCHOOL OF COMPUTER SCIENCE

PICTUREBOARD: INTERACTIVE IMAGE ARRANGEMENT USING CONTEXT-BASED SIMILARITY

M.SC. DISSERTATION

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF MASTER OF SCIENCE (M.SC.) RESEARCH TRACK IN COMPUTER SCIENCE

SUBMITTED BY TZACH YARIMI

UNDER THE SUPERVISION OF PROF. ARIEL SHAMIR, IDC

MAY 2013

I would like to send my greatest gratitudes to my advisor, Prof. Ariel Shamir from the IDC, for his innovative ideas, deep insights and many hours of helpful discussions about this paper.

Abstract

There are many possible ways to organize a collection of images on a plane. We look for arrangements where similar images are grouped based on content and not meta-data such as date or name. Such arrangements depend on a similarity measure between images which is difficult to define, but also depends on two factors: the context (the set of images) and user preferences. We present a method to arrange and browse image collections by learning the user's preferences in an interactive, fun way. Images are represented by multiple features in high dimensional space, and the distance between them is a weighted combination of the feature distances. The context determines the initial weights and is based on the relative importance of each type of feature in the given set. Using a Self-Organizing map we arrange the images in 2D, but allow the user to interact with the map by moving images to preferable locations. Our method learns the user preferences by interpreting the user's movements. The movements are translated to modification of the weights, which in turn change the map display. We show various examples and validate our approach in a user study.

CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Browsing collections of images and photographs has become a common practice. Millions of photographs are captured and uploaded to the internet every day, and many internet queries are visual in nature and return a set of images. In addition, almost everyone has a number of personal photo collections of family, friends, trips, and social events. These collections are usually arranged by the date captured, the event in which they were taken, or simply by filename. Viewing and browsing small image collections usually involves displaying the images juxtaposed on the plane. However, for such a display the meta-data of a given collection, if present, may not provide an aesthetic ordering. Our goal in this paper is to allow content-based orderings of small image collections which will be perceptually more reasonable, for better comprehension, higher effectiveness, as well as for pure aesthetic reasons. Such an ordering method can improve many daily based tasks like viewing facebook albums from single events, creating a collage of images and making an image catalogue, as well as creating more aesthetic, user-specific folder views in programs such as Shoebox presented in [31]. It can also be used to learn the metrics used for arranging a set of images from a specific set or category. Such metrics can be used to arrange similar collections in the future.

Any ordering demands a similarity (or distance) measure. However, there is no single computational measure that encapsulates perceptual image similarities. Many different measures based on color, texture, shapes and more, have been used in various applications and in different scenarios. The effectiveness of any image distance measure depends on the *context*, i.e. the actual set of images to be compared. Color differences may be significant in one set while texture in another. In this paper we define an adaptive distance measure that conforms to the context of the set by combining several distance measures together. Using this context-dependent measure we are able to present better orderings of image collections.

Still, there are times when the user may want to order the set of images in a different manner than the one suggested automatically. Creating such arrangements manually is simply done by moving similar images closer together. However, we would like to alleviate the tedious task of arranging all images manually. By moving some images closer together the user is implying that specific image features are more important than others when arranging this specific set. We present a technique to implicitly learn the user's intensions and apply them to automatically rearrange the whole image set, based only on the simple gesture of image movement that is natural for this task.

Our key idea is to use a weighted average of multiple image distance measures based on various features and to learn their relative importance for a given image collection first from the *context* – the set of images, and then from the user's *interactive gestures*. Given a set of images, we represent each one as a vector in a number of high-dimensional feature spaces. Each feature defines a sub-space and is given a weight. The distance between two images is then defined as the weighted sum of distances in all sub-spaces. Our primary goal is to learn this set of weights for all feature spaces.

Arranging a set of high-dimensional descriptors requires mapping these descriptors to a 2-dimensional space for visualization. We use a slightly modified version of the Self Organizing Map algorithm [13], where each cell contains one vector of each feature space. The distance between cells is calculated based on the weights of the feature spaces.

To account for the context of the image collection, the initial set of weights is learned from the variability of each feature in the input set – higher variability of a given feature in the collection of images results in a higher weight for this feature. Many times, the initial arrangement created using these context-dependent weights already represents the ordering of the collection well. However, our interface allows the user to modify the arrangement simply by dragging a "misplaced" image to a better position. This correction gesture implicitly triggers a change in the relative weights, which in turn affects the arrangement. The user can iterate through several such correction steps until he or she is satisfied with the arrangement.

When rearranging some images, the user may have some high level semantics in mind. Such semantics are difficult to capture using any single image feature but may be represented by a weighted combination of low level features. Our technique can be seen as a tool to search for such weights.

Providing an interactive interface for 2D arrangement of high dimensional feature vectors demands an efficient method to project and order the items. In exploratory data analysis, *ordination* is a technique that orders multi-variate objects so that similar ones are near each other and dissimilar ones are farther away. Several methods have been introduced for this purpose, such as multi-dimensional scaling (MDS) and principal component analysis (PCA) [26], but these are less suitable for interactivity. We base our interface on the well known ordination technique of Self Organizing Maps (SOM) [13]. SOM allows efficient projection and arrangement of high dimensional feature spaces to 2D (or other spaces). However, most SOM variations do not allow direct interaction or feedback. We present a method that modifies the original SOM algorithm to interactively learn the user's preferences and allows inclusion of manual corrections while updating the map. Every update, the map is created based on the high-dimensional weighted distance measure using the current weights, while the weights are modified based on the user interactions (see Figure 3.1).

We present results on various types of image collections. We concentrate on small to medium size collections as it is difficult to interact with more than a few hundred images. We conduct a set of user studies to validate some of our assumptions and illustrate the effectiveness of our method. Lastly, we use our method as a tool to explore the differences in the manner humans perceive the order in the set by examining the level of consistency of the different weights.

CHAPTER 2

PRIOR WORK

2.1 Image Visualization and Arrangement

Not many works deal directly with image arrangements. Most related works deal with content based image visualization (CBIV) and some with content-based image retrieval (CBIR) from a large database [5]. The key difference between the two problems is that CBIR deals with a relatively large image database to find a small subset of images that are similar to an input image query, while in CBIV the image collection is usually smaller and all images are displayed. Moreover, CBIR deals more with *which* results are returned, while CBIV deals more in *how* they are displayed.

Some works have tested the usage of such arrangements for simplification of daily based tasks. Rodden et al. [30] have tested the contribution of similarity based arrangements for completing the task of selecting 3 images out of 100 for a magazine article. Although the final conclusion is that similarity based arrangements did not indeed help speeding the choosing process, the presence of multiple arrangements of the same collection did improve the results. In another work, Liu et al. [23] have found that similarity based arrangements are better for image search than rank based arrangements, similar to those in current major image search engines.

Several methods have been suggested for visualizing a group of images in two, or three, dimensional space. Manovich [25] in his software ImagePlot, represents each image as a point in a 2D Cartesian system, where each axis corresponds to a one-dimensional feature space such as brightness, saturation or hue median or standard deviation. Deng et al. [6] and Barthel [1] both use SOMs for dimensionality reduction, with separate similarity metrics based on a single low-level feature space. Although these methods work well for some image collections, using a single low-level representation of images cannot represent differences in different sets of images and differences in the user preferences. Moreover, such techniques are not context-dependent: they will not work well if the given collection contains many images that are similar in terms of the single low-level feature used.

Two approaches try to overcome these problems by better representing the input images. The first approach retains the usage of a single feature space, albeit more comprehensively by extending existing high-level features or features that are specific to a certain domain. For example, Ladikos et al. [21] measure the amount of overlap between regions in image pairs in the collection and use this as a distance metric. Shrivastava et al. [34] extend the use of SIFT features by deciding which parts of the image are more important to the human observer. Rodden [29] compares arrangements based on features with increasing complexity for image search. These approaches usually work well only for specific types of image collections like images of the same building, but may have difficulties in small and diverse image collections. The second approach combines multiple low-level features. Chen et al. [3] use a predefined combination of colors, texture and shape histograms as a distance metric for a Pathfinder network, but they do not allow for user feedback. Iqbal and Aggarwal [11] combine multiple feature spaces by setting a weight for each one. However the weights are set manually by the user, a task that is hard to perform, and almost impossible when a large number of feature spaces are involved.

For CBIR, Koskela et al. [19, 20, 17] combine the results of multiple parallel SOMs, each based on a different feature space, to organize image results from large un-annotated databases. They use a relevance feedback approach to determine which areas on each SOM better reflect the user's intentions, and refine the retrieved image set by picking images from these areas for the next query iteration. In our work we combine the two approaches by using both low-level features like colors, textures, and shape, together with high-level features like SIFT [24] and SURF [2]. We also present a novel relevance-feedback mechanism to recalculate the weights of the different feature spaces based on the user's image movements.

In both approaches, a relevance feedback mechanism can be used to refine the results based on the user's feedback. Such works include [35], [10].

2.2 Feature Spaces

Much work has been done in the field of representing an image as a vector in high-dimensional feature spaces. A good survey can be found in [27]. In out work we use several feature spaces including color histograms, tiny images [36], Histogram of Oriented Gradients (HOG) [4], Textons [12], SIFT [24] and SURF [2]. A more detailed description of these works and how they are used in our work can be found in Section 3.1.

2.3 SOM

Arranging a set of high-dimensional descriptors requires mapping these descriptors to a 2-dimensional space for visualization. Many algorithms deal with this problem, including MDS, LLE [32] and Self Organizing Map. An overview can be found in [8]. We chose to use SOM because of its important features: (a) SOM maps the results to a given map with a predefined size, rather than just outputting coordinates. This helps mapping the results directly to the screen; and (b) When given a map of a certain size, the algorithm tends to use the entire map, rather than create huge gaps between the populated map cells. MDS, for example, might output half of the training set grouped in one corner, and the rest of the training set in the grouped opposite corner.

We use Self-Organizing Maps (SOM) which were first introduced in [13]. SOM was used to organize documents [14], but was also harnessed for image visualization purposes [19, 6, 1]. The SOM algorithm is very simple and goes as follows: (a) initialize an empty map; (b) randomly choose a training sample and find its best matching unit (BMU) from the map (BMU); (c) train the BMU and its neighbors to be closer to the training sample; and (d) repeat (b) until converging. Due to the high computational cost of the SOM, some improvements to the original algorithm were introduced, including Tree-Structured SOM (TS-SOM) [15, 16] and Growing-Hierarchical SOM [7]. We use a slightly modified version of the original SOM (see Algorithm 1), where each node contains a vector that combines all of the feature spaces together. The fully detailed algorithm and SOM training rule is on Chapter 3. Due to the smaller scale of image collections that can be arranged interactively, the computational cost was not large enough to justify the quality degradation implied by using hierarchical structures, and we could still achieve interactive rates with a single level SOM.

Chapter 3

Algorithm

3.1 Image Representation and Distance Metrics

Given a collection of images such as a personal photo collection, we start by calculating for every input image a set of descriptors in various feature spaces (see Section 3.1). We deliberately chose a variety of features including color, local and global structure, texture, image complexity and higher level interest points (see section 3.1). This is done not only to support diverse types of image collections but also in hope that some high level perceptual features that are difficult to capture explicitly may be captured by some combination of low level ones. Our scheme is general enough so that other feature spaces could be added as descriptors if they can support an additive '+' operation. This includes any numerical descriptors, histograms of various types etc. We also show how discrete point features such as SIFT and SURF could be utilized in our method.

Each image is therefore represented by a high dimensional vector $F \in \mathbb{R}^N$ composed of a set of feature descriptor vectors $F = (f_1, f_2, \ldots, f_k)$, where each descriptor is a vector $f_i \in \mathbb{R}^{n_i}, N = \sum_i n_i$. We also assign a weight $w_i \in [0,1]$ to each feature sub-space, where $\sum_i w_i = 1$. In each feature sub-space we define a distance measure d_i, that measures the distance between any two sub-vectors f_i^0, f_i^1 in this sub-space. We define the distance between two image vector representations F_0, F_1 as the weighted sum of all feature distances in each sub-space:

$$d(F_0, F_1) = \sum_i w_i \cdot d_i(f_i^0, f_i^1) \tag{3.1}$$

3.2 SOM Training

We define a rectangular 2D SOM whose number of units is around 50% larger than the number of images. Each unit is represented by a vector $R \in \mathbb{R}^N$ initialized to a random samples of each feature space $R = (r_1, r_2, \ldots, r_k), r_i \in \mathbb{R}^{n_i}$. The SOM training algorithm iterates stochastically through images from the collection and for each image representation vector F performs two operations:

1. Find the best matching unit (BMU) in the SOM for the image. The BMU is defined as the unit whose vector

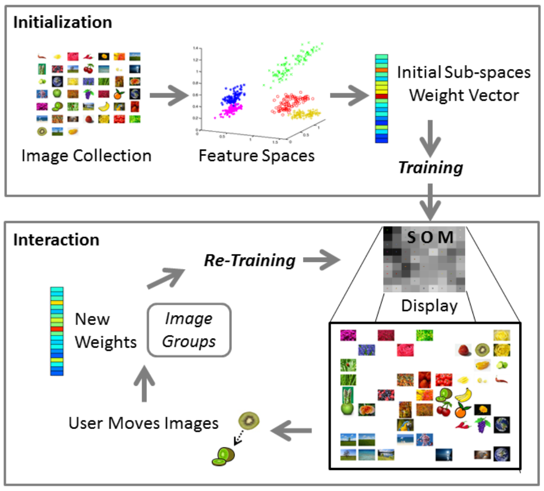


Fig. 3.1: Overview of the image collection arrangement system.

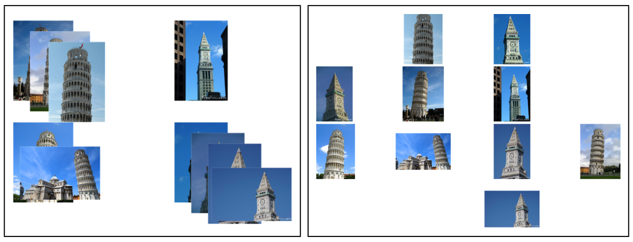


Fig. 3.2: Two possible mappings of images to screen cells: multiple images in one cell using the Hungarian algorithm, and one image per cell using the Naive method.

R is closest to F using the weighted sum of distances defined by Eq. 3.1.

2. Train the BMU and its neighborhood (of a certain area) by modifying their representative vector values to be closer to F. The training "strength" diminishes based on two factors: the 2D Euclidean distance of the trained unit from the BMU, and the learning rate.

This process continues until convergence, i.e. until the map does not change. This can be achieved by reducing the learning rate according to a desired number of iterations (we use $40n$ iterations when n is the number of images). Note that we use a somewhat different terminology than in most SOM work, to distinguish the weights in the weighted sum of Eq. 3.1. In these works the 'weights' in a SOM algorithm are what we call 'representative vectors' of the units.

3.2.1 Initialization

When the image collection is first loaded, the weights $\{w_i\}$ are initialized according to the variability of each feature in the given image collection (section 5.1). Later, these will be changed according to user interactions. This set of weights is used to train the initial SOM (Figure 3.1).

3.2.2 Mapping from SOM to the Screen

When training is done, we use the SOM to display the images on the 2D plane. The result of the training is a grid of SOM units, each contains one vector of each feature space. A method is needed in order to transform this grid into an arrangement of the input images on the screen. We first divide the screen into cells with the same number as SOM units, so each SOM unit corresponds to a single screen cell. We then map each image to the SOM cell based on its BMU, and display the image on the corresponding screen cell. We have studied three methods for mapping the images to the SOM cells:

The Naive Method This is the simplest method, where each image is mapped to its BMU in the SOM. In case when more than one image falls into the same cell, we present the images in cascading order in the cell, allowing the user to scroll between them using the mouse wheel (See the left image of Figure 3.2). While the arrangement created using this method best represents the SOM, it also allows screen cells to contain more than one image which makes the images overlap. It has been shown that people tend to dislike this overlapping, and prefer more regular, grid based arrangements [28]. To create a better looking arrangement we would like to avoid multiple images per screen cell. To accomplish this, we present two more mapping methods.

The Greedy Method In this method each image is mapped into its free BMU. This means that if more than one have the same BMU, the first image will be displayed in the cell corresponding to this BMU, while the other images will be displayed in other cells. This method is easy to understand and doesn't add any computational cost, however in a SOM with many images that fall into the same cells, it creates artefacts in the arrangement. For example two close images may be mapped into different areas of the screen.

The Hungarian Method To minimize the artefacts created by the Greedy Method, we use the Hungarian algorithm [18] to map images in the SOM units to screen cells. The Hungarian algorithm is a combinatorial optimization algorithm that solves the assignment problem in polynomial time. The input to the algorithm is a bipartite graph with weights on the edges, and the output is a perfect matching. In our case, one side of the graph represents the training samples, and the other side represents the map cells. The weights of the edges are the combined feature spaces distances between every training sample to each of the map cells of the trained SOM. The result is a mapping that minimizes this overall distance. This method gives better results than the Greedy Method while adding a computational cost. In small sets (a few hundred images), this cost is neglectable (see Figure 3.2).

3.2.3 Interaction

During interaction, we allow the user to freely move each of the images to any position on the map, bringing similar images closer to each other. After an image is moved we use information implied by the move to determine which feature spaces are more important to the user implicitly. We assume that when a user places an image next to other images, then they should be perceptually similar. The relations between the distances of these images in the various feature spaces tells us how important each feature space is. This information is used to recalculate a new set of weights that reflect this importance (Section 5.2). These weights, in turn, are used to re-train the SOM and reposition the images. The process of image repositioning and SOM retraining can be iterated several times.

Because SOM is based on random placement for initialization, retraining the map from scratch after the weights change can result in a completely different arrangement of the images. Even if the new arrangement better represents the user's perceptual intentions, the user might feel confused if the arrangement is too different from the previous one. We apply several techniques to overcome this difficulty and prevent large changes between successive SOM retrainings, while still allowing enough change to improve the arrangement (Section 5.3).

3.2.4 Fine Tuning

After the user is satisfied with the arrangement, we allow fine-tuning of the map where the user can move images without retraining the SOM. In order for the fine-tuned arrangement to be comparable to the map before the fine tuning, we allow only one image per screen cell, as in the learning mode. If the user places an image in a cell that already contains another image, the moved image will replace the old one, and the rest of the images will be moved to make room for the moved one. We decide how to move the images by locating the nearest free cell, computing the shortest path from it to the cell with the moved image, and moving each image one cell towards the free cell along this path.

3.3 Complete Algorithm

Our final algorithm for image arrangement can be seen in Algorithm 1. In the first step we transform each image to a vector in each of the feature spaces (section 3.1), calculate an initial set of weights (section 5.1) and train an initial SOM based on these vectors. We then map the SOM to the screen and display the SOM (section 3.2.2). We now allow the user to move one or more images freely on the map to correct the arrangement according to her preferences. After each move we update the proximity groups (section 5.3), calculate a new set of weights (section 5.2) and retrain the SOM using the new weights.

Algorithm 1 Interactive image arrangement algorithm.

```
Input: a collection of images
  – Initialization: –
  Calculate feature vectors for all images
  Calculate initial weights $\vec{w}$ based on Eq. 5.1
  Train initial SOM and display images
  – Interaction: –
  while user not satisfied do
    User move image(s) to a new position(s)
    Find the M closest neighbors on the 2D map
    Update proximity groups
    Calculate new weights $\vec{w}^{\text{new}}$ based on Eq. 5.2
    Pin the moved images by training their units
    – SOM retraining: –
    $t = 0$
    while SOM does not converge do
      $\vec{w} = \beta \vec{w}^{\text{new}} + (1 - \beta)\vec{w}^{\text{old}}$
      Get random training image $I$ whose feature vector is $F$
      Find the BMU of $F$ based on $\vec{w}$ and $d^*(\cdot, \cdot)$ in Eq. 5.4
      Retrain BMU's neighborhood using $F$ and Eq. 4.1
      $t = t + 1$
      Update $\eta, \beta, \theta$ according to $t$
    end while
  end while
```

Chapter 4

Feature Spaces

4.1 Choosing the Feature Spaces

To train a SOM, we need to represent each image as a fixed dimensionality feature vector, provide a distance measure between such vectors, and a method to move vectors closer to each other for training the SOM. Because the overall distance measure is defined as a weighted sum of feature distances (Eq. 3.1), each feature space must have *a*) a distance measure between two descriptor vectors; and *b*) a method to train a SOM unit to bring it closer to the training image. In addition, for the weights to truly represent the importance of each feature space, we add another requirement *c*) a feature distance measure must be normalized to the range $[0,1]$.

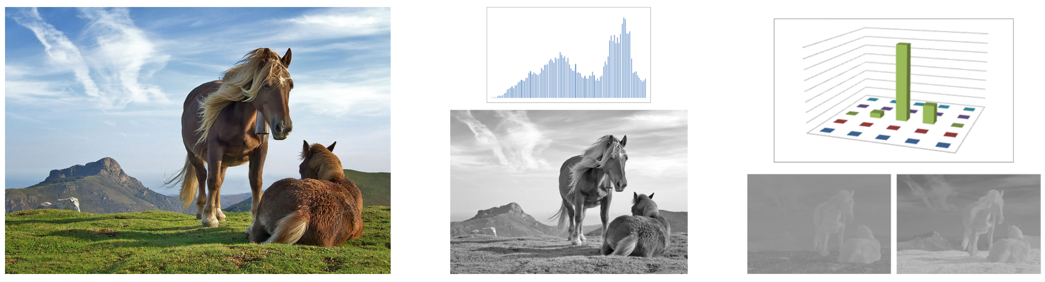
Fixed-dimension descriptors, especially histograms, are a natural choice that follow these requirements, but we show how other types of features (e.g. SIFT and SURF) could also be used for training a SOM. We aim to use feature spaces that are as diverse and independent as possible, so that each space represents a different perceptual aspect, and together they may represent more than the sum of their parts.

4.2 The Feature Spaces Used

In this section we present the feature spaces and distance measures used in our implementation, although other features could be used as well. Before representing the input images as vectors in each one of the feature spaces, we first resize each image to a maximum of 300×300 pixels by scaling, while retaining the original aspect ratio. The resizing is done for performance reasons.

4.2.1 Color Features

We use CIE-Lab color space luminance and chrominance histograms. The luminance histogram is one-dimensional and consists of 100 bins which is the full range of the luminance channel. The chrominance histogram is two-dimensional and consists of 5×5 bins for performance reasons. There exist many ways of measuring the distance between two histograms. We use the Earth Mover's Distance (EMD) [33] to calculate dissimilarity between the one-dimension luminance histograms, due to its extensive prior use for image color histograms comparison. We use the 2D EMD-L1 [**?**], which is a more efficient implementation of the EMD, as a distance measure between the color histograms.



(a) (b) (c)

Fig. 4.1: Color feature spaces: a) original image scaled to fit 300×300 pixels b) luminance channel and its histogram, c) A and B channels and their 2D histogram

4.2.2 Global Structure

In [36], a 16×16 Tiny images were used to represent the global structure of bigger images for a fast database search. We use the same idea, but instead of color images we use grayscale 16×16 images, to make this representation as independent as possible from the color histograms. The Tiny images are first normalized to have zero mean and a unit norm. We chose a simple Euclidean distance for efficiency reasons, although the 2D EMD and EMD-L1 gives better results for this type of descriptor.

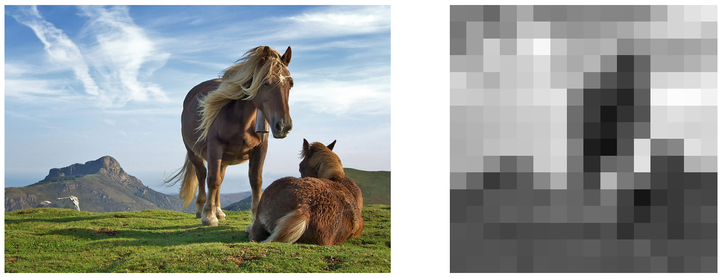

(a) (b)

Fig. 4.2: Global structure: a) original image scaled to fit 300×300 pixels, and b) 16×16 tiny image.

4.2.3 Local Structure

In [4] the histogram of oriented gradients (HOG) descriptor was used to detect humans in images. We use a similar descriptor for each input image to define local structure. We compute the gradient of each pixel of the image in the R, G and B channels using the [-1, 0, 1] gradient filter, and then select the orientation of the channel with the largest norm to be the orientation of the pixel. The orientation is an angle between 0° and 360°. We then split the image into 64 equal-sized zones and create an 18-bin histogram of the pixels' orientations (HOG) for each zone. The resulting descriptor has $64 \times 18 = 1152$ dimensions. The distance metric we use is the average of the EMDs between corresponding zones in the two images.

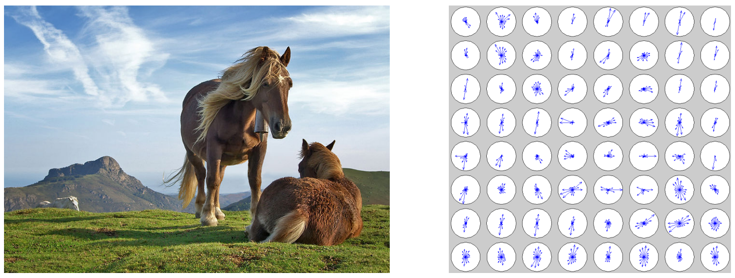

(a) (b)

Fig. 4.3: Local structure: a) original image scaled to fit 300×300 pixels, and b) Histogram of Oriented Gradients (HOG) histogram.

4.2.4 Texture

Textons [12] are a useful tool in texture analysis. In [22] multiple-texton histogram (MTH) of an RGB image were used for image retrieval. We use the following modified version of the MTH: each image is quantized into 6 levels of gray and divided into 144 equal-sized zones. We then create a 4 bin histogram for each zone, where each bin's value represents the number of occurrences of each texton in the zone. We use the 4 textons proposed in [22]. The resulting descriptor has $144 \times 4 = 576$ dimensions per image. The distance metric is the same as in HOG.

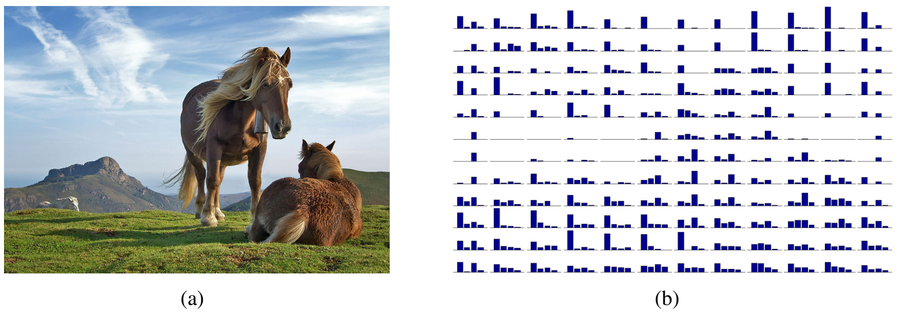


Fig. 4.4: Local structure: a) original image scaled to fit 300 × 300 pixels, and b) Textons histogram .

4.2.5 Image Complexity

We segment the image into regions following the algorithm proposed in [9] with $k = 250$ and $\sigma = 0.8$, and use the number of segments as a measure of the image complexity. The specific segmentation algorithm is not important for the results, only the fact that it is run on images of the same size, with the same parameters. Although the number of segments is a one-dimensional descriptor with a simple difference, it can still give a good indication regarding the amount of details in an image, and can be used as a feature space. A possible improvement of this feature could be dividing the image into zones and segmenting each zone individually. This will form a higher-dimension feature with the number of dimensions equals to the number of zones.

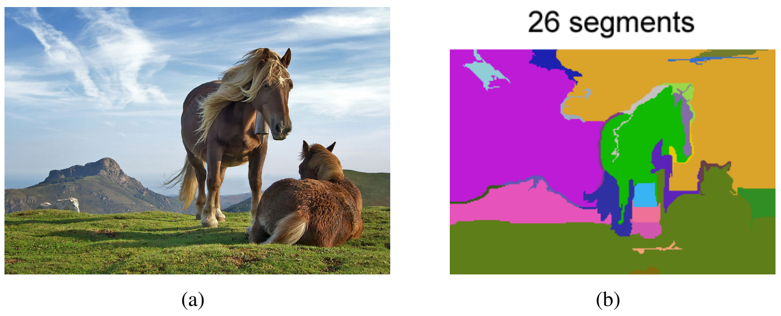


Fig. 4.5: Local structure: a) original image scaled to fit 300 × 300 pixels, and b) segmented image and number of segments.

4.2.6 High-Level Features

All of the feature spaces introduced so far can be seen as low-level descriptors of images. In order to capture some higher-level features of the images, we use two well known high-level local detectors: SIFT [24] and SURF [2]. We extract up to 1000 SIFT and SURF points from every image. However, each image can have a different number of descriptors matching different images, and it is not clear how to define such descriptors for a SOM unit. To fit our SOM framework, we must define a distance measure, as well as a way to train the SOM unit descriptor to move closer to a given training image.

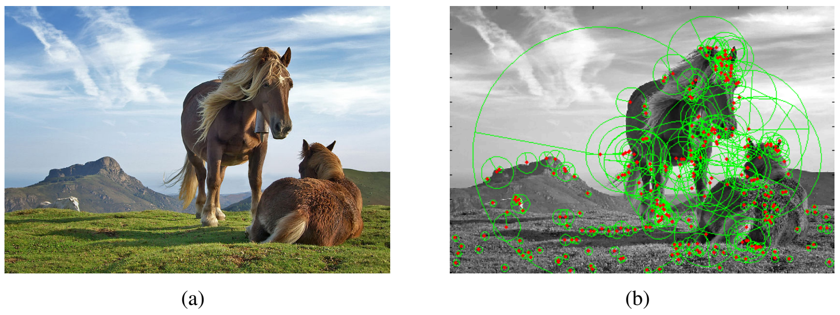


Fig. 4.6: Local structure: a) original image scaled to fit 300 × 300 pixels, and b) SIFT points of interest.

In our implementation, every SOM cell is represented by a constant number ($C = 100$) of SIFT and SURF points. The distance metric we use is $1-(m/C)$ where m is the number of matching points between the image and unit. However, we find that managing, copying and calculating distances of tens of thousands of points (of all images in the collection) is computationally expensive. Instead of using actual feature points, we calculate the descriptor feature points of all images in the collection in a pre-processing stage. Then, we build a library of feature points of all the images while discarding points that have no match. Instead of storing the points in the feature vector, we use the index of the points in the library (in fact, if no images are added to the collection after this pre-processing, the library itself can be discarded). This way, matching feature points in different images will have the same index, and the distance comparison simply becomes an index comparison in $O(n)$.

To find matching SIFT and SURF descriptors between two images, we use a method offered in SIFT [24]. For each descriptor in the first image we find the first and second closest descriptors in the second image using the Euclidean distance metric, and calculate the ratio between this two distances. If the ratio is smaller than 0.8 for SIFT or 0.7 for SURF we say that the two descriptors match, and give them the same index.

4.3 SOM Unit Training

Once the BMU of a training image is found based on Eq. 3.1, its neighborhood must be updated. Assume the BMU is at position (bx,by) on the SOM, $U(x,y)$ is a unit in its neighborhood with feature vector $R=(r_1,r_2,\dots,r_k)$, and the training image features vector is $F=(f_1,f_2,\dots,f_k)$. For quantitative feature sub-spaces, i.e. sub-spaces that support an addition '+' operation, updating the feature vector r_i in R is defined as:

$$r_i(t+1)=r_i(t)+\eta(t)\cdot\theta(x,y,t)\cdot d_i(f_i,r_i)\cdot(f_i-r_i(t)) \tag{4.1}$$

$\eta(t)=(1-t/T)\cdot\alpha$ is the learning rate that diminishes with t, where t is the iteration number, T is the total number of iterations desired, and α is a learning-rate constant. The function $\theta(x,y,t)$ diminishes with the 2D distance from the BMU. We use a uniform Gaussian with a radius that also depends on t (see also Section 5.3).

For some feature subspaces such as SIFT and SURF, the unit training must be defined differently as there is no meaning to the numerical value of the feature descriptor other than an index. For SIFT and SURF, we train a unit by randomly copying points (indexes) from the training image vector to the unit vector. Copying more points to the unit results in it being more similar to the training sample. Therefore, we can gradually reduce the number of points copied as we get farther from the BMU to its neighbors, and as the number of iteration increases during the SOM training updates.

CHAPTER 5

LEARNING

5.1 Context Dependent Weights

Analyzing different image collections and users' arrangement, we found that a specific feature space may be crucial in arranging one collection while it may hold very little meaningful information for another. Although our interface can be used to implicitly modify the weights of the feature spaces in a collection, it would be better to find which features are more important a-priori and increase their weights instead of using uniform weights for all feature spaces (see first study in Section 6.1).

We find a strong correlation between the variability of the values of a feature in a collection to its importance. More specifically, the less the values of a specific distance metric in an image collection are dispersed, the less useful the corresponding feature space is for arranging the images. For example, if all images are red in a given set, then color would not be a good feature to arrange this set. Based on this observation, we initialize the set of weights as follows:

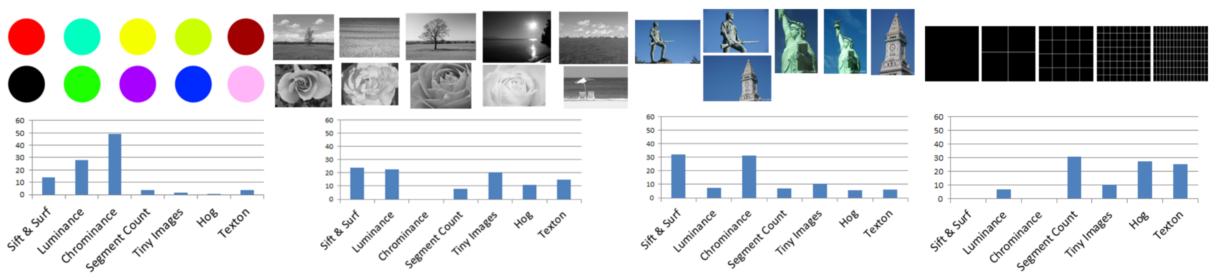
$$w_i = \frac{\sigma_{f_i}}{\sum_j \sigma_{f_j}} \tag{5.1}$$

where σ_{f_i} is the standard deviation of the distances between all images in the given collection measured in the feature sub-space f_i. This results in feature spaces with more dispersed distances having higher initial weights. Figure 5.1 shows some examples for initial weights calculation.

5.2 Interactive Weights Update

Different users may have different perspectives on a given image collection. For this reason we allow users to reposition images on the map. However, we then use these movements to learn the user's intent by modifying the weights of the feature sub-spaces based on the move, and retrain the SOM accordingly.

After an image is moved, we find the set of M closest neighbors to its new position on the 2D map using Euclidean distance. The closer the neighbor is to the moved image, the higher its contribution to the final weights



(a) (b) (c) (d)

Fig. 5.1: Initial weights calculation example: (a) An artificial set of the same shape in different colors results in high weights for color feature spaces, and low weights for texture and direction feature spaces. (b) A grayscale set results in zero weight for chrominance feature space. (c) A set with distinctive similar objects results in high SIFT and SURF weight. (d) An artificial set of a grid with a different number of cells results in higher weights for number of segments, orientation and texture feature spaces, and low weights for color and high-level feature spaces.

calculation. For each feature sub-space descriptor f_i the new weight is calculated as follows:

$$w_i = \sum_{j=1}^{M} (\frac{1}{E_j} \cdot S_i(d_i(f_i^0, f_i^j))) \tag{5.2}$$

Where E_j and $d_i(\cdot,\cdot)$ are the normalized Euclidean distance and normalized feature space distance from the moved image to the j'th closest neighbor respectively, and S_i is a sensitivity function that indicates how relevant is the specific feature space for the move in the given image collection.

We use S_i to make sure the new weights are appropriate for the specific image collection. For instance, assume again that all images in the collection are red, and the user moves an image to a new position. Even though the distance between the image and its M neighbors in chromaticity sub-space is small, we would not like to increase the weights of this feature because of the *context* of the image collection. We therefore defined the sensitivity function S_i as follows:

$$S_i(x) = \frac{|\mu_{f_i} - x|}{\sigma_{f_i}} \tag{5.3}$$

where μ_{f_i} and σ_{f_i} are the mean and standard deviation of the distance in feature space f_i between the moved image to all the other images in the set, respectively.

After calculating the new set of weights, they are normalized again to have the sum of 1. We allow for multiple consecutive image movements before recalculating the weights and rearranging the map. If more than one image movement occurred, new weights are calculated for each movement and the final weights are simply the normalized sum of all these weights.

5.3 SOM Consistency

After calculating the new set of weights, we would like to retrain the SOM. However, simply executing the SOM algorithm starting from random assignments to the units, will most probably result in a significantly different map. Although such map will better represent the user's intentions, the large differences can create confusion. In addition, when the user places an image next to another explicitly, she expects that they will remain close together in any future arrangement. However, there is no guarantee this will happen with simple SOM training. We would like the user to preserve her *mental map* of the 2D display as much as possible. On the other hand, making too few changes to the map will not reflect the insights learned from the user's interaction. To solve these issues we apply several techniques.

Map Consistency The SOM algorithm is based on a random selection of training images. This means two maps created with the same training set may look completely different. To preserve the original order as much as possible, while re-training we do not initialize the SOM to random units but use the previous map units instead.

Gradual Weight Change SOM training involves a large number of iterations. In each iteration distance calculations are used both to find the BMU and to train the map. Instead of immediately replacing the old weights with the new ones, we gradually apply them throughout the iterations by using a linear combination of the old and new weights $w_i = \beta w_i^{\text{new}} + (1-\beta) w_i^{\text{old}}$, where $0 \le \beta(t) \le 1$ grows with the iteration t.

Influence Radius Reduction The SOM training depends on an influence radius around the BMU governed by the function $\theta(x,y,t)$ from Eq. 4.1. The smaller this radius is, the less change will be made to the SOM. In each iteration of the SOM this radius reduces, making the changes to the map more local. In the initial arrangement we start with a Gaussian whose radius covers the whole map but for any successive retraining of the SOM we start with a Gaussian whose radius covers only half of the map.

Image Pinning Images that the user moved are treated somewhat differently than the other images. Moved images are expected not only to remain close to their neighbors, but also to remain in their position. To promote such behavior, we manually train the SOM unit corresponding to the moved image's new position with the image feature vector before beginning the full SOM re-training. This will increase the chance of this image's BMU to either be this position unit or a close-by unit.

Proximity Groups Our weight learning is a good technique for expressing the user's preferences globally. However, there are times when a user brings images close together but there is no weight combination that will provide a small feature distances between them. This can happen when high level semantics are used to arrange images, or simply when the user is inconsistent. In this situation, when retraining the SOM these images will most likely be driven far apart.

To solve this, we define a ‘proximity group’ for each image that can contain $g_{max} = 5$ images. When a user moves an image, we add the $g_{min} = 3$ closest neighbors in 2D into the image’s proximity group, with proximity-weights λ, that are relative to their distance from the image in the 2D space, and are normalized $\sum \lambda = 1$. If there are already g_{max} images in the group, then images with the lowest proximity-weights are replaced. We also symmetrically add the moved-image into the proximity group of each of the closest images.

Now, when searching for the BMU of an image during SOM retraining, we consider not only the distance from the image feature vector F to every unit vector R using Eq. 3.1, but also use the weighted average of all the distances from all the image’s proximity group to the unit as follows:

$$d_{\text{group}}(F,R) = \gamma \cdot d(F,R) + (1-\gamma)(\sum_{g \in G} \lambda_g \cdot d(F_g,R))$$
$$d^*(F,R) = \begin{cases} d_{\text{group}}(F,R), & \text{image with proximity group} \\ d(F,R), & \text{else} \end{cases} \tag{5.4}$$

where G is the proximity group of the image, $d(\cdot,\cdot)$ is the weighed distance of Eq. 3.1, λ_g is the normalized proximity-weight of image g in the group, and $\gamma = 0.7$ is a group weight constant. Using this method users are able to directly express local similarities between images, and keep similar images closer even after several iterations of SOM retraining.

CHAPTER 6

RESULTS

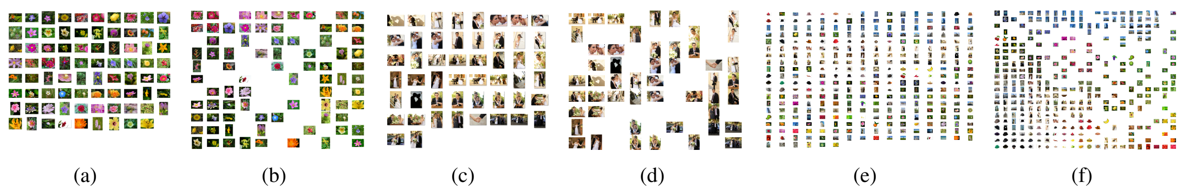


Fig. 6.1: Comparing various arrangements using PictureBoard to simple (file-name based) arrangement. Please zoom in the electronic version of the paper to see images.

We present several results of comparing input simple arrangement and the arrangement after using Picture board in Figure 6.1. More results can be seen in the accompanying video, along with examples of interaction sessions with PictureBoard.

6.1 User study

To validate the usability and benefits of PictureBoard we conduct three studies. The first study tested the automatic context-dependent weights calculation (see Section 5.1). The goal of the study was to find if the arrangements created with the context-dependent weights create better initial arrangements than ones created with uniform weights. We chose six different sets of images and arranged them automatically twice: once using uniform weights for all feature spaces, and once using the context-dependent weights. We then showed the two arrangements side by side and asked participants to choose which one looks better. The results are summarized in Figure 6.2. We had 30 participants in our study. In general, in 5 of the 6 sets, the arrangement with the context dependent weights were selected with higher probability. We combined all results together and used the Chi-Square test and can say with 95% confidence that there is a significant difference between the number of times people prefer the context-dependent weights over the uniform weights.

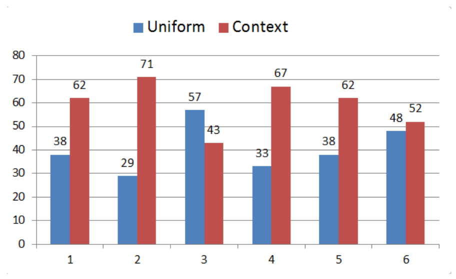


Fig. 6.2: Results of human preferences of arrangements with uniform weights (blue) vs. arrangements with context dependent weights (red) on six image collections (number denote percent). The study sets can be seen in the supplementary materials for this paper.

In our second study we wanted to assess the overall process of ordering the images with PictureBoard compared to arranging the images manually. We used three different collections of images (see Figure 6.3) and asked people to arrange them once using PictureBoard interface and once manually. There were 10 participants in this study and we compared the arrangement times and the number of image movements used in the process. The results, summarized in Figure 6.4, show that there is around 20% time savings in the arrangement. However, examining the number of movements used, we can see a drop from an average of 47 moves in manual arrangements to 10 using PictureBoard. In effect people using PictureBoard were spending more time examining the ordering and thinking and less time manually shifting images around. We conclude that ordering an image collection using PictureBoard is much less tedious, but also slightly faster than manual arrangement.

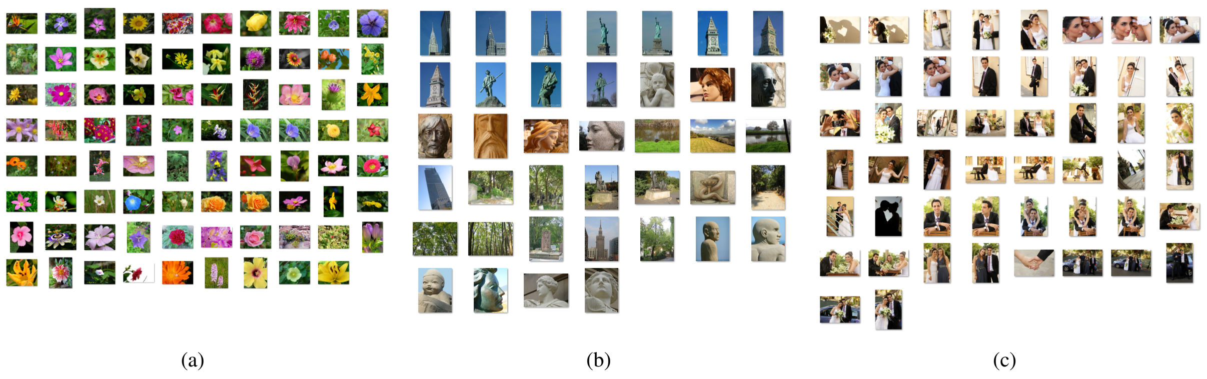


Fig. 6.3: Image collections used in the second user study.

Set #	1	2	3	Average
Number of images	79	39	50	206
Manual (sec.)	248	115	191	185
Manual (no. moves)	55	29	56	47
Learning (sec.)	205	104	136	149
Learning (no. moves)	9	13	8	10
Time saving	17%	10%	29%	20%
Moves saving	84%	65%	86%	79%

Fig. 6.4: Results of arrangement study.

PictureBoard can also be used to measure the importance different people give to various low-level image features. In our study we recorded the final weights of each participant and measured the average and standard deviation of each weight. This can show whether certain collections are perceived similarly or not by different people. If the final weights would be consistent across users, it would indicate that the collection is perceived in a similar manner by different people. If the weights would vary, it would indicate that the collection can be viewed (and arranged) in different manners. Figure 6.5 summarizes the weights of the different features for the first collection we tested (excluding testers who left the initial arrangement intact). As can be seen, most testers preferred arranging the set mostly by color, but some of them gave more weights to other feature spaces such as HOG, and number of segments. In general this type of collection is strongly perceived as color-oriented, which was less the case in the other two sets.

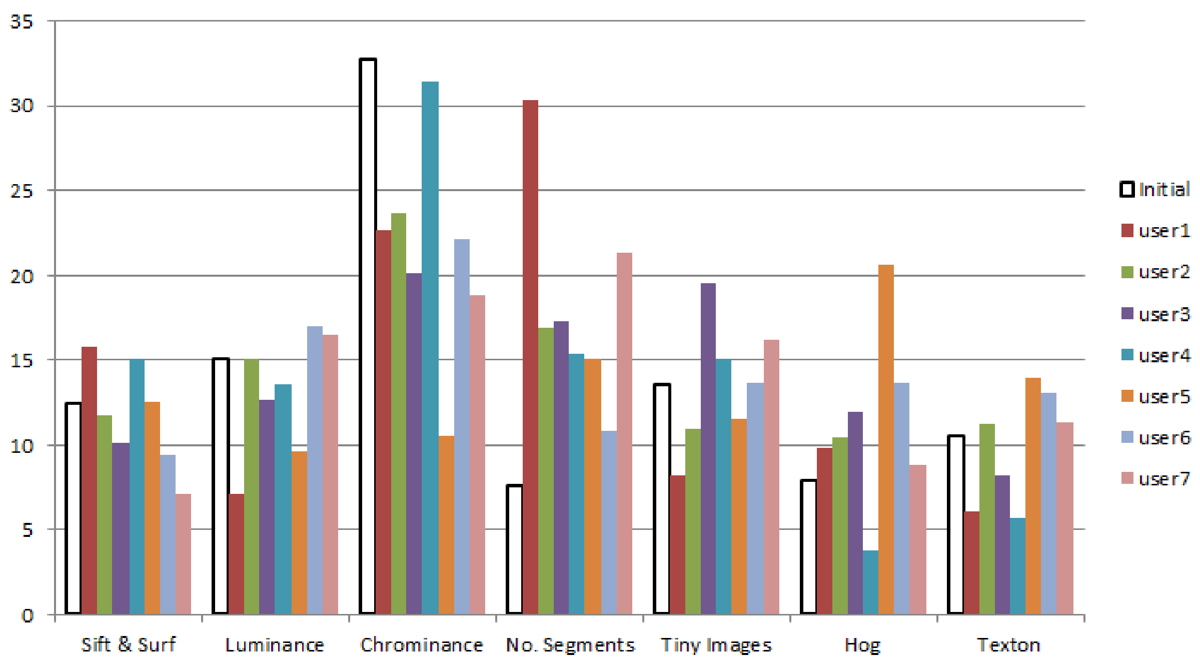


Fig. 6.5: Final feature-spaces weights after user interaction for set (1) in the second user study.

In the third study we asked users to select the best arrangement out of five arrangement of each set. We tested the following arrangement methods: *a*) manual; *b*) PictureBoard with user-defined weights and fine-tuning; *c*) PictureBoard with initial weights; *d*) PictureBoard with user defined weights (without using any of the SOM consistency preservation methods); and *e*) random. We tested three sets, for which we compared each arrangement type to all of the others. Figure 6.6 summarizes the number of time each of the arrangement type was selected as the better arrangement when compared to another arrangement type.

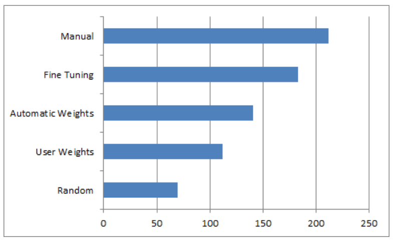


Fig. 6.6: Results of comparison of different arrangement types. Score is the number of times each type was selected when comparing to one of the other types.

We can clearly see that arrangements created by PictureBoard are better than the random arrangement, and slightly worse than the manual arrangement. It is also noticeable that the initial calculated weights outperforms the user-specific weights, although the users created the arrangements preferred the latter. We conclude that the reason for this is that user weights are specific to the user, and different users may prefer our heuristic weights.

6.2 Limitations

It is clear that capturing perceptual similarity or difference in images is a challenge. There are many high level semantics that PictureBoard cannot capture using the type of features used. For example, if the user would want to arrange a collection based on the people found in the images PictureBoard will fail. Even using a face detector would not suffice as it would be difficult to distinguish e.g. strangers from acquaintances. In terms of the interface re-arranging after every move can sometime mix back images that were correctly ordered. Similarly, even with all consistency measures we added, the fact that the basic SOM algorithm projects each time a high dimensional feature spaces to 2D can create new arrangements that confuse the user. For these reasons we added a “re-order” button that projects again the images without changing the weights. This button is used if the current arrangement does not satisfy the user. Lastly, we have found that the idea of automatic re-arrangement when moving an image can take time to get used to, as people are used to simply move images manually.

Chapter 7

Conclusions

7.1 Conclusions

We presented a method for assisting users to arrange a given collection of images. The method allows the user to put more emphasis on different perceptual aspects of the images and let the program arrange the images instead of tediously arranging them manually. Since each user may want to arrange the collection differently, we defined a way to collect low-level semantic information from the user in an interactive, fun manner. The user affects the shape of the arrangement by moving images, implicitly changing the relative importance of certain features of the images.

The results of our experiments with PictureBoard show that the proposed method and its implementation are useful for arranging images according to the user's perceptual intentions. Although using PictureBoard is a non-trivial task, after a few explanations and trials, it becomes fun and efficient, and allows a noticeable time saving, and a significant image movements saving.

In the future we would like to add more features to PictureBoard's set of features and compare the effect on the final arrangements. Such features may be of low level, but also high level such as face recognition and object recognition. We would also like to improve some of the existing features such as image segmentation to make them contain more data. It is interesting to try and capture some high level semantics using PictureBoard. Our hope is that a linear combination of low level features (which is the way PictureBoard represents the images) can approximate some high level semantics. We also want to continue investigating differences in the final weights between people. These differences may indicate perceptual image differences and it would be interesting to find a connection between the two.

BIBLIOGRAPHY

[1] Kai Uwe Barthel. Improved image retrieval using automatic image sorting and semi-automatic generation of image semantics. In *WIAMIS*, pages 227–230. IEEE Computer Society, 2008.

[2] H. Bay, T. Tuytelaars, and L. J. Van Gool. SURF: Speeded up robust features. In *ECCV*, pages I: 404–417, 2006.

[3] Chaomei Chen, George Gagaudakis, and Paul L. Rosin. Content-based image visualization. In *IV*, pages 13–18, 2000.

[4] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *CVPR*, pages I: 886–893, 2005.

[5] Ritendra Datta, Jia Li, and James Ze Wang. Content-based image retrieval: approaches and trends of the new age. In HongJiang Zhang, John R. Smith, and Qi Tian, editors, *Multimedia Information Retrieval*, pages 253–262. ACM, 2005.

[6] Da Deng, Jianhua Zhang, and Martin Purvis. Visualisation and comparison of image collections based on self-organised maps. In Martin Purvis, editor, *Australasian Workshop on Data Mining and Web Intelligence (DMWI2004)*, volume 32 of *CRPIT*, pages 97–102, Dunedin, New Zealand, 2004. ACS.

[7] Michael Dittenbach, Dieter Merkl, and Andreas Rauber. Growing hierarchical self-organizing map. In *Proceedings of the International Joint Conference on Neural Networks*, volume 6, pages 15–19, Piscataway, NJ, 2000. Technische Universitat Wien, IEEE.

[8] Daniel Engel, Lars Hüttenberger, and Bernd Hamann. A survey of dimension reduction methods for high-dimensional data analysis and visualization. In Christoph Garth, Ariane Middel, and Hans Hagen, editors, *VLUDS*, volume 27 of *OASICS*, pages 135–149. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, Germany, 2011.

[9] P. F. Felzenszwalb and D. P. Huttenlocher. Efficient graph-based image segmentation. *International Journal of Computer Vision*, 59(2):167–181, September 2004.

[10] James Fogarty, Desney S. Tan, Ashish Kapoor, and Simon A. J. Winder. Cueflik: interactive concept learning in image search. In Mary Czerwinski, Arnold M. Lund, and Desney S. Tan, editors, *CHI*, pages 29–38. ACM, 2008.

[11] Qasim Iqbal and J.K. Aggarwal. Feature integration, multi-image queries and relevance feedback in image retrieval, 2003.

[12] B. Julesz and R. Bergen. Textons, the elements of texture perception, and their interactions. *Nature*, 290:91–97, 1981.

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

[14] Teuvo Kohonen. Self-organization of very large document collections: State of the art. In L. Niklasson, M. Bodén, and T. Ziemke, editors, *Proceedings of ICANN98, the 8th International Conference on Artificial Neural Networks*, volume 1, pages 65–74. Springer, London, 1998.

[15] P. Koikkalainen and E. Oja. Self-organizing hierarchical feature maps. In *IEEE International Joint Conference on Neural Networks (4th IJCNN'90)*, volume II, pages II–279–II–284, San Diego, 1990. IEEE. Lappeenranta University of Technology.

[16] Pasi Koikkalainen. Progress with the tree-structured self-organizing map. In *ECAI*, pages 211–215, 1994.

[17] Markus Koskela. Interactive image retrieval using self-organizing maps, October 31 2003.

[18] Kuhn, H. W. The Hungarian method of solving the assignment problem. *Naval Res. Logistics Quart.*, 2:83–97, 1955.

[19] J. T. Laaksonen, J. M. Koskela, S. Laakso, and E. Oja. PicSOM: Content-based image retrieval with self-organizing maps. *Pattern Recognition Letters*, 21(13-14):1199–1207, December 2000.

[20] J. T. Laaksonen, J. M. Koskela, S. Laakso, and E. Oja. Self-organising maps as a relevance feedback technique in content-based image retrieval. *Pattern Analysis and Applications*, 4(2/3 2001):140–152, 2001.

[21] Alexander Ladikos, Edmond Boyer, Nassir Navab, and Slobodan Ilic. Region graphs for organizing image collections, 2010.

[22] Guang-Hai Liu, Lei Zhang 0006, Yingkun Hou, Zuoyong Li, and Jing-Yu Yang. Image retrieval based on multi-texton histogram. *Pattern Recognition*, 43(7):2380–2389, 2010.

[23] Hao Liu, Xing Xie, Xiaoou Tang, Zhiwei Li, and Wei-Ying Ma. Effective browsing of web image search results. Technical Report MSR-TR-2004-117, Microsoft Research (MSR), November 2004.

[24] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[25] Lev Manovich. What is visualization. *VisualStudies*, 26(1):36–49, 2011.

[26] Kanti V. Mardia, J. T. Kent, and J. M. Bibby. *Multivariate Analysis (Probability And Mathematical Statistics) Author: , Publisher*. Academic Press, 1980.

[27] G. P. Nguyen and M. Worring. Similarity based visualization of image collections, 2005.

[28] K. Rodden, W. Basalaj, D. Sinclair, and K. Wood. Evaluating a visualisation of image similarity as a tool for image browsing. In *1999 IEEE Symposium on Information Visualization (INFOVIS '99)*, pages 36–43, Washington - Brussels - Tokyo, October 1999. IEEE.

[29] Kerry Rodden. Evaluating similarity-based visualisations as interfaces for image browsing. Technical Report UCAM-CL-TR-543, University of Cambridge, Computer Laboratory, September 2002.

[30] Kerry Rodden, Wojciech Basalaj, David Sinclair, and Kenneth Wood. Does organisation by similarity assist image browsing? In *Proceedings of ACM CHI 2001 Conference on Human Factors in Computing Systems*, Sensable Navigation Search, pages 190–197, 2001.

[31] Kerry Rodden and Kenneth R. Wood. How do people manage their digital photographs? In *Proceedings of the conference on Human factors in computing systems*, pages 409–416. ACM Press, 2003.

[32] S. T. Roweis and L. K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326, December 2000.

[33] Rubner, Tomasi, and Guibas. The earth mover's distance as a metric for image retrieval. *IJCV: International Journal of Computer Vision*, 40, 2000.

[34] Abhinav Shrivastava, Tomasz Malisiewicz, Abhinav Gupta, and Alexei A. Efros. Data-driven visual similarity for cross-domain image matching. *ACM Trans. Graph*, 30(6):154, 2011.

[35] Nicolae Suditu and Francois Fleuret. HEAT: Iterative relevance feedback with one million images, 2011.

[36] A. Torralba, R. Fergus, and W. T. Freeman. 80 million tiny images: A large data set for nonparametric object and scene recognition. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 30(11):1958–1970, November 2008.

תקציר

בהינתן אוסף תמונות המכיל 3 עד כמה מאות תמונות, למשל בתיקייה במחשב האישי או בשירותים כמו facebook, הצורה הנפוצה כיום לסידור שלהן על המסך היא בסידור "רשת" לפי תאריך או שם קובץ, ולעיתים לפי מטה-מידע המצורף לתמונה (תגיות, מילות חיפוש וכד'). סידור זה לא מתחשב בדרך כלל בתוכן התמונה. ניתן לסדר את התמונות גם לפי התוכן, ע"י הטלה על המרחב. לצורך כך יש לבחור ייצוג של התמונות במרחב גבוה כלשהו, למשל היסטוגרמת צבע, ולהגדיר דרך לחשב מרחק בין הייצוג במרחב זה. לאחר מכן יש לבצע הורדת מימד, או הטלה, למרחב דו מימדי. בהינתן אוסף תמונות ספציפי, נשאלת השאלה איך לבחור את הייצוג המתאים במרחב במימד גבוה, כאשר לעתים קרובות הבחירה תלויה באוסף התמונות הספציפי ובהעדפות המשתמש הספציפי.

העבודה שלנו מנסה לפתור בעייה זו ע"י שילוב מספר ייצוגים במרחבים בעלי מימד גבוה, ושקלול פונקציות המרחק של כל מרחב לפונקציה אחת. לאחר מכן אנו מסדרים את התמונות על המסך, והמשתמש יכול "ללמד" את התוכנה על העדפות הסידור שלו ובכך לכייל את המשקלים של כל ייצוג לפי בחירתו. כמו כן אנו מציגים דרך לקביעת המשקל ההתחלתי של כל ייצוג בהתאם לסט התמונות הנוכחי.

אלגוריתם הסידור בו השתמשנו הוא Self Organizing Map, והמרחבים בהם אנו מייצגים את התמונות מייצגים מאפיינים שונים של התמונה: היסטוגרמות צבעים, טקסטורה, Tiny Image, היסטוגרמת כיוונים, מספר סגמנטים של התמונה ואפילו נקודות עניין בתמונה ע"י שימוש ב-SIFT ו-SURF.

התוכנה שכתבנו נקראת PictureBoard, והיא מאפשרת למשתמש לסדר אוסף של תמונות לבחירתו, ולאחר מכן לבצע תיקונים בסידור וללמד את התוכנה את המשקלים הרצויים. מכיוון ש-SOM הוא אלגוריתם אקראי, כל סידור שלו יכול להיראות שונה, אפילו עם אותם משקלים, ולכן יכול לבלבל את המשתמש. כדי להתגבר על בעיה זו אנו מציגים מספר שיפורים לאלגוריתם ה-SOM המקורי כך שהסידור הבא יהיה קרוב כמה שיותר לסידור הקודם, אך יכיל בתוכו מספיק שינויים כדי להגדיר אותו כסידור טוב יותר. בדקנו את עבודתנו בעזרת מספר ניסויים עם משתתפים חיצוניים.

IDC HERZLIYA | Efi Arazi School of Computer Science

המרכז הבינתחומי בהרצליה

בית-ספר אפי ארזי למדעי המחשב

סידור תמונות אינטראקטיבי בעזרת דמיון תמונות מבוסס תוכן

סיכום עבודת גמר המוגשת כמילוי חלק מהדרישות לקראת תואר

מוסמך במסלול מחקרי במדעי המחשב

על-ידי צח ירימי

העבודה בוצעה בהנחיית פרופסור אריאל שמיר

מאי 2013