Available online at www.sciencedirect.com
ScienceDirect

Journal of Visual Languages and Computing
19 (2008) 203–224

Journal of
Visual Languages
& Computing

www.elsevier.com/locate/jvlc

Interactive access to large image collections using similarity-based visualization

G.P. Nguyen*, M. Worring

Intelligent Systems Lab Amsterdam, University of Amsterdam, Kruislaan 403, 1098SJ Amsterdam, The Netherlands

Received 20 May 2005; received in revised form 5 April 2006; accepted 14 September 2006

Abstract

Image collections are getting larger and larger. To access those collections, systems for managing, searching, and browsing are necessary. Visualization plays an essential role in such systems. Existing visualization systems do not analyze all the problems occurring when dealing with large visual collections. In this paper, we make these problems explicit. From there, we establish three general requirements: overview, visibility, and structure preservation. Solutions for each requirement are proposed, as well as functions balancing the different requirements. We present an optimal visualization scheme, supporting users in interacting with large image collections. Experimental results with a collection of 10,000 Corel images, using simulated user actions, show that the proposed scheme significantly improves performance for a given task compared to the 2D grid-based visualizations commonly used in content-based image retrieval.

Keywords: Content-based image retrieval; Similarity-based visualization; Interaction

1. Introduction

Through the development of multimedia technologies and the availability of cheap digital cameras, the size of image collections is growing tremendously. Collections range from consumer pictures, to professional archives such as photo stocks of press agencies,

*Corresponding author. Tel.: +31 20 5257528; fax: +31 20 5257490.
E-mail addresses: giangnp@science.uva.nl (G.P. Nguyen), worring@science.uva.nl (M. Worring).

1045-926X/$ - see front matter © 2006 Elsevier Ltd. All rights reserved.
doi:10.1016/j.jvlc.2006.09.002

museum archives, and to scientific pictures in medicine, astronomy, or biology. Hence, large image collections are common everywhere.

The use of image collections is domain dependent. For consumer pictures, a task is finding all pictures taken of a family member in a certain event. In medicine, a doctor wants to search images similar to a given one, to diagnose a disease. In general, when working with image collections the main task is searching relevant pictures in the collection.

When the collection contains a couple of hundred images, one can find all relevant images by visually inspecting the whole collection. If the size of the collection increases to thousands or even millions of images, one needs efficient methods for searching and browsing those collections. To that end, we should note the "*semantic gap*" between the system's automatic indexing capability and the user's conceptual interpretation of the data [1]. Interaction and visualization are needed to bring the system perspective and user perspective together.

The appropriate visualization and interaction method are task dependent. For understanding the structure of the collection, visualization should allow interaction with clusters, local structures and outliers [2]. For browsing, some form of rapid serial visual presentation can be appropriate [3], or network relations between images can be visualized for easy navigation [4]. Searching requires the possibility to see overviews as well as being able to interactively zoom-in on the information [5]. Finally, annotation requires that images which should receive the same annotation are close to each other so they can be annotated with one interaction step [6]. The examples indicate the need for generic tools which can be combined to support each of these tasks.

In content-based image retrieval literature, few systems take the visualization as a tool for exploring the collection. In most query by example-based systems, a randomly selected set of images from the collection is displayed in a two-dimensional (2D) grid [1,7,8]. In this way, the user does not get an overview of the collection. As a consequence, much time is wasted in considering non-relevant images and relevant images are easily missed. Visualization offers the opportunity to guide the user in her exploration.

To guide a user, the structure of the collection is of prime importance. Thus, focus must not be on the images alone, but especially on the relations between the images. These relations are captured by a *similarity* function. Similarity is a very generic notion and can be based on *features* computed from the image content, free text descriptions, or semantic annotations. Recently advanced systems have been developed for browsing images based on similarity [2,4,9–14]. In these systems the similarity-, and thus structure, based visualization of the collection is the guide for the user.

None of the above methods explicitly addresses the problems occurring when visualizing large visual collections. The most important problem is the limited display size, not allowing to show the whole collection as images on the screen. Some systems have made an effort to relief this limitation by showing the whole collection as a point set [2]. Once the user selects a point, the corresponding image is displayed [10]. To see the visual structure of the collection, many images need to be shown simultaneously. A problem here is visibility. If small thumbnails are used, the images cannot be understood by the user. Larger thumbnails lead to substantial overlap of the images on the screen. Most of the existing systems do not take this problem into account. Exceptions are [15,16], but their treatment of the problem is rather ad-hoc. A final issue to consider is the difference between the high dimensional feature space, typically of dimension 50 or more, and the 2D display.

A mapping between the two is needed. For instance, in [15] Principal Component Analysis (PCA) is employed, whereas Rubner et al. [4] uses Multi-Dimensional Scaling (MDS). Inevitably in the projection, information on the structure of the collection is lost. In summary, there are a number of problems when visualizing large image collections. In this paper, we make them explicit.

The advanced visualization tools mentioned above make the development of systems more complex. The question arises whether it is worth the effort. This requires extensive evaluation. Evaluating the usability of a system can be done subjectively or objectively [17]. To evaluate subjectively, real users judge the performance of the system. In this direction, only Rodden [16,18], performed evaluations of similarity-based visualization. However, the use of subjective evaluation is quite expensive and cannot be repeated easily. When the task is well defined some aspects of usability evaluation can be automated [17]. Such objective evaluation has several advantages such as reducing the cost of evaluation, reducing the need for evaluation expertise and increasing the coverage of evaluated features [17]. It effectively allows to decompose the evaluation into evaluating the methodology and the design of the interface and its usability. We view objective evaluation as an important step in the development of complex systems.

Various techniques that aim at providing this type of evaluation can be found in [17] such as testing, inspection, inquiry, analytical modelling, and simulation. Simulation method is most suited for our case as it reports the performance of the system and user's interactive actions. This method uses models of the user and interface design to simulate a user interacting with the system. We employ this method making the scenario of use and criteria for success of our new approach explicit. From there, we develop a method to simulate the user actions.

The paper is organized as follows. In Section 2, we analyze the general requirements for visualizing large image collections. A set of requirements is proposed. Solutions for each requirement are then given in Section 3. From there, cost functions are established for optimal visualization in Section 4. A visualization system to illustrate the proposed theory with a collection of 10,000 Corel images and experiments based on a user simulation system are presented in Section 5.

2. Problem analysis

In this section, we analyze in detail the problems occurring when visualizing a large visual collection. From there requirements for a generic system are defined.

First of all, we give some notations and conventions used throughout the paper. An *image collection* is a set of N images, where we assume $N \gg 1000$. Each image I in the collection is represented by a feature vector f_I, for example, a color or texture histogram. The similarity of two images I, J is denoted by $\mathcal{S}_{IJ}$. Often we will rather use a distance, or dissimilarity measure D_{IJ}, which is zero whenever the images have exactly the same feature vector, and larger than zero otherwise. Thus, each image in the collection corresponds to a point in a high-dimensional feature space. The image collection, the corresponding feature vectors, and the distances between them define the *information space*. Fig. 1 shows a simple example of a 3D information space.

Apart from the data preparation step, the general scheme of a visualization system contains three steps. First, in the projection step, the information space is projected to the *visualization space* yielding a 2D view. The image collection now corresponds to a set of positions $\{\vec{y}_i\}_{i=1}^{N}$ in the 2D space. In the selection step, the system selects a subset of images

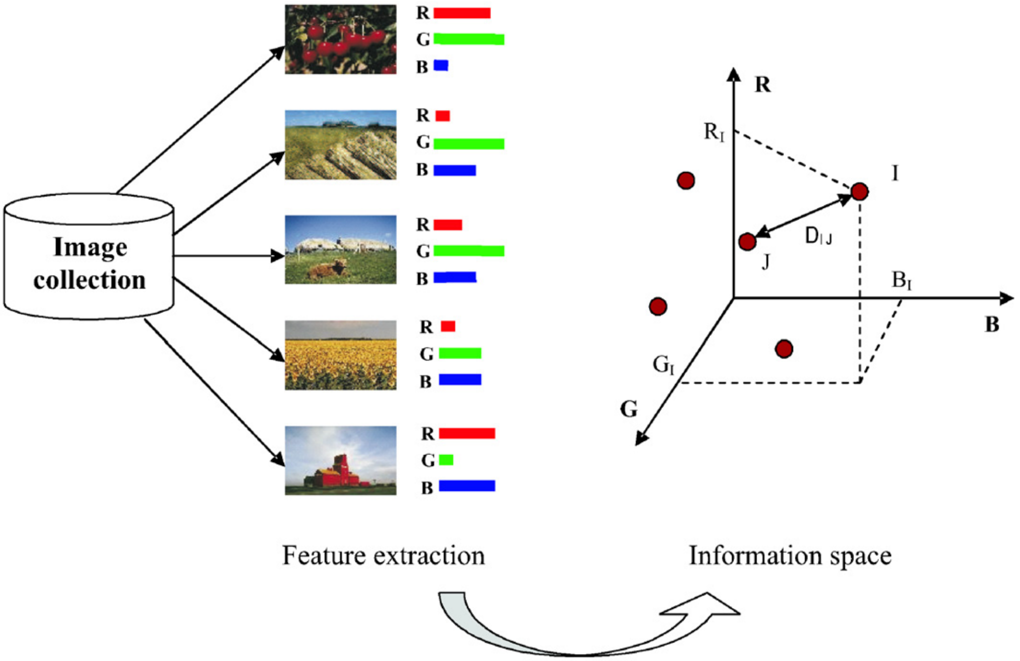


Fig. 1. An example of a three-dimensional information space based on the amount of red, green and blue in an image. Note that in practice the dimensionality is much higher.

to display. Finally, the interaction step involves the visualization of the selected set and user feedback. The overall scheme is illustrated in Fig. 2.

An obvious issue in visualizing a large collection is the *limited display size* of the visualization space. This dictates that only a restricted number of images can be shown simultaneously since the goal is to show the content of the images. Randomly selecting images is certainly not a good approach as it does not capture the distribution of images in the collection [19]. Therefore, the first requirement is:

Overview requirement: *The visualization should give a faithful overview of the distribution of images in the collection.*

Secondly, the information space often exhibits considerable structure in the form of clusters, low-dimensional manifolds, and outliers. Many examples are found in literature e.g. in [20,21]. Fig. 3 shows examples of different structures found in a set of images from a video sequence, similar to those in [21]. Another example of structure is present in a set of images of the same object in different conditions such as different light source, viewpoint, and/or orientation (see Fig. 4). This structure should be preserved in the visualization. Thus we have:

Structure preservation requirement: *The relations between images should be preserved in the projection of the information space to the visualization space.*

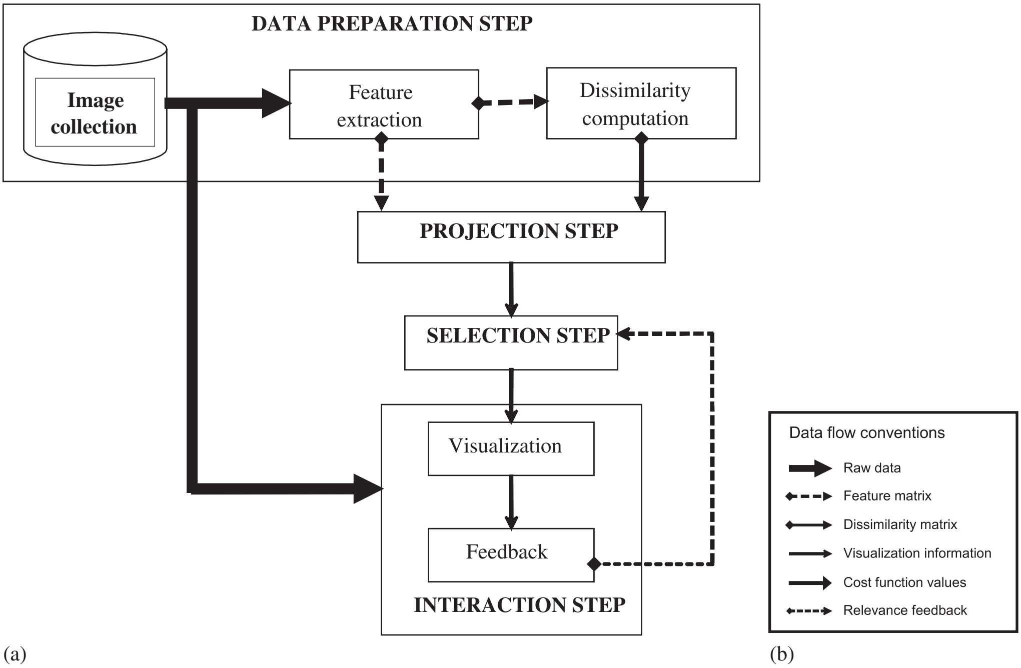


Fig. 2. General scheme of an image collection visualization system.

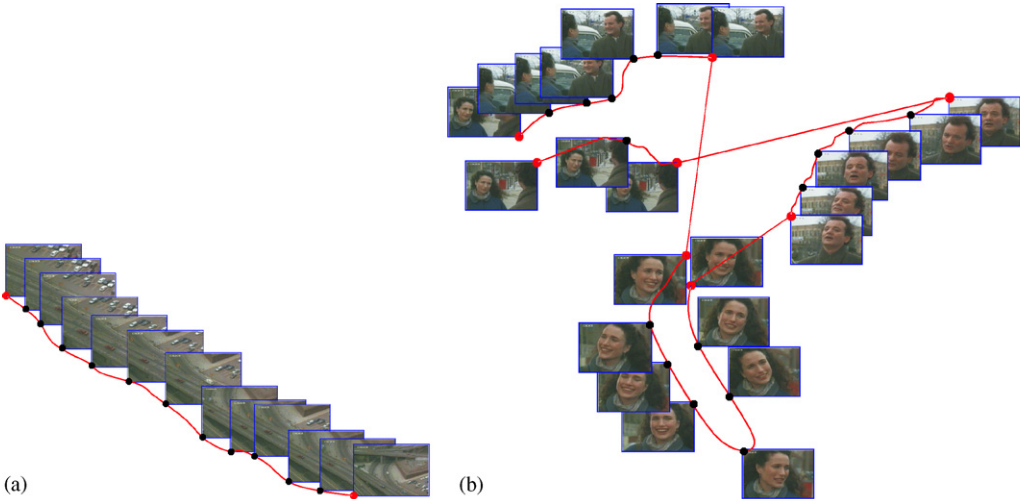


Fig. 3. (a) Linear structure of a sequence showing a car riding in a street. (b) Non-linear structure of a video sequence capturing the conversation between two people.

Finally, it must be stressed that the image itself provides important information for the user, but only when it is large enough to be understood. Now, when a set of images is displayed, they tend to overlap each other partially or fully [15,16]. The overlap between

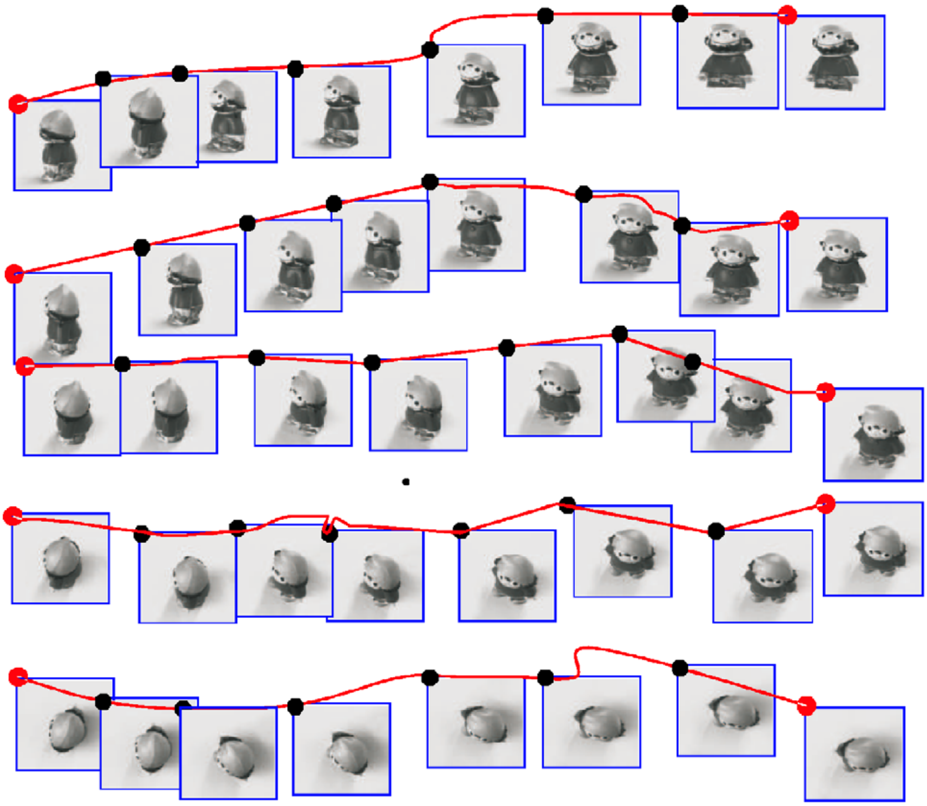


Fig. 4. A set of images of a toy taken from different orientations [20]. When the viewpoint changes in one direction, the structure is linear.

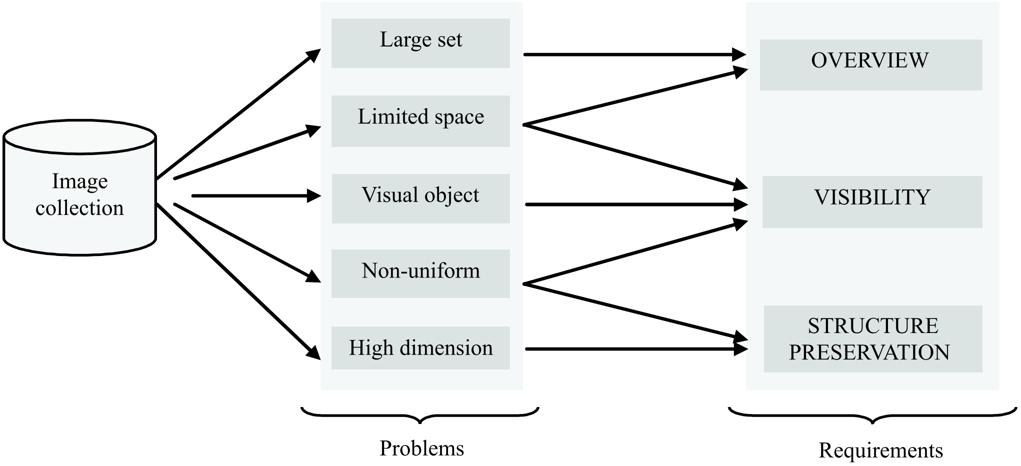


Fig. 5. Problems and requirements for visualization of large image collections.

images influences the quality of a visualization tool greatly. Due to overlap important images can be missed. Therefore, the overlap among them should be reduced as much as possible, leading to the following requirement.

Visibility requirement: *All displayed images should be visible to the extent that the user can understand the content of each image.*

So we have three general requirements: *overview*, *structure preservation*, and *visibility*. Those requirements are not independent. To increase the visibility, images should be spread out. This has a negative effect on the preservation of structure. Furthermore, more representatives yield better overviews, but the visibility decreases because overlap becomes more likely. The relations among the three requirements are illustrated in Fig. 5. To generate appropriate visualizations, we need means to balance the different requirements. In the next sections, we find appropriate cost functions for each of the requirements, which are then combined and jointly optimized.

3. Projection and selection methods

In this section, we consider different methods for projecting the information space to visualization space. As well as methods for selecting images for the overview.

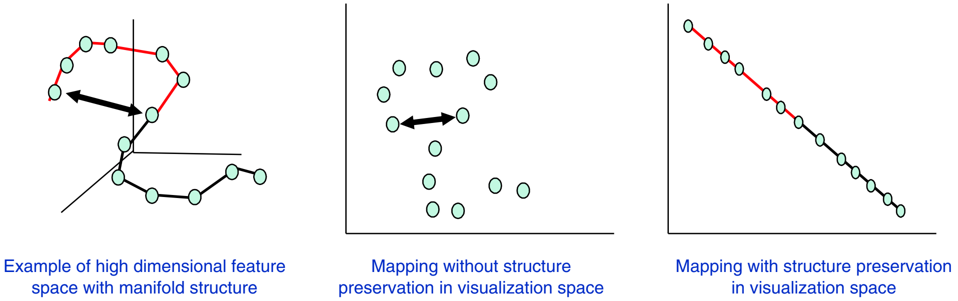
3.1. From information space to visualization

Well-known techniques used in existing systems are PCA [15] and MDS [4,16]. These methods are designed to find linear structures in the information space. If the information space contains a non-linear structure they will not satisfy the structure preservation requirement. This is illustrated in Fig. 6.

This issue is considered in new techniques, known as non-linear embedding algorithms, namely isometric mapping (ISOMAP) [22], local linear embedding (LLE) [23], and more recently stochastic neighbor embedding (SNE) [24]. The proposed mapping algorithms are able to preserve the real structure of the data and perform better than PCA and MDS. We consider non-linear techniques only.

ISOMAP was introduced in 2000 [22]. Instead of directly computing the distance between points, the authors use graph-based distance computation aiming to measure the distance along local structures. Their algorithm contains three main steps. First, the algorithm builds the neighborhood graph using t-nearest neighbors (the t closest points) or ε-nearest neighbors (all points with distance to the point less than ε). The second step uses Dijkstra's algorithm to find shortest paths between every pair of points in the graph. The distance for each pair is then assigned the length of this shortest path. After the distances are recomputed, MDS is applied to the new distance matrix.

A different approach is SNE [24], a probabilistic projection method. This method first computes the probabilities that two points take each other as neighbors, assuming a Gaussian distribution, in both the high and the 2D space. The method then tries to match the two probability distributions. Hence, it provides preservation of local geometric structure and also keeps points which are distant in the high dimensional information space distant in the visualization space. The working principle of the SNE can be briefly



Example of high dimensional feature space with manifold structure

Mapping without structure preservation in visualization space

Mapping with structure preservation in visualization space

Fig. 6. Illustration of non-linear mapping.

described as follows. Let $\mathscr{P} = P_{IJ}$ denote the probability that an image I would pick j as its neighbor in the high-dimensional space. Under the Gaussian distribution assumption, P_{IJ} is given by

$$P_{IJ} = \frac{\exp(-D_{IJ}^2)}{\sum_{L \neq I} \exp(-D_{IL}^2)}, \tag{1}$$

with $P_{II} = 0$. Note that this measure in general is asymmetric: $P_{IJ} \neq P_{JI}$.

In the 2D space, SNE initializes $\{\vec{y}_i\}_{i=1}^{N}$ at random positions. The induced probability $\mathscr{Q} = \{Q_{IJ}\}$ is then calculated for every pair of images:

$$Q_{IJ} = \frac{\exp(-\|\vec{y}_i - \vec{y}_j\|^2)}{\sum_{k \neq i} \exp(-\|\vec{y}_i - \vec{y}_k\|^2}, \quad Q_{II} = 0. \tag{2}$$

To measure the distance between these two distributions $\mathscr{P}$ and $\mathscr{Q}$, the Kullback–Leibler distance is used. This asymmetric distance is commonly used in measuring a natural distance from a "true" probability distribution, $\mathscr{P}$, to a "target" probability distribution, $\mathscr{Q}$

$$D_{\mathscr{P}\mathscr{Q}} = \sum_I \sum_J P_{IJ} \log \frac{P_{IJ}}{Q_{IJ}}.$$

The algorithm then finds the optimal positions $\{\vec{y}_i\}$ by minimizing $D_{\mathscr{P}\mathscr{Q}}$.

SNE uses direct distance computation among points, hence it can benefit from replacing this distance by the graph-based distance from ISOMAP. We therefore propose ISOSNE, a combination of ISOMAP and SNE.

Since SNE uses gradient descent methods it requires substantial computation time, especially when the size of the data reaches several thousand images. LLE [23] can be viewed as an approximation to SNE which is much faster to compute. This method first constructs the t-nearest neighborhood graph. Then, for each point $\vec{y}_i$ it computes the weights w_{ij} that optimally reconstruct $\vec{y}_i$ from its neighbors by minimizing the cost function $\sum_i \|\vec{y}_i - \sum_j w_{ij}\vec{y}_j\|^2$.

In the reference, LLE distance is computed directly, but we can also recompute distance like in ISOSNE. We denote this combination by ISOLLE.

3.2. Selection methods

To select a representative set from the collection to be used in the overview, a common method is dividing the collection into a number of groups. One image from each group is then selected as representative. This requires finding clusters based on the distance matrix.

A comprehensive overview of different clustering techniques is presented in [25]. Comparisons among different methods are given in [26–29]. They conclude that the *k*-means algorithm is one of the most successful methods because of its simplicity in implementation and its linear time complexity. We therefore employ *k*-means to select images for the overview in our system.

The *k*-means algorithm is applied after projection of the information space to visualization space. We initialize the *k* center points at random positions. The reassignment of points to the nearest center is repeated until the clustering satisfies certain requirements, or when the maximum number of iterations is reached. The image corresponding to the point nearest to the cluster center $\{m\}$ is selected as the representative of that cluster.

4. Balancing the requirements

As mentioned in the problem analysis, the three requirements are dependent on one another. In order to balance them, we first develop a cost function for each requirement. For the first two requirements, existing functions are employed, while for the last requirement, we introduce our own cost function as it has not yet been investigated in literature. From there, balancing functions are introduced and applied to realize our final visualization scheme.

4.1. Cost functions

4.1.1. Structure preservation cost function

To preserve the non-linear structure of a collection, the projection algorithm should at least map the neighbors of an image in the information space in such a way that they are neighbors in the visualization space also, which is less strict than preserving distance. In addition, users are also using comparisons of distance rather than absolute distance. The cost function used in SNE is therefore a good option for evaluating different projections. It is given by the Kullback–Leibler distance between the two distributions *P* and *Q* defined in Section 3.1:

$$\mathscr{C}_S = \sum_I \sum_J P_{IJ} \log \frac{P_{IJ}}{Q_{IJ}}. \tag{3}$$

Clearly, the lower this cost, the better the projection has preserved the relations between neighbors.

4.1.2. Overview cost function

When the images are assigned to their corresponding cluster, we need to find a cost function to evaluate the overview provided by the representative images of each cluster. Clearly this depends on the number of clusters *k* asked for, and how well the representatives cover the whole data set. A commonly used measure for the quality of the clustering is the modified Hubert statistic [27] ranging from 0 to 1, where the higher the value the better the clustering. So our overview cost function is

$$\mathscr{C}_O = 1 - \frac{r - M_p M_c}{\sigma_p \sigma_c}, \tag{4}$$

where

$$\begin{aligned}
&r = (1/M)\sum\sum D_{IJ} d(m_I, m_J),\\
&M_p = (1/M)\sum\sum D_{IJ},\\
&M_c = (1/M)\sum\sum d(m_I, m_J),\\
&\sigma_p^2 = (1/M)\sum\sum D_{IJ}^2 - M_p^2,\\
&\sigma_c^2 = (1/M)\sum\sum d^2(m_I, m_J) - M_c^2,\\
&M = k(k-1)/2,
\end{aligned}$$

where m_I is the center of the cluster containing image I and $d(m_I, m_J)$ is the distance between two cluster centers.

4.1.3. Visibility cost function

So far the solution for the requirements can be based on existing measures, mainly because they apply equally well to point-sets.

The major issue in visibility is the overlap of the images displayed. This depends on the number and size of images displayed. A few small images will not overlap, but when 1000 large images are displayed there is always overlap. It also depends on the structure of the data. If images are clustered in information space, the structure preservation requirement will dictate that a lot of overlap is present in visualization space.

To define a cost for the overlap among images displayed, we consider the overlap of two images, and combine them over all pairs. Finding the overlap between the rectangles defining two images is not difficult, however, many different cases have to be distinguished. To develop an analytical function, we make the simplifying assumption that all images have width w and height h, with $w = h$. We then represent an image as a circle, with radius $R = w/2$ (see Fig. 7a). This is a reasonable approximation as the area of the circle covers $\pi/4 \simeq 80\%$ of the image area. So, if the two images overlap outside the area of the circle and inside the image area (see Fig. 7b), the viewable area of the image is larger than 75% of the image area, which is sufficient for visibility.

The overlap between two circles i and j is given by

$$\mathscr{O}_{ij} = \begin{cases} R^2\left(2\arccos\left(\dfrac{d_{ij}}{2R}\right) - \sin\left(2\arccos\left(\dfrac{d_{ij}}{2R}\right)\right)\right) & \text{if } d_{ij} < 2R,\\ 0 & \text{otherwise,} \end{cases} \tag{5}$$

where d_{ij} is the Euclidean distance between the center points of the images I and j.

Hence, if the number of displayed images is n, the total visibility cost is defined by

$$\mathscr{C}_V = \frac{1}{n(n-1)} \sum_i^n \sum_{j \neq i}^n \frac{\mathscr{O}_{ij}}{\Pi R^2}. \tag{6}$$

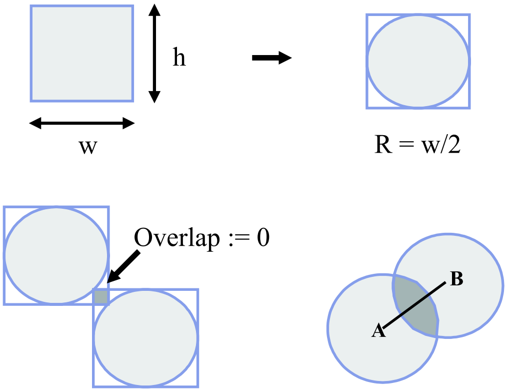


Fig. 7. Image with its corresponding enclosed circle and the overlap area between two images.

Because of the limited size of the visualization space, there is a maximum number of images which can be displayed while satisfying the visibility requirement. Let us assume an image is called viewable if its visible area occupies $t\%$ of the image and the visualization space has size $\mathscr{W}$ and $\mathscr{H}$. Then, the maximum number of displayed images with $t\%$ visible is $(\mathscr{H} \times \mathscr{W})/(h \times w \times t)$. This yields a strong constraint on the design of the visualization method.

4.2. Balancing functions

There are two main relations among the three requirements. First, the relation between overview and visibility, which is affected by the number of representative images. The more images, the better the overview, but visibility is reduced. To balance these two requirements we take a linear combination of their cost functions Eqs. (4) and (6):

$$\mathscr{C}_1 = \lambda_1 \mathscr{C}_O + (1 - \lambda_1)\mathscr{C}_V, \tag{7}$$

where $0 \leqslant \lambda_1 \leqslant 1$. Now, for given λ_1 we find the n where $\mathscr{C}_1(n)$ reaches its maximum value:

$$n_{opt} = \underset{n \in [2 \ldots n_{max}]}{\operatorname{argmax}} \mathscr{C}_1(n). \tag{8}$$

Since this step is done offline, we use a brute-force approach computing $\mathscr{C}_O$ and $\mathscr{C}_V$ for n from 2 to $n_{\max}$.

The second relation is between the visibility and the structure preservation requirement. The more visibility, in other words less overlap, the less structure is preserved. We again use a linear combination of Eqs. (3) and (6). The problem boils down to finding the best optimal positions of images in visualization space where the joint cost of overlap and disobeying structure preservation is minimal:

$$\vec{y}_{opt} = \min_{\vec{y}} \mathscr{C}_2(\vec{y}),$$

with

$$\mathscr{C}_2(y) = \lambda_2 \mathscr{C}_S(y) + (1 - \lambda_2)\mathscr{C}_V(y), \tag{9}$$

where $0 \leqslant \lambda_2 \leqslant 1$.

To find the optimum, gradient descent is applied. This requires to compute how the cost function changes when images are moved away from their positions i.e. the derivative of $\mathscr{C}_2$ with respect to $\vec{y}_i$:

$$\frac{\partial \mathscr{C}_2}{\partial \vec{y}_i} = \lambda_2 \frac{\partial \mathscr{C}_S}{\partial \vec{y}_i} + (1-\lambda_2)\frac{\partial \mathscr{C}_V}{\partial \vec{y}_i}.$$

The differentiation of $\mathscr{C}_V$ is given in [24]

$$\frac{\partial \mathscr{C}_S}{\partial \vec{y}_i} = 2\sum_J (\vec{y}_i - \vec{y}_j)(P_{IJ} - Q_{IJ} + P_{JI} - Q_{JI}).$$

Given Eq. (6), we derive

$$\frac{\partial \mathscr{C}_V}{\partial \vec{y}_i} = \frac{1}{k(k-1)\Pi R^2}\sum_J (\vec{y}_i - \vec{y}_j)\frac{-\sqrt{4R^2 - d_{ij}^2}}{d_{ij}}.$$

4.3. Final visualization scheme

Up to this point, we have analyzed the visualization requirements and how to optimize them using balancing functions. We propose a new visualization scheme which conforms the general scheme in Fig. 2 (see Fig. 8).

The data preparation step, the projection step and a part of the selection step can be prepared beforehand. Therefore, we call these steps the offline process. In this stage, features are first extracted for all images in the collection and a dissimilarity matrix is computed. Next, a projection from the information space to the visualization space is applied. After that, k-means is applied with n ranging from 2 to $n_{\max}$ (we select $n_{\max} = 300$). We then calculate $\mathscr{C}_O$ and $\mathscr{C}_V$. From there, the balancing function in Eq. (7) with given λ_1 returns the optimal number of images to be displayed in each iteration. The optimal clustering is kept for subsequent steps.

The other part of the selection step, involving the selection of the representative set, and the interaction step are part of the interactive process. In the first iteration, representatives are the cluster centers. The visualization step computes the arrangement of representative images on the screen according to the second and the third requirement. This means that the balancing function $\mathscr{C}_2$ (Eq. (9)) is optimized to find positions for all displayed images. These positions assure that relations between them are preserved as much as possible and the content of images are sufficiently visible. After the *find next* step, the system selects the set of images to display in the next iteration. The selection contains images which have not been displayed before and are closest to the corresponding center points.

5. Experiments

In this section, we present experiments to validate the different components of the visualization system. The preparation of the offline stage contains the data selection, feature selection, and dissimilarity computation. Then, we compare different mapping algorithms for the projection step. A system to demonstrate the scheme is presented in Section 4.1.3. Finally, we apply our system to a search task, comparing our approach to more traditional visualization.

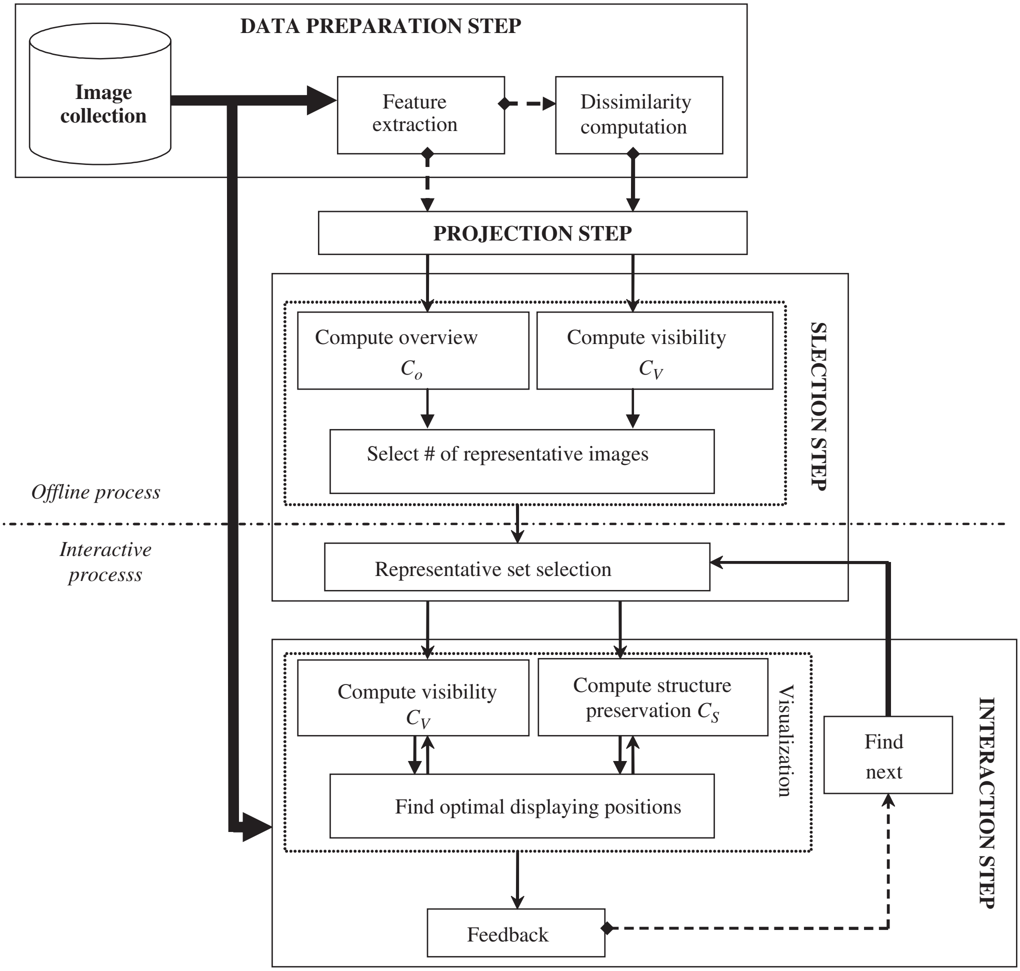


Fig. 8. Scheme of an image collection visualization system, where we combine the general scheme of Fig. 2 with balancing functions.

5.1. Data collection

We select a collection of $N = 10,000$ Corel images. There are 100 predefined categories, where each category contains 100 images. The existing categorization will be used as ground-truth for later evaluation. The images depict different scenes, and objects. Computing the dissimilarities between images strongly depends on the features and dissimilarity function chosen. Because of the large variety in images in this particular collection, no features and/or dissimilarity functions exist which correctly classify images. As in practice this is also the case and we are focussing on the interaction process, we employ simple global color histograms. In particular, hue and saturation (HS) and L^*a^*b (L^* defines lightness, a^* denotes red/green value, and b^* the yellow/blue value). We compute the histogram using 32 bins for each color channel, this means that for HS we get a histogram of 64 dimensions, and 96 in the case of L^*a^*b. Histogram intersection and Euclidean distance are used as similarity functions for HS and L^*a^*b histogram, respectively.

5.2. Comparison of projection methods

As mentioned above, we concentrate on the non-linear dimension reduction methods ISOMAP, LLE, SNE, ISOSNE, and ISOLLE. MDS is used as a baseline.

SNE and ISOSNE are expected to perform best as we have chosen Eq. (3) as the evaluation criterion. However, these two use gradient descent in finding the optimal solutions hence they have a long processing time. LLE and ISOLLE are fast as they use approximations to find the embedding. Hence, in the comparison, we also take into account time complexity. All the experiments are run on the same PC PenIV, 2 GHz. The results are in Table 1. In both experiments, MDS yields the worst performance. SNE and ISOSNE outperform the others, but require 10 times longer processing time than LLE which still has good performance. So when computations are done offline, SNE-based methods are preferred. In practical situations LLE can often be employed.

5.3. A system demonstration

In our system the projection is computed once in the offline stages so we choose ISOSNE. Fig. 9(a) and (b) shows the mapping results of ISOSNE on the given collection using the HS and L^*a^*b feature space, respectively.

For further demonstrating the system, we use the HS features. First, to find the optimal number of clusters the collection should be divided into, we apply Eq. (7) with λ equal to 0.5 and $n \in [2 \dots 300]$. From Eq. (8), we find $n_{opt} = 55$, so the collection is divided into 55 clusters.

Note that the above optimal number of images are with no overlap reduction. This means that in the visualization space, with $n_{opt} = 55$, all displayed images satisfy the visibility requirement. As in practice, one may prefer to have more images on the screen, the value of n_{opt} is used as a threshold. If there is a higher number of displayed images, we will get a better overview, but the visibility is reduced. Then we need to consider the overlap problem. Applying Eq. (9), one can increase the number of displayed images.

In the subsequent online process, in the first iteration of the visualization, images closest to the center points are selected for display. Each image represents one cluster. The second balancing function $\mathscr{C}_2$ with $\lambda_2 = 0.5$ is used to find the optimal positions for the displayed set. This process is repeated to display the subsequent sets of images.

Table 1
Results for MDS, ISOMAP, SNE, LLE, ISOSNE, and ISOLLE in preserving original structure when mapping data from high-dimensional feature space to 2D visualization space

	MDS	ISOMAP	SNE	LLE	ISOSNE	ISOLLE
$\mathscr{C}_S$ with HS feature	0.008653	0.006785	0.000247	0.000252	0.000076	0.000225
$\mathscr{C}_S$ with Lab feature	0.043584	0.004489	0.000089	0.000202	0.000063	0.000190
$\mathscr{T}$ (hour)	~1.5	~1.5	~10	~2.0	~10	~2.0

The first row gives results of mapping from HS feature space. The second row is for L^*a^*b feature space. The last row shows the computation time of each method.

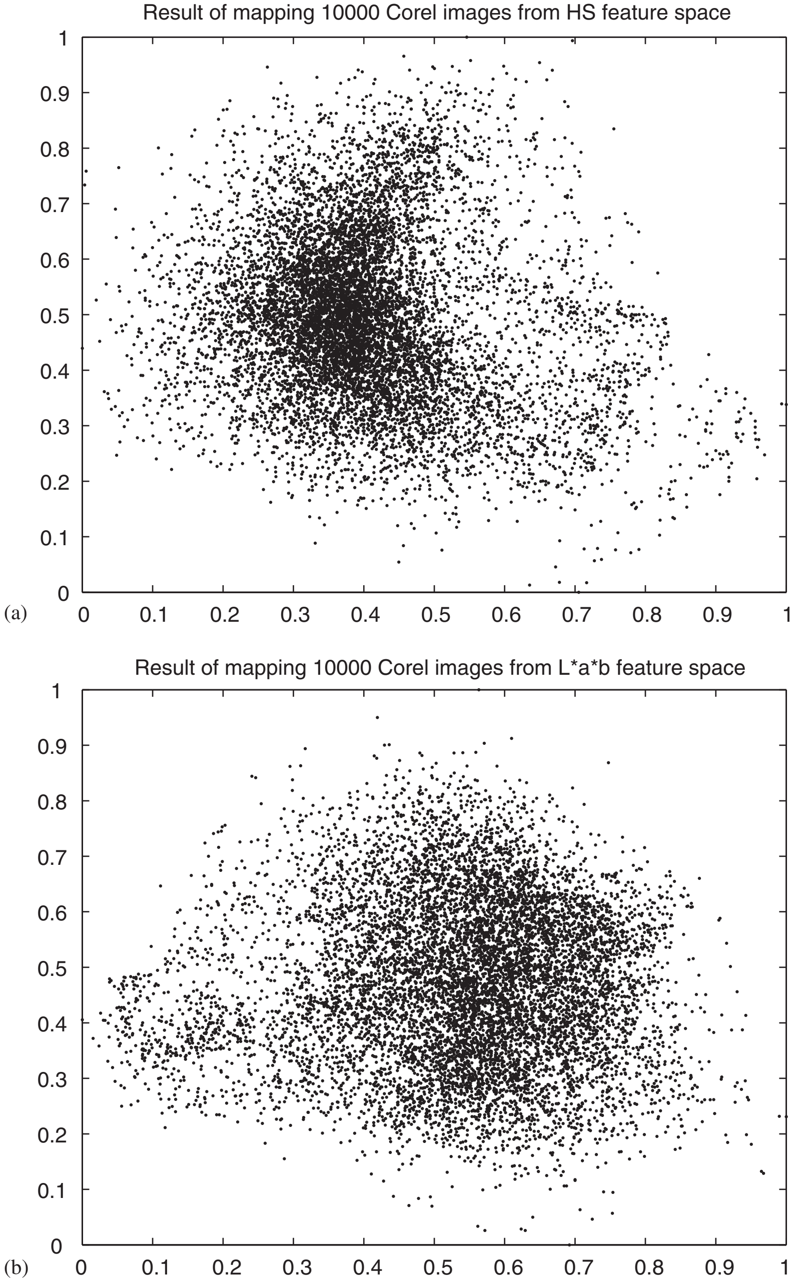


Fig. 9. Result of projecting 10,000 Corel images from 64-dimensional HS feature space and 96-dimensional L^*a^*b feature space to the two-dimensional visualization space.

In selecting the optimal positions, not only the number of displayed images, but also the value of λ_2 can affect the result. With λ_2 equal to 1.0, only structure is preserved. When λ_2 goes to 0, images are spread out loosing much of the structure. An example is shown in

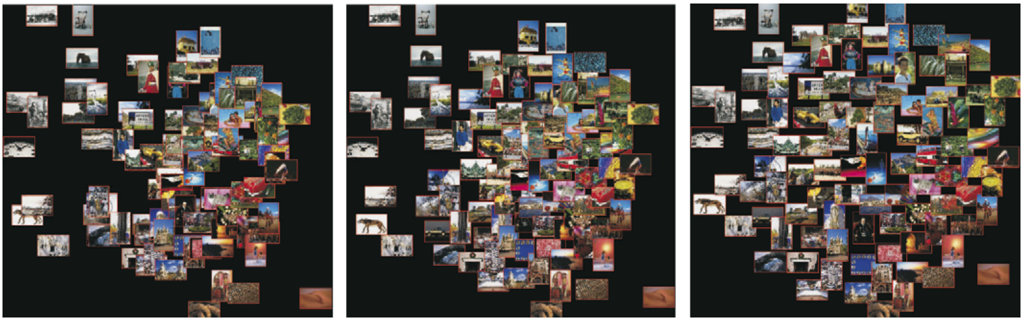


Fig. 10. Example of displaying 100 images with $\lambda_2 = 1$, 0.5, and 0, respectively.

Fig. 10. We have to notice that there is another factor affecting the selection of these parameters, which is size of the display space. We assume the visualization space is equal to the size of a standard computer screen.

We, therefore, experiment with different numbers of images, as well as different values of λ_2 to see the effect of those two parameters on the visibility and the structure preservation requirements. In order to do that, the k-means clustering is applied with $n = 50, 100, 150, 200$. After the clustering, each time a set of n representative images is displayed. The balancing function in Eq. (9) is applied with λ_2 ranging from 0 to 1. We then calculate for the currently displayed set the percentage of images visible for at least t%, with $t = \{25, 50, 75, 100\}$. The cost $\mathscr{C}_S$ is also computed for each case.

Fig. 11 shows the results for different n and t. The figures clearly illustrate the relation between number of images, λ_2, structure preservation and the visibility. With a small number of images, the system easily finds a solution for Eq. (9). For example, in case of 50 images, without any constraint on visibility ($\lambda_2 = 1$), the percentage of images 75% visible is very close to 100%. With $\lambda_2 = 0.5$ all images are visible while structure is well preserved. In contrast, with 200 images, even when $\lambda_2 = 0$ meaning no structure preservation, the total percentage of 50% visible images is not increased much. This is to be expected from the discussion in 4.1. In fact, too few images will increase the browsing and exploration time through the image collection. From the above, selecting 100 images with $\lambda_2 = 0.9$ is a good option.

5.4. Similarity based vs. 2D sequential visualization

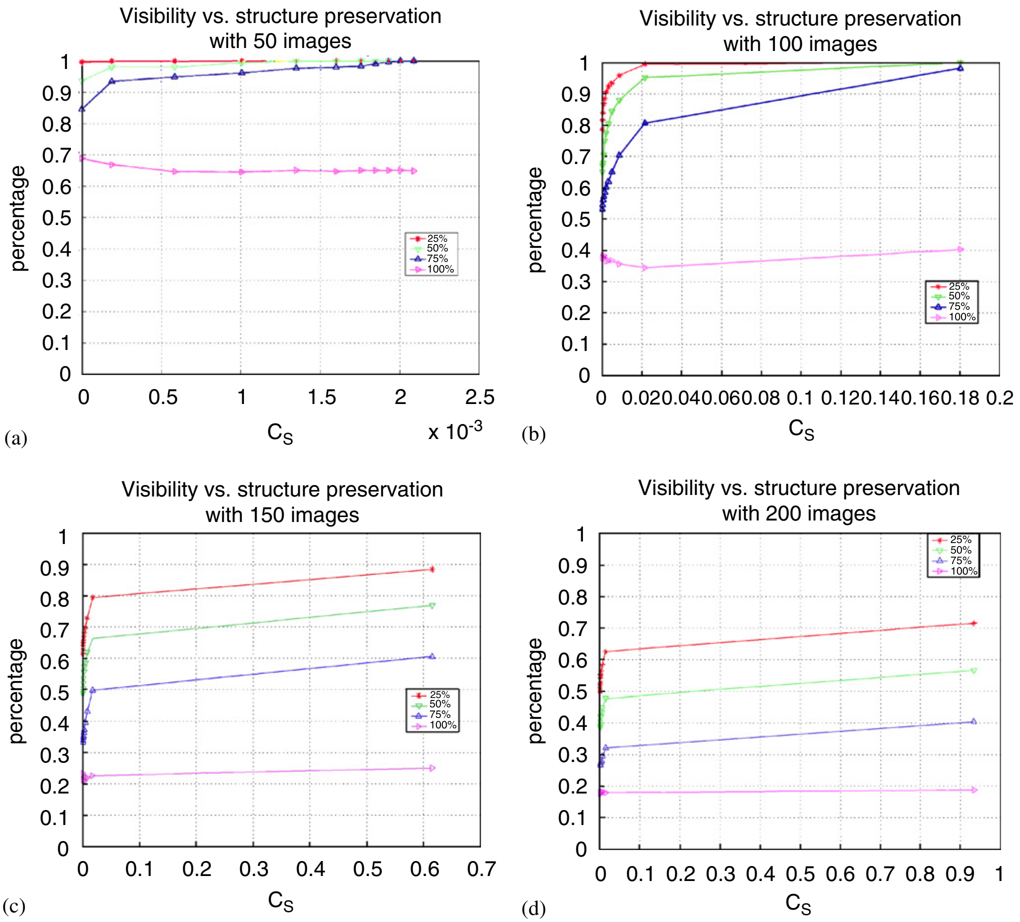
In this section, we compare traditional 2D sequential visualization with our 2D similarity-based visualization. We do so by setting up an explicit scenario of use and then we simulate the user actions.

5.4.1. Scenario setup and evaluation criteria

The scenario we use is full *database annotation*.

Database annotation: *Assigning all images in the database to their corresponding category.*

Manual database annotation is very time consuming and labor expensive, especially when the size of the data is getting larger. Now let us see how this scenario is performed



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

Fig. 11. Examples of visibility vs. structure preservation with different number of images, and different values of λ_2.

in our system. In visualization space, a set of n images, where each image represents one cluster, is displayed to the user. He/she selects an image and then goes inside the corresponding cluster. Images in that cluster are then annotated. One user action *is an interaction of the user to annotate one image a group of images*. We finally obtain the total number of *actions* to annotate the whole database, the so called *annotation effort*.

Annotation effort: *The total number of user actions needed to perform the task.*

In sequential visualization, displayed images are arranged on a grid with no relations between them taken into account. So a user action is needed for each separate image on the screen. This means that annotation effort equals the size of the collection.

In 2D similarity-based visualization, similar images are located close to each other. In the ideal case, all images in a cluster belong to the same category. In practice the cluster contains images from different categories, therefore the user draws a rectangle around each set of images belonging to the same category to annotate them. Fig. 12 shows an illustration of the process.

As we propose a user simulation scheme to implement a quantitative evaluation, we need to mimic the above defined user action. We do so by implementing an algorithm finding the number of rectangles needed to assign all images displayed to the appropriate category. Optimal search for the minimum number of rectangles can be employed. However, in reality, the user often does not draw an optimal number of rectangles to cover all images of interest. Hence, a simpler greedy search appropriately fits the user action. The pseudo code is as follows.

```
Rectangle-search(an image set M)
  For each element m in M
    If (m has not been annotated as positive)
      X = sort(neighbors(m)) on distance to m;
      R = draw_rectangle(m);
      For each element x in X, where category(x) = category(m)
        store(R);
        R = draw_rectangle(R, m); //increase size of the rectangle.
        If R contains y, where category(y) ! = category(m)
            R = store(R);
            break;
        else
            annotate(x);
```

5.4.2. Comparison

Of course the success of similarity-based visualization for annotating groups depends on how well the categories are separated from each other. When elements of a category appear in more clusters, i.e. yielding high entropy, more actions will be needed. The entropy of category *i*th is computed by the allocation of all elements in this category over the clustering [28]:

$$E_i = -\sum_j \frac{\alpha_{ij}}{\alpha_i} \log \frac{\alpha_{ij}}{\alpha_i},$$

where α_i is the number of images in category i, α_{ij} is the number of images in category i which appear in cluster j.

From the results of the previous section, we select $n = 100$, with $\lambda_2 = 0.9$, the average percentage of $t = 75\%$ visible images will be more than 80% and reaches 100% of displayed images visible for 50% and 25%. We use the above clustering result with 100 clusters. As a result of clustering, the sizes of clusters vary, therefore in the display of the cluster's content, if the size of a cluster is larger than 100, the system will show subsets of the cluster containing at most 100 images. Fig. 13 shows the annotation effort decomposed into the different categories.

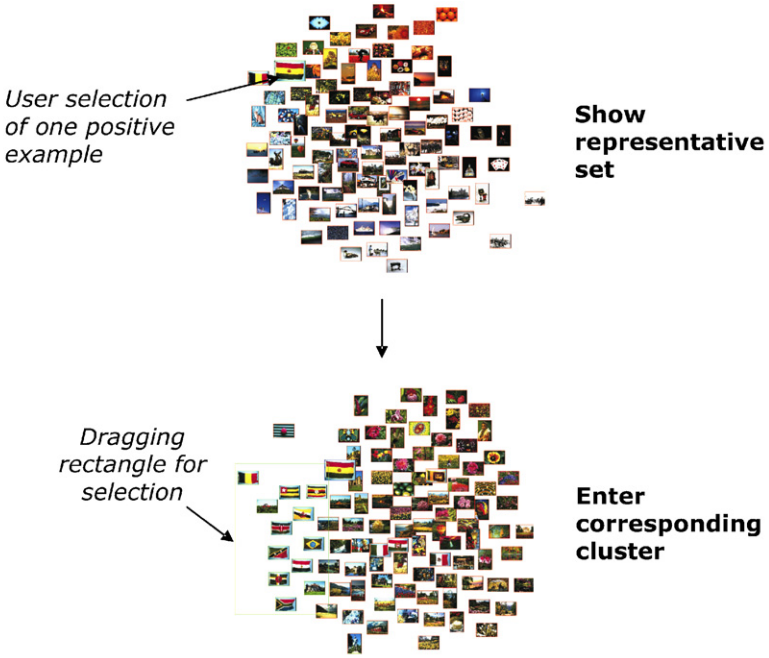

User selection of one positive example
Show representative set
Dragging rectangle for selection
Enter corresponding cluster

Fig. 12. An example to illustrate user actions during annotation process.

We observe that if the categories are reasonably separated i.e. entropy is not too large, the number of actions needed is reduced significantly. For example, in the given collection, some categories containing black and white images can be well distinguished from other colorful categories. Using color histogram as a feature, they will be placed close to each other, less actions will be needed in this case. The annotation effort for those categories reduces from 100 to only six actions. For other categories, the categorization is semantic and cannot be easily distinguished based on the color histogram only. The entropies of these categories are high making the mixture of images among different categories on the display. Therefore, more actions will be required, but in general always less than the number of actions in the baseline. On average, our system reduces the annotation effort by 20–94%. We can conclude that more complex implementation pays off, we can significantly reduce the annotation effort.

6. Conclusion

Visualization is an essential tool for exploring visual collections. To build a good visualization system, a set of related requirements should be taken into account.

In this paper, we established three general requirements for similarity-based visualization systems: overview, visibility, and structure preservation. These requirements provide the user an optimal way to interactively access a large image collection. The overview gives the user the overall look of the whole collection, and guides the user to the right search direction. The structure preservation ensures that original relations between images in the collection are kept in the visualization. Visibility is essential for interaction between the

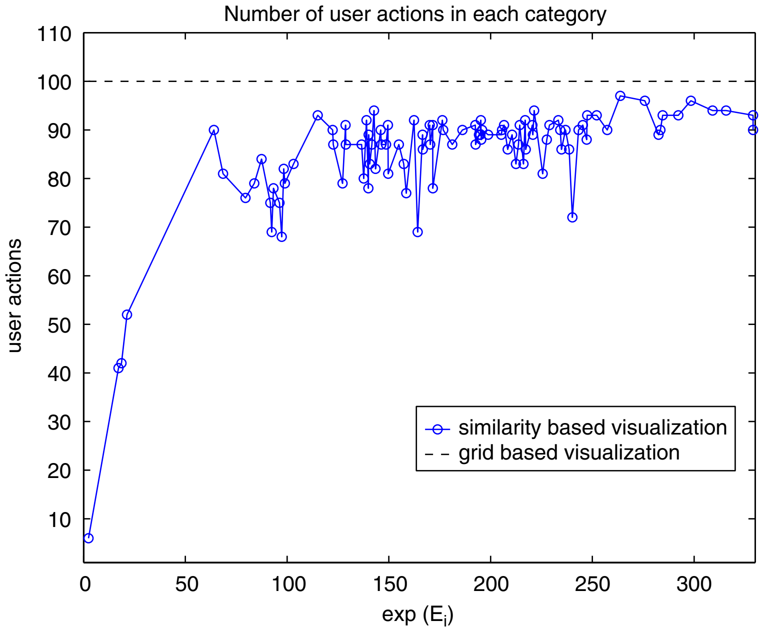


Fig. 13. Counting the number of user actions for annotation by simulation using grid-based and similarity-based visualization. The different categories are ordered by their entropy. For better viewing the result, we use an exponential function on the x-axis. The y-axis shows the annotation efforts. The baseline is the dashed line representing the actions in grid-based visualization. As mentioned in previous sections, because of the sequential display, the number of actions equals the size of the categories. In this case, there are 100 images in each category. So, annotation effort is 100 user actions. The solid line shows the result for the proposed visualization interface.

user and images displayed. As these requirements are not independent, compromises among them are needed. We proposed novel balancing cost functions and algorithms used to define the relative importance of these requirements to the overall visualization goal. Fig. 14 shows a screendump of our visualization system that takes into account the balancing between the three requirements.

Using a rather large data set of 10,000 images, we conducted experiments to evaluate the proposed framework. In a first experiment, we compared the performance of different projection methods, namely LLE, ISOMAP, SNE, ISOSNE, and ISOLLE. ISOSNE is the best option when computation time is not the limiting factor. However, for interactive performance, LLE or ISOLLE should be selected. To evaluate the interactive system as a whole, instead of doing user-based evaluation which is quite expensive and not easy to repeat, an objective user model is built. In particular we defined a *database annotation* scenario. The proposed visualization scheme reduces the total annotation effort significantly ranging from small reduction to 16 times lower effort depending on the separation of the different categories.

Not only in the given scenario, but for different tasks such as target search, category search, category annotation, or example-based search, one could apply the three requirements and balancing functions to build an optimal system by simply changing the scenario.

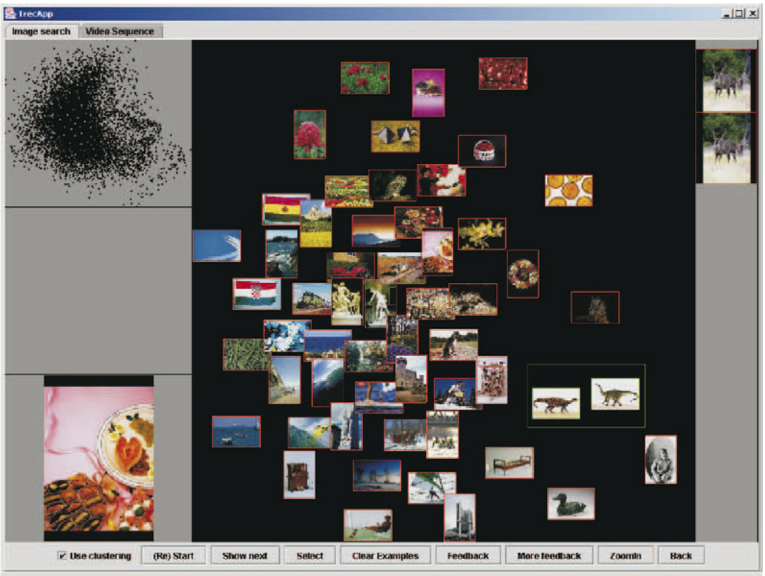


Fig. 14. A screendump of the visualization system. The upper-left corner shows the whole collection as a point set with red (or grey in non-color print) points representing the currently displayed set. The bottom-left corner shows an enlarged version of the currently selected thumbnail. The main screen shows the representative set as images. The drawn rectangle is on the screen an example of a user selection of a group of images by dragging the mouse.

Acknowledgments

The authors would like to thank J.J. Verbeek for his valuable discussions and suggestions. The work is within the ImIk project (Interactive disclosure of Multimedia Information and Knowledge) sponsored by IOP.

References

[1] A. Smeulders, M. Worring, S. Santini, A. Gupta, R. Jain, Content-based image retrieval at the end of early years, IEEE Transactions on Pattern Analysis and Machine Intelligence 22 (12) (2000) 1349–1380.

[2] C. Chen, G. Gagaudakis, P. Rosin, Content-based image visualization, IEEE International Conference on Information Visualization, 2000, pp. 13–18.

[3] O. de Bruijn, R. Spence, Rapid serial visual presentation: a space-time trade-off in information presentation, Proceedings of Advanced Visual Interfaces, 2000, pp. 189–192.

[4] Y. Rubner, C. Tomasi, L.J. Guibas, The earth mover's distance as a metric for image retrieval, International Journal of Computer Vision 40 (2) (2000) 99–121.

[5] C. Ahlberg, B. Schneiderman, Visual information seeking: tight coupling of dynamic query filters with starfield displays, in: CHI Conference: Human Factors in Computing Systems, 1994.

[6] G.P. Nguyen, M. Worring, Similarity based visualization of image collections, in: The Seventh International Workshop on Audio–Visual Content and Information Visualization in Digital Libraries, 2005, 22pp.

[7] R.C. Veltkamp, M. Tanase, A survey of content-based image retrieval systems, in: O. Marques, B. Furht (Eds.), Content-based Image and Video Retrieval, Kluwer Academic Publishers, Dordrecht, 2002, pp. 47–101.

[8] J.Z. Wang, J. Li, G. Wiederhold, Simplicity: semantics-sensitive integrated matching for picture libraries, IEEE Transactions on Pattern Analysis and Machine Intelligence 23 (9) (2001) 947–963.

[9] G. Caene, G. Frederix, A.A.M. Kuijk, E.J. Pauwels, B.A.M. Schouten, Show me what you mean! pariss: a cbir-interface that learns by example, Proceedings of the Fourth International Conference on Advances in Visual Information, 2000, pp. 257–268.

[10] L. Cinque, S. Levialdi, A. Malizia, K.A. Olsen, A multidimensional image browser, Journal of Visual Language and Computing 9 (1998) 103–117.

[11] W. Leeuw, R. Liere, Visualization of multidimensional data using structure preserving projection methods, Data Visualization: The State of the Art Editors, 2003, pp. 213–224.

[12] M. Nakazato, T.S. Huang, 3D MARS: immersive virtual reality for content-based image retrieval, Proceedings of International Conference on Multimedia and Expo, 2001, p. 12.

[13] S. Santini, A. Gupta, R. Jain, Emergent semantics through interaction in image databases, IEEE Transactions of Knowledge and Data Engineering 13 (3) (2001) 337–351.

[14] C. Vertan, M. Ciuc, C. Fernandez-Maloigne, V. Buzuloiu, Browsing image databases by 2d image similarity scatter plots: updates to the iris system, Proceedings of International Conference Communications, 2002, pp. 397–402.

[15] B. Moghaddam, Q. Tian, N. Lesh, C. Shen, T.S. Huang, Visualization and user-modeling for browsing personal photo libraries, International Journal of Computer Vision 56 (1–2) (2004) 109–130.

[16] K. Rodden, W. Basalaj, D. Sinclair, K. Wood, Does organisation by similarity assist image browsing, ACM Conference on Human Factors in Computing Systems, 2001, pp. 190–197.

[17] M. Ivory, M. Hearst, The state of the art in automating usability evaluation of user interfaces, ACM Computing Surveys 33 (4) (2001) 470–516.

[18] K. Rodden, W. Basalaj, D. Sinclair, K. Wood, Evaluating a visualization of image similarity as a tool for image browsing, Proceedings of the 1999 IEEE Symposium on Information Visualization, 1999, p. 36.

[19] E. Pekalska, R.P.W. Duin, P. Paclik, Prototype selection for dissimilarity-based classifiers, Pattern Recognition 39 (2) (2006) 189–208.

[20] G. Peters, B. Zitova, C. Malsburg, How to measure the pose robustness of object views, Image and Vision Computing 20 (4) (2002) 249–256.

[21] R. Pless, Image spaces and video trajectories: using isomap to explore video sequences, IEEE International Conference on Computer Vision, 2003, pp. 1433–1440.

[22] J.B. Tenenbaum, V.D. Silva, J.C. Langford, A global geometric framework for nonlinear dimensionality reduction, Science 290 (5500) 2000, 2319–2322.

[23] S. Roweis, L. Saul, Nonlinear dimensionality reduction by locally linear embedding, Science 290 (5500) 2000, 2323–2326.

[24] G. Hinton, S. Roweis, Stochastic neighbor embedding, Neural Information Processing Systems 15 (2002) 833–840.

[25] A.K. Jain, M.N. Murty, P.J. Flynn, Data clustering: a review, ACM Computing Surveys 31 (3) (1999) 264–323.

[26] U. Brandes, M. Gaertler, D. Wagner, Experiments on graph clustering algorithms, Proceedings of the 11th European Symposium Algorithms, Lecture Notes in Computer Science, vol. 2832, Springer, Berlin, 2003, pp. 568–579.

[27] R.C. Dubes, How many clusters are best?—an experiment, Pattern Recognition 20 (6) (1987) 645–663.

[28] J. He, A.H. Tan, C.L. Tan, S.Y. Sung, On quantitative evaluation of clustering systems, Clustering and Information Retrieval (2003) 105–134.

[29] Y. Zhao, G. Karypis, Criterion functions for document clustering: experiments and analysis, Technical Report tr 01-40, Department of Computer Science, University of Minnesota, Minneapolis, MN, 2001.