Interactive Search by Direct Manipulation of Dissimilarity Space

Giang P. Nguyen, Marcel Worring, *Member, IEEE*, and Arnold W. M. Smeulders, *Senior Member, IEEE*

***Abstract*—In this paper, we argue to learn dissimilarity for interactive search in content based image retrieval. In literature, dissimilarity is often learned via the feature space by feature selection, feature weighting or by adjusting the parameters of a function of the features. Other than existing techniques, we use feedback to adjust the dissimilarity space independent of feature space. This has the great advantage that it manipulates dissimilarity directly. To create a dissimilarity space, we use the method proposed by Pekalska and Duin, selecting a set of images called prototypes and computing distances to those prototypes for all images in the collection. After the user gives feedback, we apply active learning with a one-class support vector machine to decide the movement of images such that relevant images stay close together while irrelevant ones are pushed away (the work of Guo *et al.*). The dissimilarity space is then adjusted accordingly. Results on a Corel dataset of 10000 images and a TrecVid collection of 43907 keyframes show that our proposed approach is not only intuitive, it also significantly improves the retrieval performance.**

***Index Terms*—Active learning, dissimilarity learning, interactive image search, visualization.**

I. Introduction

INTERACTIVE search tasks in content-based image retrieval (CBIR) are classified into three types: association search, target search, and category search [27]. In search by association, the user starts with no specific aim other than interesting findings. Target search aims at finding one specific image. And category search ideally finds all images belonging to a specific class or complying to a given information need. In any of the three tasks, during the search process, the system aims at finding relevant images while discarding irrelevant ones. To do so, a (dis)similarity measure is needed to compare images. As dissimilarity is the inverse of similarity they can be mapped easily and in the following, we will just use the most appropriate term.

Dissimilarity between images strongly depends on the context of the search. Prior to the interaction, the system's definition of dissimilarity is based on the objective image content, whereas during the interaction the user judges similarity based on a subjective interpretation of the semantic content. This contrast is known as the semantic gap [27]. Imagine we have two sets of pictures, one of "dogs" and the other of "birds". In a search task looking for images of the "animal" category, images in these two sets are to be taken as members of the same class. However, if the task is searching images of "pets", the pictures with "dogs" are relevant, but the pictures with "birds" may not be relevant as it depends on what type of bird appears in the pictures. Only the user knows exactly what she is searching for and the system needs to learn the dissimilarity based on the user's relevance feedback.

In literature, many different methods have been developed to learn dissimilarity from relevance feedback. For an overview, see [22] and [34]. When feedback is given, dissimilarity is commonly learned via feature space [1], [14], [12]. When a small set of specific features is used, it works for a narrow domain only. In contrast, a large set of generic features provides the possibility to find dissimilarities among images of any kind. However, such a large set of features leads to a high computational load, especially when the dimension of the feature space goes to thousands of features. And what is more, a large set needs a large set of training examples. For interactive search, immediate response is important, hence for interactive search in broad domains learning dissimilarity based on the feature space is not ideal.

Let us reconsider the above example. It is difficult to define effective features which would assign "dogs" and "birds" to the "animal" group. However, if the user points out that target images are similar to an example picture of a "dog" and an example of a "bird", we might group them based on the observation that they are close to either one of them. Defining concepts on the basis of examples is far more intuitive for the user than defining it in terms of the features of the images. In our approach, rather than considering the feature space, we will focus on the *dissimilarity space*, where images are represented by their relations to other images.

A similar approach has been applied in [2]. In this reference, the authors also consider dissimilarity space as a substitute for the feature space. They create dissimilarity spaces following the technique of Duin and Pekalska [7], [18] for different classes of features such as color or texture. They first select a set of images, named prototypes. The dissimilarity space is created such that images are represented by their relative dissimilarities to the prototypes. They then explore the optimal way of fusing these spaces over the feature spaces. Although the retrieval process is done on the dissimilarity spaces, these spaces are not updated, hence the interactive learning of dissimilarity still strongly depends on the initial feature spaces. As indicated, this makes it difficult to learn the semantic target classes.

Manuscript received July 24, 2006; revised June 11, 2007. The work was supported in part by IOP Man-Machine Interaction (MMI) within the ImIk project (Interactive disclosure of Multimedia Information and Knowledge) and in part by the BSIK project MultimediaN. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Yong Rui.

The authors are with the Intelligent Systems Lab Amsterdam, Informatics Institute, University of Amsterdam, 1098 SJ Amsterdam, The Netherlands (e-mail: giangnp@science.uva.nl; worring@science.uva.nl; smeulders@science.uva.nl).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TMM.2007.906586

To let the user define similarity through interaction, a visualization interface, we call the *manipulation space*, is necessary. In such an interface, the user is able to manipulate and select images to express his similarity definition via relevance feedback. In literature, there are different approaches for displaying and browsing an image collection [3], [26], [6], [11]. The traditional approach for visualizing image collections in search systems is the "page of thumbnails". This visualization can show the ranking of a set of images, but not the relations among them. More advanced is [26] where the feature space is projected to two dimensions and the interface allows users to give feedback by changing the relative position of images. In this method relations are only implicitly used in the visualization. Relations are explicit in [3] and [11], where the existence of discrete relations between images are presented as a graph. The method in [16] does not use discrete relations, but visualizes the similarity among the images. More methods along this line can be found in the reference. The user may interact in the space by labelling images as relevant and/or irrelevant, by giving dissimilarity scores, or by moving relevant images close to one another. In this paper, we consider the labelling of relevant and irrelevant images only. By doing this, the user gives direct feedback to adjust the dissimilarity among images. This visualization method is best suited for our approach, as the visualization directly shows the current interpretation of dissimilarity the system has. Interaction is very intuitive as the user interacts with dissimilarity directly, which is not possible in feature based approaches.

In this paper, we integrate the information available from interaction as deep into the definition of (dis)similarity as one can. We do not only learn class membership from the user's interaction, but also iteratively update the dissimilarity space itself using the technique in [9] with a small modification. This means that when the user changes the layout of the image set displayed in the manipulation space, the layout of images in the dissimilarity space is also adjusted to fit the feedback. Images are presented to the user with their relations reflecting the system's definition of dissimilarity. This definition is obtained from the dissimilarity space. The user either agrees or disagrees with the current relations. If not, he can give feedback to show his opinion on how the relations should be. The changes on the manipulation space will be directly mapped to update the dissimilarity space.

The paper is organized as follows. In Section II, we will describe in more detail existing research in learning dissimilarity. Next, in Section III, we present our approach to learn the dissimilarity through updating the dissimilarity in manipulation space. Results of the system for two different image collections are shown in Section IV. Finally, conclusions are presented in Section V.

II. Background and Related Work

In this section, we introduce notation and give an overview of the literature on learning dissimilarity.

A. Basic Notation

Given a collection of images $\mathcal{I} = \{I_1, I_2, \ldots, I_n\}$, the r-dimensional feature space $\mathcal{F}$ is defined in which each image $I_i \in \mathcal{I}$ is represented by a feature vector $\vec{F_i}$ of length r. Dissimilarities $S(.)$ are derived from the feature vectors. They are computed between every pair of images I_i and I_j and stored in a matrix $\mathcal{S} = [S(I_i, I_j)]_{i=1,n;j=1,n}$. Furthermore, let $\vec{W} = (w_1, w_2, \ldots, w_r)$ denote a set of r values weighting the dimensions in $\mathcal{F}$. Finally, let $\vec{\zeta}$ denote a set of parameters steering the dissimilarity function.

When the user is interacting with the system he has a goal which can be defined as a set of desired images $\mathcal{I}_+ \subset \mathcal{I}$ to be found. Given this goal, learning of dissimilarity can be viewed as an iterative process where the system learns to identify the set $\mathcal{I}_+$ from feedback given by the user.

B. Methods for Learning Dissimilarity

In most existing methods, learning is done via feature space either by feature selection [1], [14], feature weighing [12], [33], or using a parameterized function of the features [24], [9].

In general, for existing methods, the learning of dissimilarity in iteration $t+1$ from analyzing the feedback of the user in iteration t can be formulated as

$$S^{t+1}(I_i, I_j) = f(\vec{F_i}, \vec{F_j} | \vec{W}^t, \vec{\zeta}^t) \quad (1)$$

where $\vec{W}^0$ and $\vec{\zeta}^0$ are start values.

In general, not all features are equally important. Hence, in the feature weighting approach weights are set for each feature. Feature selection is a special case of feature weighting, where the weights of the eliminated features are set to 0. As the update changes $\vec{W}$ only, (1) is rewritten as

$$S^{t+1}(I_i, I_j) = f(\vec{F_i}, \vec{F_j} | \vec{W}^t). \quad (2)$$

In [33], the authors concentrate on exploring the distribution of the data set. A subspace of the feature space is found, and a quadratic similarity functions is learnt. From there, the dissimilarity matrix between images is updated. A similar approach is presented in [1], where the authors propose a weighted Minkowski similarity that continuously learns the weight of each feature based on positive and negative examples. The dissimilarity matrix is adjusted on the basis of the new set of weights. In [14], the similarity is recalculated by selecting a subspace. Updating is based on the configuration of images on the screen resulting from the user's manipulation of the position of images on the screen. The system re-estimates the layout of the images, such that the similarity function gets closer to the user's desire. A similar approach but with dynamic selection of the feature subspace is followed in [12]. In this reference, starting with a number of features, a dynamic function is proposed which determines at each step the optimal number of features. From there, the weighted and non-weighted perceptual dynamic function is built based on the Minkowski distance. In [8], the authors propose a system which allows the user to score similarity between given pairs of images. The system then learns the similarity coefficient from the user feedback, predicting the similarity among the images which were not displayed. Within the same school of thought, work has been reported in [4], [10], [25], and [30]. In general, this approach requires a large set of features in order to select a subset best representing the semantic similarity between images. For interactive search, the selection of a large number of features leads to a high computational load, especially with complicated dissimilarity functions. In addition, the need for a large number of examples in learning leads to tedious interaction.

The parameter based approaches form another class of methods. In these approaches, the set of vectors in feature space does not change during learning, but rather a parameterized function of the features is adjusted to fit the user's feedback. Equation (1) is specialized as

$$S^{t+1}(I_i, I_j) = f(\vec{F_i}, \vec{F_j} | \vec{\zeta^t}). \tag{3}$$

For example, in [24], the authors introduce an interface where the user adjusts the similarity between images in the manipulation space by moving them around. New positions of images displayed are used as relevance feedback. Using Fuzzy Feature Contrast and Tversky's similarity measure, the authors define a similarity function where $\vec{\zeta}$ contains around 100 different parameter dimensions. Based on user feedback, the system then adjusts the set of parameters in the dissimilarity function such that the dissimilarity decreases between images which according to the user are close. The large number of parameters requires a large number of training examples and thus substantial user interaction. In [9], given a set of images as query examples, a restricted similarity measure is formulated which recalculates dissimilarity between all images and queries depending on their positions compared to the classification boundary. The boundary is characterized by a parameter set. Given a set of positive and negative examples, SVM and AdaBoost are used to learn a classification boundary. The top ranked images are then labelled as positive or negative examples to repeat the refinement of dissimilarity.

The parameter based approaches do not require a large set of features. However, to effectively learn the dissimilarity matrix, either the features should be well chosen or the system should have a wide range of parameters.

The use of features in learning dissimilarity has a major limitation as it strongly depends on the choice of features. With the lack of efficient features for general search tasks, a new approach that learns dissimilarity with less dependence on features should be considered.

III. Direct Manipulation of Dissimilarity

In this section, we present our approach in learning dissimilarity by updating the dissimilarity space based on user's relevance feedback without going back to the feature space. Using the same notation as in (1) our method can be described as

$$S^{t+1}(I_i, I_j) = f(S^t(I_i, I_j)), \text{ with } S^0(I_i, I_j) = f(\vec{F_i}, \vec{F_j}). \tag{4}$$

An overview of our proposed approach is in Fig. 1. First, a dissimilarity matrix is obtained by comparing feature vectors in the feature space $\mathcal{F}$. A projection from the high dimensional space creates a manipulation space $\mathcal{M}$ (to be discussed in Section III-B). Images are presented in $\mathcal{M}$ to the user for interaction and feedback. A set of images, named the prototype

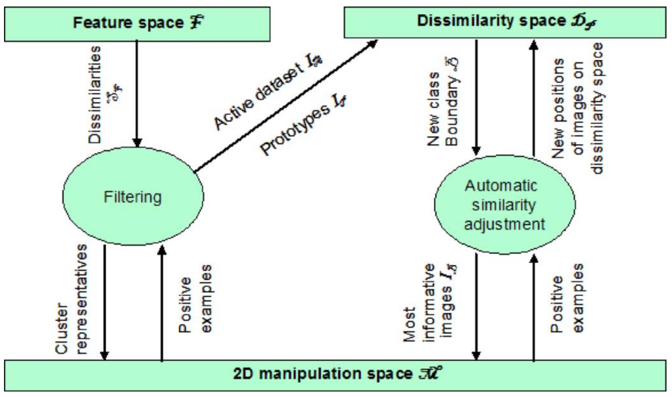


Fig. 1. Schematic overview of the proposed approach.

set $\mathcal{I_P}$, is selected. When sufficient prototypes have been found, a dissimilarity space $\mathcal{D_P}$ is created (Section III-A). This starts the learning process. The manipulation space $\mathcal{M}$ is now a direct projection of $\mathcal{D_P}$. In this learning phase, $\mathcal{D_P}$ is iteratively adjusted using feedback and active learning. The adjustment is presented in Section III-C. In each iteration, a set of most informative images is returned. The user then labels positive images for another round of feedback. The learning phase finishes when the user stops the search.

Clearly the above defines a complex system. Several of the constituent components have been studied before in literature and have their own merit in various search scenarios. However, it is their complex interplay which yields ample opportunities for fully supporting the user in complex search scenarios. In the following sections, we elaborate on the components and their relations.

A. Creation of the Dissimilarity Space

To create a prototype-based dissimilarity space, we employ the method proposed by Pekalska [18]. In the reference, the goal is classification of numeric data. They do not consider user interaction or relevance feedback. We extend the methods to interactive image search.

The first step is to select a set $\mathcal{I_P} \subset \mathcal{I}$ of p prototype images

$$\mathcal{I_P} = \{I_{P_1}, I_{P_2}, \ldots, I_{P_p}\}. \tag{5}$$

The role of the prototypes is to create a dissimilarity space where relevant and irrelevant images are well separated. Hence, careful selection of prototypes is important. The mapping from dissimilarity matrix to dissimilarity space by selection of prototypes is equivalent to choosing a set of columns (or rows) in the dissimilarity matrix. With $\mathcal{D_P}$ denoting the dissimilarity space, and Φ the mapping from a dissimilarity matrix $\mathcal{S}$ to $\mathcal{D_P}$

$$\Phi : \mathcal{S} \overset{\mathcal{I_P}}{\longmapsto} \mathcal{D_P}. \tag{6}$$

This means that for each image I_i, we have a p-dimensional vector

$$\mathcal{D}_i = \left(S(I_i, I_{P_1}), S(I_i, I_{P_2}), \ldots, S(I_i, I_{P_p})\right). \tag{7}$$

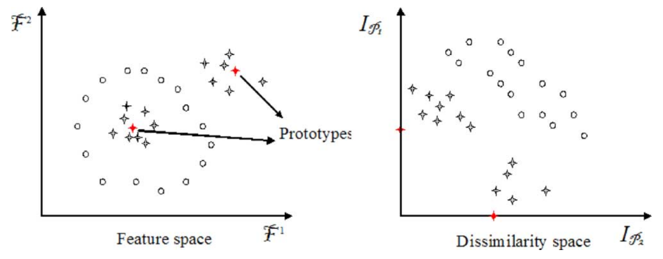


Fig. 2. Example to illustrate the creation of a dissimilarity space. The stars and circles represent the relevant and irrelevant classes, respectively. (a) For simplicity, images are represented in a 2-D feature space $\mathcal{F} = \{\mathcal{F}^1, \mathcal{F}^2\}$, with dissimilarities among them obtained by the unweighed Euclidean distance between feature vectors. Clearly, relevant images are clustered into different area, which make the separation difficult. Two images $\mathcal{P}_1$ and $\mathcal{P}_2$ are selected as prototypes. (b) Distances to $I_{\mathcal{P}_1}$ and $I_{\mathcal{P}_2}$, define the 2-D dissimilarity space. In the dissimilarity space, it becomes much easier to classify the two classes.

Therefore, dissimilarities between all images in $\mathcal{I}$ to $\mathcal{I}_\mathcal{P}$ are represented by a matrix with size $n \times p$. The collection $\mathcal{I}$ then builds up a p-dimensional dissimilarity space $\mathcal{D}_\mathcal{P}$, named *prototype-based dissimilarity space*

$$\mathcal{D}_\mathcal{P} = \begin{bmatrix} S(I_1, I_{P_1}), & S(I_1, I_{P_2}), & \ldots, & S(I_1, I_{P_p}) \\ S(I_2, I_{P_1}), & S(I_2, I_{P_2}), & \ldots, & S(I_2, I_{P_p}) \\ \vdots & & & \vdots \\ S(I_n, I_{P_1}), & S(I_n, I_{P_2}), & \ldots, & S(I_n, I_{P_p}) \end{bmatrix}. \quad (8)$$

An illustration of creating a dissimilarity space is shown in Fig. 2.

In $\mathcal{D}_\mathcal{P}$, the similarity $S_\mathcal{P}(I_i, I_j)$ of two images I_i and I_j is defined by the Euclidean distance between D_i and D_j, where D_i and D_j are p-dimensional dissimilarity vectors of the two images [see (7)].

Prototype selection is an important step in creating a dissimilarity space. In [19], the authors compare different ways of selecting sets of prototypes such as random selection, cluster analysis by K-centers, and clustering by mode seeking. They experimented with different datasets. Conclusions are that a systematic selection of prototypes performs better than a random selection and the K-center approach performs well in general. We follow the same direction, and aim to find a set of prototypes $\mathcal{I}_\mathcal{P}$ such that the mapping Φ preserves the information in the similarity matrix $\mathcal{S}$ as good as possible.

The method in [18] aims for classification, where training set and test set have been defined beforehand. In interactive search, we are not able to select a set of prototypes as we do not know which images the user will search for. In practice, there are cases where the user starts the search with relevant and/or irrelevant images, but in general this set of images is not a good set of prototypes. For CBIR, a strategy is needed for finding a good set of prototypes.

The browsing for prototypes is performed in manipulation space, where the user directly interacts with images. The selection of relevant images as prototypes makes the learning on dissimilarity space simpler [2] (see example on Fig. 2). We therefore focus on images selected as relevant by the user. In particular, the system shows a set of images to the user, who will select relevant images when they appear. Because of the limitation of the display screen, only a small set of images can be displayed at a time. We denote the set of images displayed in iteration t as $\mathcal{I}_D^t$. This set should be carefully chosen as it affects the resulting $\mathcal{I}_\mathcal{P}$, most importantly $\mathcal{I}_D^t$ should give an adequate overview of the whole collection [16]. First, the collection is divided into a set of clusters $\{I_{C_k}\}_{k=1,M}$ using a clustering algorithm. The system selects an image from each cluster, which is called the representative element of that cluster, for display. M is chosen such that the representatives are giving an overview of the collection $\mathcal{I}$, while assuring that the M images still fit the screen. If a relevant image (i.e., an element of $\mathcal{I}_+$) is present in $\mathcal{I}_D$, the user will select that image as a prototype, i.e.,

$$\mathcal{I}_\mathcal{P}^{t+1} = \mathcal{I}_\mathcal{P}^t \cup (\mathcal{I}_D^t \cap \mathcal{I}_+), \qquad \mathcal{I}_\mathcal{P}^0 = \emptyset. \quad (9)$$

Once images currently on display are inspected, the system will choose a new set of representatives to display.

From a practical point of view, the size n of the image collection is usually large. Hence, sequentially visiting all images is a time consuming procedure and should be avoided when there are clusters not containing any relevant image. To speed up the process, we add a filter to the browsing. Rather than visiting all elements in cluster I_{C_i}, the cluster is divided into m subclusters using the same clustering algorithm. The value of m is selected such that

$$m = \left\lceil \frac{\|\mathcal{I}_{C_i}\|}{n^*} \right\rceil \quad (10)$$

with $\|\mathcal{I}_{C_i}\|$ the size of the cluster. For example, if the cluster $\mathcal{I}_{C_i}$ contains 100 elements and n^* is set equal 20, the number of sub-clusters is $m = 5$. Increasing the value of n^* will give a finer clustering and hence requires more browsing time. Coarse clustering with a high value of n^* increases the chance of missing relevant images.

As in the above, centers of the sub-clusters are chosen as the representatives for that cluster, i.e., one of the m sub-centers is sequentially selected to represent the cluster. Therefore, instead of visiting all n images, the user only considers Mm representative images. For each cluster, when m representatives have been visited and no relevant image is found, the cluster has a high probability of being irrelevant to the search. Hence, it is eliminated from the collection. On the contrary, if one of the representatives is relevant, with the expectation that more relevant images could be inside the corresponding cluster, the system keeps the cluster.

When a cluster is removed, there is space for other representative images to go on display. The unexplored part of the collection $\mathcal{I}_U$ contains images which are not in the removed clusters $\mathcal{I}_R$ and not in the kept clusters $\mathcal{I}_K$

$$\mathcal{I}_U^{t+1} = \mathcal{I}_U^t \backslash (\mathcal{I}_K^t \cup \mathcal{I}_R^t), \qquad \mathcal{I}_U^0 = \mathcal{I}. \quad (11)$$

In each iteration, the system needs to cluster $\mathcal{I}_U^t$. This step cannot be done offline as it depends on the user actions. Hence, a fast and computationally inexpensive clustering algorithm is used namely competitive learning [23]. With new clusters available, the browsing is continued until a predefined number of prototypes has been found, denoted by iteration ∞. It should be noted here that the competitive learning has a random initialization step and hence the prototypes may vary when different starting points are chosen. In practice, we expect the difference in their utility to cover the space limited.

At this point, we have found a set of prototypes $I_{\mathcal{P}}^{\infty}$ and effectively reduced the collection to an active set $\mathcal{I}_A$

$$\mathcal{I}_A = \mathcal{I} \backslash \mathcal{I}_R^{\infty} \tag{12}$$

with increased chance of finding relevant images in a later stage. For notational convenience the ∞ is dropped in the following sections.

B. The Manipulation Space

To provide relevance feedback, the 2-D manipulation space $\mathcal{M}$ is needed in which the user interacts with the images. A projection Ψ from the high dimensional space, being it feature space or dissimilarity space, to two dimensions is needed. Similarity-based visualization, [14], [21], [16], [20] works on dissimilarity space directly:

$$\Psi : \mathbb{S} \mapsto \mathcal{M}. \tag{13}$$

With $\mathbb{S}$ a matrix containing dissimilarities, which can be either the original dissimilarity space or the prototype-based dissimilarity space. In similarity-based visualization, the position of images in $\mathcal{M}$ is chosen such that distances $\mathcal{S}_{\mathcal{M}}$ reflect, as faithfully as possible, the dissimilarities in dissimilarity space $\mathbb{S}$. This makes these methods well suited for our task.

In [16], we have compared five different projection techniques namely Isomap (Isometric mapping), Stochastic Neighbor Embedding (SNE), Local Linear Embedding (LLE), Multi-Dimensional scaling (MDS), and a new method ISOSNE which replaces the MDS step used in Isomap by SNE. The projection method giving the best performance in terms of preserving original distances is ISOSNE, so this is used as Ψ in this paper.

ISOSNE contains two main steps (see [16] for more details). A k nearest neighbor graph in $\mathbb{S}$ is created first. Distances between elements are then defined as the distance along the shortest path in the graph which connects them. To find a proper value for k, we have tested with many different values using a number of different artificial datasets. An appropriate value for k turned out to be ten neighbors. The second step is to project the thus obtained distances to the 2-D manipulation space. To preserve the distances, the algorithm optimizes a cost function $\mathbf{C}$ measuring the difference between the probabilities of elements being at a certain distance in $\mathcal{M}$ and the equivalent probabilities in the original space, denoted as $P^{\mathcal{M}}$ and $P^{\mathcal{O}}$, respectively. Based on Kullback–Leibler distance, $\mathbf{C}$ is computed as

$$\mathbf{C} = \sum_i \sum_j P_{ij}^{\mathcal{O}} \log \frac{P_{ij}^{\mathcal{O}}}{P_{ij}^{\mathcal{M}}} \tag{14}$$

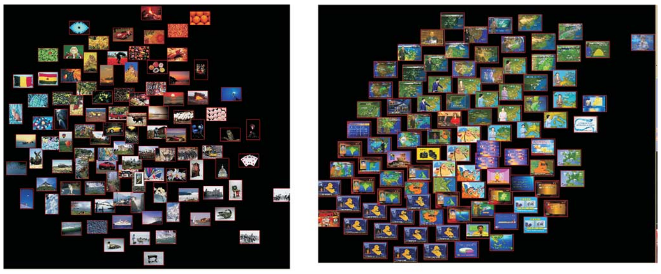

(a) (b)

Fig. 3. Two examples of similarity-based visualization of images in the manipulation space. The layout of images is such that similar images are close. This allows for efficient interaction as images of the same class are likely to be grouped on the screen and can be selected by one user interaction.

where the probability distributions are calculated as follows:

$$P_{ij}^{(.)} = \frac{\exp(-S_{(.)}^2(I_i, I_j))}{\sum\limits_{l \neq i} \exp(-S_{(.)}^2(I_i, I_l))} \tag{15}$$

with $S_{(.)}$ denoting similarity in the high-dimensional original space, or a Euclidean distance in 2-D between image I_i and I_j in $\mathcal{M}$.

To find the optimal placement of images in manipulation space, they are first initialized at random positions. These positions are then adjusted using gradient descent to reduce the cost function $\mathbf{C}$. When $\mathbf{C}$ is optimized, the distances are optimally preserved, but images might overlap one another, which makes interaction difficult. To reduce the overlap between images, we take the result obtained as the starting point for a subsequent optimization, balancing preservation of the original distances and the visibility of images. Fig. 3 shows two examples of displaying images in manipulation space with similarity preservation and overlap reduction.

C. Automatic Adjustment of Dissimilarity Space

At this point, the initial dissimilarity space is in place. The user has selected the set of prototypes $\mathcal{I}_{\mathcal{P}}$ treated as query examples to start the search process. The search task is now to find other relevant images using these examples.

Different learning strategies can be employed [34]. We select active learning with support vector machines (SVMs) for its capability of boosting retrieval results [31], [34], [17]. As in interactive search, there is an unbalance between the size of the category searched for and the size of the collection, we follow [13], [5], and [15] and use one-class SVM.

The prototypes I_{P_i} are used as positive examples. The one-class SVM defines a boundary $\mathcal{B}$ covering as much as possible the positive examples. The distance $d_{\mathcal{B}}(.)$ to the boundary is taken as a measure of the probability that an images belongs, or does not belong, to the search goal. Let us define:

- $\mathcal{B}_+$ the set of images inside $\mathcal{B}$ predicted to be relevant to the search.
- $\mathcal{B}_-$ the set of images outside $\mathcal{B}$ predicted to be irrelevant to the search.
- $\mathcal{I}_{\mathcal{B}}$ the set of images closest to $\mathcal{B}$ according to distance function $d_{\mathcal{B}}(.)$.

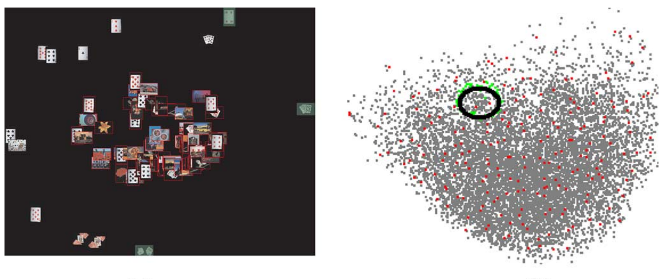

(a) (b)

Fig. 4. (a) Example of images closest to the border, aiming at solving the search task "images in the Card category". (b) View of the full manipulation space, with an indication of where the region displayed in (a) is located.

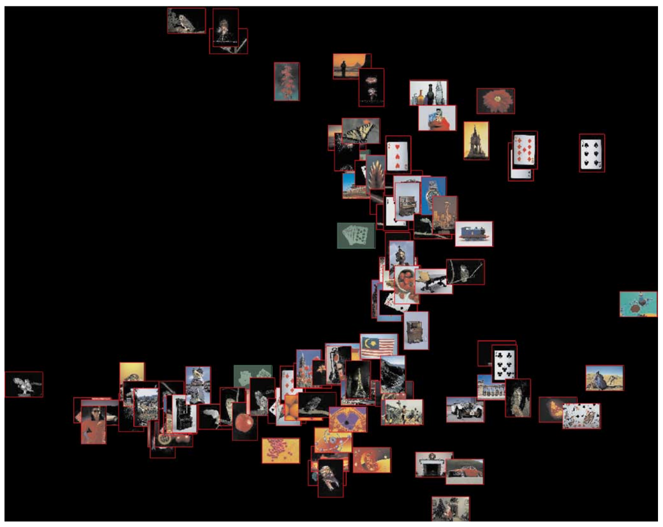


Fig. 5. Example of images inside the boundary, again looking for "images in the Card category".

In the next iterations, to improve the search, the system aims at refining $\mathcal{B}$. The refinement is such that it eliminates irrelevant images from $\mathcal{B}_+$ and adds new irrelevant images to $\mathcal{B}_-$. To do so, the images in $\mathcal{I}_\mathcal{B}$ are chosen as display set I_D^t as they are the ones for which classification is most uncertain. Feedback on those yields the most information for improving the classification boundary. This is known as the "close-to-boundary" feedback approach [31], [17]. Fig. 4 shows an example of images closest to the boundary, and Fig. 5 shows images in $\mathcal{B}_+$.

The user will label relevant images if they exist. Unlabelled images are treated as irrelevant and removed from the collection. Or more formally defined

$$I_{\mathcal{B}_+^t} = I_D^t \cap I_+ \tag{16}$$

$$I_{\mathcal{B}_-^t} = I_D^t \backslash I_{\mathcal{B}_+^t}. \tag{17}$$

Based on the feedback given, the SVM is recomputed on the new set of positive examples to update the boundary

$$\mathcal{B}_+^{t+1} = (\mathcal{B}_+^t \cup I_{\mathcal{B}_+^t}) \backslash I_{\mathcal{B}_-^t}. \tag{18}$$

The process is repeated until the user stops the search.

The above was applied to a fixed $\mathcal{D}_\mathcal{P}$, the result of the projection of the dissimilarity matrix $\mathcal{S}_\mathcal{F}$ obtained in the feature space

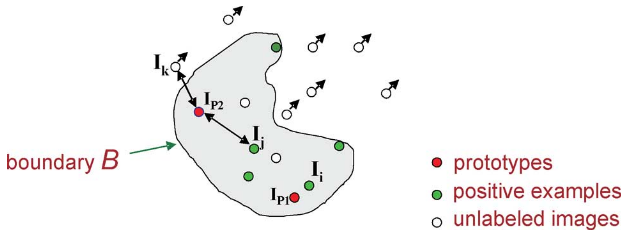


Fig. 6. Illustration of similarity update. Assume a dissimilarity space created from 2 prototypes I_{P_1}, I_{P_2}. Let image I_j be less similar to $\mathcal{I}_\mathcal{P}$ when compared to image I_i. After updating as indicated, image I_i will be more similar to $\mathcal{I}_\mathcal{P}$ than I_j. When simply using distance to the boundary, this is not the case. Images $I_k \notin \mathcal{B}_+$ will be pushed away from I_{P_1} and I_{P_2}.

$\mathcal{F}$ [(6)]. When more feedback from the user comes in, the similarity itself might also have to be adjusted to better reflect the user's target similarity. To do so, we adapt the update method from [9].

As indicated, $\mathcal{B}$ defines the currently predicted class boundary. For the dissimilarity update we make the assumption that the prediction is correct. For all images $I_i \in \mathcal{B}_+$ the representation $\mathcal{D}_\mathcal{P}(I_i)$ is kept constant. In contrast, for $I_i \in \mathcal{B}_-$, the representation is altered. In the ideal case where $\mathcal{B}$ perfectly covers all relevant images, all other irrelevant images should be pushed away from the boundary. Of course, in practice, relevant images can be misclassified. Pushing these images far away from the boundary makes it difficult to retrieve them later. Hence, as the likelihood of being relevant depends on the distance to the boundary $d_\mathcal{B}(.)$, we require that

$$\forall I_i, I_j \in \mathcal{B}_- : \text{if } d_\mathcal{B}(I_i) < d_\mathcal{B}(I_j) \Rightarrow \mathcal{S}_\mathcal{P}(I_i, I_P) < \mathcal{S}_\mathcal{P}(I_j, I_P) \tag{19}$$

and use this in the gradual change of the dissimilarity space based on the user feedback.

The method in [9] (see Fig. 6) satisfies the above constraints by using the following update function:

$$\mathcal{S_P}^{t+1}(I_i, I_P) = \begin{cases} \mathcal{S_P}^t(I_i, I_P), & \text{if } I_i \in \mathcal{B}_+, \\ \max\limits_{I_j \in \mathcal{B}_+} \mathcal{S_P}^t(I_j, I_P) + d_\mathcal{B}(I_i), & \text{otherwise.} \end{cases} \tag{20}$$

Hence, after learning with SVM and based on the new positions of images with respect to the boundary, their distances to the prototypes are changed. This leads to the adjustment of the dissimilarity space. These changes assure that irrelevant images will be pushed away, while the system keeps relevant images in the vicinity of the prototypes. In the next iteration the new similarity helps in better classification. As in each interaction the dissimilarity space adjusts itself without going back to feature space, the further the learning process, the less the feature space influences the characteristics of the dissimilarity space.

IV. Experiments

A. Setup

We now present experiments to show the performance of our proposed approach. In the introduction, three difference search tasks are listed. Target search can be considered as a special case of category search, where the size of the search category equals

1. In that case, our approach can not be applied as there it is unnecessary to find positive examples. Associative search is hard to evaluate objectively as the user's goals might fluctuate considerably during the search. We therefore concentrate on category search here.

The first experiment concentrates on the filtering component in the browsing phase of the proposed system. We will evaluate whether adding this component to the system will speed up the search for relevant images.

The second experiment considers the creation of the dissimilarity space. As described in Section III-A, to create $\mathcal{D}_{\mathcal{P}}$ we need to determine the prototype set $\mathcal{I}_{\mathcal{P}}$ and the dissimilarity between images and prototypes $\mathcal{S}(I_i, I_{P_i})$. At the beginning of the search, two spaces are available, namely the feature space $\mathcal{F}$ and the manipulation space $\mathcal{M}$.

To create a dissimilarity space, both spaces can be used. We have the following two options for creating a dissimilarity space:

$$\Phi^1 : \mathcal{S}_{\mathcal{F}} \xmapsto{\mathcal{I}_{\mathcal{P}}} \mathcal{D}_{\mathcal{P}}^r \tag{21}$$

$$\Phi^2 : \mathcal{S}_{\mathcal{M}=\Psi(\mathcal{S}_{\mathcal{F}})} \xmapsto{\mathcal{I}_{\mathcal{P}}} \mathcal{D}_{\mathcal{P}}^2 \tag{22}$$

where $\mathcal{D}_{\mathcal{P}}^r$ is the dissimilarity space based on r-dimensional prototypes, and $\mathcal{D}_{\mathcal{P}}^2$ the dissimilarity space based on 2-D prototypes in manipulation space. This experiment leads to the choice of the proper dissimilarity space $\mathcal{D}_{\mathcal{P}}$.

In the final experiment, the goal is to compare the search performance in the selected dissimilarity space against the feature space. For a fair comparison, the starting points are the same for both approaches.

In the experiments, objective evaluation is employed where all user actions are simulated. To that end, we could use the advanced user actions defined in [16], where groups of images on display are selected with one interaction step. For this paper, however, we restrict ourselves to one-by-one selection of positive and negative examples displayed as this makes the comparison to other methods easier.

We select two different image collections. The first one contains images from a Corel collection. This collection includes a large number of pre-categorized images, where the category can be used as ground truth. We select 100 nonoverlapping categories that are not too abstract. For example, images in the "Canada" category were not selected as images can range from flags to landscapes, and famous people. Each of the categories contains 100 images. In total, the selected set has 10000 images.

The second collection is obtained from the TrecVid 2005 benchmark [28]. This set contains 43 907 images, extracted from news video archives. We take the 29 different categories defined in [32] such as boat, basketball, car, and chair to classify the collection . Other than the Corel collection, an image in this collection may be in more than one category. The number of images in each category varies from tens to thousands.

For these two image collections, we extract the contexture feature set introduced in [32]. This feature set is evaluated as effective in learning the categorization of images. The authors define 15 proto-textures. They learn the probability that an image contains the proto-textures. For the two collections in our experiment, eight different parameter settings are used (spatial scale $\sigma = 1$, $\sigma = 3$ and different region sizes with ratios of 1/2 and 1/6 for the x and y dimensions of the image). For each image, we extract a feature vector containing 15 probabilities for eight parameter values leading to a feature space of 120 dimensions. To obtain the manipulation space, ISOSNE is applied to project the feature space to 2-D space. The Euclidean distance is used for comparing two feature vectors.

For comparison we define a baseline, based on displaying pictures without any dissimilarity (or feature) computation, where the system in each iteration displays a set of n_D randomly chosen images. Relevant images are selected if they are present in the displayed set. The baseline is calculated as the number of relevant images n^{t+1} likely to be found at iteration $t+1$. We have

$$n^{t+1} = n^t + \frac{n_+ - n^t}{n - n^t} * n_D \tag{23}$$

where n is the size of the collection and n_+ is the total number of relevant images.

For the Corel collection, we have $n = 10000$, $n_D = 100$, and the value $n_+ = 100$ for each category. At the first iteration, the user will find one relevant image out of hundred on average. For the TrecVid2005 collection we average over all categories to obtain the baseline.

As the goal in category search is to find as many relevant images as possible, recall is the most important measure. Therefore, we report recall values R based on the top-ranked 100 images $\mathcal{I}^{100}$

$$R = \frac{\|\mathcal{I}^{100} \bigcap \mathcal{I}_+\|}{\|\mathcal{I}_+\|}. \tag{24}$$

where $\|.\|$ denotes the size of a set. From there, we calculate the relative improvement measuring the improvement of a method over the baseline. Let a method X, in iteration t, yield a recall value R_X^t, the baseline at the same iteration returns a recall R_B^t. Thus, the relative improvement is

$$\phi^t(X, B) = 100 \frac{R_X^t - R_B^t}{R_B^t}. \tag{25}$$

In the following experiments, we perform the search for all the queries, and finally average the results.

B. Experiments on Prototype Selection

To see the efficiency of the filtering, we test with the two given collections above. We report at each iteration the number of images removed, number of images kept, and the number of relevant images falling into removed clusters. Finally, we average results over all categories, which is 100 for the Corel, and 29 for the TrecVid. The comparisons are between browsing through the collection with and without the filtering component. Selection of representative samples for a cluster by random selection is also examined. For a fair comparison, we average 10 different runs for the random approach.

TABLE I
RESULTS FOR BROWSING THE COREL COLLECTION

	iterations	elements kept	relevant missed
no filtering	184	100%	0%
random subcluster	20	60%	13%
selected subcluster	23	61%	5%

TABLE II
RESULTS FOR BROWSING THE TRECVID COLLECTION

	iterations	elements kept	relevant missing
no filtering	> 500	100%	0%
random subcluster	46	67%	17%
selected subcluster	41	64%	8%

We select $n^* = 10$ for our experiments. Tables I and II show results, where "elements kept" represents the number of images in the collection after the filtering. "Relevant missed" is the number of relevant images that are accidentally removed during the filtering process.

Of course, browsing without filtering will not miss any relevant image as all are visited. The drawback is that the total number of iterations needed is 10 times higher than in the other two approaches. On average, for the Corel collection, normal browsing requires 184 iterations to check the whole collection, while the proposed approach needs only 23 iterations. For the TrecVid collection, the difference is even more significant with the number of iterations reducing to 41 while without filtering over 500 iterations are needed.

With the proposed approach, the number of missed relevant images is always smaller than random selection of representatives. Moreover, it is observed that the filtering can reduce the size of the collection significantly without loosing many of the relevant images. In the Corel collection, the size is reduced by 39%, while missing 5% of the relevant images. Reducing the size of the collection will certainly speed up the search. Because the collection is reduced by removing a number of irrelevant images, the chance of retrieving relevant images becomes higher, hence an in increase in recall is expected. The same holds for the TrecVid collection, with a reduction of 36% of the size of the collection on average missing 8% of relevant images.

From this experiment, the conclusion is that adding filtering during browsing does indeed support the search process by reducing the size of the collection while keeping the chance of missing any relevant images at an acceptable level. Moreover, the clustering used in our experiments is sufficiently fast for interaction. Even for the TRECVID collection with over 40 000 images the clustering step takes less than one second on a regular PC.

C. *Experiment on the Creation of Dissimilarity Space*

To create the projection of the 120-dimensional feature space $\mathcal{F}$ to $\mathcal{M}$, the 2-D manipulation space, we apply ISOSNE as presented in Section III-B. We compute two dissimilarity spaces $\mathcal{D}_{\mathcal{P}}^{120}$ and $\mathcal{D}_{\mathcal{P}}^{2}$ [see (21) and (22)]. To do so, first the prototype set $\mathcal{I}_{\mathcal{P}}$ is selected. In principle the number of prototypes could be taken quite large, but in practice users will typically be willing to select a limited number. We therefore test with $p = 5$ and $p = 10$. Smaller values are not suited as then the prototypes cannot cover the whole space.

On each dissimilarity space, prototypes I_{P_i} are used as initial positive examples. The SVM produces a ranked list based on distances to the boundary. Recall values are reported based on the 100 top-ranked images.

Figs. 7 and 8 show the performance on $\mathcal{D}_{\mathcal{P}}^{120}$ and $\mathcal{D}_{\mathcal{P}}^{2}$. Based on (25), for both collections, the performance of learning on dissimilarity space is on average a 60% improvement over the baseline.

Because of the projection of $\mathcal{I}$ from 120 dimensions to two dimensions for creating the manipulation space, distances between images cannot be kept perfectly even though the projection is optimal in keeping those. If the distance is not preserved on $\mathcal{M}$, performance of the $\mathcal{D}_{\mathcal{P}}^{2}$ will get worse compared to $\mathcal{D}_{\mathcal{P}}^{120}$. However, it is interesting to observe from the results that the search performance on $\mathcal{D}_{\mathcal{P}}^{2}$ is always better than on $\mathcal{D}_{\mathcal{P}}^{120}$ for both five or ten prototypes. These results show that the ISOSNE performs very well in preserving distances between images. Moreover, because ISOSNE extracts the structure of the collection by first computing the graph-based distance, it takes an advantage over the direct distance computation on the feature space. In other words, dissimilarity between images in the feature space is computed by directly comparing two feature vectors, whereas in the manipulation space, as a results of ISOSNE, dissimilarity is obtained by preserving a graph-based distance on the feature space. That explains why the performance of learning on $\mathcal{D}_{\mathcal{P}}^{2}$ is better than learning on $\mathcal{D}_{\mathcal{P}}^{120}$.

For the selection of a dissimilarity space, we prefer using $\mathcal{D}_{\mathcal{P}}^{2}$, more so because it establishes a direct link between distances in dissimilarity space and manipulation space.

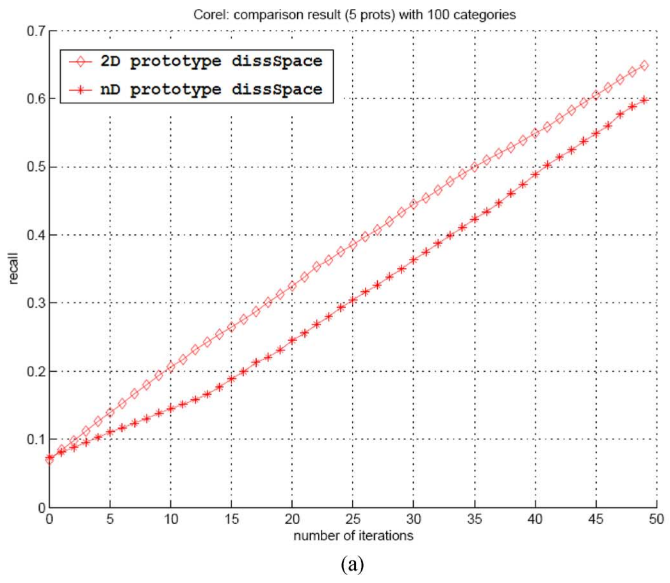
Note that these figures consider a category search task and not a typical websearch. For the later, the number of iterations would be excessive, but in professional applications they are quite acceptable. This is also reflected in the interactive search task in TRECVID which allows users a maximum of 15 min of search time for one information need.

D. *Experiment on Direct Manipulation of Dissimilarity Space vs. Indirectly via Feature Space*

We compare two ways of updating the dissimilarity matrix, via $\mathcal{D}_{\mathcal{P}}^{2}$ as we propose, or via feature space $\mathcal{F}$ [1], [14], [12], [33]. The baseline is defined in Section IV-A.

Results are shown in Figs. 9 and 10 for the Corel and TrecVid dataset, respectively. The figures show that with a small number of prototypes, the dissimilarity space is not able to maintain the distances between images. Therefore, the improvement of learning on the dissimilarity space is smaller than learning via the feature space. With $p = 10$ prototypes, the dissimilarity space covers the image collection better. Hence, on average it gives a higher improvement. Higher numbers of prototypes might increase this further, but this would make the user effort in creation of the initial dissimilarity space much higher.

With a smaller number of prototypes, i.e., a smaller number of initial positive examples, the performance of the baseline is worse than with a higher number of examples. From the figures, it is observed that average performance of learning via feature space will get worse when starting from more initial examples. Because the prototype set $\mathcal{I}_{\mathcal{P}}$ is chosen such that it is distributed over the collection when p gets higher, this set better covers the



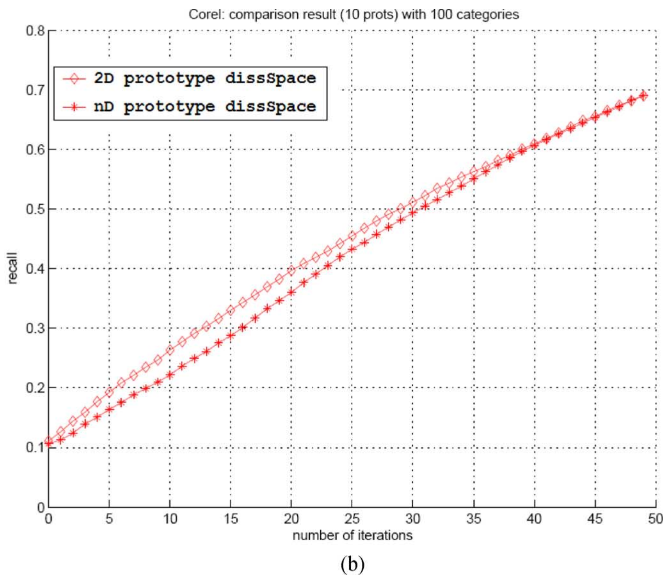
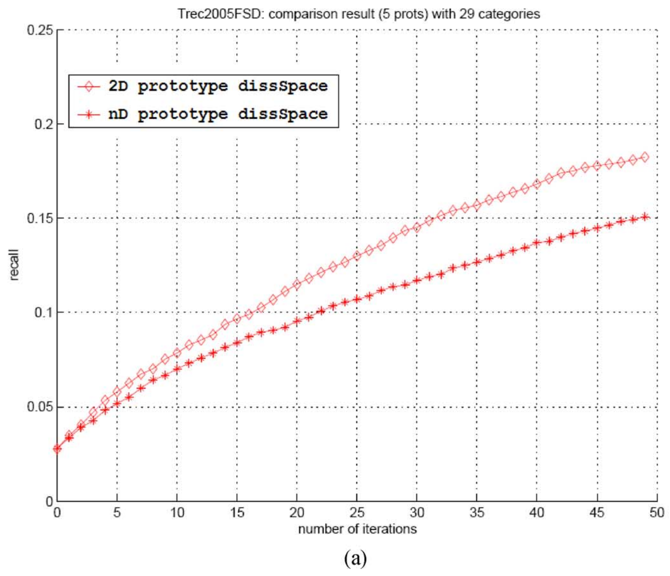


Fig. 7. Using different dissimilarity spaces with the Corel collection averaged over 100 categories. (a) Dissimilarity space created by five prototypes. (b) Dissimilarity space created by ten prototypes.



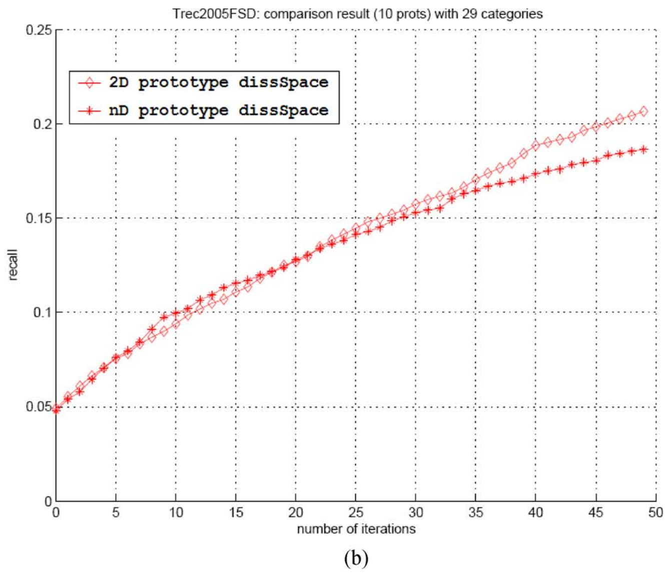


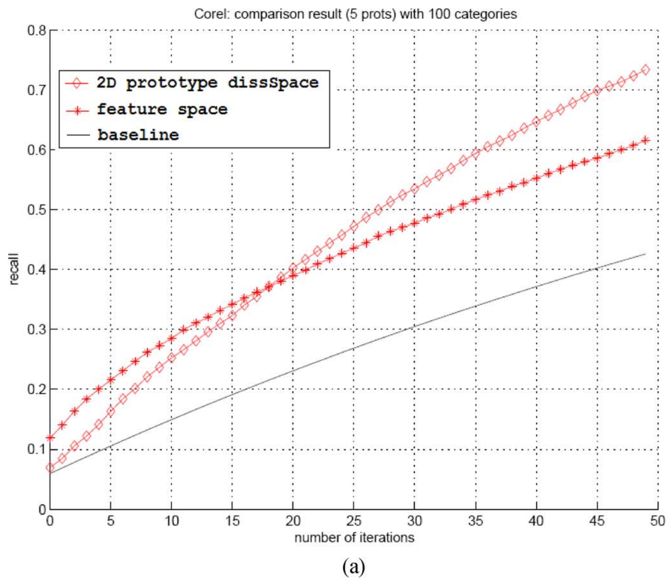
Fig. 8. Using different dissimilarity spaces with the TrecVid collection averaged over 29 categories. (a) Dissimilarity space created by five prototypes. (b) Dissimilarity space created by ten prototypes.

collection. In the feature space, this means that the more prototypes, the broader the boundary $\mathcal{B}$. This leads to a higher number of irrelevant images inside $\mathcal{B}$. This is a main disadvantage of using a feature space since they are not capable of capturing a semantic categorization which might be composed of several visual more low-level categories. When a dissimilarity space is created from $\mathcal{I}_{\mathcal{P}}$, the set of initial examples groups positive images together. Therefore, the performance of learning on dissimilarity space is improved.

V. Conclusion

In this paper, we have proposed a new approach for interactively learning dissimilarity. Our main motivation is that representing images by their relations to others is closer to the perceptual meaning of those images, which is not always easy to obtain using feature representations.

In our approach, different from existing techniques [1], [14], [12], [33] we directly learn the dissimilarity space from the user's feedback using a set of prototypes. This means that instead of collecting a large set of features or choosing problem-specific features, only the relations between images are used. The dimensionality of the dissimilarity space depends on the number of selected prototypes. This number should not be too many as they are provided as positive examples ($\ll$ number of features). By doing so, we avoid the computational problem occurring for large sets of features and the difficulty in selecting effective features in interactive category search.



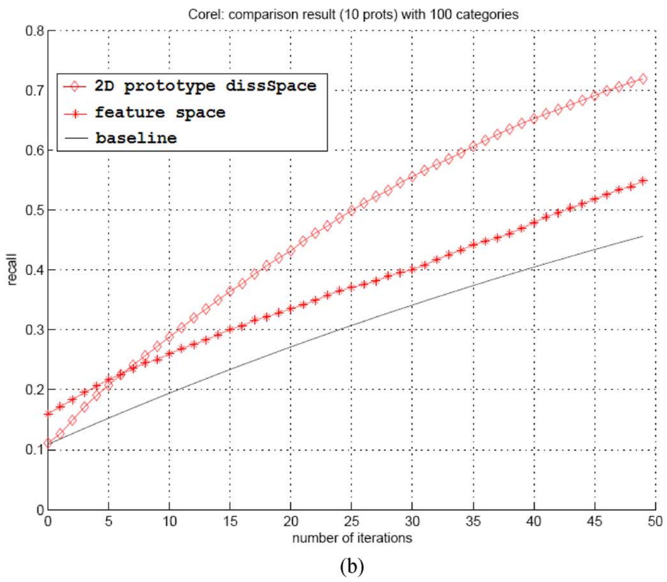
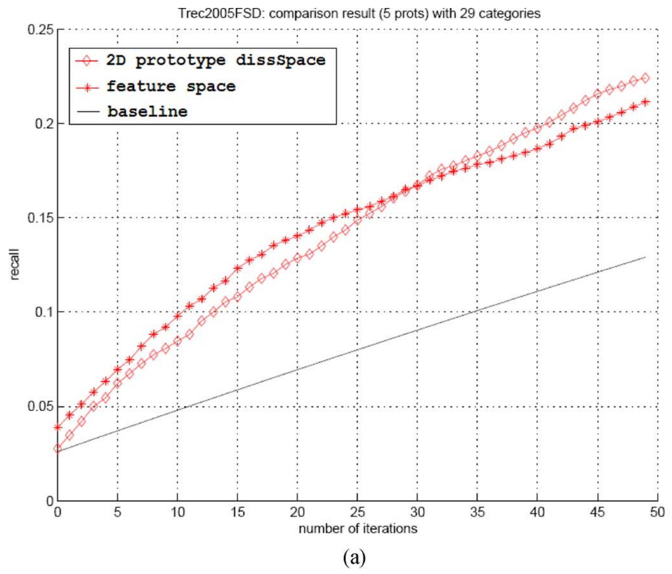


Fig. 9. Comparison of direct learning on dissimilarity spaces and learning via feature space with the Corel collection averaged over 100 categories. The results are evaluated by recall. (a) Dissimilarity space created by five prototypes. (b) and (d) Dissimilarity space created by ten prototypes.



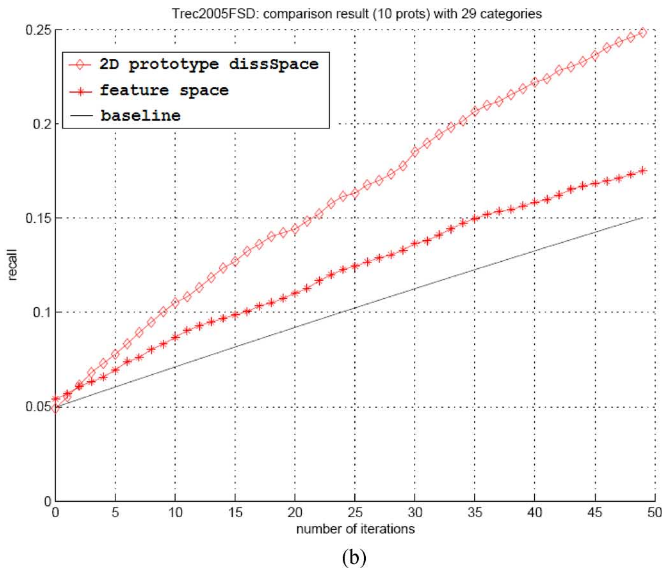


Fig. 10. Comparison of direct learning on dissimilarity spaces and learning via feature space with the TrecVid collection averaged over 29 categories. The results are evaluated by recall. (a) Dissimilarity space created by five prototypes. (b) Dissimilarity space created by ten prototypes.

We have demonstrated by experiment that learning in this dissimilarity space $\mathcal{D}_{\mathcal{P}}$ in general gives a better performance than learning in feature space $\mathcal{F}$, when a reasonable number of prototypes $\mathcal{I}_{\mathcal{P}}$ is being used, which in our case is ten. With a smaller number of prototypes, the dissimilarity space created will not be able to represent the relations between images. Therefore, it fails to improve over learning in the original feature space.

For the selection of prototypes, we present a browsing technique with hierarchical clustering and filtering components. With large image collections, this browsing strategy can speed up searching for relevant images, and in the mean time filter the collection by removing irrelevant clusters. This means that we are able to get a better chance of finding relevant images. In our experiments, we showed that the number of iterations needed to browse through the collection reduces by a factor of ten for both the Corel and the TrecVid collection.

We present a method to update the dissimilarity space using relevance feedback and active learning with one-class SVM. In this adjustment, we aim for assuring the closeness of relevant images to the classification boundary, while pushing away irrelevant ones. The results show that our proposed approach with ten prototypes improves over the learning in feature space. After 50 iterations, we gain an average relative improvement over the baseline of 60% for both the Corel and the TrecVid collection.

Finally, we have implemented an interactive search system, based on the proposed methodology (part of the MediaMill search system [29]). Fig. 11 shows a screenshot.

In conclusion, interactive learning in dissimilarity space rather than via feature space has great potential. Instead of the difficult task of defining a good feature space for general search tasks, indication of a set of prototypical examples, i.e., prototypes, is sufficient. With a limited number of well selected

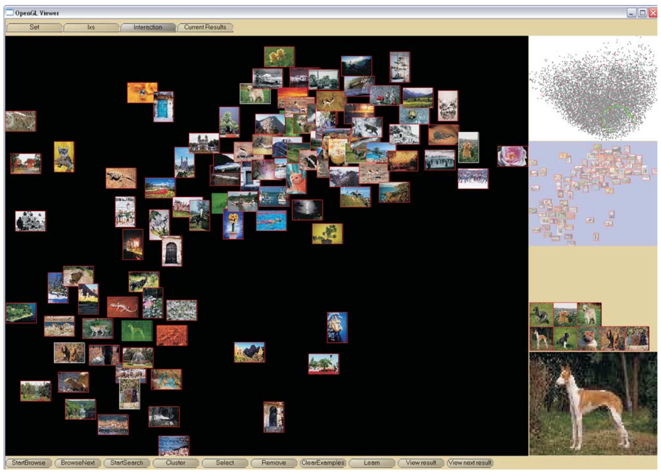


Fig. 11. Screen shot of the system while searching the Corel collection. The system contains four parts. The main window represents the part of manipulation space displayed, where the user interacts with images. This screenshot captures one step of the learning stage with displayed images being the ones closest to the classification boundary. The top-right window gives an overview of the whole manipulation space. The bottom-right corner window shows the current image at full size. During the interaction, relevant images selected by the user will be kept in the main window on the right side. For selection of positive examples, the 2-D similarity-based visualization shows the advantage over-the-grid based display. Instead of selecting one image at a time, in 2-D similarity-based visualization similar images stay close together, therefore the user can select a group of images at a time. Furthermore, as distances among images on the screen have a direct relation to distances in the dissimilarity space, the interaction is very intuitive.

prototypes, interactive retrieval performance on this space improves over the performance in feature space in terms of recall and the required number of interactions. Even more, as the manipulation space reflects the structure of the collection in the dissimilarity space, user interaction in the manipulation space will directly influence the adjustment of the dissimilarity space, allowing for intuitive interaction.

Acknowledgment

The authors thank J. van Gemert for providing the contexture features.

References

[1] B. Bhanu, J. Peng, and S. Qing, "Learning feature relevance and similarity metrics in image databases," in *Proc. IEEE Workshop on Content-Based Access of Image and Video Libraries*, 1998, p. 14.

[2] E. Bruno, N. M. Loccoz, and S. M. Maillet, "Learning user queries in multimodal dissimilarity spaces," in *Proc. 3rd Int. Workshop on Adaptive Multimedia Retrieval*, 2005.

[3] I. Campbell, "The Ostensive Model of Developing Information Needs," Ph.D. dissertation, University of Glasgow, Glasgow, U.K., 2000.

[4] A. Carkacioglu and F. Y. Vural, "Learning similarity space," in *Proc. Int. Conf. Image Processing*, 2002.

[5] Y. Chen, X. S. Zhou, and T. S. Huang, "One-class SVM for learning in image retrieval," in *Proc. Int. Conf. Image Processing*, 2001, vol. 1, pp. 34–37.

[6] K. Cox, "A unified approach to indexing and retrieval of information," in *SIGDOC '94: Proc. 12th Annu. Int. Conf. Systems Documentation*, 1994, pp. 176–181.

[7] R. P. W. Duin, D. Ridder, and D. M. J. Tax, "Experiments with a featureless approach to pattern recognition," *Pattern Recognit. Lett.*, vol. 18, pp. 1159–1166, 1997.

[8] I. El-Naqa, Y. Yang, N. P. Galatsanos, R. M. Nishikawa, and M. N. Wernick, "A similarity learning approach to content based image retrieval: Application to digital mammography," *IEEE Trans. Medical Imag.*, vol. 23, no. 10, pp. 1233–1244, Oct. 2004.

[9] G. D. Guo, A. K. Jain, W. Y. Ma, and H. J. Zhang, "Learning similarity measure for natural image retrieval with relevance feedback," *IEEE Trans. Neural Netw.*, vol. 13, no. 4, pp. 811–820, Jul. 2002.

[10] X. He, O. King, W. Y. Ma, M. Li, and H. J. Zhang, "Learning a semantic space from user's relevance feedback for image retrieval," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, no. 1, pp. 39–48, Jan. 2003.

[11] D. Heesch and S. Rüger, "Image browsing: A semantic analysis of NN^k networks," in *Proc Int. Conf Image and Video Retrieval*, 2005, pp. 609–618, LNCS 3568.

[12] B. Li and E. Y. Chang, "Discovery of a perceptual distance function for measuring image similarity," *ACM Multimedia J. (Special Issue on Content-Based Image Retrieval)*, vol. 8, no. 6, pp. 512–522, 2003.

[13] L. M. Manevitz and M. Yousef, "One-class SVMs for document classification," *J. Mach. Learn. Res.*, vol. 2, pp. 139–154, 2004.

[14] B. Moghaddam, Q. Tian, N. Lesh, C. Shen, and T. S. Huang, "Visualization and user-modeling for browsing personal photo libraries," in *Int. J. Comput. Vis.*, 2004, vol. 56, pp. 109–130.

[15] G. P. Nguyen and M. Worring, "Optimization of interactive visual similarity-based search," *ACM Trans. Multimedia Comput., Commun., and Applic.*, 2006.

[16] G. P. Nguyen and M. Worring, "Interactive access to large image collections using similarity based visualization," *J. Vis. Lang. and Comput.*, 2006.

[17] H. T. Nguyen and A. W. M. Smeulders, "Active learning using preclustering," in *Proc. 21st Int. Conf. Machine Learning*, 2004.

[18] E. Pekalska and R. P. W. Duin, "Dissimilarity representations allow for building good classifiers," *Pattern Recognit. Lett.*, vol. 23, pp. 943–956, 2002.

[19] E. Pekalska, R. P. W. Duin, and P. Paclik, "Prototype selection for dissimilarity-based classifiers," *Pattern Recognit.*, vol. 39, no. 2, pp. 189–208, 2006.

[20] K. Rodden, W. Basalaj, D. Sinclair, and K. Wood, "Does organisation by similarity assist image browsing?," in *ACM Conf. Human Factors in Computing Systems*, 2001, pp. 190–197.

[21] Y. Rubner, C. Tomasi, and L. J. Guibas, "The earth mover's distance as a metric for image retrieval," *Int. J. Comput. Vis.*, vol. 40, no. 2, pp. 99–121, 2000.

[22] Y. Rui, T. S. Huang, M. Ortega, and S. Mehrotra, "Relevance feedback: A power tool for interactive content based image retrieval," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 8, no. 5, pp. 644–655, Sep. 1998.

[23] D. E. Rumelhart and D. Zipser, "Feature discovery by competitive learning," *Cog. Sci.*, vol. 9, pp. 75–112, 1985.

[24] S. Santini, A. Gupta, and R. Jain, "Emergent semantics through interaction in image databases," *IEEE Trans. Knowl. Data Eng.*, vol. 13, no. 3, pp. 1041–4347, May/Jun. 2001.

[25] S. Santini and R. Jain, "Similarity measures," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 21, no. 9, pp. 871–883, Sep. 1999.

[26] S. Santini and R. Jain, "Integrated browsing and querying for image databases," *IEEE Multimedia*, vol. 7, no. 3, pp. 26–39, Jul.-Sep. 2000.

[27] A. W. M. Smeulders, M. Worring, S. Santini, A. Gupta, and R. Jain, "Content-based image retrieval at the end of early years," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 12, pp. 1349–1380, Dec. 2000.

[28] C. G. M. Snoek, J. van Gemert, J. M. Geusebroek, B. Huurnink, D. C. Koelma, G. P. Nguyen, O. de Rooij, F. J. Seinstra, A. W. M. Smeulders, C. J. Veenman, and M. Worring, "The MediaMill TRECVID 2005 semantic video search engine," in *Proc. 3th TRECVID Workshop*, 2005.

[29] C. G. M. Snoek, M. Worring, J. van Gemert, J. M. Geusebroek, D. Koelma, G. P. Nguyen, O. de Rooij, and F. Seinstra, "Mediamill: Exploring news video archives based on learned semantics," in *Proc. ACM Multimedia*, 2005.

[30] D. M. Squire, "Learning a similarity-based distance measure for image database organization from human partitionings of an image set," in *Proc. 4th IEEE Workshop on Applications of Computer Vision (WACV'98)*, 1998, p. 88.

[31] S. Tong and E. Chang, "Support vector machine active learning for image retrieval," in *ACM Int. Conf. Multimedia*, 2001, vol. 9, pp. 107–118.

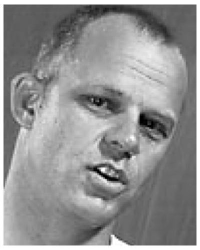
[32] J. van Gemert, J. M. Geusebroek, C. J. Veenman, C. G. M. Snoek, and A. W. M. Smeulders, "Robust scene categorization by learning image statistics in context," in *CVPR Workshop on Semantic Learning Applications in Multimedia (SLAM)*, 2006.

[33] H. Ye and G. Xu, "Similarity measure learning for image retrieval using feature subspace analysis," in *Proc. 5th Int. Conf. Computational Intelligence and Multimedia Applications*, 2003, pp. 131–136.

[34] X. S. Zhou and T. S. Huang, "Relevance feedback in image retrieval: A comprehensive overview," *Multimedia Syst.*, vol. 8, pp. 536–544, 2003.

Giang P. Nguyen received the B.Sc. degree (Hons.) in applied mathematics and informatics in 1999 and the M.Sc. degree in 2001, both from Hanoi University of Science, Hanoi, China. She received the Ph.D. degree from the Faculty of Science, University of Amsterdam, Amsterdam, The Netherlands, in December 2006.

She was with the ISIS Group at the Faculty of Science, University of Amsterdam, since 2001. Currently, she holds a Postdoctoral position in the Computer Vision and Media Technology Group, Aalborg University, Aalborg, Denmark. Her research interests lie in multimedia retrieval, information visualization and interactive search.



Marcel Worring (M'03) received the M.Sc. degree (Hons.) from Vrije Universiteit, Amsterdam, The Netherlands, in 1988 and the Ph.D. degree from the University of Amsterdam in 1993, both in computer science.

He is currently an Associate Professor at the University of Amsterdam. His interests are in multimedia search and systems. He leads several multidisciplinary projects covering knowledge engineering, pattern recognition, image and video analysis, and information space interaction, conducted in close cooperation with industry. In 1998, he was a Visiting Research Fellow at the University of California, San Diego. He has published over 100 scientific papers and serves on the program committee of several international conferences.

Dr. Worring is the Chair of the IAPR TC12 on Multimedia and Visual Information Systems and the General Chair of the 2007 ACM International Conference on Image and Video Retrieval.

Arnold W. M. Smeulders (SM'79) received the M.Sc. degree in physics from the Technical University of Delft, Delft, The Netherlands, in 1977 and the Ph.D. degree in medicine (on the topic of visual pattern analysis) from Leiden University, Leiden, The Netherlands, in 1982.

He is the Scientific Director of the Intelligent Systems Lab Amsterdam, University of Amsterdam, and of MultimediaN, the Dutch public-private partnership, and of the ASCI National Research School. The ISIS Group at the Faculty of Science, University of Amsterdam, concentrates on theory, practice, and implementation of multimedia information analysis, including image databases and computer vision, and has an extensive record in co-operations with Dutch institutions and industry in the area of multimedia and video analysis. He also participates in the EU-Vision, DELOS, and MUSCLE networks of excellence. His research interests are in cognitive vision, content-based image retrieval, learning and tracking, and the picture-language question. He has written 300 papers in refereed journals and conferences and has advised 28 Ph.D. students.

Dr. Smeulders is a Fellow of the International Association of Pattern Recognition and an Associate Editor of the *International Journal of Computer Vision* and the IEEE TRANSACTIONS ON MULTIMEDIA.