Adaptive Image Retrieval using a Graph Model for Semantic Feature Integration

Jana Urban and Joemon M. Jose
Dept. of Computing Science, University of Glasgow
Glasgow, UK
{jana,jj}@dcs.gla.ac.uk

ABSTRACT

The variety of features available to represent multimedia data constitutes a rich pool of information. However, the plethora of data poses a challenge in terms of feature selection and integration for effective retrieval. Moreover, to further improve effectiveness, the retrieval model should ideally incorporate context-dependent feature representations to allow for retrieval on a higher semantic level. In this paper we present a retrieval model and learning framework for the purpose of interactive information retrieval. We describe how semantic relations between multimedia objects based on user interaction can be learnt and then integrated with visual and textual features into a unified framework. The framework models both feature similarities and semantic relations in a single graph. Querying in this model is implemented using the theory of random walks. In addition, we present ideas to implement short-term learning from relevance feedback. Systematic experimental results validate the effectiveness of the proposed approach for image retrieval. However, the model is not restricted to the image domain and could easily be employed for retrieving multimedia data (and even a combination of different domains, eg images, audio and text documents).

Categories and Subject Descriptors: H.3.3 [Information Storage and Retrieval]: Information Search and Retrieval—*relevance feedback, retrieval models*

General Terms: Retrieval Models, Experimentation, Performance

Keywords: semantic features, image retrieval, relevance feedback, random walks, fusion

1. INTRODUCTION

Ever since the deficiencies of primitive content-based features were realised, interest has turned to *"semantic features"* and *"semantic retrieval"*. Semantic features are now the ultimate goal in order to facilitate effective retrieval of visual data, but what are they? Smeulders et al state that *"Semantic features aim at encoding interpretations of the image which may be relevant to the application."* [15, p. 1361]. There are two important points to note in this assertion. Firstly, semantics are about *interpretation*, and secondly the interpretation is to a large degree domain or *context dependent*. An image by itself usually has no intrinsic meaning. The meaning is bestowed upon the image by a human observer regarding the context of both the observer and the image.

The goal of the semantic approach is to replace the low-level feature space with a higher-level semantic space, which is closer to the abstract concepts the user has in mind when looking for an image. Since the endeavour of obtaining semantic features directly from the visual attributes was unfruitful, mining for semantic concepts from a knowledge-base has been the focus of research to this end. Most of the existing attempts towards semantic features can be broadly categorised in two classes: *annotation-based* [7, 12] and *user-based* [18, 3, 4]. This distinction arises from the nature of the knowledge-base used: the first method relies on an (at least partially) annotated image corpus from which semantic concepts can be learnt and propagated to other images, whereas the latter learns semantic concepts from the user directly. While there is a number of general concepts that can universally be agreed upon, e.g. an 'indoor' vs. 'outdoor' classification, there are more subtle meanings that are subject to the observer's interpretation, e.g. 'a romantic scene'. The major difference in the two approaches hence lies in the interpretation context considered for deciphering the image's meaning. It should become obvious that the annotation-based approach can only succeed in taking very general concepts into consideration, as opposed to user-based approaches that are tailored to the user's expectations and interpretations.

Our approach is an example of the latter in that contextual information is mined from user interaction. We have developed a system, *EGO*, that encourages its users to manage their retrieval results on a workspace provided in the interface [20]. While searching for images, the creation of groupings of related images is supported, inciting the user to break up the task into related facets to organise their ideas and concepts. The system can then assist the user by recommending relevant images for selected groups. Previous user experiments have shown that *EGO* helps to overcome the query formulation problem and leads to a more effective and enjoyable search experience compared to a state-of-the-art relevance feedback interface [22].

In this work, we use the groupings created in the user experiments to infer a semantic feature. Our underlying assumption is that all objects (images) in one group share some semantic concept (user-, usage-, and task-dependent), eg images of snowy mountains, images with high visual contrasts, images that could be used as background on the front of a flyer. Instead of trying to label these concepts, however, we simply record that there is a semantic relation between those images in a group. We refer to these relationships as *peer information*. Appropriately recorded, the peer information can be used to implement long-term learning of semantic concepts in the system.

MIR'06, October 26–27, 2006, Santa Barbara, California, USA.

In addition to the peer information, low-level visual features and textual annotations are further sources of information for the retrieval (and recommendation) system. However, the combination of different feature modalities is a big challenge in multimedia retrieval [11, 6, 19]. Most state-of-the-art systems treat each feature individually and fuse the result lists to obtain the final results. However, the method of fusion is far from obvious and such systems fail to capture dependencies between the features. Even worse, such systems have difficulties in exceeding the performance of a text-only system in information retrieval tasks [11]. Instead of a late fusion of results, we propose to integrate the different modalities in a single graph and use the theory of random walks [10] to calculate retrieval results.

In our model, images, terms, and visual features are represented as nodes in an *Image-Context Graph* (ICG). The links between nodes represent: (1) image attributes (relations between images and their features); (2) intra-feature relations (feature similarities); and (3) semantic relations (peer information). We describe a retrieval model based on random walks, that can retrieve both top matching images as well as terms to a query (consisting of both image examples and terms). In addition, we show how short-term relevance feedback learning can be integrated in our model by adapting the link weights in the ICG. The main contributions of this paper are:

- We propose a group-based contextual feature (peer information) based on mining usage information while searching in a multimedia collection.
- We show how the peer information can be integrated with already existing low-level visual features and textual annotation in a graph model.
- We define various learning strategies in the graph model.
- Through systematic experimental results the effectiveness of the proposed approach is validated and learning strategies are investigated.

The remainder of this document is organised as follows. Section 2 reviews related work. We detail the graph-model and explain the mathematical background in Section 3. Section 4 introduces the baseline systems used in the evaluation. It consists of three separate retrieval models for each feature modality, whose results are combined using a rank-based list aggregation method. We outline the experimental methodology in Section 5, followed by the experimental results in Section 6. Finally, we summarise and conclude the paper in Section 7.

2. RELATED WORK

The theory of Random Walks has been applied to information retrieval in the form of Google's famous PageRank algorithm [1]. The idea can be sketched as follows. Imagine a random surfer on the Web choosing to follow a link on each page at random. Occasionally, the surfer gets stuck in a dead end or in cycles, or simply gets bored. At these points, he may randomly jump to another page on the Web not following any links. The goal of a page's PageRank score is to reflect its quality depending on the number of other pages linking to it based on the random surfer model. The PageRank algorithm can be viewed as a random walk on the Web graph. The mathematical details will be elaborated in Section 3.1.

2.1 Random Walks in the Image Domain

Graph-based modelling techniques have recently found their way into the image domain. The two most closely related approaches include its application for relevance feedback learning [3, 4] and for image captioning [12]. Han et al have proposed to model the relationships between images based on their co-selection in relevance feedback sessions [3]. The ratio of the frequency of two image being labelled as positive examples in the same retrieval session over the total frequency of them having been selected together (as positive or negative samples) determines the weight of the link between these two images. The calculation of a semantic similarity measure between two images is based on the overall correlation as determined by analysing the resulting graph (referred to as the image link network). An overall similarity measure is defined as a weighted linear combination of the semantic similarity and the low-level feature similarity. In contrast, the theory of random walks is explicitly employed on an image graph in which links between image nodes are also constructed from relevance feedback information in [4]. Here the graph is constructed by adding two special nodes to the graph: a positive absorbing node and a negative absorbing node. Each positively labelled image receives a link to the positive absorbing node, while negative examples are directly linked to the negative absorbing node. As this approach is not discriminating between query session, it can only be used for short-term learning.

The second application of random walks in the image domain is to automatically learn annotations for previously unlabelled images [12]. A graph, called GCap, is constructed, which contains one node per image, a node for each image region per image, and a node for all terms in the vocabulary. Images are connected to its region nodes and the terms it is annotated with. Further, regions are linked to their k-nearest neighbours. Given an unlabelled image, ie an image node I_i that does not have any links to a term node in the graph, a random walk is performed to compute the most probable terms for this image. These are found by calculating the long-term (stationary) probabilities that a random walker finds himself at a particular node given that it randomly restarts the walk from i. The top t terms with the highest stationary probability are returned as the suggested labels.

The semantic link approaches [3, 4] only model the information gained from relevance feedback which has to be combined with feature-based similarity values in a further step, while the image captioning approach [12] only models image-feature similarities without a facility of adaptation to relevance feedback. We propose to model both the image-feature relations as well as inter-image (or semantic) relations together. Hence there are two vital ingredients to our approach: (1) the feature integration of semantic as well as low-level features using a graph-model, and (2) a learning strategy in the graph model. The latter incorporates two levels of feedback to implement short- and long-term learning from user feedback. By adding links between images that are grouped together the semantic network is iteratively constructed and enforced by using adaptive link weights, thus implementing a *long-term learning* strategy. Further, we show how *short-term learning* can be achieved by introducing feature weights to ensure that those links to feature nodes with a strong feature weight are favoured over feature links with small weights given a particular query.

3. THE IMAGE-CONTEXT GRAPH

The problems addressed in this paper are (a) how to capture and model personalised usage information to improve retrieval performance, and (b) how to integrate this information with other features (visual and textual) to model interdependencies between features.

The idea is to represent images and all their attributes (features) in a graph. The graph consists of a number of layers of vertices: vertices for all images in the collection, and one layer of vertices per implemented feature. These layers will contain both visual and textual features. There are two different types of edges connect-

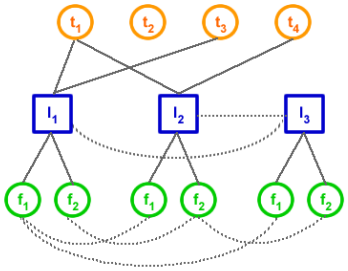

Figure 1: An example image-context graph

ing vertices: edges representing a "contains" relationship (ie edges between the image vertices and their attributes), and edges representing the similarity amongst vertices in the same layer ("similarity edges"). These edges are constructed based on the similarity between features (similarity between visual feature vectors, similarity between terms) or semantic relationships/co-occurrences of images. Thus the graph represents the images in context and in the following it is referred to as the *Image-Context Graph*, or *ICG*. An example graph containing three image nodes $(I_1,..,I_3)$, four term nodes $(t_1,...,t_4)$, and two types of visual features (f_1,f_2) is depicted in Figure 1.

The general recommendation problem (or retrieval problem for that matter) can be stated as: Given a query, consisting of image examples and/or terms, compute the most similar images to recommend to the user. In the ICG, this translates to: given a start set of vertices in the graph, compute those image vertices that are most likely to be reached starting from the start set.

A solution to this problem can be found in the theory of Random Walks. The likelihood of passing a node in the ICG is given by calculating the stationary distribution of the Markov chain induced by the ICG. By setting the restart vector to the nodes representing the query items, we can stage a Random Walk with Restarts on the ICG. This is equivalent to computing a query-biased "PageRank" of the ICG as will be explained in the following section.

3.1 Mathematical Background

A random walk is a finite-state Markov chain that is time-reversible. Markov chains are frequently used to model physical and conceptual processes that evolve over time, for example the spread of disease within a population or the modelling of gambling. An introduction to Random Walks and Markov chains can be found in [10].

Let the Markov chain $\mathscr{M}$ consist of a finite number of states, say $N=\{1,2,...,n\}$, and probabilities of a transition occurring between states at discrete time steps. The (one-step) *transition probability* p_{ij}, denotes the conditional probability that $\mathscr{M}$ will be in state j at time $t+1$ given that it was observed in state i at time t. In general, p_{ij}^k denotes the probability that $\mathscr{M}$ proceeds from state i to state j after k transitions. The *transition probability matrix* $P=[p_{ij}]$ is often used to represent $\mathscr{M}$. The stationary distributions $\pi^T=[\pi_1,\pi_2,...,\pi_n]$ represent the long-run proportion of time the chain $\mathscr{M}$ spends in each state. π is also referred to as the steady state probability vector. Markov chains are often represented as a graph, or state transition diagram G. Finally to make the connection to PageRank: the PageRank scores are equivalent to the stationary distribution π of the Markov chain associated with the Web graph.

3.1.1 Calculating π

In general the stationary distribution, π, of a Markov chain can be found by solving the following eigenvector problem:

$$\pi = \overline{\overline{P}}^T * \pi \tag{1}$$

A unique stationary distribution is guaranteed to exist, iff $\overline{\overline{P}}$ is a stochastic, irreducible matrix [8].

In the PageRank model, a transition probability matrix P is built from the hyperlink structure of the Web. To create a stochastic, irreducible matrix, Brin and Page suggested to eliminate dangling pages (pages with no outlinks) by linking them to all other pages in the Web [1]. This is achieved by replacing 0^T rows of the sparse matrix P with dense vectors, that is the uniform vector $\frac{1}{n}e^T$ initially or a more general probability distribution over all pages v^T. This stochastic fix can be modelled implicitly by the following transformations (see [8]):

$$\overline{P} = P + a\ v^T \tag{2}$$

$$\overline{\overline{P}} = (1-\alpha)\overline{P} + \alpha\ e\ v^T \tag{3}$$

where a is a vector whose elements $a_i=1$ if row i in P corresponds to a dangling node, and 0 otherwise; e the vector of all 1s; $0 \leq \alpha \leq 1$; and v representing a general probability distribution over the nodes—often referred to as the personalisation or restart vector.

Substituting $\overline{\overline{P}}$ in Equation 1 then leads to:

$$\pi = ((1-\alpha)P + ((1-\alpha)a + \alpha\ e)v^T)^T\pi \tag{4}$$

$$\pi = (1-\alpha)(P + a\ v^T)^T\pi + \alpha\ v \tag{5}$$

with the constraint that π is normalised, such that $|\pi|=1$ and thus $e^T\pi=1$. α is the probability of restarting the random walk from any of the nodes in v.

3.1.2 Parameters of the PageRank Model

α. The value of α denotes the probability of a surfer choosing to jump to a new Web page (teleportation), while they choose to click on hyperlinks with probability $(1-\alpha)$. A small α places more emphasis on the hyperlink structure of the graph and much less on the teleportation tendencies, and also slows convergence of the iterative computation of PageRank. Originally $\alpha=0.15$ was proposed [1].

In the image annotation graph of [12] a value of $\alpha=0.65$ was found to be better suited, which they could explain by a relationship to the estimated diameter of the graph.

The personalisation vector v^T. Instead of the uniform distribution $\frac{1}{n}e^T$, a more general distribution $v^T>0$ can be used in its place. v^T is often referred to as *personalisation vector* or *restart vector* in random walk terms.

The personalisation vector also allows PageRank to be made query-sensitive. The original PageRank assigns a score to a page proportional to the number of times a *random* surfer would visit that page, if they surfed indefinitely, following all outlinks with equal probability or occasionally jumping to a random new page chosen with equal probability. If we change the probability distribution given by the personalisation vector v^T then we can introduce a certain bias that the surfer jumps to pages with high probability in v^T.

3.2 Constructing the ICG

Let G be the ICG and V the set of vertices in G and E the set of edges. Then $G = (V, E)$. The graph will be stored in the form of its adjacency matrix M.

3.2.1 The Nodes

There are three types of nodes: image nodes $\mathscr{I}$, term nodes $\mathscr{T}$, and feature nodes $\mathscr{F}$, and $V = \mathscr{I} \cup \mathscr{T} \cup \mathscr{F}$:

- Let $\mathscr{I}$ denote the set of all *image nodes* in G. Add one node per image to the set of image nodes. I_i denotes the node for image i.
- Let $\mathscr{T}$ denote the set of all *term nodes* in G. Add one node for every term in the vocabulary to $\mathscr{T}$. t_i denotes the node for term i.
- Construct the set of *visual feature nodes* $\mathscr{F}$ by adding one node per low-level visual feature for each image. If the number of implemented visual features is v (which is 6 in our case), then $|\mathscr{F}| = v \times |\mathscr{I}|$. f_{ij} denotes the node for the j-th feature of image i.

3.2.2 The Edges

There are two types of edges: attribute edges and similarity edges. The first type of edges link images to their attributes, the second type of edge links nodes of the same feature type (term and visual feature nodes) based on the similarity between these nodes. A special type of similarity edges are peer edges between image nodes themselves, which are created based on users' groupings of images.

Attribute Edges Each image node I_i is linked to all its features. Thus an edge is created to each of its visual feature nodes $f_{i1}, \ldots f_{iv}$. For the textual features, an edge is created between an image node I_i and a term node t_j if image i is annotated with term j.

Similarity Edges Similar to [12], we propose to create edges between visual features based on their nearest neighbours. Consider a feature node f_{il} representing the l-th feature of image i, then compute the top k nearest neighbours by calculating the similarity score between the feature vector $\overrightarrow{f_{il}}$ and the feature vector $\overrightarrow{f_{jl}}$ for all other images j ($0 < j < |\mathscr{I}|$). This allows for an adaptive definition of closeness without having to fix a threshold value.

A similar idea could be applied to the term nodes choosing a similarity measure between terms based on relationships between terms (eg using WordNet) or a collection-based analysis. Since the number of terms contained in an image (annotations) is typically very low (compared to text documents), a collection-based analysis is probably not very significant. Instead we adopt a simple similarity measure $sim(t_i, t_j) = 1$ if $i = j$ and 0 otherwise. Using this similarity measure, we will obtain an edge that links each term node to itself.

Peer Edges Finally, the edges between the image nodes themselves are based on user feedback. For each group created by a user, edges are created connecting all the images in that group. An edge between two images i and j has a weight, which generally reflects the frequency of these images co-occurring in groups. However, the weight can also be reduced by negative feedback (see below). These edges represent high-level semantic relationships between images based on their usage.

3.3 Evaluating a Query

The objective of retrieval in the graph is to find those image nodes $\in \mathscr{I}$ that are closest (or best connected) to the query nodes. The overview of the algorithm is as follows. First, the restart vector is built from the query nodes. Then, a Random Walk with Restarts is performed on the graph to estimate the stationary probability distribution π. Finally, the image nodes are returned to the user sorted in descending order by their steady state probability scores. Algorithm 1 shows an overview of these steps.

Algorithm 1 Calculating the query results based on a Random Walk on ICG

Require: Query consisting of image examples and query terms; M the adjacency matrix of the ICG; constant $0 < \alpha < 1$;
Ensure: $||\pi||_1 = 1$ (L_1 norm of π)

1: Initialise personalisation vector v.
2: M' = normalise(M).
3: Initialise $\pi^0 = v$
4: Set $k = 0$ the number of iterations.
5: **while** not converged **do**
6: $\quad \pi^k = (1-\alpha) * M' * \pi^{k-1} - \alpha * v$
7: $\quad$ Normalise π^k.
8: $\quad$ k = k+1
9: **end while**
10: **return** Image documents sorted by their π values after convergence.

Construction of the restart/personalisation vector. Assume a query contains a number of image examples and a set of terms. The personalisation vector v is initialised, such that $v(u) = \frac{1}{q}$ for all nodes u representing the image examples and terms, where q is the size of the query. The remaining elements are set to 0. Choosing the personalisation vector this way ensures that these nodes are favoured in the following Random Walk computation.

Calculating π. Recall from Section 3.1.1 (cf Equation 1) that the stationary distribution, π, of a Markov chain can be found by solving the eigenvector problem: $\pi = \overline{\overline{P}}^T * \pi$. In the ICG, there are no dangling nodes due to the way the ICG is constructed, so the transformation to create a stochastic, irreducible matrix representing the ICG (cf Equation 2) can be simplified to:

$$\overline{\overline{P}} = (1-\alpha)P + \alpha \; e \; v^T \tag{6}$$

And the calculation of π can be achieved by:

$$\pi \quad = \quad (1-\alpha)M'\pi + \alpha \; v \tag{7}$$

where $M'(= P^T)$ is the column normalised adjacency matrix of the ICG. α is the probability of restarting the random walk from any of the nodes in v.

The estimation of π is solved in the iterative algorithm detailed in Alg 1. The algorithm converges if two consecutive estimates π^k and π^{k+1} are reasonably close together, ie $|\pi^k$-$\pi^{k+1}| <$ threshold. The threshold is set to 10^{-6}.

Returning the query results. Finally, we choose the top r image nodes (ie the elements, $\pi(u_i)$, from π, where $1 \leq i \leq |\mathscr{I}|$) and present them to the user.

3.4 Relevance Feedback

In this section we show how both long- and short-term learning can be implemented in the ICG to create a retrieval system that adapts to its users. On the one hand, relevance feedback is used to build up the semantic or peer network (the subgraph consisting of image nodes and the edges between them) over time. On the other hand, short-term learning is implemented by computing a set

of feature weights used to adapt the transition probabilities from image nodes to feature nodes on a per-query basis.

3.4.1 Long-Term Learning: Adding Peer Links

In our application feedback is provided in terms of grouping images. Images which are put in the same group receive a co-occurrence edge in the graph, thus the weight of an edge between two particular images reflect the frequency of these images being grouped together. Similarly if an image is selected as negative example for a particular group, the weight of the edges between the negative image and all other images in the group will be discounted. Hence, over time semantic relations are strenghtened which reflect the usage context of the images.

Positive Feedback is always a result of adding an image to a group. Therefore, the newly added image receives a new link to all other images in the group (and vice versa). If a link already exists between the new image and another group image, its link weight is increased by 1. With usage over time, the link weights between semantically related images are strengthened.

Negative Feedback is incurred if either an image is specifically labelled as not belonging to a certain group, implicitly ignored in the recommendation set for a group in three consecutive turns, or deleted from a group in which it previously resided.

There are two strategies to deal with negative feedback: discount the link weight by a constant factor, eg $\frac{1}{5}$, or decrement the link weight by 1. The former changes the weights drastically in favour of the most recent feedback. For instance, if two images are in the same 10 groups together but are regarded as unrelated in the 11th group, the resulting link weight is 2 ($10 * \frac{1}{5}$). This might not be a desirable outcome with regards to the long-term learning capabilities of the ICG. However, one could also implement an overlay of the graph in which the most recent feedback changes the current link weights by a high discount factor so that it will take a short-term view. Only the original link weights of the ICG will be stored and the overlay discarded after a query session.

3.4.2 Short-Term Learning: Adjusting Link Weights

Visual Feature Weights. Given the ICG is constructed from v visual features, then each image node is connected to exactly v feature nodes (in addition to possible term and other image nodes). Now, the importance of the specific visual feature will depend on the current query. Therefore, the probability of moving from an image node to a visual feature node (and vice versa) should change depending on the importance of that particular feature.

In [14] an optimised framework is presented for calculating visual feature weights, when only positive feedback is considered (also used in the baseline system). The same inter-feature learning algorithm can be employed in the ICG to change the link weights between image and feature nodes. Essentially, an optimal solution for the visual feature weights $\overrightarrow{u}$ is derived that minimises the summed distances between positive feedback examples and the query. The feature weights are indirectly proportional to the sum of weighted distances[1] between the query and all relevant images under feature j (for $j = 1, .., v$):

$$u_j \propto \frac{1}{\sqrt{\sum_{i=1}^{P} \mathrm{rel}_i \, d_j(p_i, q)}} \tag{8}$$

[1]As the link weights are probabilities (ie the higher its weight the more likely that link will be followed) the distances used in the computation of feature weights are converted to similarities by $1/d$.

where $\mathrm{rel}_i \in [0,1]$ is the relevance score of the i-th example. The feature weights are subject to normalisation, ie $\sum_{j=1}^{v} u_j = 1$.

During the normalisation phase before calculating π, all outgoing links from an image node to their feature nodes are re-weighted with the corresponding link weight, such that $l' = u_j * l$, where l is the link weight.

Overall Feature Weights. A final modification on link weights is made to influence the overall importance of the three high-level features: visual, textual and peers. Assume there are weights w_v, w_t, and w_p for the three high-level features. To implement overall feature weighting, all outgoing links from an *image* node are weighted with the corresponding feature weight depending on their link type: $l' = w_i * l$, where l is the link weight and i is the feature type.

These weights could either be specified explicitly by the user or obtained automatically. There are two possibilities of how to calculate the weights automatically on a per-query basis. First, the weights can be calculated based on the similarity between the query items considering the three features separately. For this, construct a visual, term, and peer query based on the image examples and terms provided in the query (cf Section 4). The visual feature weight is then proportional to the sum of similarity scores between these queries and the query items. The disadvantages of this method is that it uses the original indices and similarity computations rather than the graph representation to determine feature weights. In addition, the peer-, feature-, and term-scores are not readily comparable, which has to be addressed for example by a min-max normalisation of the scores.

Alternatively, the graph structure could be used to determine the similarity between query nodes based on the three individual features. A solution would be to perform a random walk for each type of feature: one in which the restart vector is set to the term nodes contained in the query for the term feature; then to the image nodes for the peer feature; and finally to the visual feature nodes connected to the query images for visual feature. Again the weights would be proportional to the sum of resulting ranks (ie similarity scores). Let π_t be the stationary distribution of the random walk started from the term nodes and let sim_t denote the overall term similarity of the query. Define π_v, π_p, and sim_v, sim_p similarly. Then $sim_t = \sum_{u \in Q} \pi_t$, and $w_t = \frac{sim_t}{sim_t + sim_v + sim_p}$. The obvious problem with this approach is that three random walks have to calculated before the actual random walk is calculated.

Since both these adaptive methods have drawbacks which cannot easily be overcome, we only present results using a set of predefined feature weights in this paper to determine the effect of such weights.

4. THE BASELINE SYSTEM

In order to evaluate the graph-based method of feature integration, it is compared to a traditional separatist approach, which is the most prevalent technique for multimedia retrieval applications where a number of features are available for indexing [11]. This baseline treats all three features individually to compute three separate result lists. The lists are finally combined using the rank-based *Voting Approach* [21] in this paper.

4.1 Visual Features

The images are represented according to the hierarchical object model used in [14]. In this model an image is represented by a set of feature vectors, one for each distinct feature implemented. The distance between an object x in the database and a given query rep-

resentation q is computed in two steps. First, the individual feature distances g_i (for i in $1..v$, where v is the number of features) are computed by the generalised Euclidean distance,

$$g_i(q,x) = (\vec{q}_i - \vec{x}_i)^T W_i (\vec{q}_i - \vec{x}_i) \quad (9)$$

where $\vec{q}_i$ and $\vec{x}_i$ are the i-th feature vectors of the query q and the database object x respectively, and W_i the *feature transformation matrix* used for weighting the feature components. W_i is a $K_i \times K_i$ real symmetric full matrix, where K_i is the i-th feature dimension. The second step is then to combine the individual distances to arrive at a single distance value d. This is achieved by a linear combination between $\vec{g}(q,x) = [g_1(q,x), ..., g_I(q,x)]^T$ and a feature weight vector $\vec{u}$,

$$d(q,x) = \vec{u}^T \vec{g}(q,x) \quad (10)$$

4.1.1 Relevance Feedback by Learning a Transformed Feature Space

We use the optimised learning framework proposed in [14] for calculating the weights in the matching function, when only positive feedback is considered. Due to the hierarchical object model, it distinguishes between component and feature weights. Three steps of computation are required to determine the optimal feature weights. First, a query q to represent the training samples is chosen. Second, the intra-component weights W_i are computed for each feature i. Finally, we can find the inter-feature weights $\vec{u}$ that best capture the features' importance in the training samples.

The *optimal query vector* $\vec{q}_i$ (for the i-th feature) is calculated as the centroid of the P positive examples specified by the user. The *optimal feature component weights* are given by the feature space transformation matrix $W_i = \det(C_i)^{\frac{1}{K_i}} C_i^{-1}$, where C_i is the covariance matrix of the P positive examples. Finally, the *optimal feature weights* $\vec{u} = [u_1, .., u_v]$ are solved by, $u_i = \sum_{j=1}^{v} \sqrt{\frac{f_j}{f_i}}$, where $f_i = \sum_{p=1}^{P} \mathrm{rel_p} g_{pi}$ and $\mathrm{rel_p}$ is the relevance score of example p. The total distance of a database image to the optimal query q is then computed by Equations (9) and (10).

4.2 Peer Feature

To model the group context in its most simplistic form we count the number of co-occurrences between images. If two images belong to the same group, their co-occurrence score is incremented by one. This information can be represented in a square matrix M, whose rows and columns are the images in the collection. The entry $m_{i,j}$ denotes the number of groups that contain both image i and image j. The diagonal of M is set to 1. M is symmetric, thus $m_{i,j} = m_{j,i}$, since if image i co-occurs with image j, then image j also co-occurs with image i. In other words, images i and j are considered peers.

We can also interpret this information parallel to traditional textual IR, if we consider the images in the collection the vocabulary with which our documents (the same images) are annotated. Each image is then represented by a term-vector that encodes its peers. First of all, we assign each term j a weight in document i:

$$w_{i,j} = \mathrm{tf}_{i,j} \times \mathrm{idf}_j \quad (11)$$

where traditionally $\mathrm{tf}_{i,j}$ is the term frequency (how often term j appears in D_i), $\mathrm{idf}_j = \log_2 \frac{N}{\mathrm{df}_j}$ the inverse document frequency, and df_i the document frequency (how often the term appears in the whole collection), and N the number of documents in the collection. In the peer index, $\mathrm{tf}_{i,j}$ measures how often two images i and j co-occur, while idf_j measures the general importance of image j depending on how many times this image co-occurs with other images in the collection.

Algorithm 2 Query processing with inverted index

```
for every query image q in Q do
    retrieve the postings list for q from the inverted index
    for each peer d indexed in the postings list do
        score(d) = score(d) + w_{d,q}
    end for
end for
Normalise scores.
Sort documents according to normalised scores.
```

The similarity between two documents (images) is traditionally computed based on the cosine between their term-vectors. The *cosine similarity* between a query vector Q and a document D_i is defined as

$$sim(Q, D_i) = \frac{\sum_{j=0}^{V} w_{Q,j} \times w_{i,j}}{\sqrt{\sum_{j=0}^{V} (w_{Q,j})^2 \sum_{j=0}^{V} (w_{i,j})^2}} \quad (12)$$

where V is the total number of terms, and $w_{Q,j}$ is the weight of term j in the query.

However, since the vocabulary in this case is very large ($V = N$) and the vectors can be expected to be very sparse, the exact vector-space model is expensive to implement. Instead of storing the whole matrix M, we create an inverted index, in which each term (image) is stored with its postings list. The postings list for a particular term is a list of documents that contain this term (together with the term weight in the document). The posting list thus contains a reference to all peers of a given image. Instead of having to compare N vectors given a particular query, the inverted index facilitates a fast computation of the relevant results. The algorithm is specified in Algorithm 2. The normalisation discards the effect of document length and document scores. The exact normalisation of scores is expensive again, since it requires to access *every* term (see denominator in Equation 12). To approximate the effect of normalisation, we can base it on the number of terms in a document [9]. The document scores in Algorithm 2 are then normalised by

$$\mathrm{score}(d) = \frac{\mathrm{score}(d)}{\sqrt{\text{number of terms in d}}} \quad (13)$$

Since the peer index is symmetric, the number of terms of a document is equal to the number of documents containing the term, which is given by the postings list size.

4.2.1 Relevance Feedback

4.2.1.1 Short-term Learning in an Inverted Index.

Rocchio's algorithm is widely used in text retrieval to incorporate relevance feedback [13]. The objective of this method is to move the query point closer towards relevant documents (in P) and further away from non-relevant documents (in N). In the case of an inverted index Rocchio's method can be implemented efficiently by adjusting the weights for each query term $t \in Q \cup P \cup N$:

$$w'_{Q,t} = \alpha\ w_{Q,t} + \beta \left(\frac{1}{|P|} \sum_{t \in P} w_{Q,t} \right) - \gamma \left(\frac{1}{|N|} \sum_{t \in N} w_{Q,t} \right) \quad (14)$$

where the parameters α, β, γ are typically chosen experimentally.

4.2.1.2 Long-term Learning.

In addition to adjusting the query term weights on a per-query basis, the overall peer weights in the inverted index are updated. So whenever an image is added to a group, the new image is added to

the postings list of all group images and all group images are added to the postings list of the new image. If it an image already exists in a postings list, its weight is incremented by the relevance score (typically 1). In reverse, if an image is considered a negative example to the current group, the weights between the negative example and the group images are decremented. Parallel to the long-term learning strategy in the ICG (Section 3.4.1) other negative strategies could be implemented.

4.3 Textual Feature

Similarly to the peer feature, the textual annotations are stored in an inverted index. The querying and relevance feedback process are the same as for the peer feature, with the exception of the long-term learning facility.

4.4 Combination

In the *voting approach (VA)* each feature is treated as a voter producing its own individual ordering of candidates (images). The final combined list is computed based on the *median rank aggregation* method proposed in [2]. It assumes a number of independent voters that rank a collection based on the similarity to a query. The aggregation rule then sorts the database objects with respect to their median of the ranks they receive from the voters. The idea of the MEDRANK algorithm can be sketched as follows. Assume each voter produces a ranked list. From each list, access one element at a time, until a candidate is encountered in the majority of the lists, place this candidate as the top ranked of the final list. The second candidate will be placed second top, and so on. Continue until top k candidates are found, or there are no more candidates.

5. EXPERIMENTAL METHODOLOGY

5.1 Image Dataset

We employed a subset of the Corel collection (CD 1, CD 4, CD 5, and CD 6 of the Corel 1.6M dataset), containing 12800 photographs in total.

5.1.1 Features

Visual Features The following 6 low-level colour, texture and shape features are implemented (feature dimension in brackets):
Colour: Average RGB (3), Colour Moments (9)[17];
Texture: Co-occurrence (20), Autocorrelation (25), and Edge Frequency (25) [16];
Shape: Invariant Moments (7) [5].

Textual Annotation Approximately 7700 images contain annotations (obtained from [24]). Both the keyword as well as the description field are used to annotate the images.

Peer Information Previous user experiments are the basis for the peer information used in this work [22]. The resulting groups from each query session in these experiments are identified in the user logs and inserted as links into the ICG and in the peer index used in the baseline.

5.1.2 Tasks

10 tasks representing different semantic concepts have been chosen and manually labelled for this evaluation (number of ground-truth images in brackets): Task 1—mount-ainous landscapes (549 images), Task 2—elephants (113), Task 3—tigers (103), Task 4—animals in the snow (220), Task 5—African wildlife (865), Task 6—underwater world (402), Task 7—skiing (100), Task 8—caves (200), Task 9—snowy mountains (244) and Task 10—autumn trees (203). Tasks 2 and 3 are almost identical to existing Corel categories with annotations. Tasks 7 and 8 are also derived from Corel categories, but none of their images are annotated. Further, tasks 1–6 have been used in previous user experiments [22], which are the basis for the usage information employed. Images belonging to these tasks contain on average 35 peer links. So some tasks benefit from annotations, some from peers, while others cannot use either.

5.2 Performance Measures

The performance is based on the traditional measures *precision* and *recall* in Information Retrieval [23]. We are primarily concerned with the quality of the recommendations, that is how many of the r returned images are relevant, where typically $r <= 100$ to allow the user to see all recommendations on one screen. The *precision* after the r-th image retrieved, $P(r)$, measures its composition of relevant and non-relevant images, and hence provides a good indication for this. $P(r)$ values are in the range $[0,1]$ (corresponding to 0-100%). The *recall* value measures how many of the total available relevant images are returned (also in the range $[0,1]$). The total number of relevant images found becomes an important performance measure when running the recommendation system over a number of feedback iterations.

6. RESULTS

To establish the retrieval effectiveness of the proposed approach, we compare different variations of the ICG to the separatist approach. The individual baselines are referred to as IND_v, IND_t, IND_p for the visual, textual and peer features, respectively. IND_{vt} denotes the combination of visual and textual features, while the combination of all three features is referred to as IND. The parameters of the ICG are fixed to $alpha = 0.6$ and $k = 25$ in these experiments (based on initial experiments not reported here).

The results are based on the average performance over 2999 queries in total (one query per ground-truth item per task). Table 1 compiles these results based on precision at rank 10, 20, 50, and at the rank of the number of relevant images per task, P(NR), and recall at rank 10, 50, 100 and at 0.5 precision, R(P05). First of all, the individual baselines IND_v,IND_t, and IND_p were considered. The textual feature on its own outperforms all other features both in terms of precision as well as recall[2]. The visual features are especially poor. Next, the baseline is compared to the ICG when no peer information is available (ie no previous interaction has been recorded). The results in Table 1 show that while IND_{vt}'s performance is initially better than ICG's performance, ICG outperforms IND_{vt} when considering a larger result set. Finally, the peer information is added to the baseline and the ICG. The results are shown in the last two columns of Table 1. This time ICG_p significantly[3] outperforms the baseline. Also note that ICG *without* peer information eventually manages to retrieve more relevant images than the baseline *with all three features* (P(NR), R(100), R(P05)), although the baseline is more precise up to the top 20 (P(10), P(20)).

Figure 2 shows the distribution of P(NR) values. It is interesting to see that the individual baselines contain many outliers. This shows that they can perform really well for some queries (ie for queries whose result set is annotated or ones that were previously captured by peer information). However, if this is not the case, the individual feature cannot contribute any relevant results and a combination is necessary.

Moreover, we would like to draw attention to a hypothetical comparison to the GCap approach proposed in [12]. An extension of

[2]The dominance of textual features has also been shown on the TrecVid corpus [11].

[3]Statistical significance was calculated with the paired-sample t-test, which resulted in a significant difference of $p < 0.01$ between ICG_p and all other methods.

Table 1: Comparison between various baselines and ICG with and without peer information

Method	IND_v	IND_t	IND_p	IND_{vt}	ICG	IND	ICG_p
P(10)	0.18	0.58	0.29	0.50	0.42	0.58	*0.62*
P(20)	0.16	0.56	0.28	0.44	0.41	0.54	*0.59*
P(50)	0.14	0.51	0.28	0.36	0.40	0.48	*0.57*
P(NR)	0.09	0.24	0.02	0.21	0.29	0.26	*0.39*
R(10)	0.01	0.02	0.01	0.01	0.01	0.02	0.02
R(50)	0.01	0.03	0.02	0.02	0.02	0.03	0.03
R(100)	0.03	0.14	0.07	0.08	0.12	0.12	*0.15*
R(P05)	0.00	0.18	0.02	0.10	0.24	0.18	*0.36*

GCap for retrieval has already been discussed in [12], but it has not been shown how it would perform experimentally. If we consider that the basic graph construction before any user interaction is recorded is almost analogous with the GCap graph (apart from the fact that regions of images are used as visual nodes, while we propose to use the individual global features), we can also consider *ICG* as the baseline. Our results show that *ICG* (and thus GCap) performs well in comparison to the separatist approach. However, adding the peer information causes a dramatic increase in performance over all baselines. The long-term learning facility is thus essential for improving retrieval effectiveness.

6.1 Incorporating Relevance Feedback

After having compared the performance of one-shot queries, the next objective is to look at the performance over multiple relevance feedback iterations. The setup of these runs is the following: for each task 200 queries consisting of 3 example images are issued to the system and relevance feedback is performed over a total of 20 relevance feedback iterations. All of the images in the recommendation set, ie those amongst the top 10 retrieval results, are chosen for feedback. Both positive as well as negative feedback is used. The peer information is reset after each query.

The main results are compiled in Figures 3, 4 and 5, showing the development of P(10), P(100) and R(100) over RF iterations. In terms of precision[4], ICG_p outperforms the baseline *IND*. However, recall can only be improved by a significant margin after the 15th iteration. Even if the users might not always be prepared to ask for this many recommendations, the same effect will be noticed once the group size becomes sufficiently large. In this simulated setup, only a small number of images are added to the group in each iteration. In reality, the user can populate groups much faster resulting in larger groups for which ICG_p will return the better results. In order to verify this assumption we need to compare the performances with varying group size (number of query images).

We also looked at the alternative feedback strategy, which discounts negative feedback links by a factor of 5 (see Section 3.4.1), referred to as ICG_{pd}. This variation does not have a noticeable effect compared to the decrementing feedback strategy implemented in ICG_p. This is probably due to the fact that the peer information is still relatively sparse so that peer links do not have large weights in the first place. In this case, decrementing by 1 or dividing by a factor both have a similar effect.

ICG without peers shows an interesting behaviour. While its initial performance is close to ICG_p, it quickly drops off after the 10th iteration. Initially, both methods are able to find similar images based on annotations or visual features. After that the peer links are particularly useful to navigate to related images which are not necessarily similar and can therefore continue to retrieve relevant results.

[4]Please note that relevant images already found are not returned in following iterations, hence the drop in precision over iterations.

Table 2: Average group size after 20 RF iterations

GS	IND	ICG_p	ICG_{pw}	$ICG_{w:p}$	$ICG_{w:t}$	$ICG_{w:v}$
Task1	152.31	*189.10*	189.06	188.87	189.30	189.28
Task2	102.28	73.18	73.16	73.16	74.08	73.18
Task3	87.72	50.61	50.66	50.41	*65.08*	50.60
Task4	113.31	*146.33*	146.12	125.74	*153.13*	145.82
Task5	162.26	*185.74*	185.92	185.86	186.42	185.98
Task6	149.46	*189.22*	187.45	188.93	190.08	189.22
Task7	8.86	36.84	*43.26*	37.68	36.96	36.40
Task8	7.66	*42.48*	28.94	43.73	42.99	41.92
Task9	82.72	*88.94*	88.94	88.94	88.84	88.95
Task10	18.57	27.80	27.34	27.96	27.54	27.85
Avg	88.52	103.00	102.10	101.13	*105.44*	102.92

6.2 Introducing Feature Weights

As has been elaborated in Section 3.4, short-term learning can be implemented by adjusting link weights in the *ICG*. In the following, we present the results for feature weights on two different levels: (1) visual feature weights for weighting the importance of the visual nodes, and (2) overall feature weights between the three high level features represented in the graph.

6.2.1 Visual Feature Weights

The results of incorporating visual feature weights, referred to as ICG_{pw}, are plotted in Figures 3, 4 and 5. Overall, the visual feature weighting does not have a large effect in the graph. Table 2 reveals the average number of images found after 20 iterations for each task. Note that the maximum group size that can theoretically be reached a this point is 193 (3 initial images plus 10 images per iteration) and the minimum number of relevant images found for a particular task. These results show that for the two visual tasks, Task 7 and 8, which do not contain any annotations or peer information, the weighting actually influences the performance, albeit in a different way for the two tasks. The weights lead to an increase in the group size for Task 7, while the group size drops in comparison to ICG_p for Task 8[5]. Task 7 (skiing) can be better captured by the implemented visual features, since the images are very homogeneous in colour.

Nevertheless, it seems that the graph structure is the crucial factor in the random walk computation. We are currently investigating if boosting the visual feature weights by a factor could change this.

6.2.2 Overall Feature Weights

The final modification on link weights can me made to influence the overall importance of the three high-level features: peers, textual, and visual. We experimented with three sets of feature weights: $ICG_{w:p}$ denotes the weight ratio 3:1:1 between peer, text, and visual features, $ICG_{w:t}$ the ratio 1:3:1, and $ICG_{w:v}$ the ratio 1:1:3. The results are shown in Figures 6 and 7. It can be seen that the weights do not influence the performance early on in the feedback session. Later, the text feature is dominant, while strengthening the visual weights does not have an impact on performance either way. However, if the peer weights are emphasised it actually hurts the performance in the long run. The peer information is collected over time, involving a number of different users performing different tasks. Hence, it does not capture exactly the semantic relationships relevant for a new task and therefore can also lead the

[5]The same phenomenon was observed for the other performance measures, which cannot be discussed due to space limitations.

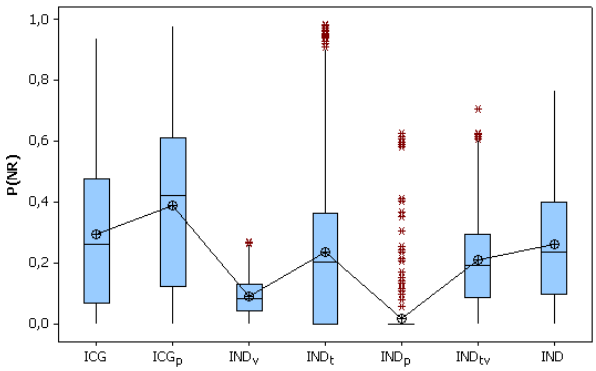

Figure 2: P(NR) for ICG and baselines

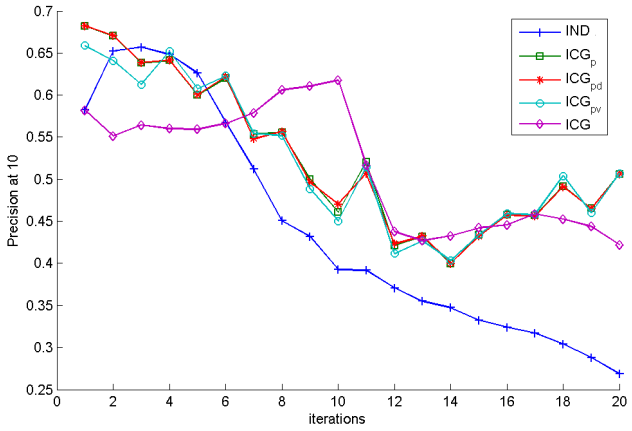

Figure 3: Precision at 10 over RF iterations

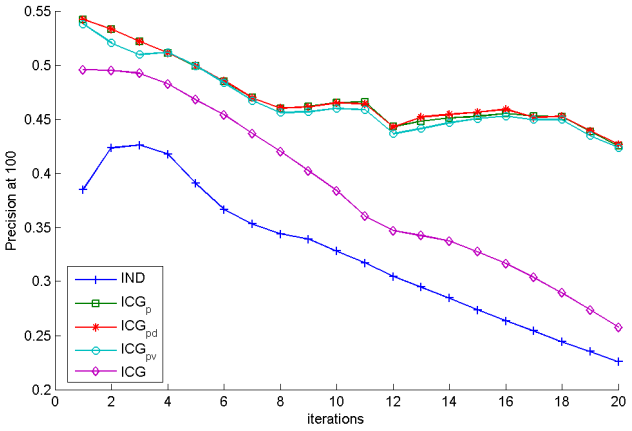

Figure 4: Precision at 100 over RF iterations

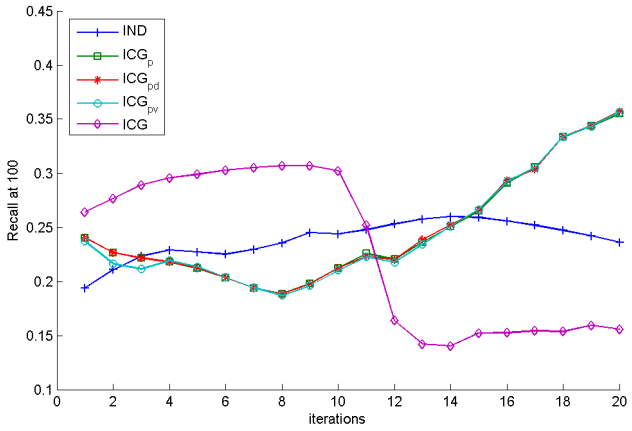

Figure 5: Recall at 100 over RF iterations

random walker in the wrong direction after the relevant peers have been found. The tasks for which this was the case here are Tasks 3 and 4 as is conveyed by the per-task results in Table 2. Other performance measures indicate that the textual weights can also boost the performance of Task 2. For example, the precision at 100 is 0.27 for $ICG_{w:t}$, compared to 0.19 for $ICG_{w:p}$ and 0.22 for ICG_p. Tasks 2–4 contain almost exclusively images with annotations, and therefore it is not surprising that relevant images can be found quicker by increasing the textual feature weight. Furthermore, Tasks 2 and 3 contain a lot of irrelevant peer links (since they are a subset of Task 5, but not all images relevant for Task 5 are also relevant for Tasks 2 and 3). All other tasks show little differences in performance.

Instead of simply using the feature weights as a scaling factor for updating link weights, we can also envisage a more drastic weighting technique. To ensure that a peer link is chose with probability w_p, a feature-attribute link with probability w_v, and a term-attribute link with probability w_t, all link weights of a particular feature i have to sum to the feature weight w_i. Therefore, during the normalisation stage of the adjacency matrix of the ICG, the first $|\mathscr{I}|$ columns of the feature-context matrix are normalised, such that all peer links sum to w_c. Similarly, all feature-attribute links sum to w_f, and all term-attribute links sum to w_t (in the first $|\mathscr{I}|$ columns). The remaining columns sum to 1.

These results show that knowledge about the tasks and dataset is necessary if one wants to exploit the relative importance of high-level features. An adaptive or interactive learning strategy for setting weights is very desirable. Further study is needed in order to find a solution to this problem. Nevertheless, the results in the previous sections have shown that, even without short-term learning capabilities, the graph represents the similarities and relationships between images very well, as its performance is significantly better than without the peer information and also than the baselines.

6.3 Computational Comparison

Another issue worth mentioning is the computational costs involved with these methods. Retrieval in the ICG requires (1) normalisation of the graph matrix, and (2) solving the random walk to find the stationary distribution. The former takes about 0.7 sec on average on a quad 3.2Ghz Xeon processor system with 4GB of RAM (The machine was also supporting three other processes simultaneously during most of the total experiment run.). The latter varies with the parameter α: the larger α is, the quicker the algorithm converges. For $\alpha = 0.1$ it takes on average 1.7 sec, for $\alpha = 0.6$ the solving time is 0.5 sec, and for $\alpha = 0.9$ it comes down to 0.2 sec. The total query time is around 2 sec for the ICG with α set to 0.6, which is the same as *IND* if only the top 100 results are merged. The costs for *IND* spiral dramatically if we attempt to merge more results. For example merging the complete result set, ie 12,800 images, took approximately 200 sec.

Preliminary runs on a much larger collection consisting of almost 40,000 images (created by adding Corel CDs 7 and 8) suggest that the query time of *ICG* ($\alpha = 0.6, k = 10$) increases to approximately 22 sec (1 sec for normalisation and 21 sec for solving). However, for this collection size there is a considerable increase in retrieval

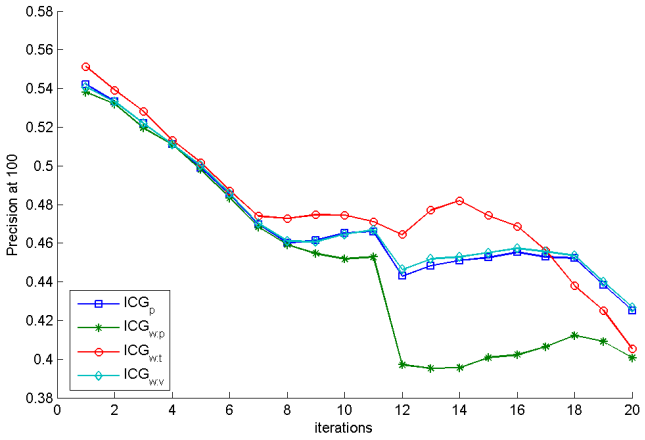

Figure 6: Precision at 100 of weighted ICG over RF iterations

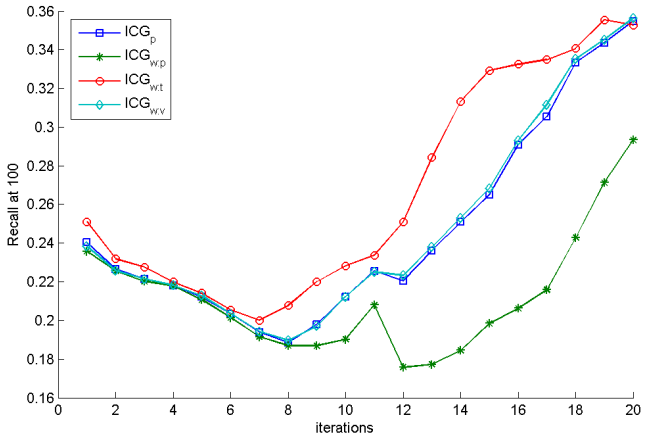

Figure 7: Recall at 100 of weighted ICG over RF iterations

effectiveness, too. For instance, *IND* achieves 0.21 precision at 10, compared to 0.45 for ICG_p for the 10 tasks used previously. Also recall at 100 more than doubles from 0.07 to 0.17. The superior performance justifies a further investigation of optimised algorithms for computing the random walk. This has been studied in the Web domain extensively, considering that there the algorithm has to deal with billions of documents [8].

7. CONCLUSIONS

We have introduced a model to learn semantic relationships between images obtained from user interaction as a contextual feature that allows long-term learning in an image retrieval and management environment. It is implemented in a graph-based model, the ICG, that encodes visual, textual and peer features together. The theory of random walks is employed to compute retrieval results.

Results of a simulated experiment have shown that the ICG is successful at integrating various features that are otherwise difficult to compare for adaptive image retrieval. ICG generally outperforms the baseline methods, which treat each feature separately for retrieval and then merge the final results. In particular, including the peer information significantly improves performance. This long-term learning capability is therefore an improvement over the simple graph model proposed by [12] for learning image annotations. Further, we have proposed and experimented with various short-term learning strategies that influence the link weights in the graph for the current query session. This improved the retrieval performance for certain tasks. The question still remains of how such weights could be adapted automatically.

8. ACKNOWLEDGMENTS

The research leading to this paper was supported by the European Commission under contract FP6-027026, Knowledge Space of semantic inference for automatic annotation and retrieval of multimedia content—K-Space.

9. REFERENCES

[1] S. Brin and L. Page. The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems*, 30(1–7):107–117, 1998.

[2] R. Fagin, R. Kumar, and D. Sivakumar. Efficient similarity search and classification via rank aggregation. In *Proc. of the ACM SIGMOD Int. Conf. on Management of Data*, pages 301–312, 2003.

[3] J. Han, M. Li, H. Zhang, and L. Guo. A memory learning framework for effective image retrieval. *IEEE Trans. Image Processing*, 14(4):511–524, 2005.

[4] J. He, H. Tong, M. Li, W.-Y. Ma, and C. Zhang. Multiple random walks and its application in content-based image retrieval. In *MIR '05: Proc. of the 7th ACM SIGMM Int. Workshop on Multimedia Information Retrieval*, pages 151–158,, 2005.

[5] M.-K. Hu. Visual pattern recognition by moment invariants. *IEEE Trans. Information Theory*, 8(2):179–187, Feb. 1962.

[6] G. Iyengar, P. Duygulu, S. Feng, P. Ircing, S. P. Khudanpur, D. Klakow, M. R. Krause, R. Manmatha, H. J. Nock, D. Petkova, B. Pytlik, and P. Virga. Joint visual-text modeling for automatic retrieval of multimedia documents. In *Proc. of the ACM Int. Conf. on Multimedia*, pages 21–30, New York, NY, USA, 2005.

[7] J. Jeon, V. Lavrenko, and R. Manmatha. Automatic image annotation and retrieval using cross-media relevance models. In *Proc. of the Annual Int. ACM SIGIR Conf. on Research and Development in Information Retrieval*, pages 119–126, New York, NY, USA, 2003.

[8] A. N. Langville and C. D. Meyer. Deeper inside pagerank. *Internet Mathematics*, 1(3):335–400, 2004.

[9] D. L. Lee, H. Chuang, and K. Seamons. Document ranking and the vector-space model. *IEEE Softw.*, 14(2):67–75, 1997.

[10] L. Lovasz. Random walks on graphs: A survey. *Combinatorics, Paul Erdos is Eighty*, 2:353–398, 1993.

[11] NIST. *Proc. of the TREC Video Retrieval Evaluation Conference (TRECVID2005)*, Gaithersburgh, MD, USA, Nov. 2005.

[12] J.-Y. Pan, H.-J. Yang, C. Faloutsos, and P. Duygulu. GCap: Graph-based automatic image captioning. In *Proc. of the 4th Int. Workshop on Multimedia Data and Document Engineering (MDDE 04), in conjunction with CVPR 04*, 2004.

[13] J. J. Rocchio. Relevance feedback in information retrieval. In *The SMART retrieval system: experiments in automatic document processing*, pages 313–323. Prentice-Hall, Englewood Cliffs, US, 1971.

[14] Y. Rui and T. S. Huang. Optimizing learning in image retrieval. In *IEEE Proc. of Conf. on Computer Vision and Pattern Recognition (CVPR-00)*, pages 236–245, Los Alamitos, June 2000.

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

[16] M. Sonka, V. Hlavac, and R. Boyle. *Image Processing, Analysis, and Machine Vision*. Thomson-Engineering, Toronto, Canada, 2nd edition, 1998.

[17] M. Stricker and M. Orengo. Similarity of color images. In *Proc. of the SPIE: Storage and Retrieval for Image and Video Databases*, volume 2420, pages 381–392, Feb. 1995.

[18] Z. Su and H.-J. Zhang. Relevance feedback in CBIR. In *Sixth Working Conference on Visual Database Systems (VDB'02), May 29-31, 2002, Bisbane, Australia*, pages 21–35, 2002.

[19] H. Tong, J. He, M. Li, C. Zhang, and W.-Y. Ma. Graph based multi-modality learning. In *Proc. of the ACM Int. Conf. on Multimedia*, pages 862–871, New York, NY, USA, 2005.

[20] J. Urban and J. M. Jose. EGO: A personalised multimedia management tool. In *Proc. of the 2nd Int. Workshop on Adaptive Multimedia Retrieval*, pages 3–17, 2004.

[21] J. Urban and J. M. Jose. Evidence combination for multi-point query learning in content-based image retrieval. In *Proc. of the IEEE Sixth Int. Symposium on Multimedia Software Engineering (ISMSE'04)*, pages 583–586, Dec. 2004.

[22] J. Urban and J. M. Jose. Evaluating a workspace's usefulness for image retrieval. *ACM Multimedia Sytems Journal (Special Issue on User-Centered Multimedia)*, 2006. accepted for publication.

[23] C. J. van Rijsbergen. *Information Retrieval*. Butterworth, London, 2nd edition, Jan. 1979.

[24] Berkley's list of Corel CDs, and image annotations. Available from http://elib.cs.berkeley.edu/corel/, last accessed in July 2006.