Accepted version of paper published in International Conference on Multimedia Retrieval (ICMR), 2015.
DOI: http://dx.doi.org/10.1145/2671188.2749336

Effective, Efficient, and Scalable Unsupervised Distance Learning in Image Retrieval Tasks

Lucas Pascotti Valem[1], Daniel Carlos Guimarães Pedronette[1],
Ricardo da S. Torres[2], Edson Borin[2], Jurandy Almeida[3]
[1]Dept. of Statistic, Applied Math. and Computing, Universidade Estadual Paulista (UNESP), Rio Claro, Brazil
[2]Institute of Computing, University of Campinas (UNICAMP), Campinas, Brazil
[3]Institute of Science and Technology, Federal University of São Paulo (UNIFESP), São José dos Campos, Brazil

ABSTRACT

Various unsupervised learning methods have been proposed with significant improvements in the effectiveness of image search systems. However, despite the relevant effectiveness gains, these approaches commonly require high computation efforts, not addressing properly efficiency and scalability requirements. In this paper, we present a novel unsupervised learning approach for improving the effectiveness of image retrieval tasks. The proposed method is also scalable and efficient as it exploits parallel and heterogeneous computing on CPU and GPU devices. Extensive experiments were conducted considering five different public image collections and several descriptors. This rigorous experimental protocol evaluates the effectiveness, efficiency, and scalability of the proposed approach, and compares it with previous methods. Experimental results demonstrate that high effectiveness gains (up to +29%) can be obtained requiring small run times.

Categories and Subject Descriptors

H.3.3 [**Information Search and Retrieval**]: Search process

General Terms

Experimentation, Performance

Keywords

content-based image retrieval; unsupervised learning; effectiveness; efficiency; scalability

1. INTRODUCTION

Advances in image acquisition technologies coupled with sharing and storage facilities have triggered a huge growth of image collections. Given the increasing amount of available digital images, the development of effective and efficient systems for indexing and organizing this visual content is mandatory. In this scenario, the use of Content-Based Image Retrieval (CBIR) systems for dealing with this challenge constitutes one of the most promising approaches.

The general goal of CBIR systems consists in retrieving a collection of relevant images according to their visual similarity to query patterns defined by users. A common retrieval task, for example, consists in retrieving similar images to a given query image by considering visual properties such as shape, texture, and color. In general, CBIR approaches extract low-level features of images generating a feature vector which represents them. The similarity between images is computed according to the distance between their correspondent feature vectors, using a distance measure (e.g., the Euclidean distance).

The creation of novel visual features and distance measures has supported the development of CBIR approaches for decades. This development, however, has not been addressing properly the *semantic gap* problem, which consists in the difficulties in mapping low-level features to high-level concepts. More recently, aiming at dealing with the semantic gap and improving the search effectiveness performance, other stages of the retrieval pipeline not directly related to low-level feature extraction have been considered [21].

In this scenario, unsupervised post-processing methods capable of computing a more effective distance measure among images have been proposed [16, 38]. The general objective of these methods consists in replacing pairwise similarities by more global affinity measures [39]. Several methods have been employed with this purpose, such as diffusion process [38, 39], graph-based learning methods [37], and iterative re-ranking approaches [28, 29]. Other methods improve the distance measures by capturing and exploiting the intrinsic manifold structure of the datasets [2, 15].

In fact, different approaches have demonstrated the capability of producing relevant gains in terms of quality of image searches. Nevertheless, most of the proposed approaches are evaluated considering only effectiveness aspects, ignoring efficiency and scalability properties. Focusing on real-word applications, however, the three aspects should be considered at the same time. In addition to the quality of the retrieval process, the time spent to obtain the results and the capability of handling growing image collections are also indispensable.

This paper presents a novel unsupervised distance learning method that considers these three relevant aspects. The proposed RL-Recommendation algorithm exploits the information encoded in the ranked lists through unsupervised recommendations among images. While the effectiveness

ICMR'15, June 23–26, 2015, Shanghai, China.

DOI: http://dx.doi.org/10.1145/2671188.2749336.

gains are comparable with other similar approaches, the algorithm requires very low computation efforts as it uses only a portion of the ranked lists. The use of only top positions of ranked lists allows for taking advantage of existing indexing schemes, making it suitable for large image collections. In addition, we also designed and implemented a parallel solution of the algorithm that considers heterogeneous computing using CPUs and GPUs.

An extensive set of experiments was conducted, considering five public datasets of different sizes and several image descriptors. Different experiments were conducted aiming at assessing the effectiveness, efficiency, and scalability of the proposed approach. The experimental evaluation demonstrates that the proposed method can achieve significant effectiveness improvements in several image retrieval tasks. Experimental results also show that the gains can be obtained for various size of image collections with very small run times. The proposed RL-Recommendation algorithm is also evaluated in comparison with other recently proposed methods and several state-of-the-art approaches considering a shape dataset commonly used for benchmarking.

The paper is organized as follows: Section 2 introduces the problem definition; Section 3 presents the proposed unsupervised learning algorithm; Section 4 shows the experimental results; finally, Section 5 discusses the conclusions and presents possible future work.

2. PROBLEM FORMULATION

This section aims at providing a formal definition of image retrieval model and the unsupervised distance learning problem considered. Let $\mathcal{C}$=$\{img_1, img_2, \ldots, img_n\}$ be an image collection, where n is the size of the collection $\mathcal{C}$.

Let $\mathcal{D}$ be an image descriptor that can be defined [9] as a tuple (ϵ, ρ), where ϵ: $\hat{I} \to \mathbb{R}^m$ is a function, which extracts a feature vector $v_{\hat{I}}$ from an image in $\hat{I}$; and ρ: $\mathbb{R}^m \times \mathbb{R}^m \to \mathbb{R}$ is a distance function that computes the distance between two images according to the distance between their corresponding feature vectors. The value of $\rho(\epsilon(img_i), \epsilon(img_j))$ defines the distance between two images img_i and img_j. The notation $\rho(i,j)$ is used for denoting this distance along the paper for simplicity and readability purposes.

The distance $\rho(i,j)$ among all images img_i, $img_j \in \mathcal{C}$ can be computed to obtain a squared $n \times n$ distance matrix A, such that $A_{ij} = \rho(i,j)$. The distance matrix A is used as the input for various unsupervised learning algorithms, but often cause scalability difficulties for large image collections, leading to storage and time complexity of at least $O(n^2)$.

An alternative representation of retrieval results is based on ranked lists. Based on the distance function ρ, a ranked list τ_q can be computed in response to a query image img_q. The ranked lists can contain information from the entire collection, and especially their top positions are expected to contain the most relevant images related to the query image. Therefore, one suitable strategy for speeding up the searching process consists in considering a subset of the L most similar images, where $L \ll n$ is the number of images at top positions of the ranked list. This is a useful strategy specially for large collections, where n is very large, and therefore τ_q is very expensive to compute.

The ranked list τ_q=$(img_1, img_2, \ldots, img_L)$ can be defined as a permutation of the image collection $\mathcal{C}_s \subset \mathcal{C}$, which contains the most similar images to query image img_q, such that and $|\mathcal{C}_s| = L$. A permutation τ_q is a bijection from the set $\mathcal{C}_s$ onto the set $[L] = \{1, 2, \ldots, L\}$. For a permutation τ_q, we interpret $\tau_q(i)$ as the position (or rank) of image img_i in the ranked list τ_q. We can say that, if img_i is ranked before img_j in the ranked list of img_q, that is, $\tau_q(i) < \tau_q(j)$, then $\rho(q,i) \leq \rho(q,j)$.

Taking every image $img_i \in \mathcal{C}$ as a query image img_q, we can obtain a set $\mathcal{R} = \{\tau_1, \tau_2, \ldots, \tau_n\}$ of ranked lists for each image of the collection $\mathcal{C}$. The objective of this work consists in proposing an algorithm for redefining the set of ranked lists $\mathcal{R}$ producing a more effective set $\mathcal{R}_e$. Therefore, the unsupervised distance learning algorithm can be defined as a function f_u, such that:

$$\mathcal{R}_e = f_u(\mathcal{R}). \tag{1}$$

The objective of function f_u is to exploit the set $\mathcal{R}$ by performing unsupervised recommendations based on information encoded in the top positions of ranked lists.

3. RL-RECOMMENDATION ALGORITHM

The proposed RL-Recommendation (*Ranked Lists-Recommendation*) algorithm is mainly based on the concept of supporting unsupervised recommendations among images. Recommendation approaches, originally created for automatically selecting items that match personal preferences, are simulated by the proposed algorithm in an unsupervised way. The recommendations are performed based on information encoded in ranked lists, in which images at top positions are recommended to each other. In this scenario, the recommendation means that the distance between two images should be decreased, and therefore, they should be moved up in the ranked lists of each other.

The presented method is closely related to the recently proposed Pairwise Recommendation [28] algorithm. However, our approach differs in important aspects. While the Pairwise Recommendation [28] requires the distance among all images in a dataset as data entry, our method requires only the top positions of ranked lists, which can be obtained taking advantage of indexing strategies. In addition, the proposed RL-Recommendation algorithm dispenses the use of clustering steps and requires a low number of iterations for convergence. As a result, the algorithm requires much less computing power being designed for parallel computation and becoming suitable for ever-growing real-world datasets.

The RL-Recommendation algorithm can be broadly described considering four main steps:

1. **Computing the Sparse Distance Matrix:** the expected input of the algorithm consists of the set ranked lists $\mathcal{R}$, which ensures scalability properties of the algorithm. However, since the unsupervised recommendations require distance scores, they are computed based on ranked lists. Only the distances among images at top positions of ranked lists are considered, leading to a sparse distance matrix A.
2. **Computing the Cohesion Measure:** the cohesion [28] measure provides an unsupervised estimation of effectiveness of ranked lists. The motivation is based on the conjecture that effective ranked lists have more authority for making recommendations. The cohesion measure is also used as a convergence criterion: the recommendations are performed while the cohesion of ranked lists are increasing.

3. **Performing Unsupervised Recommendations:** the top positions of ranked lists represent the information with the higher accuracy provided by image descriptors. This information, exploited for creating a top-k *image profile*, supports the unsupervised recommendations. If two images are contained in this profile, it constitutes an indication of similarity, producing a recommendation which reduces the distance between them.
4. **Sorting Ranked Lists:** the recommendations change the distances among images. Therefore, ranked lists must be updated to reflect the new ranking. A sorting procedure is performed aiming at updating the ranked lists according to the new computed distances.

Steps 2-4 are repeated while the average cohesion of ranked lists continues increasing above a given threshold. The parameter k, which defines the top-k positions used for cohesion and recommendations, is incremented at each iteration. Next sections describe in details each step and present the parallel solution proposed.

3.1 Computing the Sparse Distance Matrix

The expected input of the algorithm (as discussed in Section 2) consists of the set of ranked lists. In fact, since the unsupervised recommendations require distance scores, we propose to estimate distances based on the information given by the ranked lists. We consider only the distances among images at top-L positions of ranked lists, leading to a very sparse distance matrix A (with approximately $n \times L$ values used). The objective is to ensure the scalability of the algorithm.

The distance between two images is computed based on the sum of the reciprocal references at their ranked lists. Formally, given two images img_q and img_i, their distance $\rho(q,i)$ is defined as $\rho(q,i) = \tau_q(i) + \tau_i(q)$. According to this formulation, the distance matrix A (such that $A_{qi} = \rho(q,i)$) can be easily computed processing all ranked lists by summing up the reciprocal positions. However, the references are not symmetric and the ranked lists do not contain all images (only top L positions). Therefore, we can observe situations where img_i is in the ranked lists of img_q but the inverse is not true ($img_i \in \tau_q$ but $img_q \notin \tau_i$).

An alternative solution is proposed in Algorithm 1. In the first part (Lines 1-6), the ranked lists are processed and all pairs of images, which present references, have their distances set to $2 \times L$ (even if the references are not reciprocal). In the second part (Lines 7-11), each ranked list reference produces a decrement of L and an increment of the reference position ($\tau_q(i)$). In this way, for pairs of images which refer to each other at top L positions, the initial values are replaced by their respective positions. For pairs in which only one image refers the other, one of the two position remains L. The algorithm has the complexity of $O(n \times L)$.

Algorithm 1 Sparse Distance Matrix Computing.

Require: Blank matrix A and set of ranked lists $\mathcal{R}$
Ensure: Processed sparse distance matrix A

```
    //Sparse Distance Matrix Computing - Part I
 1: for all img_q ∈ C do
 2:   for all img_i ∈ τ_q do
 3:     A_qi ← 2 × L
 4:     A_iq ← 2 × L
 5:   end for
 6: end for
    //Sparse Distance Matrix Computing - Part II
 7: for all img_q ∈ C do
 8:   for all img_i ∈ τ_q do
 9:     A_qi ← A_qi + τ_q(i) − L
10:   end for
11: end for
```

3.2 Cohesion Measure

A cohesion measure [28] is used to provide an unsupervised estimation of the effectiveness of ranked lists. High cohesion scores indicate that ranked lists have more authority to recommend than others. The cohesion measure aims at assessing the quality of ranked lists by analyzing how images refer to each other in their ranked lists. The objective consists in evaluating the density of references among images at top positions of a given ranked list. For highly-effective ranked lists, the images at top positions are expected to refer to each other at the top positions of their ranked lists.

Let $k(\tau_i)$ be a subset of a ranked list τ_i with its top-k positions (or k-Nearest Neighbors). Let $img_j \in k(\tau_i)$ be an image of this subset, and let $k(\tau_j)$ be a subset of the ranked list of image img_j. Finally, let $img_p \in k(\tau_j)$ be an image at position $\tau_j(p)$ of the ranked list τ_j. The cohesion $c(\tau_i, k)$ of a ranked list τ_i can be defined as follows:

$$c(\tau_i, k) = \frac{\sum_{img_j \in k(\tau_i)} \sum_{img_p \in k(\tau_j)} w(\tau_j(p)) \times s(k(\tau_i), img_p)}{\sum_{img_j \in k(\tau_i)} \sum_{img_p \in k(\tau_j)} w(\tau_j(p))} \tag{2}$$

where function s determines if the image img_p (that belongs to subset $k(\tau_i)$) also belongs to subset $k(\tau_k)$ and is defined as follows:

$$s(k(\tau_i), img_p) = \begin{cases} 1, & \text{if } img_p \in k(\tau_i) \\ 0, & \text{otherwise.} \end{cases} \tag{3}$$

The function w takes as input the position of an image in a ranked list and gives high weights to images at the first positions ($w(p) = 1/p$). Notice that, if all referenced images are in the subset $k(\tau_i)$, we have a perfect cohesion. In this situation, the function s assumes the value 1 for all images and therefore cohesion is set to 1.

The cohesion measure is used not only for estimating the authority of ranked lists in making recommendations, but also for defining the convergence criterion and the stop condition of the algorithm. Regarding the convergence criterion, the average cohesion of all ranked lists is computed for each iteration. The algorithm is iteratively executed until the variation of cohesion is smaller than a threshold ϵ. Let $\bar{c}$ denotes the average cohesion of all ranked lists and let the superscript $^{(t)}$ denotes the current iteration, the algorithm is executed while the follow criterion is met:

$$(\bar{c}^{\,(t)} - \bar{c}^{\,(t-1)}) \geq (\bar{c}^{\,(t)} \times \epsilon). \tag{4}$$

3.3 Unsupervised Recommendations

The unsupervised recommendations [28] are associated with decreases of the distances among images and are defined based on information encoded in the ranked lists. The main idea behind the unsupervised recommendation given by a ranked list τ_i is: *"the image img_y is recommended to image img_x, if both img_x and img_y are in the top-k positions of the ranked list of τ_i."*

Each recommendation is associated with a different weight, which defines how much the distance should be de-

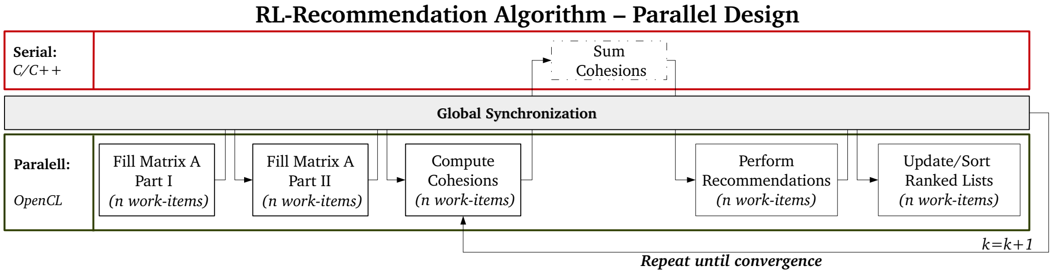

Figure 1: Design of the Parallel RL-Recommendation Algorithm.

creased. The recommendation weight considers the position of images in ranked lists and the authority estimation of the ranked lists, given by the cohesion measure. Algorithm 2 presents the unsupervised recommendation approach for a given ranked list τ_i.

Algorithm 2 Unsupervised Recommendations.

Require: Distance matrix A, ranked list τ_i and cohesion c_i
Ensure: Updated matrix A
1: **for all** $img_x \in k(\tau_i)$ **do**
2: $\quad w_x \leftarrow 1 - (\tau_i(x)/k)$
3: $\quad$ **for all** $img_y \in k(\tau_i)$ **do**
4: $\qquad w_y \leftarrow 1 - (\tau_i(y)/k)$
5: $\qquad w \leftarrow c_i \times w_x \times w_y$
6: $\qquad \lambda \leftarrow 1 - min(1, \alpha \times w)$
7: $\qquad A_{xy} \leftarrow min(\lambda A_{xy}, A_{yx})$
8: $\quad$ **end for**
9: **end for**

Values w_x and w_y (Lines 2 and 4) represent the weight assigned to images img_x and img_y in the recommendation. The weights are computed based on the position of images in the ranked lists: for images at top positions of the ranked list a higher weight is assigned. The weights associated with the first positions indicate where it is more likely to find the most similar images, that is, positions that represent more reliable recommendations. In Line 5, the weight w of a recommendation is computed. That represents the *reputation* of the recommendation. For computing w, we consider w_x, w_y and the cohesion c_i computed for the ranked list τ_i.

The λ coefficient combines information of positions and cohesion of the ranked list and is used for determining how the distances between img_x and img_y should be decreased. The constant α aims at adjusting the impact of recommendations. By increasing the value of α, the distances among images will decrease faster. A *min* function (Line 7) avoids negative values limiting the coefficient λ to 0.

3.4 Parallel Design

This section discusses the parallel design of the RL-Recommendation algorithm. We adopted the OpenCL standard, which is an open cross-platform, low-level API for parallel heterogeneous computing. The OpenCL code is executed on a computational *device*, which can be a CPU, GPU, or other accelerator. OpenCL supports both *data-parallel* and *task-parallel* programming models, as well as hybrid models.

Parallel pieces of code are defined in a *kernel*, which is a function declared in an OpenCL Programming Language (a subset of C language, with some restrictions and special keywords). The kernels are executed on an OpenCL *device* and each instance of a kernel running on a compute unit is called a *work-item*. The same code is executed in parallel by different *work-items*, and each *work-item* executes the code with different data.

Figure 1 illustrates the design of the Parallel RL-Recommendation Algorithm using six distinct kernels. Different kernels in OpenCL ensure global synchronization among all work-items, i.e., all work-items of a given kernel should finish so that the next can be started.

The first two kernels compute the distance matrix, as discussed in Section 3.1. Two kernels are used since all work-items of "*Part I*" should finish before the start of "*Part II*." The third kernel computes the cohesion measure for all ranked lists. Each work-item computes the cohesion for a given ranked list. The cohesion of ranked lists are serially summed up for computing the average cohesion used for checking the convergence criterion.

The unsupervised recommendations are computed by the fourth kernel. We also have n work-items, each one processing a ranked list. Although the recommendations processing are independent, different ranked lists can update concurrent positions of the Distance Matrix A causing the lost of some updates. A possible solution is to ensure exclusive access, but the overhead associated with fine grained synchronization in OpenCL is significant. Therefore, as suggested by other works [30], we allow direct updates to matrix A because the loss of some updates has a very low impact on the effectiveness of the algorithm (as demonstrated by confidence intervals of Section 4.4).

The last kernel executes a sorting procedure for updating the ranked lists. Since most of changes occur only in the beginning of the ranked lists [30], we use the insertion sort algorithm, whose complexity tends to be linear when the input is almost sorted.

Regarding memory transfers, the algorithm's data should be available in the device memory before the execution of kernels starts. The input and output data are composed of the set of ranked lists, which should be transferred before the beginning of the execution and copied back after its conclusion. In this work, we consider two memory transfer models defined by the OpenCL standard: write buffer and map buffer. For the write buffer model, the main memory transfer should be explicitly transferred to the device memory. On the other hand, the map buffer model consists in copying only memory pointers. Both strategies are evaluated in our experiments (Section 4.4).

4. EXPERIMENTAL EVALUATION

A large experimental evaluation was conducted for assessing the effectiveness, efficiency, and scalability properties of the proposed method. Section 4.1 describes datasets and image descriptors considered. Section 4.2 discusses the impact of parameters. Section 4.3 presents the results of the effectiveness evaluation, while Section 4.4 presents the efficiency evaluation. Section 4.5 discusses scalability aspects.

Table 1: Datasets and images descriptors used in the experimental evaluation.

Dataset	Size	Type	General Description	Descriptors	Effectiv. Measure
Soccer [36]	280	Color Scenes	Dataset composed of images from 7 soccer teams, containing 40 images per class	Border/Interior Pixel Classification (BIC) [32], Auto Color Correlograms (ACC) [14], and Global Color Histogram (GCH) [33]	MAP (%)
MPEG-7 [18]	1,400	Shape	A well-known dataset composed of 1400 shapes divided in 70 classes. Commonly used for evaluation of unsupervised distance learning approaches.	Segment Saliences (SS) [10], Beam Angle Statistics (BAS) [3], Inner Distance Shape Context (IDSC) [19], Contour Features Descriptor (CFD) [27], Aspect Shape Context (ASC) [20], and Articulation-Invariant Representation (AIR) [12]	MAP (%), Recall@40
Brodatz [6]	1,776	Texture	A popular dataset for texture descriptors evaluation composed of 111 different textures divided into 16 blocks	Local Binary Patterns (LBP) [25], Color Co-Occurrence Matrix (CCOM) [17], Local Activity Spectrum (LAS) [34]	MAP (%)
N-S [24]	10,200	Objects/ Scenes	Composed of 2,550 objects or scenes. Each object/scene is captured 4 times from different viewpoints, distances, and illumination conditions	ACC [14], BIC [32], Color and Edge Directivity Descriptor (CEED) [7], Fuzzy Color and Texture Histogram (FCTH) [8], Joint Composite Descriptor (JCD) [40], Scale-Invariant Feature Transform (SIFT) [22]	N-S score
ALOI [11]	72,000	Objects	Images from 1,000 classes of objects, with different viewpoint, occlusion, and illumination conditions.	ACC [14], BIC [32], GCH [33], Color Coherence Vectors (CCV) [26], Local Color Histograms (LCH) [23]	MAP (%)

4.1 Datasets, Descriptors, and Experimental Setup

The experimental evaluation was conducted on five different datasets with diverse characteristics and size ranging from 280 to 72,000 images. The evaluation also used 18 different local and global descriptors, considering shape, color, and texture properties. Table 1 summarizes information about datasets and descriptors.

All images of each dataset are considered as query images in the effectiveness evaluation. The Mean Average Precision (MAP) was used as effectiveness measure for most of datasets, except for the N-S collection [24]. For the MPEG-7 [18] dataset, the Recall@40 is also considered in addition to MAP. For the ALOI dataset, we used the BP-Tree [1] indexing structure for the computation of the ranked lists.

The efficiency evaluation experiments considered the average run time of 10 executions and 95% confidence intervals. The hardware environment is composed of a CPU Intel Xeon CPU E3-1240 and a GPU AMD Radeon HD 7900 Series. The software environment is given by the operating system Linux 3.11.0-15 - Ubuntu 12.04 and OpenCL 1.2 AMD-APP.

4.2 Impact of Parameters

The RL-Recommendation algorithm considers three parameters: *(i)* k: number of initial neighbors; *(ii)* α: a constant that defines the weight of recommendations; and *(iii)* ϵ: the threshold parameter used for the convergence criterion. A set of experiments were conducted for evaluating the influence of different parameter settings on the retrieval scores for defining the best parameters values.

The first experiment aims at analyzing the impact of parameters k and ϵ. We computed the MAP scores ranging the parameter k in the interval $[0, 20]$ and the parameter ϵ from 0.005 to 0.02. The MPEG-7 [18] dataset and the CFD [27] shape descriptor were used in the experiment. The analysis of the variation of the MAP according to k and ϵ is shown in the surface illustrated in Figure 2. We can observe small variations on MAP scores, from 90% to 92%, which demonstrates the robustness of the proposed method for different parameters settings. The best effectiveness results were obtained for $k = 8$ and $\epsilon = 0.0125$. We used $\alpha = 2$, as suggested in [28]. These values were used in most of the experiments. Only the N-S [24] and ALOI [11] datasets used $k = 4$ and $k = 40$, respectively. Both datasets have very small and large number of images per class.

The second experiment analyzes the constant L, which defines the size of ranked list used as input. As previously discussed, the RL-Recommendation algorithm does not require the use of the entire ranked list. An experiment was conducted using the MPEG-7 [18] dataset for verifying the impact of the size of ranked lists on the effectiveness results. We computed the MAP scores ranging L in the interval $[50, 1400]$. The results for three different descriptors are shown in Figure 3.

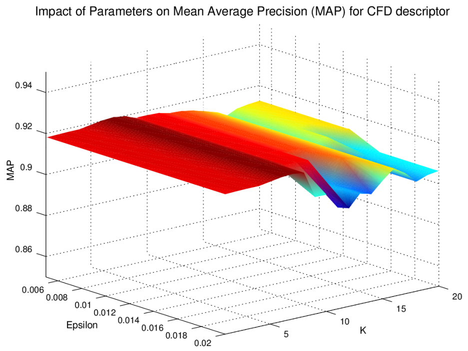

Figure 2: Impact of parameters k and ϵ on effectiveness.

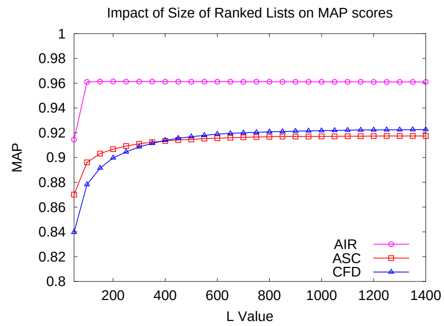

Figure 3: Impact of size of ranked lists L on effectiveness.

The curve behavior reveals that a small subset of the dataset (low L values) is enough for producing good effectiveness results. In addition, for values of L greater than 400, the MAP scores stop increasing expressively for all the descriptors. We used the value of $L = 400$ for most of experiments.[1]

[1]The value of L used for the Soccer [36] dataset is limited by the dataset size ($L = 280$). For the ALOI [11] and N-S datasets we used $L = 7200$ and $L = 200$, respectively.

Table 2: Effectiveness evaluation of the proposed RL-Recommendation algorithm considering various datasets and descriptors (MAP as score).

Descriptor	Dataset	Original MAP	Pairwise Recom. [28]	RL-Recom. Serial	RL-Recom. Parallel GPU	Gain
SS [10]	MPEG-7	37.67%	39.90%	48.68%	48.64% ± 0.0062	+29.22%
BAS [3]	MPEG-7	71.52%	77.65%	79.58%	79.57% ± 0.0047	+11.27%
IDSC [19]	MPEG-7	81.70%	86.83%	88.80%	88.78% ± 0.0067	+11.86%
CFD [27]	MPEG-7	80.71%	91.38%	91.39%	91.37% ± 0.0055	+13.23%
ASC [20]	MPEG-7	85.28%	91.80%	91.34%	91.32% ± 0.0050	+7.11%
AIR [12]	MPEG-7	89.39%	95.50%	96.12%	96.12% ± 0.0071	+7.53%
GCH [33]	Soccer	32.24%	32.35%	34.38%	34.44% ± 0.0340	+6.64%
ACC [14]	Soccer	37.23%	40.31%	41.23%	41.20% ± 0.0239	+10.74%
BIC [32]	Soccer	39.26%	42.64%	45.15%	45.17% ± 0.0693	+15.00%
LBP [25]	Brodatz	48.40%	51.92%	51.26%	51.24% ± 0.0047	+5.91%
CCOM [17]	Brodatz	57.57%	66.46%	64.34%	64.32% ± 0.0059	+11.76%
LAS [34]	Brodatz	75.15%	80.73%	79.71%	79.71% ± 0.0031	+6.07%

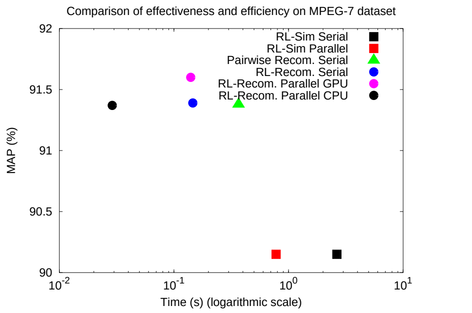

Figure 4: Effectiveness and efficiency analysis on the MPEG-7 [18] dataset.

Table 3: Effectiveness evaluation on the N-S [24] dataset, considering the N-S score.

Descriptor	Type	Original Score	RL-Recom.	Gain
ACC [14]	Color	3.36	3.53	+5.06%
BIC [32]	Color	3.04	3.15	+3.62%
CEED [7]	Color/Text.	2.61	2.72	+4.21%
FCTH [8]	Color/Text.	2.73	2.80	+2.56%
JCD [40]	Color/Text.	2.79	2.88	+3.23%
SIFT [22]	Local	2.54	2.88	+13.39%

Table 4: Effectiveness evaluation on the ALOI dataset [11], considering MAP as score.

Descriptor	Original MAP	Baseline: RL-Sim [13]	RL-Recom.	Gain
ACC [14]	44.15%	46.12%	50.11%	+13.50%
BIC [32]	71.95%	78.84%	80.35%	+11.67%
CCV [26]	47.77%	50.96%	53.52%	+12.04%
GCH [33]	50.87%	53.14%	55.81%	+9.71%
LCH [23]	58.85%	66.03%	72.63%	+23.42%

4.3 Effectiveness Evaluation

This sections aims at assessing the effectiveness of the proposed algorithm. A large set of experiments was conducted, considering various datasets and several descriptors.

Table 2 presents the MAP results for three datasets, considering shape, color, and texture features. We report the MAP scores for both serial and parallel GPU implementation of the RL-Recommendation algorithm. For the parallel GPU execution, the result is an average of 10 executions with the respectively 95% confidence interval. Notice that the confidence intervals are very small, indicating a low variation among different executions. The relative gain is computed based on serial execution. Very significant gains are observed for most of descriptors, ranging from +5.92% to +29.22%. For comparison purposes, we also reported the MAP scores of the Pairwise Recommendation [28] algorithm. The effectiveness results of the RL-Recommendation are significantly superior for most of the descriptors.

The effectiveness results considering the N-S [24] dataset are presented in Table 3. For this dataset, the N-S retrieval score between 1 and 4 is computed. This score corresponds to the number of relevant images among the first four image returned (the highest achievable score is 4). The N-S [24] is a very challenging dataset for unsupervised learning algorithms due to the small number of images per class (only 4). Despite this characteristic of the dataset, the RL-Recommendation achieved gains ranging from +2.56% to +13.39%.

Table 4 presents the MAP scores considering the ALOI [11] dataset. The ALOI [11] dataset is the biggest considered in the experiments and used indexing structures of the ranked lists computation. The gains obtained by the RL-Recommendation are also very significant for this dataset ranging from +9.71% to 23.42%. We also report the MAP scores for a recent baseline, the RL-Sim [13] algorithm. Notice that the RL-Recommendation effectiveness results are superior for all the descriptors.

A joined effectiveness and efficiency analysis conducted on the MPEG-7 [18] dataset is presented in Figure 4. The Pairwise Recommendation [28] and the RL-Sim algorithms [29, 31] are considered as baselines. The position of algorithms in the graph is given by the MAP score and the run time. Therefore, an ideal algorithm, with high effectiveness and low run time, is positioned at the top-left corner of the graph. Notice that the RL-Recommendation Algorithm (serial and parallel) occupies the best positions.

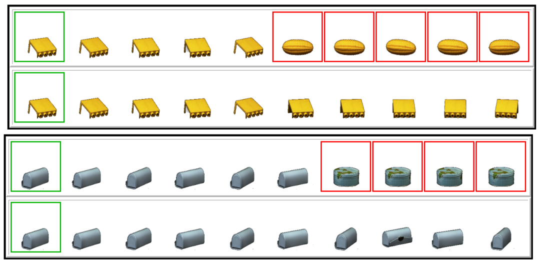

Figure 5: Impact of the RL-Recommendation on the ALOI [11] dataset: two ranked lists before and after the algorithm (query images with green borders and wrong results with red borders).

Two visual examples of the impact of the RL-Recommendation algorithm on ranked lists are illustrated in Figure 5. The query images are presented in green borders and wrong results in red borders. The first line represents the original retrieval results and the second line, the results after the algorithm execution.

Finally, we also evaluate our method in comparison with several other state-of-the-art unsupervised learning methods. We use the MPEG-7 [18] and the Bull's Eye Score (Recall@40), commonly used for evaluation and comparison of post-processing methods. Table 6 presents the results. We can observe that the RL-Recommendation achieved comparable effectiveness results, despite of the low computational efforts required.

4.4 Efficiency Evaluation

We conducted a set of experiments aiming at evaluating the efficiency of the proposed RL-Recommendation algorithm. Several aspects were evaluated, considering various datasets, the serial and OpenCL parallel implementations,

Table 5: Efficiency evaluation: runtime (in seconds) of the RL-Recommendation for different devices and datasets.

Algorithm	Exec.	Device	Soccer [36]	MPEG-7 [18]	Brodatz [6]	N-S Dataset [24]
Pairwise Recom. [28]	Serial	CPU	0.1149 ± 0.00018	0.3663 ± 0.00094	0.6672 ± 0.00140	14.802 ± 0.11059
RL-Recommendation	Serial	CPU	0.0607 ± 0.00000	0.1462 ± 0.00021	0.1108 ± 0.00102	0.1868 ± 0.00018
RL-Recommendation	Parallel	GPU[1]	0.1380 ± 0.00642	0.1401 ± 0.00250	0.1004 ± 0.00412	**0.0582 ± 0.00633**
RL-Recommendation	Parallel	GPU[2]	0.1538 ± 0.01056	0.2438 ± 0.00371	0.2376 ± 0.00326	0.3754 ± 0.00604
RL-Recommendation	Parallel	CPU[1]	0.0131 ± 0.00100	0.0319 ± 0.00043	0.0299 ± 0.00129	0.1166 ± 0.00085
RL-Recommendation	Parallel	CPU[2]	**0.0128 ± 0.00104**	**0.0290 ± 0.00075**	**0.0284 ± 0.00114**	0.1149 ± 0.00055

Memory Transfer Model: [1]Write Buffer; [2]Map Buffer.

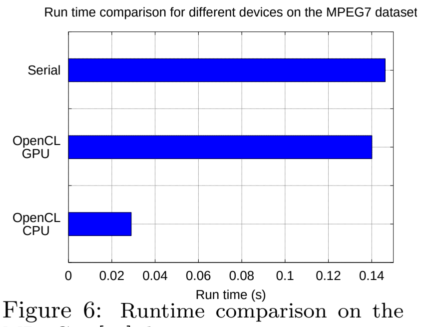

Figure 6: Runtime comparison on the MPEG-7 [18] dataset.

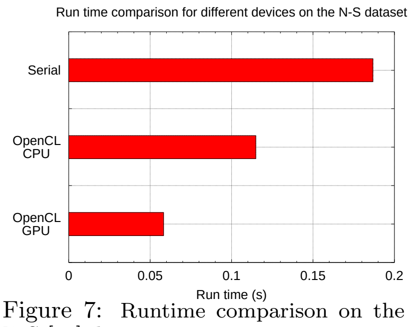

Figure 7: Runtime comparison on the N-S [24] dataset.

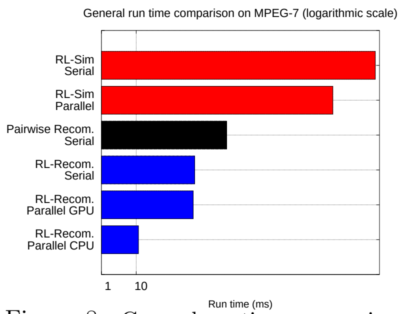

Figure 8: General runtime comparison on the MPEG-7 [18] dataset.

Table 6: Comparison of post-processing methods on the MPEG-7 [18] dataset - Bull's Eye Score (Recall@40).

Shape Descriptors		
DDGM [35]	-	80.03%
CFD [27]	-	84.43%
IDSC [19]	-	85.40%
SC [5]	-	86.80%
ASC [20]	-	88.39%
AIR [12]	-	93.67%
Post-Processing Methods		
Algorithm	**Descriptor(s)**	**Score**
Locally C. Diffusion Process [38]	IDSC	93.32%
Shortest Path Propagation [37]	IDSC	93.35%
Mutual kNN Graph [16]	IDSC	93.40%
RL-Sim [29]	CFD	94.13%
RL-Recommendation	**CFD**	**94.38%**
RL-Recommendation	**ASC**	**94.40%**
Locally C. Diffusion Process [38]	ASC	95.96%
Self-Smoothing Operator [15]	SC+IDSC	97.64%
Co-Transduction [4]	SC+IDSC	97.72%
Self-Smoothing Operator [15]	SC+IDSC+DDGM	99.20%
Pairwise Recommendation [28]	CFD+IDSC	99.52%
RL-Recommendation	**AIR**	**99.78%**
Tensor Product Graph [39]	AIR	99.99%

different devices (CPU, GPU), and memory transfer models. The reported results do not consider the OpenCL build and environment time, since the build can be executed once off-line and the environment time is constant independently of dataset sizes. We also present a comparison with other two unsupervised learning algorithms: the Pairwise Recommendation [28] and the RL-Sim algorithms [29, 31].

Table 5 presents the average run time and confidence intervals for the RL-Recommendation algorithm considering different criteria. For comparison, the run time for the Pairwise Recommendation [28] algorithm is also reported. The best performance for each dataset is highlighted in boldface. As we can observe, the performance results of the RL-Recommendation are very superior to the Pairwise Recommendation [28] algorithm. Considering only the serial executions, the RL-Recommendation is up to 79.3× faster, for the N-S [24] dataset. Considering the parallel implementations, the results are still more significant. The speedup obtained over the serial implementation ranges from 3.2× (N-S [24] dataset) to 5× (MPEG-7 [18] dataset). We can also observe that the performance of Parallel GPU in comparison with Parallel CPU increases according to size of the dataset, indicating that GPU devices can be better exploited for large amounts of computation. The overall run time for the whole N-S [24] dataset (with 10,200 images) is only $0.0582s$ for the Parallel GPU execution.

Figures 6 and 7 illustrate a comparison between the serial and parallel implementations for the N-S [24] and MPEG-7 [18] datasets, respectively. For the parallel implementations, different devices (CPU, GPU) are considered. Significant performance were obtained by the parallel implementation. Figure 8 presents a more general comparison of the RL-Recommendation (both serial and parallel) with baselines. The run time of Pairwise Recommendation [28] (serial) and the RL-Sim [29, 31] (serial and parallel) are reported. The MPEG-7 [18] dataset was considered for the experiment. Notice that, even using a logarithmic scale, the run time of the proposed RL-Recommendation (in blue) algorithm is significant smaller than other considered approaches.

4.5 Scalability Evaluation

This section evaluates the scalability of the proposed algorithm. We conducted an experiment varying the size of the ranked lists used as input and analyzing the behavior of the algorithm. The size of the ranked lists is defined by the constant L. Therefore, this constant defines an important trade-off control between effectiveness and efficiency. The ALOI [11] dataset and the LCH [23] descriptor were considered. We ranged the value of L from 70 to 7000, reporting for the average time of the unsupervised distance learning per ranked list. We also reported the same results for the RL-Sim [13] algorithm, as a baseline.

Figure 9 shows the results. We can observe very small average times for growing values of L for the RL-Recommendation Algorithm. This behavior enables the use of the algorithm in different datasets of any sizes, demonstrating its scalability characteristics. Even larger values of L were not considered because the effectiveness gains stabilizes after certain values of L (discussion in Section 4.2).

5. CONCLUSIONS

In this paper, we have presented a novel unsupervised

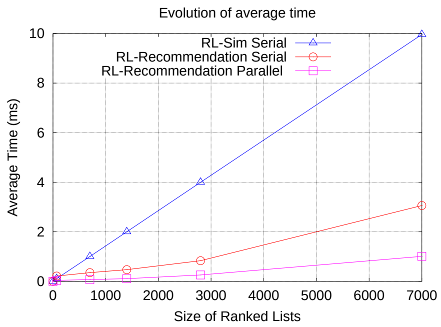

Figure 9: Scalability analysis: impact of size of ranked lists on average time for the ALOI dataset [11].

learning algorithm for image retrieval tasks. The main motivation consists in exploiting information of ranked lists for improving the effectiveness of CBIR tasks. The proposed approach differs from previous works, as it considers at the same time effectiveness, efficiency, and scalability issues. In addition to the significant effectiveness gains, the algorithm requires low computation efforts, presenting very positive efficiency and scalability properties. We have also exploited parallel computing for heterogeneous environments. We have conducted a large set of experiments on various public datasets and several descriptors. The results and comparisons with other recent state-of-the-art approaches demonstrate the effectiveness and efficiency of the proposed method. Future work includes the use of the proposed method for combining different descriptors and different modalities (e.g., visual and textual descriptors).

6. ACKNOWLEDGMENTS

The authors are grateful to São Paulo Research Foundation - FAPESP (grants 2013/08645-0, 2014/04220-8, and 2013/50169-1), CNPq (grants 306580/2012-8 and 484254/2012-0), CAPES, AMD, and Microsoft Research.

7. REFERENCES

[1] J. Almeida, R. da S. Torres, and N. J. Leite. BP-tree: An efficient index for similarity search in high-dimensional metric spaces. In *CIKM*, pages 1365–1368, 2010.
[2] J. Almeida, D. C. G. Pedronette, and O. A. B. Penatti. Unsupervised manifold learning for video genre retrieval. In *CIARP*, pages 604–612, 2014.
[3] N. Arica and F. T. Y. Vural. BAS: a perceptual shape descriptor based on the beam angle statistics. *Pattern Recognition Letters*, 24(9-10):1627–1639, 2003.
[4] X. Bai, B. Wang, X. Wang, W. Liu, and Z. Tu. Co-transduction for shape retrieval. In *ECCV*, volume 3, pages 328–341, 2010.
[5] S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *PAMI*, 24(4):509–522, 2002.
[6] P. Brodatz. *Textures: A Photographic Album for Artists and Designers*. Dover, 1966.
[7] S. A. Chatzichristofis and Y. S. Boutalis. Cedd: color and edge directivity descriptor: a compact descriptor for image indexing and retrieval. In *ICVS*, pages 312–322, 2008.
[8] S. A. Chatzichristofis and Y. S. Boutalis. Fcth: Fuzzy color and texture histogram - a low level feature for accurate image retrieval. In *WIAMIS*, pages 191–196, 2008.
[9] R. da S. Torres and A. X. Falcão. Content-Based Image Retrieval: Theory and Applications. *Revista de Informática Teórica e Aplicada*, 13(2):161–185, 2006.
[10] R. da S. Torres and A. X. Falcão. Contour Salience Descriptors for Effective Image Retrieval and Analysis. *Image and Vision Computing*, 25(1):3–13, 2007.
[11] J.-M. Geusebroek, G. J. Burghouts, and A. W. M. Smeulders. The amsterdam library of object images. *International Journal of Computer Vision*, 61(1):103–112, 2005.
[12] R. Gopalan, P. Turaga, and R. Chellappa. Articulation-invariant representation of non-planar shapes. In *ECCV*, volume 3, pages 286–299, 2010.
[13] D. C. Guimarães Pedronette, J. Almeida, and R. Da S. Torres. A scalable re-ranking method for content-based image retrieval. *Information Sciences*, 265:91–104, May 2014.
[14] J. Huang, S. R. Kumar, M. Mitra, W.-J. Zhu, and R. Zabih. Image indexing using color correlograms. In *CVPR*, pages 762–768, 1997.
[15] J. Jiang, B. Wang, and Z. Tu. Unsupervised metric learning by self-smoothing operator. In *ICCV*, pages 794–801, 2011.
[16] P. Kontschieder, M. Donoser, and H. Bischof. Beyond pairwise shape similarity analysis. In *ACCV*, pages 655–666, 2009.
[17] V. Kovalev and S. Volmer. Color co-occurence descriptors for querying-by-example. In *ICMM*, page 32, 1998.
[18] L. J. Latecki, R. Lakmper, and U. Eckhardt. Shape descriptors for non-rigid shapes with a single closed contour. In *CVPR*, pages 424–429, 2000.
[19] H. Ling and D. W. Jacobs. Shape classification using the inner-distance. *PAMI*, 29(2):286–299, 2007.
[20] H. Ling, X. Yang, and L. J. Latecki. Balancing deformability and discriminability for shape matching. In *ECCV*, volume 3, pages 411–424, 2010.
[21] Y. Liu, D. Zhang, G. Lu, and W.-Y. Ma. A survey of content-based image retrieval with high-level semantics. *Pattern Recognition*, 40(1):262 – 282, 2007.
[22] D. Lowe. Object recognition from local scale-invariant features. In *ICCV*, pages 1150–1157, 1999.
[23] H. Lu, B. Ooi, and K. Tan. Efficient image retrieval by color contents. In *ADB*, pages 95–108, 1994.
[24] D. Nistér and H. Stewénius. Scalable recognition with a vocabulary tree. In *CVPR*, volume 2, pages 2161–2168, 2006.
[25] T. Ojala, M. Pietikäinen, and T. Mäenpää. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *PAMI*, 24(7):971–987, 2002.
[26] G. Pass, R. Zabih, and J. Miller. Comparing images using color coherence vectors. In *ACM-MM*, pages 65–73, 1996.
[27] D. C. G. Pedronette and R. da S. Torres. Shape retrieval using contour features and distance optmization. In *VISAPP*, volume 1, pages 197 – 202, 2010.
[28] D. C. G. Pedronette and R. da S. Torres. Exploiting pairwise recommendation and clustering strategies for image re-ranking. *Information Sciences*, 207:19–34, 2012.
[29] D. C. G. Pedronette and R. da S. Torres. Image re-ranking and rank aggregation based on similarity of ranked lists. *Pattern Recognition*, 46(8):2350–2360, 2013.
[30] D. C. G. Pedronette, R. da S. Torres, E. Borin, and M. Breternitz. Efficient image re-ranking computation on GPUs. In *ISPA*, 2012.
[31] D. C. G. Pedronette, R. da S. Torres, E. Borin, and M. Breternitz. Rl-sim algorithm acceleration on GPUs. In *SBAC*, 2013.
[32] R. O. Stehling, M. A. Nascimento, and A. X. Falcão. A compact and efficient image retrieval approach based on border/interior pixel classification. In *CIKM*, pages 102–109, 2002.
[33] M. J. Swain and D. H. Ballard. Color indexing. *International Journal on Computer Vision*, 7(1):11–32, 1991.
[34] B. Tao and B. W. Dickinson. Texture recognition and image retrieval using gradient indexing. *JVCIR*, 11(3):327–342, 2000.
[35] Z. Tu and A. L. Yuille. Shape matching and recognition - using generative models and informative features. In *ECCV*, pages 195–209, 2004.
[36] J. van de Weijer and C. Schmid. Coloring local feature extraction. In *ECCV*.
[37] J. Wang, Y. Li, X. Bai, Y. Zhang, C. Wang, and N. Tang. Learning context-sensitive similarity by shortest path propagation. *Pattern Recognition*, 44(10-11):2367–2374, 2011.
[38] X. Yang, S. Koknar-Tezel, and L. J. Latecki. Locally constrained diffusion process on locally densified distance spaces with applications to shape retrieval. In *CVPR*, pages 357–364, 2009.
[39] X. Yang, L. Prasad, and L. Latecki. Affinity learning with diffusion on tensor product graph. *PAMI*, PP(99):1, 2012.
[40] K. Zagoris, S. Chatzichristofis, N. Papamarkos, and Y. Boutalis. Automatic image annotation and retrieval using the joint composite descriptor. In *PCI*, pages 143–147, 2010.