Fast Interactive Information Retrieval with Sampling-Based MDS on GPU Architectures

Hasmik Osipyan ✉, April Morton, and Stéphane Marchand-Maillet

University of Geneva, Department of Computer Science, Geneva, Switzerland
hasmik.osipyan@etu.unige.ch
{april.morton,stephane.marchand-maillet}@unige.ch

Abstract. Relevance feedback algorithms improve content-based image retrieval (CBIR) systems by effectively using relevant/non-relevant images labeled by users. The main constraint of these algorithms is the update time for large datasets. Opening the graphics processing units (GPUs) to general purpose computation provides an opportunity for performing parallel computation on a powerful platform. In this paper, we suggest a fast interactive interface for CBIR which includes the conventional ranked list view along with two additional views based on fast k-means clustering and fast sampling-based multidimensional scaling (SB-MDS) on a multi-core GPU architecture. We study the performance and efficiency of our framework on a collection of Maya syllabic glyph images. Experimental results show the improvement of retrieval performance at interactive speeds.

Key words: Image Retrieval, Relevance Feedback, Visualisation, User Interaction, GPU Architecture, CUDA

1 Introduction

Relevance feedback methods aim to improve CBIR systems by using information regarding relevant/non-relevant samples extracted from users through interactive visual interfaces [6]. Many recent techniques view relevance feedback as a machine learning problem, where relevant/non-relevant samples train a learner to classify images in the database as positive/negative [28]. Due to the statistical nature of many relevance feedback techniques, it is important that the system gains a sufficient sample of relevant/non-relevant images [26]. In order to obtain these labeled samples, standard relevance feedback interfaces request user input by displaying a ranked list of relevant images and providing an option for the user to mark chosen images as relevant/non-relevant [12]. Other relevance feedback interfaces include similarity-based visualisations that preserve distances between data items in order to give the user a better understanding of the dataset and/or allow him to directly mark images as relevant/non-relevant [23].

Due to time and resource constraints, users are typically limited to viewing a pre-determined subset of images in a ranked list or similarity-based grid. Though conventional visualisations greatly aid the user in providing relevance

feedback, their ability to deliver a sufficient amount of relevant items to a relevance feedback algorithm greatly depends on the number of relevant items in the first r retrieved or displayed results. Thus, there exists a need within a reasonable time frame to develop complementary visualisations in interactive information retrieval systems that potentially include more relevant images in initial results. This is a hard problem for large datasets, as classical methods for mapping high n-dimensional data into a lower dimensional space, via MDS, take $O(n^3)$ time [27]. Thus, it is important to consider both readability and speed as important factors when designing an interactive visual interface supporting a relevance feedback system. Fortunately, the increase in performance of graphics processing units (GPUs) combined with the use of the compute unified device architecture (CUDA) technology [17] allows users to perform parallel computations for different non-graphical problems. However, achieving a good utilization of a GPU with CUDA technology requires not only careful implementation, informed by a detailed understanding of the performance characteristics of the underlying hardware, but also the right usage of all the capabilities provided by these units.

In this paper, we suggest a fast interactive visual interface for relevance feedback which combines the standard ranked list view with two additional views that depend on the fast k-means and fast sampling-based multidimensional scaling (SBMDS) algorithms implemented on a GPU architecture. In combination, these views allow the user to choose from a greater amount of relevant results which are visually more informative and perform at interactive speeds because of the GPU architecture advantages. To illustrate the effectiveness of our fast interactive CBIR interface based on a GPU architecture, we assess the retrieval performance, visual informativeness, and speed by applying our system to a Maya syllabic glyph dataset. Results show that visual informativeness and speed are generally greatly improved, while retrieval performance is improved for specific queries. We assess the underlying factors behind the improved cases and provide recommendations regarding when and how to best use our fast interactive visual interface for relevance feedback.

The rest of this paper is organized as follows. Section 2 reviews related work in relevance feedback, k-means clustering, and MDS. Section 3 describes in detail the implementation of SBMDS on a GPU architecture. Finally we present the proposed fast interactive CBIR interface and experimental results in section 4 and conclude in section 5.

2 Related Work

Within the field of relevance feedback there are three common approaches including the query movement [19], weighted distance [7], and machine learning approaches [28]. The original query updating approach proposed by Rocchio [21] finds an optimal query by re-weighting the query vectors in a way which maximizes the difference between the average vector of the relevant/non-relevant documents. Das *et al.* [7] and Aksoy *et al.* [1] use a weighted distance approach

where the weights are the ratios of the standard deviations of the feature values for the whole database to the standard deviations of the feature values from the images selected as relevant by the user. Vasconcelos *et al.* [25] investigate Bayesian learning methods for relevance feedback, Ratsch *et al.* [20] take advantage of boosting techniques, and Tian *et al.* [24] utilize positive/negative labeled feedback samples for Support Vector Machine (SVM) learning. In our experiments, we follow the same weighted approach as Aksoy *et al.* [1] and rely on the L_1 distance function.

In order to extract relevant image labels from the user most systems rely on a ranked list view where the user can select all relevant documents within the first r results [8]. Melo *et al.* [15] implement a grid-based data visualisation with relevance feedback to support user interaction. Doloc-Mihu [8] also include similarity based-views of retrieved items using the Kamada-Kawai (KK) and Force-Directed (FD) algorithms. Though both views provide a good starting point for obtaining relevant images, the ranked list view may not contain a sufficient amount of relevant images and the similarity-based projections can be cluttered or very slow for large datasets.

In the last decade, the classical MDS algorithm and its variants have become very popular general data analysis techniques for visualisation [4]. Based on the needs of processing large-scale information data, different approaches have been developed to decrease computational complexity of existing classical MDS algorithms. Morrison *et al.* [16] show the effectiveness of their approach based on the iterative MDS spring model by reducing the computational complexity from $O(n^2)$ to $O(n\sqrt[2]{n})$. Yang *et al.* [27] suggest a new approximation method for the MDS algorithm by dividing the original matrix into sub-matrices and then combining the sub-solutions to obtain a final solution which reduces the time complexity to $O(nlogn)$. Although the above methods greatly reduce the time complexity of conventional MDS algorithms, more recent methods implemented on the GPU architecture reduce the time complexity even further. Park *et al.* [18] suggest a CUDA-based implementation of the classical MDS algorithm which computes results more than 100x faster than MDS general solutions on the CPU. Glimmer [11], another implementation on the GPU architecture, uses a stochastic approach [5]. Though the iterative approaches are more flexible because they find a stress-optimal configuration, they do not always guarantee an optimal solution because it is difficult to find an appropriate stopping point. Therefore, choosing a conventional classical MDS algorithm over the iterative MDS algorithm depends on the data and application [3].

The k-means algorithm minimizes the distance from each vector to it's nearest cluster centroid. The standard k-means algorithm, Lloyds algorithm [14], is an iterative refinement approach that greedily minimizes the sum of squared distances between each point and its assigned cluster center. Unfortunately, Lloyd's algorithm for k-means costs $O(nk)$ for a dataset consisting of n items partitioned into k clusters and thus is very expensive for applications with large nk. In order to resolve this problem, Arthur et. al [2] suggest an alternative seeding technique for selecting the cluster centroids that is $O(\log k)$. In addition to the many k-

means clustering techniques performed on the CPU, there exist many implementations developed on the GPU to further accelerate the algorithm. Farivar *et al.* [10] show over a 13x performance improvement compared to a baseline CPU implementation. Li *et al.* [13] suggest two different approaches for low-dimensional and high-dimensional datasets and show 3-8x faster results compared to the best reported GPU-based algorithms.

In our paper, we implement a fast interactive interface for CBIR that combines a new SBMDS algorithm on a GPU architecture with an existing k-means clustering algorithm on the GPU [13]. We use the asynchronous data transfer and stream techniques in CUDA to improve execution time and show that SBMDS is approximately 20x faster than CUDA-based fast multidimensional scaling (CFMDS).

3 Sampling-Based MDS on GPU Architectures

MDS is a set of techniques for analyzing large-scale data and providing a visual representation of proximities among data objects. Classical MDS algorithms take an $n \times n$ square matrix D containing all possible dissimilarities/similarities between n data objects and map these dissimilarities into a lower dimensional Euclidean space. The goal of classical MDS is to minimize the loss function

$$\sum_{i,j}(\|x_i - x_j\|) - d_{ij})^2 \rightarrow min, \tag{1}$$

which measures the lack of fit between the distances of the lower dimensional objects $\|x_i - x_j\|$ (where $\|\cdot\|$ is the Euclidian distance) and the dissimilarities d_{ij} of the full dimensional data objects.

In order to compute classical MDS, the input matrix D is first converted into a dot product matrix as shown in equation (2) below

$$J = I - \frac{1}{n}[1], B = -\frac{1}{2}JD^2J. \tag{2}$$

Next, the m largest positive eigenvalues $\gamma_1, \gamma_2, \ldots, \gamma_m$, and corresponding eigenvectors $\epsilon_1, \epsilon_2, \ldots, \epsilon_m$, of matrix B are extracted. The final low-dimensional data points $x_1, x_2, \ldots, x_n$ are computed from equation 3 below, where γ_m is the diagonal matrix of the m eigenvalues and E_m is the matrix of m eigenvectors

$$x = E_m\sqrt{\gamma_m}. \tag{3}$$

The time complexity $O(n^3)$ for solving the eigenpairs is the main bottleneck of the original MDS algorithm. Thus, the demand of working with large scale datasets creates a need for new algorithms with a smaller time complexity than the $O(n^3)$ complexity obtained by the classical MDS algorithm. Based on this

demand, Yang *et al.* [27], suggest a sampling-based fast approximation to the conventional MDS algorithm (SBMDS) which requires $O(n \log n)$ time.

The main idea behind the SBMDS algorithm is to first partition the $n \times n$ square input matrix D into $\frac{n}{p} \times \frac{n}{p}$ sub-matrices $D_1, D_2, \ldots, D_p$ and then apply the classical MDS algorithm to each sub-matrix D_i. Then, an alignment matrix M of size $sp \times sp$ ($s = 1 +$ estimated dimensionality of the dataset) is created by sampling s points from each of the D_i sub-matrices. After the MDS algorithm is run on the matrix M, an affine mapping A_i is computed by solving the linear least squares problem

$$A_i dMDS_i = mMDS_i. \tag{4}$$

In equation (4), $dMDS_i$ is the MDS solution for the sampled points from the sub-matrix D_i and $mMDS_i$ is the MDS solution for the matrix M. Equation (4) gives the mapping between D_i and M which is then applied to the rest of D_i to obtain the coordinates for all $\frac{n}{p}$ points. This process is applied recursively until the size of D_i is optimal to run MDS on.

There are two problematic areas of the SBMDS approach including: a) finding the optimal size for the sub-matrices, $\frac{n}{p}$, and b) finding the optimal number of sub-matrices to be joined together at each conquer step. Choosing the optimal size greatly affects the implementation on CUDA and on GPU architectures, where better results are obtained only for large-scale datasets with large sub-matrices. This is primarily because breaking down a dataset into very small portions generally has an inverse affect on program performance.

The main difference between the CPU and GPU at an architectural level is the memory hierarchy. The GPUs are able to better hide the memory latency as compared to the CPUs, which explains why GPUs have smaller cache sizes. Thus, only a few blocks can fit at any time in the cache. Consequently, the GPU memory organization is quite different than that of the CPUs. As a result of the extreme difference in access latencies between on-chip and off-chip memory, CUDA encourages programs to be written with usage of on-chip memory. In our work, we use this abstraction to get faster results compared to previous implementations and demonstrate the influence of shared memory on the performance of the algorithm. The two disadvantages of shared memory usage are the small amount of memory and accessibility only for threads within one block. Therefore, shared memory is not suitable for all types of algorithms and thus requires more effort from the programmer.

To fit the SBMDS algorithm on the GPU architecture we use two kernels (Fig. 1). In the first kernel, $MDSonGPU()$, MDS is calculated for each sub-matrix and, in the second kernel, $AffineMapping()$, the affine mapping is computed by solving the least squares problem.

Then, the data is loaded back from the GPU to the CPU. With this technique, the MDS and affine mapping calculation are done instantly. They are located in the local memory and since accessing the local memory is much faster than accessing global memory, the performance is improved. Both kernels are built on

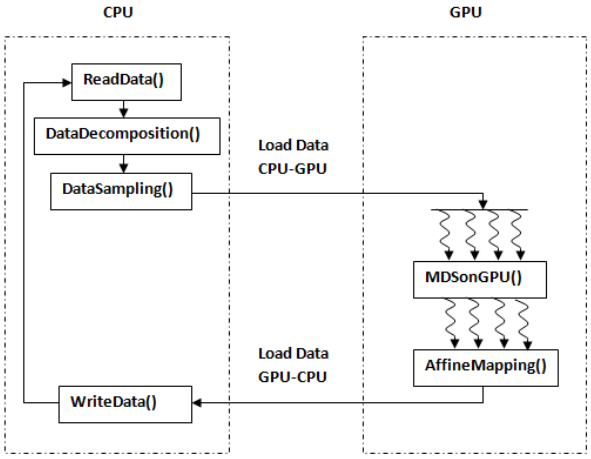

Fig. 1. SBMDS algorithm implementation on GPU architecture.

the GPU using $2D$ dimensional blocks with sizes 32 and $2D$ grid with sizes equal to the size of the sub-matrix divided by the block size. In our implementation, for gaining better performance, we also use other techniques in addition to CUDA including streams, asynchronous data transfers by using $cudaMemcpyAsync$ function and the CULA library [9].

4 Experimental Results

In this section, we explain in detail our fast interactive interface for relevance feedback and show the experimental results.

Consider the dataset $\mathbf{X} = \{x_1, \ldots, x_n\}$, the query image x_q, an arbitrary distance function d, the number of clusters in the k-means algorithm k and the number of iterations in the relevance feedback algorithm k_{rf}. Our interface consists of the query along with three individual visualisations (Fig. 2). Our goal is to allow the user to chose from a greater amount of relevant results, provide a more informative visual overview of the data, and update both views at interactive speeds.

The conventional ranked list view (Fig. 2.2-B) includes a list of the r closest images to the query x_q given the distance function d during iteration k_{rf} of the feature re-weighting relevance feedback algorithm. It shows the top 70 retrieval results, given the L_1 distance function, before the first round of relevance feedback.

After computing a 2-D fast SBMDS projection on a GPU architecture for the entire re-weighted dataset during iteration k_{rf} of a relevance feedback algorithm, a small subset of images are selected (Fig. 2.2-C). In order to automatically select this small subset of images, which should contain diverse images close to the query, we use the k-means clustering algorithm. The distance function d is

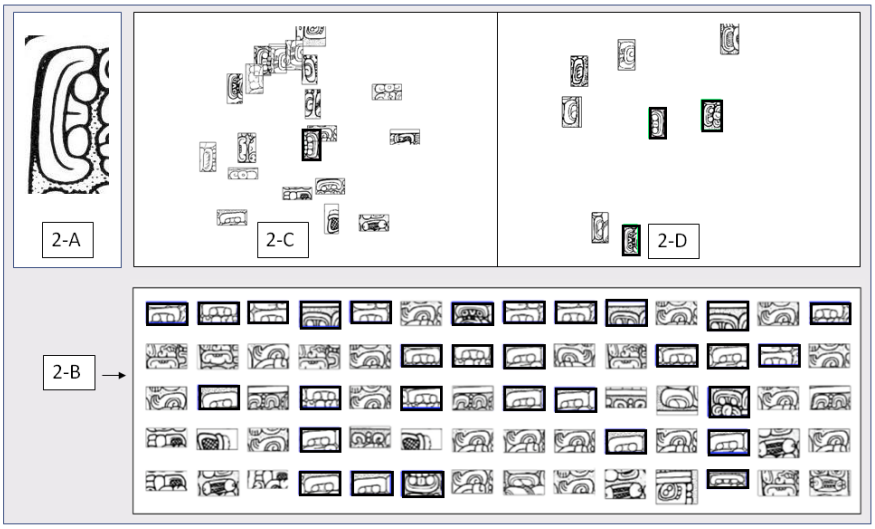

Fig. 2. Our visual interface for fast interactive CBIR: 2-A) The query image of a Mayan glyph from a Mayan syllabic dataset. 2-B) The ranked list of retrieval results in descending order. 2-C) The cluster representative view, where the glyph with black border is marked as relevant. 2-D) The cluster member view corresponding to the glyph with black border marked as relevant.

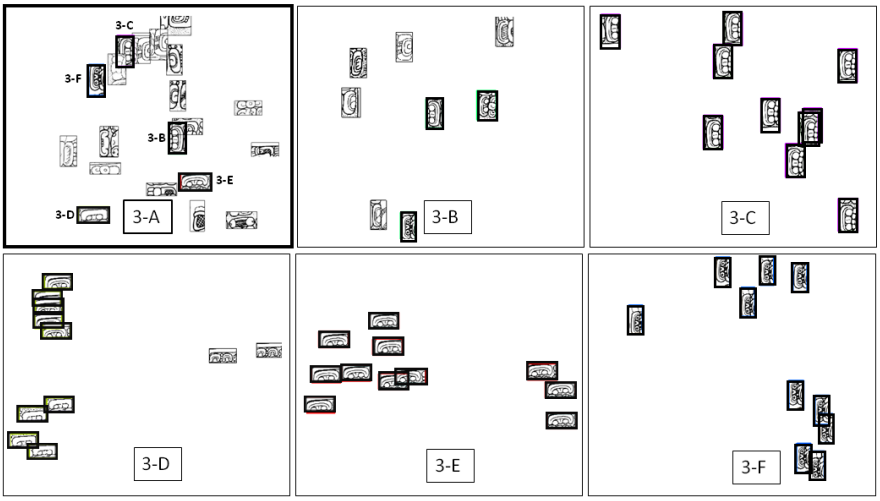

Fig. 3. The original cluster representative view along with each of the cluster member views: 3-A) The marked images represent all relevant images in the cluster representative view. 3-B-3-F) Represent the corresponding cluster member views and their marked relevant images (from the relevant images marked in Fig. 3-A).

used to determine the n_c nearest centroids to the query x_q, and is again used to select each of the n_c "distinct representative images" to be displayed for each of these centroids. Fig. 2.2-C shows an example of the cluster representative view where the user has highlighted one relevant item out of the $n_c = 20$ distinct representative images for the centroids obtained after performing k-means for $k = 260$. In addition, Fig. 3-A shows the same n_c items as in Fig. 2-C but with all five relevant images highlighted in various colors.

Given a chosen image from the cluster representative view, we automatically select a subset of images coming from the selected image's corresponding cluster to display in the cluster member view. The number of images automatically selected from each cluster, denoted n_s, is determined by the user. Then the distance function d is used to determine the n_s nearest images to the true cluster centroid, excluding the distinct representative image itself. In the case where the given cluster does not have the requested number of images to be viewed, the remaining lowest-ranked images in the ranked list view are selected to ensure that the user is still able to view n_s images. Fig 2-D shows the n_s images of the cluster member view corresponding to the relevant image with the black border selected in Fig 2-C. As in the first cluster representative view, the user can simply use the cluster member view as a way to understand the relationships among the selected items, or can choose to select relevant images out of this view rather than the ranked list view. In addition, Figs. 3-B-3F show each of the cluster member views corresponding to the relevant images marked in 3-A, along with their highlighted relevant images. Out of a total of 70 displayed images, there are 48 relevant images. Thus, we can see that in this example our complementary views allow the user to select a much higher sample of relevant images than the 30 relevant samples viewed in the first 70 results of the ranked list.

We evaluate our system by assessing the retrieval performance, visual effectiveness, and speed of the fast SBMDS and k-means implementation on the GPU on a dataset containing Mayan hieroglyphs originating from ancient Mesoamerica [22]. The task here is the careful interactive investigation of the database in view of supporting unknown glyph decipherment by visual and interpretative associations made by expert epigraphers. Each instance in the Mayan hieroglyphic dataset is a binary shape of a syllabic hieroglyph segmented from a large inscription. The dataset contains 6,240 images uniformly distributed over 24 visual classes where each class contains 10 different instances with 25 synthetic variations of each instance. Even if this dataset is of "modest" size, the potential number of associations that should be investigated at various levels makes this task of "large-scale" complexity.

To describe the shapes we first compute their HOOSC descriptors [22] and then perform k-means clustering to estimate a visual vocabulary of 1,000 visual words. From this visual vocabulary, we compute a normalized distribution of quantized descriptor indices to obtain a bag-of-visual words for each image. All our experiments are run under Windows 7, Intel Core i5-3230M CPU, 2.60 GHz equipped with the NVIDIA GT 730M video card (384 CUDA cores, 2 SM, 2GB global memory, 48KB shared memory).

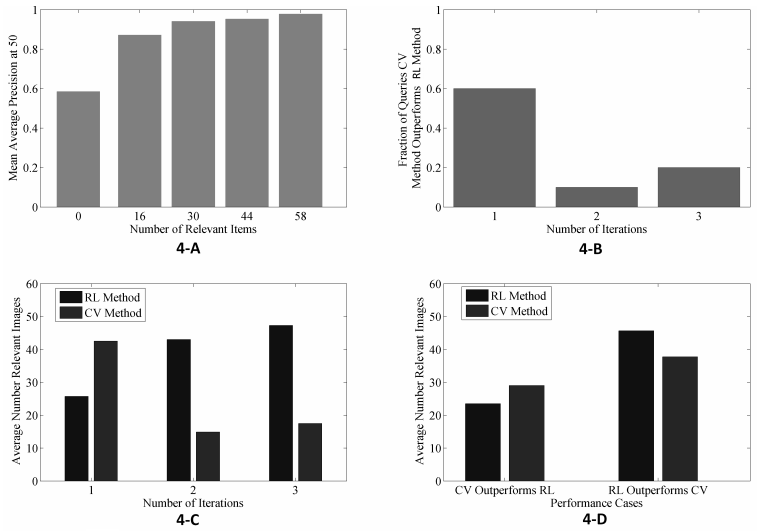

Fig. 4. 4-A) Number of relevant items versus the mean average precision at 50 over all queries in the experiment. 4-B) Number of iterations versus the percentage of queries when the CV method achieves greater precision than the RL method. 4-C) Number of iterations versus the average number of relevant items marked within CV/RL methods. 4-D) Average number of relevant items during the first iteration for CV/RL methods when the CV method achieves greater precision than the RL method and the RL method achieves greater precision than the CV method.

To implement our retrieval performance experiments we use the feature re-weighting approach for relevance feedback described in [1] using both the ranked list view for obtaining relevant images and our combined cluster representative and cluster member views for obtaining relevant images. All feature vectors are compared using an L_1-distance function which empirically performs better on the proposed datasets. In addition, to be consistent, we use the same L_1-distance function for all computations requiring a distance function. For brevity, from this point forward we denote the ranked list-based method for obtaining relevant glyphs as the RL method and our combined view-based method as the CV method.

To carry out the experiment, the original 10 instances from each class are used as queries and each of the corresponding 25 synthetic copies are excluded from retrieval results. In addition, in order to simulate user behavior for selecting relevant images we use a pseudo-relevance feedback technique in both the ranked list implementation and our new visual implementation. More specifically, when using the RL method to extract relevant images we automatically select all relevant images within the top $r = 70$ retrieval results during each iteration of relevance feedback. When using the CV method we also automatically select all relevant images out of $r = 70$ (to ensure comparability) and thus let $n_c = 20$ and

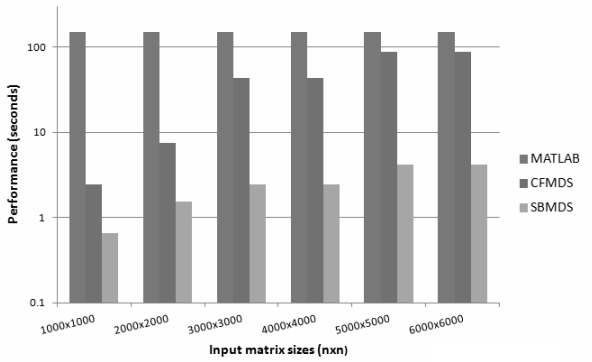

Fig. 5. Performance of SBMDS on GPU architecture compared with MATLAB and CFMDS (log scale seconds)

let $n_s = \left\lfloor \frac{50}{q} \right\rfloor$, for each of the q chosen images in the cluster representative view. As above, in the case that a given cluster does not have n_s images to be viewed, we simply evenly display additional images in the remaining clusters. In addition, if there is a remainder when computing n_s before rounding down we also evenly take these from other eligible clusters. This ensures that we always have $r = 70$ distinct displayed images for the user to sort through in both the RL view and the CV view. 4. Fig 4-A shows that the mean average precision increases as the number of relevant items selected by the user increases. In addition, 4.4-B shows that our method performs better 60% of the time during the first iteration and Figs. 4-C-4-D show that during this first iteration the CV method results in a much higher average number of relevant images. Thus we can conclude that our method performs better and is thus recommended during the first iteration because it allows the user to mark a greater amount of relevant images than the RL input method.

To evaluate the visual effectiveness of our interactive visual interface, we compare our combined cluster representative and cluster member views with traditional MDS projections and determine that our combined view is more distinct and thus makes it easier for the user to both understand relationships among items and select relevant glyphs.

To evaluate the speed of the fast SBMDS algorithm on the GPU architecture, we compare the final results with a known implementation (CFMDS [18]). The total number of samples for the alignment matrix is defined as $\frac{3}{2} \times \frac{\text{No. of data points}}{p}$ based on the graphics hardware's available memory. The optimal number of sub-matrices is chosen after 10 independent executions and for different input sizes, the best results are obtained for different p (1000x1000, 2000x2000, 3000x3000 - p = 10; 4000x4000, 5000x5000, 6000x6000 - p = 30). All the results are obtained, keeping the accuracy between our algorithm and the CFMDS implementation. Therefore, in the results for each dataset, we include the results for the best matched number of sub-matrices obtained by cross validation. Fig. 5 shows the

performance of three different implementations in terms of speed (MATLAB sequential version, CFMDS and SBMDS). We can see that SBMDS produces results approximately 20x faster than CFMDS, which is itself 100x faster than the sequential implementation. Also, we compare the accuracy of the results of SBMDS on the GPU architecture and the sequential version and find the error is approximately $\varepsilon < 0.001$. Therefore, we preserve the accuracy while significantly improving the speed. Also, this approach is scalable because running the GPU implementation on 1.2 million random objects is up to 50x faster depending on the right choice of the size of each sub-matrix.

5 Conclusion

Although relevance feedback algorithms improve CBIR systems, the update time for these algorithms can be a bottleneck. Thus, opening the graphics hardware to general purpose computation can improve the speed of algorithms used in an interactive CBIR setting by enabling parallel computation on a powerful platform. We present a fast interactive interface for CBIR based on implementations of fast k-means clustering and a new fast SBMDS on GPU architectures. The evaluation is performed using standard datasets of Mayan hieroglyphs. This complex experimental setup demonstrates the effectiveness of our fast interactive interface at interactive speeds. We also show that, using GPU architecture for the SBMDS algorithm results in much better performance compared to the existing algorithms on the CPU as well as on the GPU.

Acknowledgments

This work is jointly supported by the Swiss National Science Foundation (SNSF) via the project MAAYA (grant CR21I2L_144238) and the European COST Action on Multilingual and Multifaceted Interactive Information Access (MUMIA) via the Swiss State Secretariat for Education and Research (SER grant C11.0043).

References

1. Aksoy, S., Haralick, R., Cheikh, F., Gabbouj, M.: A weighted distance approach to relevance feedback. In: IAPR International Conference on Pattern Recognition. pp. 812 – 815 (2000)
2. Arthur, D., Vassilvitskii, S.: k-means++: The advantages of careful seeding. In: Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms. pp. 1027–1035. Society for Industrial and Applied Mathematics (2007)
3. Borg, I., Groenen, P., Mair, P.: Applied Multidimensional Scaling. SpringerBriefs in Statistics, Springer (2012)
4. Buja, A., Swayne, D.F., Littman, M.L., Dean, N., Hofmann, H., Chen, L.: Data visualization with multidimensional scaling. Journal of Computational and Graphical Statistics (2008)

5. Chalmers, M.: A linear iteration time layout algorithm for visualising high-dimensional data. In: Proceedings of the 7th Conference on Visualization '96. IEEE Computer Society Press, Los Alamitos, CA, USA (1996)
6. Chen, Y., Zhou, X.S., Huang, T.S.: One-class svm for learning in image retrieval. In: Image Processing, 2001. vol. 1, pp. 34–37. IEEE (2001)
7. Das, G., Ray, S., Wilson, C.: Feature re-weighting in content-based image retrieval. In: Image and Video Retrieval, pp. 193–200. Springer (2006)
8. Doloc-Mihu, A., Raghavan, V.V., Karnatapu, S., Chu, H.C.H.: Interface for visualization of image database in adaptive image retrieval systems (airs)
9. EM Photonics: CULA Reference Manual (2011)
10. Farivar, R., Rebolledo, D., Chan, E., Campbell, R.: A parallel implementation of k-means clustering on gpus. In: WorldComp 2008. Las Vegas, Nevada (2008)
11. Ingram, S., Munzner, T., Olano, M.: Glimmer: Multilevel mds on the gpu. IEEE Transactions on Visualization and Computer Graphics 15(2), 249–261 (2009)
12. Kumar, K.K., Bhutada, S., Balaram, V.: An adaptive approach to relevance feedback in cbir using mining techniques. In: Proceedings of International Conference on Electronics. vol. 80 (2012)
13. Li, Y., Zhao, K., Chu, X., Liu, J.: Speeding up k-means algorithm by gpus. In: CIT. pp. 115–122. IEEE Computer Society (2010)
14. Lloyd, S.: Least squares quantization in pcm. IEEE Trans. Inf. Theor. 28(2), 129–137 (2006)
15. Melo, D.O., Lopes, A.A.: Data visualization and relevance feedback applied to information retrieval. In: Proceedings of the Sixth Workshop on Ph.D. Students in Information and Knowledge Management. pp. 27–32. ACM, NY, USA (2013)
16. Morrison, A., Ross, G., Chalmers, M.: Fast multidimensional scaling through sampling, springs and interpolation. Information Visualization 2(1), 68–77 (2003)
17. Nvidia: CUDA: Compute Unified Device Architecture. Reference Manual (2008)
18. Park, S., Shin, S.Y., Hwang, K.B.: Cfmds: Cuda-based fast multidimensional scaling for genome-scale data. BMC Bioinformatics 13(S-17), S23 (2012)
19. Peng, J.: Multi-class relevance feedback content-based image retrieval. Computer Vision and Image Understanding 90(1), 42–67 (2003)
20. Ratsch, G., Scholkopf, B., Mika, S., Muller, K.R.: SVM and boosting: One class. GMD-Forschungszentrum Informationstechnik (2000)
21. Rocchio, J.J.: Relevance feedback in information retrieval. Prentice-Hall, Englewood Cliffs NJ (1971)
22. Roman-Rangel, E., Marchand-Maillet, S.: Stopwords detection in bag-of-visual-words: The case of retrieving maya hieroglyphs. In: ICIAP (2013)
23. Thomee, B., Huiskes, M.J., Bakker, E., Lew, M.S.: An exploration-based interface for interactive image retrieval. In: ISPA. pp. 188–193. IEEE (2009)
24. Tian, Q., Hong, P., Huang, T.S.: Update relevant image weights for content-based image retrieval using support vector machines. In: ICME. pp. 1199–1202. IEEE (2000)
25. Vasconcelos, N., Lippman, A.: Learning from user feedback in image retrieval systems. In: NIPS. pp. 977–986 (1999)
26. Wu, Y., Zhang, A.: A feature re-weighting approach for relevance feedback in image retrieval. In: Image Processing. vol. 2. IEEE (2002)
27. Yang, T., Liu, J., Mcmillan, L., Wang, W.: A fast approximation to multidimensional scaling. In: CIMCV (2006)
28. Zhou, Z.H., Chen, K.J., Dai, H.B.: Enhancing relevance feedback in image retrieval using unlabeled data. TOIS 24(2), 219–244 (2006)