Efficient Query Refinement in Multimedia Databases *

Kaushik Chakrabarti
University of Illinois
kaushikc@cs.uiuc.edu

Kriengkrai Porkaew
University of Illinois
kporkaew@cs.uiuc.edu

Sharad Mehrotra
University of California
sharad@ics.uci.edu

Large repositories of multimedia objects containing digital images, video, audio and text documents are becoming common. There is an increasing application need to search these repositories based on their content. Examples include online shopping (e.g., "Find me all shirts that look like *this* one), medical applications (e.g., "Find all tumors similar to a given pattern"), face recognition, trademark searches, audio retrieval and content-based video browsing/searching. To address this need, we are building the *Multimedia Analysis and Retrieval System (MARS)*, a system for effective and efficient content-based searching and browsing of large scale multimedia repositories [2]. MARS represents the content of multimedia objects by a collection of features (e.g. color histograms, cooccurence texture, color layout and shape for images; keywords for text). The user poses a query by submitting an example and requesting for a few objects that are "most similar" to the submitted example. The similarity between any two objects is computed by first computing their similarities based on the individual features and then combining them to obtain the overall similarity [2]. MARS uses specialized indices and advanced query processing algorithms to efficiently return the top matches to the user [1, 2].

An important aspect of multimedia similarity retrieval is that of query refinement. Due to the subjective nature of multimedia retrieval, rarely do the answers to the "starting" query satisfy the user's need right away. Rather, among the answers returned, the user may find one or more objects that are closer to what she had in mind than the original example which becomes the basis for a second query (referred to as "refined" queries). The user may also specify the relevance levels (e.g., very relevant, relevant) for the new query objects. Not only does the query object(s) change, but the function used to compute the similarity is also changed (by the system) in order to better capture the user's perception of similarity and hence satisfy her information need better. The latter is achieved by using *inter-feature* and *intra-feature* weights. While inter-feature weights capture the relative importance of the various features, intra-feature weights capture the relative importance of various dimensions within each feature. The task of the refinement model is to learn the weights that best capture the user's perception of similarity. The second task of the refinement model is to modify the query to reflect the new set of query objects. There are two ways to modify the query, namely represent the set of query objects by a single point (the weighted centroid) or by *multiple* representative points. We refer to the former model as Query Point Movement (QPM) and the latter as Query Expansion (QEX) model [3]. Recent work shows that query refinement techniques significantly improve the quality of answers and the answers get progressively better with more iterations of feedback [3].

While there has been a lot of research showing the effectiveness of query refinement, there exists no work on how to implement query refinement efficiently in a multimedia database system. We explore such approaches in this paper. The proposed approaches are independent of the refinement model used (e.g., QPM or QEX) and hence work for all models. We assume that each feature is indexed using a multidimensional index structure (called the F-index). A similarity query can then be answered by executing a k-NN query on each F-index and merging the individual feature results to obtain the final results. Our first contribution is to generalize the notion of similarity queries and allow multiple query points in a query (referred to as *multipoint queries*). This generalization is necessary since refined queries cannot be always expressed as single point queries. We develop a k-NN algorithm that can handle multipoint queries and show that it performs significantly better than the naive approach (i.e. execute several single point queries using the 'single-point' k-NN algorithm and merge results).

The second and the main problem we address is how to evaluate refined queries efficiently. A naive approach is to treat a refined query just like a starting query and execute it from scratch. We observe that the refined queries are *not* modified drastically from one iteration to another. As a result, most of the execution cost can be saved by appropriately exploiting the information generated during the previous iterations of the query. We propose 3 techniques, namely Reuse (RU), Full Reconstruction (FR) and Selective Reconstruction (SR), that *cache* certain information during the execution of the previous iterations of the query and use that cached information to save the execution cost (both I/O and CPU costs) during the subsequent iterations. We define notions of I/O optimality and CPU optimality and evaluate the three schemes in terms of these criteria. We find that although RU is significantly better than the naive approach, it is not I/O optimal. So we propose FR and show that it is I/O optimal and is significantly more efficient compared to RU. However, FR is not CPU optimal. We finally propose SR and show that SR, like FR, is I/O optimal, and, under certain conditions, also CPU optimal. Even though SR is not always CPU optimal, it is always better than FR in terms of CPU cost and is hence the best technique. Our experiments show that the above techniques speed up the execution of refined queries by several orders of magnitude compared to the naive technique.

*This work was supported in part by the National Science Foundation under Grant No. IIS-9734300, in part by the Army Research Laboratory under Cooperative Agreement No. DAAL01-96-2-0003 and in part by NSF/DARPA/NASA Digital Library Initiative Program under Cooperative Agreement No. 94-11318.

References

[1] K. Chakrabarti and S. Mehrotra. The hybrid tree: An index structure for indexing high dimensional feature spaces. *Proc. of the 15th International Conference on Data Engineering (ICDE)*, March 1999.

[2] M. Ortega, Y. Rui, K.Chakrabarti, S. Mehrotra, and T. Huang. Supporting ranked boolean similarity queries in mars. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 1998.

[3] K. Porkaew, K. Chakrabarti, and S. Mehrotra. Query refinement for multimedia similarity retrieval in MARS. *ACM Multimedia Conference*, 1999.