An Experimental Study on the Performance of Visual Information Retrieval Similarity Models

Horst Eidenberger and Christian Breiteneder
Institute of Software Technology and Interactive Systems
Vienna University of Technology
Vienna, Austria
{eidenberger, breiteneder}@ims.tuwien.ac.at

Abstract–**This paper is an experimental study on the performance of the two major methods for macro-level similarity measurement: linear weighted merging and logical retrieval. Performance is measured as the average query execution time for a significant number of tests. The two models were implemented in the standard version (as they are applied in a number of prototypes) and in an optimized version. The results show, that optimized logical retrieval clearly outperforms optimized linear weighted merging.**

Keywords–content-based image retrieval; content-based visual retrieval; visual information retrieval; query optimization; similarity measures; triangle inequality; experimental study

I. INTRODUCTION

Content-based retrieval of information from visual media (images and video; CBIR) has been an area of increasing interest and research in the past years ([6]). Up to now, one of the major problems of most CBIR approaches has been bad performance in terms of query execution time. Similarity measurement in CBIR systems is essentially based on distance measurement of feature vectors that have been previously extracted from visual media. Most used distance functions are L_1 and L_2 metrics, e.g. the city block distance and the Euclidean distance. These distance functions have a complexity of at least $O(n)=n$, n being the size of the feature vectors. Most query acceleration approaches follow one of three directions:

1. Indexing of feature data. Indexing method include tree techniques (quadtree, R-tree, etc.) and gridfiles. They suffer from the drawback that most of them support only one inherent distance measure (mostly Euclidean distance) and therefore have to be implemented fore each group of features with common distance measure separately. Additionally, most of them become increasingly ineffective for high-dimensional data.
2. Complexity reduction of feature vectors prior or after the feature extraction process. This includes coarse representation of features (reduced scales or number of histogram-bins, etc.) and redundancy reduction (e.g. factor analysis).
3. Occlusion of media objects to minimize the number of distance comparisons. The most well-known approach from this area is using the triangle inequality (the fourth metric axiom) to exclude dominated media objects (see [1]).

In the paper we investigate methods from the third area: occlusion of media objects that are based on the similarity models used in most CBIR systems. We will compare the performance of the linear weighted merging model (LWM) and the logical retrieval model (LR) in form of a simple and an optimized algorithm (see Section II for details on LWM and LR). The rest of the paper is organized as follows. Section II points out relevant related work, Section III describes the algorithms we used in our experiments, Section IV is a brief sketch on the test environment we used and Section V describes the experimental results.

II. RELATED WORK

Subsequently, we will outline the CBIR macro-level similarity measurement process and earlier work on CBIR query acceleration. In [3] we define macro-level similarity measurement as the process that extracts a result set for a query from a given distance space. Distance space is defined as the vector space that is derived from feature space by measuring the distance of media objects to given query examples with distance functions (micro-level similarity measurement). In feature space, media objects are represented as numerical feature vectors.

The two most widely applied methods for macro-level similarity measurement are: (1) linear weighted merging (LWM) and (2) logical retrieval (LR). LWM is a two step process. In the first step, a position value is calculated for each media object (according to equation 1) and in the second, the media objects are ordered by this position value and the first n elements are selected as the result set.

$$Position\,value_{Object} = \sum_{i=1}^{F} w_i d_i \tag{1}$$

In equation 1 d_i and w_i are the distance value and weight for feature i (of F) and the given media object. This equation is a simplified version of the formula given in [6]. It is implemented by most CBIR prototypes (e.g. QBIC, [4]). In opposition to LWM, the result set size in LR is not constant and depends on the given media collection. In LR, each query is a logical expression of terms c_i of the form given in equation 2. The parameters t_{i1}, t_{i2} are thresholds for the minimum

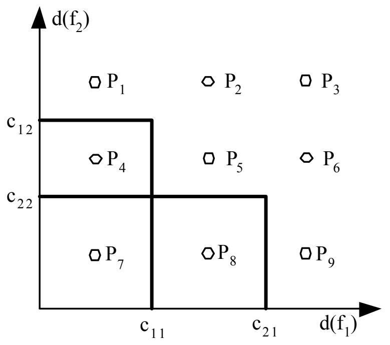

Figure 1. LR conditions and possible locations of media objects P_i.

respective maximum distance of a media object for a certain feature. A media object is added to the result set, if the query expression evaluates to *true* for its distance values. For example, this method is implemented in MARS ([5]).

$$t_{i1} \le d_i \le t_{i2} \tag{2}$$

In our earlier work, we implemented a simplified version of LR (called QueryModels), where conditions may only be *and*-connected. For this approach we implemented a heuristic optimization technique that tries to order the conditions of an expression in a way, that those conditions (distance functions) are evaluated first that cut off most non-similar media objects and/or use the fastest distance functions. Because of the *and*-connection, consecutive conditions have to take only those media object into account that have not been cut off by prior conditions. This method lead to a reduction of the average query execution time of *66%* (see [2] for details). Subsequently, we will introduce a simple optimization technique for LR expressions that contain the logical operators *and*, *or* and *not*.

III. USED ALGORITHMS

We tested four different algorithms for macro-level similarity measurement: (1) LWM with simple optimizations (referred to as LWM), (2) LWM with triangle inequality optimization (LWM+), (3) LR with no optimization (LR) and (4) LR with a simple optimization technique (LR+). In the tests we used no indexing or complexity reduction techniques, because these methods are optimizations on the micro-level and can be applied to any of the four algorithms. The test plan was as follows:

1. Select query parameters (query example, features, weights, result set size, etc.). The details concerning this step will be described in Section IV.
2. Calculate the result sets for LMW and LWM+.
3. Derive an LR expression from the LWM result set that represents exactly the same result set as the LWM algorithm.
4. Calculate the result sets for LR and LR+.

The used LWM algorithm has the following form (pseudo-code):

```
FOR EACH mo {
   pv:= 0;
   FOR EACH feature {
      dist:= CALC DISTANCE FROM qe TO mo;
      pv:= pv + dist*weight(feature);
      IF pv > distanceSum(n) THEN {
         GOTO break;
      }
   }
   ADD pv to distanceSum;
break:
}
rs:= FIRST n ELEMENTS BY distanceSum;
```

In this algorithm, *qe* is the query example, *mo* is a media object, *pv* is the (partial) position value of *mo* and *distanceSum* is a vector of media object position values (always sorted in ascending order). This algorithm differs from the standard LWM in one point: whenever the partially calculated position value exceeds the position value of the n-th element (result set border), the calculation for this media object is aborted and calculation continues with the next media object.

The LWM+ algorithm uses the triangle inequality technique (TRIQ) for query optimization. The TRIQ is an occlusion technique that can only be applied on distance functions that fulfill the metric axioms (see [1] for details on TRIQ). We use two distance measures that are both metric: city block distance and Euclidean distance. Based on [1] we use equation 3 (joint cutoff criterion) to occlude media objects. In this equation, *r* is a reference object, *q* is the query example, *x* is an arbitrary media object and min_d is the distance value of the n-th element in the result set. All *d(x,r)* values can be calculated *before* the querying process.

$$|d(x,r) - d(q,r)| > \min_d \tag{3}$$

Because we are using multiple features and want to retrieve more than one media object, we use an adapted version of TRIQ. The final LWM+ algorithm looks as follows:

```
FOR EACH mo {
   pv:= 0;
   FOR EACH feature {
      IF |refDist(mo)-refDist(qe)| *
         weight(feature) >
         (distanceSum(n)-pv) THEN {
         GOTO break;
      }
      dist:= CALC DISTANCE FROM qe TO mo;
      pv:= pv + dist*weight(feature);
      IF pv > distanceSum(n) THEN {
         GOTO break;
      }
   }
   ADD pv to distanceSum;
break:
}
rs:= FIRST n ELEMENTS BY distanceSum;
```

The similarity measurement for a media object is terminated as soon as it becomes clear that the position value will exceed the position value of the n-th element in *distanceSum*. Using the TRIQ, this can be done prior to the distance calculation.

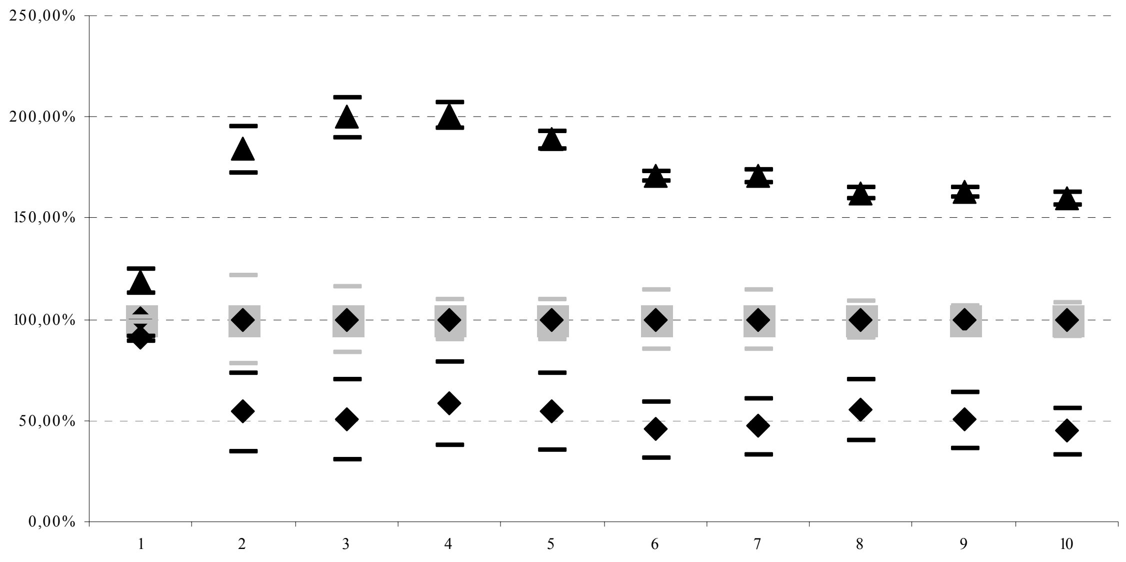

Figure 2. Results for varying number of queried features. The triangles show the average performance of LR, the diamonds stand for the LR+ performance and the gray squares and the diamonds within them depict the performance of LWM resp. LWM+. The lines above and below the icons show the standard deviations.

In our earlier work we have shown that LR is a more flexible model for macro-level similarity measurement than LWM. It is possible to derive an LR expression for arbitrary LWM result sets with the following algorithm:

```
expression:=();
FOR i:=n to 1 DO {
   FOR j:=i-1 to 1 DO {
      IF distVector(i) NOT EXCEEDS
         distVector(j) THEN {
         GOTO break;
      }
      IF distVector(i) EXCEEDS
         distVector(j) THEN {
         DEL distVector(j) FROM expression;
      }
   }
   ADD distVector(i) TO expression;
break:
}
```

Here, *expression* is a vector of all conditions and *distVector(i)* is the distance vector for media object *i*. The idea of the algorithm is to take each LWM result set element *i*, check, if it is included in the expression derived so far and – if not – add *i or*-connected to *expression*. Each added distance vector defines an *f*-dimensional *and*-connected **cube** (*f* features, a cube consists of one LR condition for each feature) where the distance values are the t_{f2} thresholds of LR conditions (see Section II) and the t_{f1} are all *0*. Figure 1 shows an example for two features: (c_{11},c_{12}) and (c_{21},c_{22}) are cubes of conditions and the result set consists of $\{P_4, P_7, P_8\}$. Additionally, the LWM to LR conversion algorithm checks, if new conditions dominate existing ones and – if yes – eliminates the dominated ones. LR querying based on the derived expression is done with the following algorithm:

```
FOR EACH mo {
   distVector:=();
   FOR EACH feature {
      dist:= CALC DISTANCE FROM qe TO mo;
      ADD dist to distVector;
   }
   FOR EACH condition {
      IF distVector EXCEEDS expression THEN {
         GOTO break;
      }
   }
   ADD mo TO rs;
break:
}
```

The optimized LR+ algorithm uses the same algorithm but adds an additional *and*-connected cube of conditions to *expression*. This cube consists of one condition for each feature, where t_{f2} is the maximum value of all threshold values for this feature in *expression* and t_{f1} is always *0*. For the example in Figure 1 the cube of conditions (c_{21},c_{12}) is added. Thus the media objects P_1, P_2, P_3, P_6 and P_9 can be occluded very fast. The next section describes the test environment and test data for these algorithms.

IV. Test environment

The querying algorithms were implemented in Perl and evaluated on a DOS computer. Perl was chosen, because it allows rapid prototyping and effective statistical analysis. DOS was chosen, because querying performance was tested by the average query execution time and therefore using a single user, single task operating system was the proper choice.

We did about 50000 tests on one to ten features (equally distributed) and up to 10000 artificial feature vectors with length between one and 32 elements (equally distributed). The artificial feature vectors were normalized to *[0,1]* and consisted of equally distributed (45%), normally distributed (50%) and negative exponentially distributed (5%) columns of random numbers. The reference values for LWM+ were calculated prior to the tests. Two distance functions were used: city block distance and Euclidean distance. Each artificial feature was bound to one distance function (equally distributed).

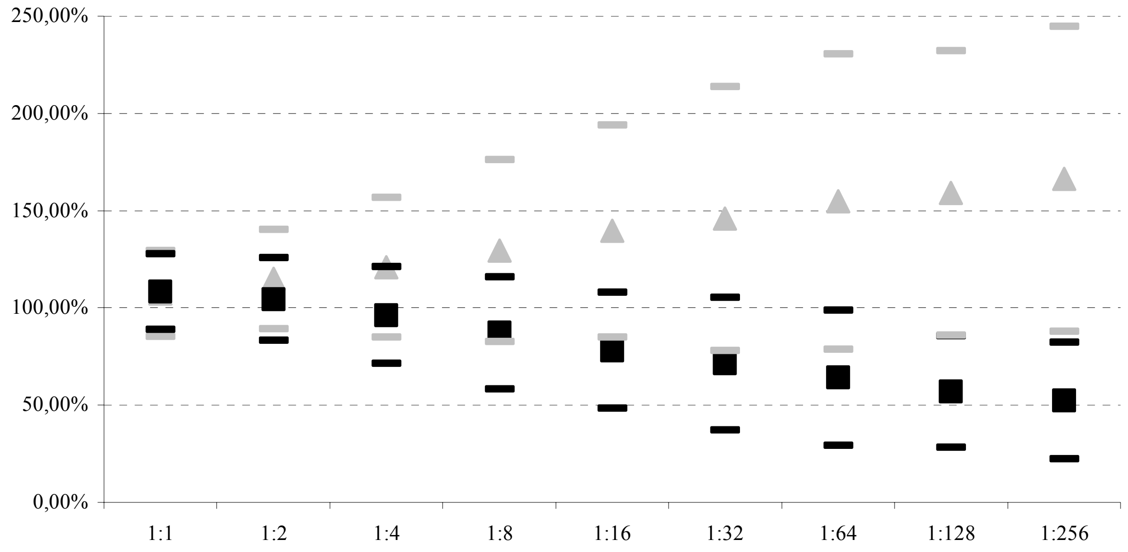

Figure 3. Results for varying size of result set. The grey triangles show the average performance of LR and the black squares depict the LR+ performance.

Each query was done with one random selected query example and random selected weights. The result set size was fixed to 32 elements for all tests. The next section describes the results for our experiments.

V. Experiments and results

First we tested the four algorithms performance for a varying number of features. We did 40000 queries on one to ten features. Figure 2 shows the results. The triangles show the average performance of LR, the diamonds stand for the LR+ performance and the gray squares and the diamonds within them depict the performance of LWM respective LWM+. All performance values are relative to LWM. The lines above and below the icons show the standard deviations relative to the average values.

This test revealed that using the TRIQ has hardly any effect on query execution time (smaller than 1%). Additionally, it showed that the performance of LWM is always better than LR and that LR has a very small standard deviation. Using the simple optimization in LR+ reduces the query execution time to about 50% of LWM. Because this is a heuristic approach, the standard deviation of LR+ is bigger than of LWM. The better performance of LR+ seems to be independent from the number of queried features.

In the second experiment, we tested the algorithms behavior for a varying relation of result set size and queried collection size. This is interesting because at least the performance of LR+ could be dependent on this relation. We did 7200 queries with a varying number of features and relations from 1:1 to 1:256. Figure 3 shows the results. This time, LWM and LWM+ were omitted. Still, the performance values are relative to LWM (100%). This test showed that only for relations of result set size to queried collection size of bigger than 1:4, the performance of LR+ is worse than LWM (above 100%). For relations lower than 1:4 LR+ outperforms LWM and reaches an average query execution time of about 50% at a relation of 1:256. In this test, the standard deviation for LR was significantly worse than in the first test. This is because queries on varying numbers of features were mixed. The overall performance of the tested algorithms (compared to LWM, 100%) is: LWM+: 99.9%, LR: 172% and LR+: 55%.

VI. Conclusion

In this paper we compared the query execution performance of two methods for macro-level similarity measurement: linear weighted merging (LWM) and logical retrieval (LR). We implemented each algorithm in a standard and an optimized version. Additionally, we implemented a conversion algorithm that generates LR expressions from LWM result sets. About 50000 tests were performed.

The major result of this study is, that optimized LR clearly outperforms LWM in terms of query execution time. In our earlier work we showed that this is true for the quality of retrieval results as well. Thus, there is – from our point of view – no reason to use LWM in CBIR systems any longer.

VII. References

[1] J. Barros, J. French, and W. Martin, "Using the triangle inequality to reduce the number of comparisons required for similarity based retrieval," Proc. SPIE Conf. on Storage and Retrieval for Image and Video Databases, San Jose CA, USA, pp. 392-403, 1996.

[2] C. Breiteneder, and H. Eidenberger, "Performance-optimized feature ordering for content-based image retrieval," Proc. European Signal Processing Conference, Tampere, Finland, 2000.

[3] H. Eidenberger, and C. Breiteneder, "Macro-level similarity measurement in VizIR," Proc. IEEE Multimedia Conf. & Expo, Lausanne, Switzerland, 2002.

[4] M. Flickner, H. Sawhney, W. Niblack, J. Ashley, Q. Huang, B. Dom, M. Gorkani, J. Hafner, D. Lee, D. Petkovic, D. Steele, and P. Yanker, "Query by image and video content: the QBIC system," *IEEE Computer*, vol. 28, no. 9, pp. 23-32, 1995.

[5] M. Ortega, R. Yong, K. Chakrabarti, K. Porkaew, S. Mehrotra, and T.S. Huang, "Supporting ranked boolean similarity queries in MARS," *IEEE Transactions on Knowledge and Data Engineering*, vol. 10, no. 6, pp. 905-925, November 1998.

[6] A.W.M. Smeulders, M. Worring, S. Santini, A. Gupta, and R. Jain, "Content-based image retrieval at the end of the early years," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 12, pp. 1349-1380, December 2000.