A PARALLEL CBIR IMPLEMENTATION USING PERCEPTUAL GROUPING OF BLOCK-BASED VISUAL PATTERNS

Shyi-Chyi Cheng[1], *Wei-Kan Huang*[2], *Yu-Jhih Liao*[2], *and Da-Chun Wu*[2]

[1]Department of Computer Science and Engineering, National Taiwan Ocean University, Taiwan
[2]Department of Computer and Communication Engineering, National Kaohsiung First University of Science and Technology, Taiwan

ABSTRACT

This paper proposes a parallel solution to retrieve images from distributed data sources using perceptual grouping of block-based visual patterns. The method of grouping visual patterns into image model based on generalized Hough transform is one of the most powerful techniques for image analysis. However, real-time applications of this method have been prohibited due to the computational intensity in similarity searching from a large centralized image collection. A query object is decomposed into non-overlapped blocks, where each of them is represented as a visual pattern obtained by detecting the line edge from the block using the moment-preserving edge detector. A voting scheme based on generalized Hough transform is proposed to provide object search method, which is invariant to the translation, rotation, scaling of image data. In this work, we describe a heterogeneous cluster-oriented CBIR implementation. First, the workload to perform an object search is analyzed, and then, a new load balancing algorithm for the CBIR system is presented. Simulation results show that the proposed method gives good performance and spans a new way to design a cost-effective CBIR system.

Index Terms—Hough transforms, object detection, information retrieval, distributed computing, load modeling.

1. INTRODUCTION

Content-based image retrieval (CBIR) is extremely desirable in many applications with an explosion of the amount of multimedia data available to people. Currently, most CBIR systems are based on the centralized computing servers which respond to extract features from images, index images with feature vectors, and retrieve images relevant to a query. One of the shortcomings is that a centralized CBIR server is computational intensive and difficult to scale up. As reported by Smeulders et al. [1], one of promising future trends in CBIR includes distributed computing on data collection, data processing, and information retrieval.

Cluster-based distributed computing offers a good cost effective solution to extend the centralized system model, given its excellent scalability, fault tolerance and flexibility attributes [2,3]. However, the nature of clusters is heterogeneous and requires load distributions that consider each node's computational features [3]. Although many methods on load balancing are proposed in the literature to improve the performance of a computer cluster, it is still not totally solved, particularly for parallel multimedia systems.

Much of the research effort for CBIR systems aims at seeking effective features to represent images in feature vectors for discriminating elements among the global database. Both of the low-level features including color, shape, or texture-based primitives and high-level semantic concept are explored to perform similarity searches [1,4]. From the retrieval accuracy point of view, correct features are essential to apply pattern recognition techniques in CBIR systems. On the other hand, complex features are potential expensive and hard to scale up with the ever-increasing sizes of the data collections associated to a centralized CBIR system. One of the most common approaches followed to reach a cost effective and scalable parallel CBIR system has been to exploit the parallelism of a similarity search method with effective and efficient image representation in order to achieve a reasonable user response times.

Nowadays, two approaches to distributed CBIR systems are tested in the related works: a cluster [3] and a peer-to-peer network [5,6]. The key to success for a cluster-based CBIR system is the design of load balancing operations preventing the coexistence of idling and overloaded processors. Peer-to-peer (P2P) applications provide a good way to span fully decentralized and distributed CBIR services. In a purely decentralized CBIR architecture, all peers perform exactly the same tasks; however, in a cluster, we might have a master node to invoke queries to several slave nodes and collect each slave node's retrieval results. It is possible to further implement a peer in a P2P network as a cluster when the shared image collection of the peer is large. Thus, each approach has its own advantages and does not contradict with each other.

As mentioned above, a dynamic and distributed load balancing algorithm indeed affects the performance of a distributed CBIR system. Only a few of the load balancing operators focus on distributed CBIR systems though load balancing for a cluster has been extensively studied. For instance, Bosque et al. include a load balance operator based on the number of running tasks and the computational power of each node in their parallel CBIR system [3]. The problem is that the computational power of a node is highly dynamic and hard to obtain in the running time. In addition, the complexity of an image which implies different computational load in image analysis should be considered in the design of load balancing for a CBIR system.

The use of perceptual grouping to extract structure gives the system an edge over content-based image retrieval systems that retrieve images containing structural objects based purely upon low-level features. Different approaches to group visual patterns extracted from image blocks, regions, or objects have been proposed to offer a semantic-based representation for image understanding and analysis applications [7]. In our previous work, we suggested using generalized Hough transform (GHT) to represent visual patterns existing in image blocks [8]. A visual

pattern is obtained by detecting a line in a square block using the moment-preserving edge detector. A voting scheme based on generalized Hough transform (GHT) and visual patterns is also proposed to provide object search method, which is invariant to the translation, rotation, reflection of image data, and hence, invariant to orientation and position.

The method of grouping visual patterns into image model based on generalized Hough transform is one of the most powerful techniques for image analysis. However, real-time applications of this method have been prohibited due to the computational intensity in similarity searching from a large centralized image collection. In this work, we describe a heterogeneous cluster-oriented CBIR implementation. First, the workload to perform an object search is analyzed, and then, a new load balancing algorithm based on workload tracking is presented. The main contributions of the proposed distributed CBIR system include: (1) the parallelism of the voting scheme of GHT is exploited to speedup the matching process; (2) a systematic way to analyze workload for object search is proposed; (3) a load balancing algorithm based on workload tracking is included in our system. Computer simulation results show that the proposed method gives good performance and spans a new way to design a cost-effective CBIR system.

.2. OBJECT SEARCH USING GENERALIZED HOUGH TRANSFORM

The problem of object-based image retrieval can be modeled by GHT on the basis of block information. It can cope with objects, moving with translation, rotation, and scaling. GHT was initially proposed to represent an object with arbitrary shape using a so-called R-table [9]. The shape of the object can be described by the geometric relationship between the object reference point X^R and the edge points Xs of the object boundary. To locate the pattern in an image, each of the edge points is considered and the corresponding locations of the reference point are calculated. Assume that the shape, scale s, and rotation τ of the desired region are known. A reference point, $O(x^R, y^R)$, is chosen at any location inside the sample region. Then an arbitrary line intercepting the boundary template at the point $M(x,y)$ can be constructed at this reference point aiming in the direction of the region border. The edge direction ϕ, the distances of the reference point to region border, r, and the angle α for the segment OM are then determined as a geometrical template. Next, a reference table, R-table, is constructed using ϕ as a key and (r,α) as data. In the matching phase, the 4D parameter space $s-\tau-x-y$ is investigated. When the figure with the scale s and rotation τ is the object for extracting, the edge direction ϕ_i, which is searched in keys of R-table, is determined for an edge point (x_e, y_e) in the target image. By using the values of (r_i,α_i) that are indexed by ϕ_i, the candidate coordinates (x^R, y^R) of the reference point, are determined by the following expression:

$$(x^R, y^R) = (x_e + r(\phi)s\cos(\alpha(\phi)+\tau), y_e + r(\phi)s\sin(\alpha(\phi)+\tau)) \cdot \quad (1)$$

Then a voting process for increasing the value of the coordinates (s,τ,x,y) of the 4D parameter space is executed. Calculating candidate coordinates of the reference point and voting process to coordinates are executed for all possible values of s and τ.

After all the edge points of the target image have been processed, the 4D accumulator array will contain high peak value for locations where many edge points coincide with many query object edges. High peaks correspond to reference point locations where instances of the query object occur in the target image.

In our previous work, we modeled the boundary of the query object as a set of disjoint edge blocks. Each edge block B, shown in Fig. 1, is represented as a vector (h_1, h_2, l, θ) where h_1 and h_2 are the two representative gray (color) values, l is the edge translation, and θ is the orientation angle of the edge in B. The solution to the edge detection problem in a given block is analytic and this means that the edge detection process can be performed very fast for large-database applications with no need for special hardware. In this case, Eq. (1) can also be used to determine the reference point of the query object from the target image, except for replacing the coordinates of edge points with the center coordinates of an edge block.

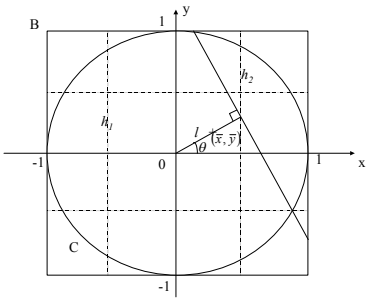

Fig. 1. An edge model in a 4 x 4 block B. The circle C is inscribed in B, and $(\bar{x}, \bar{y})$ are the coordinates of the centers of gravity of the gray (color) values inside C.

The computational complexity of GHT would be very high if we apply all edge information to locate the query object in the target image using (1). However, the parallelism of GHT is very high since we can concurrently perform the process to match every edge block of a query object with those of a target image. In addition, the affine transformation parameters (s^*, τ^*, x^*, y^*) which relate a query object with a target image are actually capable of being determined independently. Based on these two observations, in this work, we implement a parallel CBIR system using GHT.

Let N and M be the number of edge blocks obtained from a query object and a target image, respectively. Given an edge block $B = (h_1, h_2, \theta, l)$ be part of boundary of a query object Q, and the edge block $\hat{B} = (\hat{h}_1, \hat{h}_2, \hat{\theta}, \hat{l})$ be the corresponding block of B in the target image. Blocks B and $\hat{B}$ form a match pair if

$$\varepsilon(B, \hat{B}) = |h_1 - \hat{h}_1| + |h_2 - \hat{h}_2| < \xi \quad (2)$$

where ξ is a predefined threshold. The match pair (B, $\hat{B}$) will have a vote on the $s-\tau-x-y$ parameter space according to their block size ratio, the difference between θ and $\hat{\theta}$, and the difference between l and $\hat{l}$. During a query session, $N \times M$ edge pairs are tested to determine if they are match pairs, which are then used to determine the affine parameters that align the query object with the target image. The discussion about the object search method is out of the scope of this paper and the details can be found in [8]. In this work, we focus on its parallel implementation.

3. THE PROPOSED PARALLEL IMPLEMENTATION

Image browsing and query by example provide us different ways to access a large image database. In general, users of a CBIR system are required either to follow the summarized topics of an image database for browsing the image content or to specify queries by keywords, similarity, sketching an object, and painting a rough image as the query for retrieval. For a distributed CBIR system, it is very time consuming to broadcast a query to every local node and collect and rank the retrieval results. In this implementation, the master process computes the visual patterns of the input image and broadcasts them to the n slave processes, each of them is supposed to contain an image collection characterized with the same topic. The slave processes then proceed to perform the object search procedure using GHT mentioned above. Once each comparison has been performed, the slave processes forward similarity results to the master process for ranking and display the k nearest neighbors of the query to the user as the retrieval results. The workload for every slave process is obviously unbalanced since the size of image collection stored in every slave node is different.

Before the discussion of our load balancing operator, the amount of workload to perform an object search is first calculated. Performing the object search process, quite a lot of block pairs, each of them consisting of an edge block of the query image and an edge block of the target image are used to determine the optimal affine transformation parameters (s,τ,x,y). However, the voting scheme of GHT puts heavy burden on the computing units. Actually, the four parameters can be determined separately. In this work, we first estimate the optimal τ value, and then other parameters.

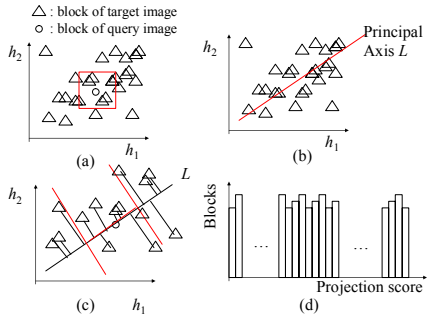

Fig. 2. Locate the relevant blocks for a query block in the $h_1 - h_2$ space: (a) the blocks within the square are relevant; (b) find out the principal axis L of blocks of the target image using PCA; (c) project each block onto L and redefine the relevant blocks in terms of projection scores; (d) index the target image using the projection score histogram.

Note that not all the block pairs are qualified to have a vote on the parameter space in order to reduce the noise of the global voting result. The question is: is it possible to skip the non-relevant comparison on block pairs? Given a query block B_q, Fig. 2 shows the four steps procedure to look after the relevant blocks through applying the principal component analysis (PCA) on the $h_1 - h_2$ space of the target image. For each block B in the target image, we project it on the principal axis L according to the values of h_1 and h_2 of B and obtain a project score p_B. The target image T is then indexed as a projection score histogram H with each element defined as

$$H = \{H_i\}, p_{\min} \le i \le p_{\max} \tag{3}$$

where H_i represents the number of blocks in T with the same projection score i, and $p_{\min}$ and $p_{\max}$ are the minimal and maximal principal scores of T, respectively. Let p_q be the projection score of a query block q. We re-formulate the condition to define that a target image block B with the projection score p_B is relevant with respective to q if

$$| p_q - p_B | \le \lambda \tag{4}$$

where λ is a pre-defined threshold. Based on (4), it is simple to prove that the workload for the slave node S to determine the affine parameters for matching the query Q with the target image T is

$$WL_S = \sum_{i \in (p_{\min}, p_{\max})} H_i^{(Q)} \sum_{j \in (i-\lambda, i+\lambda)} H_j^{(T)} \tag{5}$$

where $H^{(Q)}$ and $H^{(T)}$ are the projection score histograms of Q and T based on the principal axis L of T, respectively.

Based on the projection score histograms, we design a distributed load balancing operation using the proposed workload tracking technique. To avoid a large amount of data move and message passing, a fully distributed load balancing operator is essential for a CBIR system with a large image collection. As mentioned above, we distribute the global image collection among the slave processes where each of them contains images with the same semantic topic. However, this results in the possibility of load unbalancing.

There are three steps in the proposed load balancing scheme. First, all the images are clustered into several classes using the well-known c-mean clustering algorithm according to their projection score histograms. Second, for each pair of slave processes S_1 and S_2, compute its similarity measurement as

$$Sim(S_1, S_2) = \frac{| C(S_1) \cap C(S_2) |}{| C(S_1) \cup C(S_2) |} \tag{6}$$

where $C(S_1) \cap C(S_2)$ and $C(S_1) \cup C(S_2)$ represents the number of images within S_1 andS_2 that are clustered into the same classes in terms of projection score histograms and the total number of images within S_1 and S_2, respectively. Finally, for each slave process, we maintain a list of neighboring slave processes in the descending order in terms of the similarity measurement. During a query answering session, once a slave process S_i has completed all its assignments, it requests a database image for comparison from the slave process S_j who resides in the head of the neighboring list. S_j might decide not to migrate one of its assignments to S_j if the data move cannot reduce the overall response time. In this case, S_i will modify its neighboring list by removing S_j from the head of the list and adding it again into the tail of the list. On the contrary, S_j might decide to admit the request of S_i by choosing an image whose workload is similar to the estimated workload of S_i if the following condition is satisfied:

$$T_j > T_i,\ T_i \approx RTT + \frac{\overline{T_i} W_{LB}}{\overline{WL_i}} + \frac{W_{LB}}{B_{ij}},\quad T_j \approx \frac{W_j - W_j^{(a)}}{\overline{WL_j}} \overline{T_j} \tag{7}$$

where $\overline{T_i}$ ($\overline{T_j}$) is the average running time to complete an assignment for S_i (S_j), RTT is the response time for the requesting process, B_{ij} is the communication bandwidth between S_i and S_j, $\overline{WL_i}$ ($\overline{WL_j}$) is the average workload of S_i (S_j), and W_{LB} is the actual workload for the moved image, $W_j^{(a)}$ is total amount of workload that has been completed for S_j. The underlying idea of the proposed load balancing scheme is that the jitter in computational loading for a fast slave process should be minimized in order to meet its computational power. Thus, the proposed load balancing scheme is

fully distributed and consider the heterogeneity of slave processes in a cluster.

4. EXPERIMENTAL RESULTS

In order to evaluate the proposed approach, a series of experiments was conducted on a local area network (LAN) with its nodes connected with each other by a 100 Mbps Ethernet. In this section, we experimentally compare our distributed CBIR system with and without the proposed load balancing operation. A color image database consisted of 20,000 scenery images, coming from Corel's CorelPhoto image databases, was used. The size of each image in the database is 384 x 256.

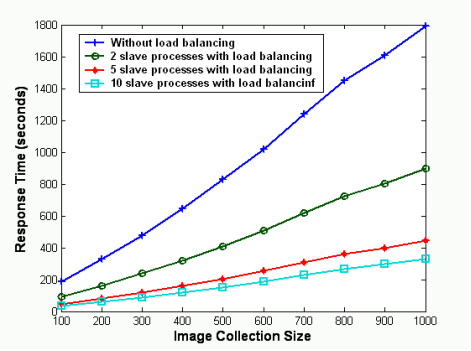

Fig. 3. Performance comparison for the CBIR system with and without load balancing under the situation of querying a specific slave process.

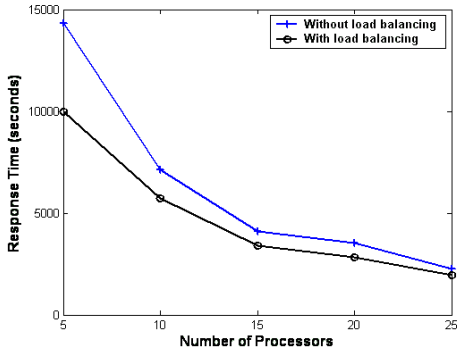

Fig. 4. Performance comparison for the CBIR system with and without load balancing under the situation of uniformly distributing the global database among slave processors.

It is possible for the distributed CBIR system to be in a very load unbalance situation due to the fact we initially partition the global image database into several parts which are then distributed among the slave processes according to the semantic topics. Considering that the user gives a query example and attempts to retrieve similar images in terms of visual characteristics under a specific semantic topic. In this case, only the slave process with the corresponding semantic topic is responsible to answer the query and others are idle at first. The matching jobs are then gradually distributed among the slave processes by performing the proposed load balancing operation. However, this would lead to a large amount of data communication. Fig. 3 shows the performance comparison with and without load balancing under this condition by varying the sizes of image collection of the slave process from 100 to 1000. The average speedup achieved for the cluster with 2, 5, and 10 slave nodes are 2.00, 4.02, and 5.46, respectively. When the proposed load balancing operation is performed, the overhead of message passing limits the eventual speedup achieved.

In the second test, we assume that the global image database is uniformly distributed among the slave nodes and all the slaves are responsible to answer a specific query. In this case, the communication overhead among slave nodes is small as compared with computing time for slave nodes to complete their matching jobs. Fig. 4 shows the performance comparison for the CBIR system with and without the proposed load balancing operation under this condition. The reductions in response times rang from 31% with 5 nodes to 15% with 25 nodes. As the number of nodes increases, the differences in response time decreases. Since the size of image collection in the case of large number of nodes is small, the performance of the system is hard to improve using the proposed load balancing algorithm. Actually, Fig. 4 reflects the worst case performance of the system.

5. CONCLUSION

In this paper we have presented a parallel implementation for an object search method based on the perceptual grouping of block-based visual patterns. A matching strategy based on the visual-pattern codes also makes the retrieval process robust and accurate, and the method using the modified generalized Hough transform to find the correct geometry transformation parameters in object searches without the main disadvantage of high-computational complexity for traditional generalized Hough transform. Furthermore, the proposed load balancing method reduces the system response time and scales up the system with the ever growing amount of image collection.

References

[1] A. W. Smeulders, M. M. Worring, A. Gupta, and R. Jain, "Content-Based Image Retrieval at the End of the Early Years," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 22 no. 12, pp1349-1380, 2000.

[2] G. Bell and J. Gray, "What's Next in High-Performance Computing?," *Commun. ACM*, vol. 45 no 2, pp91-95, 2002.

[3] J. L. Bosque, O. D. Robles, L. Pastor, and A. Rodriquez, "Parallel CBIR Implementations with Load Balancing Algorithms," *J. Parallel Distributing*, vol. 66, pp1062-1075, 2006.

[4] .P. John Eakins, "Towards Intelligent Image Retrieval," *Pattern Recognition*, vol. 1 no. 35, pp3-14, 2002.

[5] I. King, C. H. Ng, and K. C. Sia, "Distributed Content-based Visual Information Retrieval System on Peer-to-Peer Networks," *ACM Trans. Information Systems*, vol. 22, no. 3, pp477-501, 2004.

[6] P. Androutsos, D. Androutsos, and A. N. Venetsanopoulos, "A Distributed Fault-Tolerant MPEG-7 Retrieval Scheme on Small World Theory," *IEEE Trans. Multimedia*, vol. 8, no. 2, pp278-288, 2006.

[7] S.-C. Zhu, "Statistical Modeling and Conceptualization of Visual Patterns," *IEEE Trans. On Pattern Analysis and Machine Intelligence,* vol. 25, no 6, pp691-712., 2003.

[8] S.-C. Cheng, C.-T. Kuo, and H.-J. Chen, "Invariant Image retrieval Using Block-Based Visual Pattern Matching," *IEEE Int. Conf. Image Processing (ICIP'06),* USA, Oct. 2006.

[9] D. Ballard, "Generalizing the Hough Transform to Detect Arbitrary Shapes," *Pattern Recognition,* vol. 13, no. 2, pp111-122, 1981.