Private Content Based Image Retrieval

Shashank J, Kowshik P, Kannan Srinathan and C.V. Jawahar
International Institute of Information Technology
Hyderabad, India
{shashank_j,kowshik}@students.iiit.ac.in, {srinathan,jawahar}@iiit.ac.in

Abstract

For content level access, very often database needs the query as a sample image. However, the image may contain private information and hence the user does not wish to reveal the image to the database. ***Private Content Based Image Retrieval*** *(PCBIR) deals with retrieving similar images from an image database without revealing the content of the query image – not even to the database server. We propose algorithms for PCBIR, when the database is indexed using hierarchical index structure or hash based indexing scheme. Experiments are conducted on real datasets with popular features and state of the art data structures. It is observed that specialty and subjectivity of image retrieval (unlike SQL queries to a relational database) enables in computationally efficient yet private solutions.*

1. Introduction

With the emergence of a number of practical vision systems, security of visual information is becoming important. For instance, if an organization has developed a novel and practical algorithm for face detection, it may not want to reveal the algorithm publicly but still may wish the algorithm to be widely used for maximizing the organization's profit. At the same time the user may also demand that the organization learns nothing about her input images. How can the organization and the user achieve their objectives? This question was raised and answered adequately by [1] and subsequently by [2]. In this paper, we ask an equally important but markedly different question. *Is it possible for an image database to respond correctly without knowing the query image?*

Recent years have seen a rapid increase in image collections. The earliest effort in organizing image collections was done by annotating each image with its description and forming a text based retrieval system. Due to diversity in content and increase in the size of the image collections, annotation became both ambiguous and laborious. With this, the focus shifted to Content Based Image Retrieval(CBIR) [11, 14], in which images are indexed according to their visual content. A query for such a system can either be a sample image or a part of it. Query by example aids the user in expressing his query more accurately. This decreases the *semantic gap* between the user and the system, thereby circumventing the cardinal limitation of the conventional annotation based systems.

Nowadays, the users are increasingly concerned about the privacy of their query. Some users may not want to reveal the content in their query image during retrieval, not even to the database. An example application is the logo patent search. All the logos that have been registered are stored in a public database. An organization would like to see if its new logo is very similar to any of the existing logos to avoid copyright infringements. Evidently for the fear of leakage of ingenuity in their newly proposed logo, the organization would not like to reveal it to the database. Another important application is in the medical field. A central database stores different types of CT images which have been annotated with the diagnosis and other important information. A patient would like to query the database using his CT image to obtain the relevant information. Since medical reports tend to be very private, they should not be revealed to the database.

Private Content Based Image Retrieval (PCBIR) deals with the retrieval of similar content *without* revealing the content of the query image to the database. PCBIR is achieved by exchange of messages between the user and the database. These messages collectively help the user in inferring the required information from the database but prohibit the database from knowing the user's interest. In this sense, PCBIR is similar to private information retrieval (PIR) schemes that allow a user to obtain the data stored at a specific address in a database whilst keeping the database oblivious of the address. PIR originated from the work of Chor, Goldreich, Kushilevitz and Sudan [6]. They proposed an algorithm which needed replication of databases while Chor and Gilbao [5] achieved the same using only two databases. Kushilevitz and Ostrovsky [10] and Cachin *et. al* [4] independently showed that PIR can be achieved

978-1-4244-2243-2/08/$25.00 ©2008 IEEE

without the replication of the database. However, all these works are limited to linear databases. Contrastingly, in recent years, most image datasets are indexed hierarchically. *In practice, the address corresponding to the correct answer (of a query) is typically unknown a priori to the user.* PIR is concerned about point queries, that is the value at a particular position, while PCBIR deals with a similarity search.

Since image retrieval is fundamentally a similarity search, PCBIR can be more efficiently solved (than PIR) as shown in Section 4. The lower bound on server side computation for private retrieval of arbitrary information is obviously $\Omega(n)$. This is because if the server does not read the entire database, something about the query is guaranteed to be revealed! However, in the case of *images*, concepts appear in the form of clusters. Consequently, it is enough for the server to just touch all the clusters for the query's privacy. Note that for optimizing the privacy, server should have the *identical* access patterns in all the clusters.

The Blind Vision proposed by Shai Avidan and Butman [1] applies secure multi-party techniques to vision algorithms. The setting in [1] is completely different from what it is in this work. Note that using secure multi-party computation, it is possible to solve even the PCBIR problem; however, we do not proceed in that direction because (1) General secure multi-party protocol based solution to PCBIR is usually quite inefficient when compared to tailor-made solution for the same (like the one proposed in this paper). (2) PCBIR requires privacy in only one direction. The whole database is often public while query is private.

This is probably the first attempt towards ensuring privacy in image retrieval. The traditional CBIR research focuses on feature selection, efficient indexing as well as high performance in the form of recall and precision. PCBIR is concerned with efficient retrieval under the privacy constraint without trading the recall and precision. The approach may also be viewed as a secure computation of a similarity measure.

2. Design of the Basic Algorithm

In Content Based Image Retrieval [14, 11] indexing of images is done based on the *content* in them. The content can be described by features like color, texture, shape etc. These feature representations are extracted from the images. Many of the laboratory image retrieval systems follow a flat-file storage with a linear search for retrieval. However for large databases, the retrieval speed with such a scheme becomes a concern. Moreover, the dimensionality of the feature vectors is high in image databases. To tackle the problem of high dimensionality, large databases employ tree-like or hierarchical data structures. They offer fast and efficient retrieval. These structures consist of nodes which in turn give rise to child nodes. Usually the images are stored in leaf nodes while each intermediate node contains information regarding the data stored in the subtree under it. For example, in the case of R-tree, the intermediate nodes contain the information of *Minimum Bounding Rectangle* of the subtree. The information stored at a node is useful for traversal and updating.

Given a query image by the user, its feature vector is extracted to query the index structure. During querying, one traverses the index structure by taking decision at each node to decide which child node(s) we need to traverse next. This decision is taken using the query's feature vector and the data at the node. This decision making function is usually a threshold function or a distance metric, depending on the index structure used. When a leaf is encountered, the data in it is used to get the results. Finally the database returns the results to the user. Backtracking might have to be employed in certain structures to attain the results.

The above discussed scheme is for a general CBIR system in which the user gives the query image to the database thus losing the privacy of his query. We intend to develop a method in which the database does not learn anything about the query but the user gets the results for his query. To achieve this, it is sufficient to keep the path of the traversal unknown to the database. In order to keep the path unknown, the user should not reveal node(s) being accessed.

The database takes the decision at each node using the query's feature vector and the information at the node. Since the user does not want to reveal the query, the decision at each node should be taken at the user's end. For this, the user needs the information at the node. Let us assume for instance that the user can obtain the information at a node without the knowledge of the database. With the query's feature vector already available with the user and with the information received, the user can decide which child node to access next. The user employs the same technique to get the data at these child nodes without the knowledge of the database. This is done recursively until a leaf is reached, keeping the traversal path unknown to the database. We shall explain the algorithm for a simple hierarchical structure – *the binary search tree.*

2.1. Basic Algorithm

1. The user extracts the feature vector of the query image, say f_{query}.
2. The user first asks the database to send the information at the root node.
3. Using f_{query} and the information received, the user decides whether to access the left subtree or the right subtree.
4. In order to get the data at the node to be accessed, the user frames a query Q_i where i indicates the level in

which the node occurs.(Please note that the root is at level 0)

5. The database returns a reply A_i for the query Q_i.
6. The user performs a simple function *f(A_i)* to obtain the information at the node. If the node is a leaf node, user adds the information to the *results* else `goto` step 3.

From the steps 4–6 the user obtains the information privately from the database, while step 3 is the decision making step. Since the decision is being taken at the user's end and the database does not know the nodes being accessed, we ensure the privacy of the query. CBIR can be understood as a special case of PCBIR wherein Q is the query image and A is the set of similar images computed at the server. On the other hand, if Q is $NULL$ and A is the whole database, it reduces to a trivial solution of PCBIR where the user downloads the whole database and searches at the client side. However, our solution is a substantial improvement over the trivial one so much that it is feasible in practice as demonstrated in Section 4.

We shall now explain how the exchanged messages, Q_i and A_i, are framed. The formulations of Q_i and A_i use the concept of Quadratic Residuosity Assumption (QRA) [10]. The user selects a natural number $N = p \cdot q$, where p,q are large prime numbers and a set Z_N^* is defined as

$$Z_N^* = \{x | 1 \leq x \leq n, gcd(N, x) = 1\} \quad (1)$$

A natural number y is called a Quadratic Residue (QR), if $\exists x \mid x^2 = y \bmod N$ and x,$y \in Z_N^*$ else y is called a Quadratic Non-Residue (QNR).

Fact 1.[10] For any x,y in Z_N^* it follows that, if exactly one of them is a QNR, $x \times y$ is a QNR. In other words $QR \times QNR$ is QNR while $QR \times QR$ as well as $QNR \times QNR$ are QR

A sufficiently large set Y_N is constructed with equal number of QRs and QNRs taken from Z_N^*. We now formally state QRA.

Assumption 1 (QRA). Given a number $y \in Y_N$, it is predictably **hard** to decide whether y is a QR or a QNR. By hard, we mean no known *efficient* algorithm exists.

The query Q_i is filled with numbers from the set Y_N. Due to the properties of Z_N^*, Y_N and Assumption 1, Q_i is *well defined* for the user but not for the database. Since the p and q are known only to the user, user can easily distinguish QRs from QNRs, while the database cannot.

2.2. Indexing with Binary Search Tree

In [10], authors describe a PIR solution for linear databases. We suitably adapt their solution to Binary Search Tree (BST) here. In a BST, there are 2^i nodes at level i. All the nodes at a level can be viewed as an array. The node that

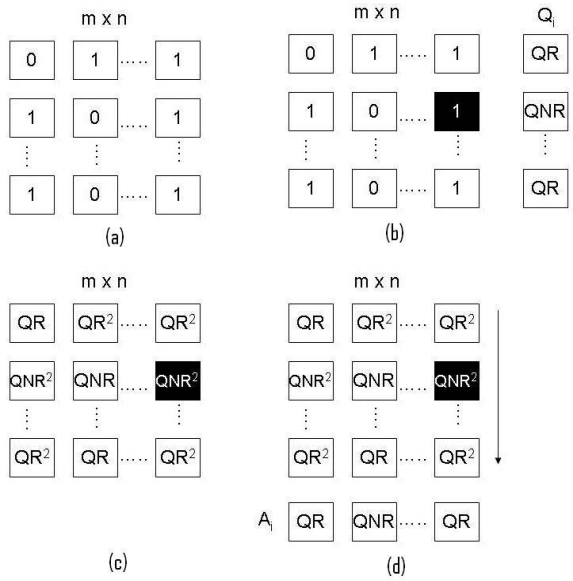

Figure 1. An example of formulation of Q_i and A_i

the user wants to access can be any one of the 2^i nodes at a level i.

Take the first bit in all the nodes and form a 2D array(say D) of dimension $m \times n$. Since only one bit is being considered from each node, $m \cdot n = 2^i$. If the user wants to access the l *th* node in a level then he wants to access the bit at $(l/n, l \bmod n)$ in D. Let $(l/n\ ,\ l \bmod n)$ be (x, y).

1. The user selects a natural number N and constructs the sets Z_N^* and Y_N.
2. To get the bit at (x, y), the user frames a query Q_i of length m, where the xth value is a QNR and the rest are QRs. The user sends Q_i to the database.(see Figure 1(b))
3. The database receives Q_i and computes A_i of length n as $A_i[k] = \prod_{j=1}^{n} w(j, k)$ where $w(j, k) = (Q_i[j])^2$ if $D_{j,k} = 1$ and $Q_i[j]$ if $D_{j,k} = 0$.(see Figure 1(c))
4. The user receives A_i and checks if $A_i[y]$ is a QR or QNR. If it is a QR then the value is 1 else 0.(from Fact 1)

The above protocol is run p times at each level to retrieve the p bits of information at the node. Figure 1(a) shows the binary data framed as a $m \times n$ matrix. Figure 1(b) shows how the user forms his query. The user is interested in the black node , hence places a QNR at the position corresponding to the row in which it appears as indicated in step 2 of the algorithm. Figure 1(c) shows the calculation of $w(i, j)$. Figure 1(d) shows multiplication along the columns to obtain A_i.

Privacy of the query is achieved by keeping the path of traversal unknown to the database. Notice that the path includes the leaf node as well. Since the algorithm proceeds

by privately retrieving the contents of successive nodes in the path, it terminates by privately retrieving the contents of the leaf node, which is the result of the user's query. Hence the result of query is privately retrieved.

Since the database does not know the positions of the QNRs in Q_i , the database is unaware of the node that the user is accessing. At each level the communication complexity is O(max(m,n)), since Q_i is of length m and A_i of length n. If m=n=$\sqrt{2^i}$ the communication complexity is O($\sqrt{2^i}$) at a level i. Hence the net communication complexity is $\sum_{i=0}^{logn} \sqrt{2^i} = O(\sqrt{n}logn)$. We remark that, analogous to the QRA-based PIR, any of the faster and later PIR schemes, when similarly adapted would serve our purpose. We prefer the QRA-based PIR for its simplicity. It is a straightforward task to extend to other PIR schemes.

3. Private Retrieval with Popular Index Structures

For fast and efficient retrieval many index structures have been proposed in both metric space and vector space. Structures in Vector space [3] include R-tree, KD-tree and SS-Tree while BK-Tree, VP-Tree have been proposed in metric space. More recently, indexing algorithms based on visual words have been proposed by [12] and [13]. Most of these have a hierarchical structure.

3.1. Hierarchical Structures

The algorithm explained in Section 2.2 can be easily extended to the other tree structures as it only needs minor modifications in the computational logic at the user end. The computational logic implies decision that needs to be taken at each node as to which of child nodes need to be considered next. The decision includes verification of a constraint for the child nodes. This constraint may involve comparison between feature vectors or calculation of a distance metric. The decision making policy and the constraint for any index structure rely on the query feature vector and the data at the node. With the query feature vector available with the user and the data at a node that he privately obtained from the server, user can easily simulate the decision making process.

Since the number of nodes at a level vary from structure to structure, the query length varies too. The user needs to maintain information about the levels and the number of nodes in them to frame a proper Q_i at a level i. The format of the information received varies with the type of information present at the node. For example in case of KD tree, at each node we store the *split dimension* and *split value*. In Vocabulary Tree [12], the nodes store the cluster centers of the child nodes. If the structure of the information is known to the user, he can easily reconstruct it from the received bits.

The algorithm also supports multiple branching from a node which is quite common in image retrieval structures. There is a possibility that we may have to access multiple nodes at the same level. Since nodes to be accessed are known to the user, he can query multiple times at a level. The generic algorithm for private retrieval in hierarchical data structures is provided below.

Algorithm 1 PCBIR in Hierarchical Structures

1: The user extracts the feature vector of the query sample, say F_q.
2: User gets the data at the root element and decides which child node(s) to move to next by using the constraint and the decision policy.
3: If the node of interest is at level i, the user frames Q_i of length equal to number of nodes at level i, with a QNR at the node of interest and sends it to the database.
4: Database receives Q_i and forms the reply A_i as explained in Section 2.
5: The user receives the reply A_i from the database and obtains the information at the node of interest.
6: **If** the node is a leaf node **then** the user
Considers the data at the node to compute results.
else
Applies the decision policy to find the child nodes to move next.
7: For every child node identified in step 6, recurse from step 3.

The recently proposed state of the art indexing structure, Vocabulary Tree proposed by Nister [12] has a hierarchical structure. It employs SIFT features which are extracted from the images. They are hierarchically clustered to yield a vocabulary of visual words as well as a vocabulary tree. The images are then indexed using this tree with each node storing a list of images that visit it. Since the vocabulary tree is hierarchical in nature, the above private retrieval algorithm can be applied to it.

3.2. Hash-based Retrieval

Hashing techniques like *LSH* [7] are widely used for image retrieval. A hashing technique employs the use of a hash function *h* which is used to divide the images in a database into bins. All the images with the same output value for the hash function are placed in a single bin. The number of bins is affected by the hash function and its range of input and output values. Given a query image, the hash function is applied to it and is mapped to a bin depending on the output value. Only the images in that bin are retrieved as the results of the query and thereby increasing the performance.

The PCBIR algorithm can be applied to hashing techniques also. We can treat the bins as an array of nodes

which is similar to the array of nodes in a level in a hierarchical indexing scheme. The data in the bin to which the query image is mapped by the hash function is to be retrieved. The algorithm for private retrieval using hashing techniques is given below.

Algorithm 2 PCBIR in Hash Tables

1: The user applies the hash function on the query and determines the bin in to which it falls.(lets say it is i)
2: The user frames and sends a query Q (as described in Section 2) to obtain the information at node i.
3: The database replies to the query Q with an answer A.
4: The user obtains the information at node i from A.

The steps 2,3 and 4 are the steps of PCBIR algorithm. Thus we treat the hash bins as nodes at a level and apply the basic PCBIR algorithm to retrieve the data at a specific node. In other words, we treat the hash bins as a tree with one level hierarchy.

LSH is also similar to the general hashing technique except that it uses multiple hash functions, so as to ensure that for each function, the probability of collision is much higher of objects which are close together. LSH has been widely used in various applications in vision [8, 9]. Given a query image, it is hashed using all the hash functions and the k-nearest neighbors are computed from the images retrieved by each hash function. It is similar to performing N simple hash retrievals. By applying the above algorithm individually to each of the N hash functions, privacy can be obtained in LSH.

3.3. Hash Tree Based Retrieval

As discussed earlier in Section 1, if the server does not read the entire database, it knows that the query is not related to that part that it has not read. Thus a private retrieval scheme has to be $\Omega(n)$ in server side computation. However this theoretical limit can be surpassed with minimal trade off in privacy by using hash tree based indexing techniques. All images in a hash bin are similar. We can take advantage of this property to lower the computational complexity. For example consider a hash function that distributes images into bins of flowers, mountains, buildings etc. We again apply a hash function within each bin to construct a hash tree. For example flowers can be further classified into roses, sunflowers, lilies etc. The obtained hash bins can be further hashed, thus forming a *hash tree* (hierarchical hashing), as shown in Figure 2.

In non-private retrieval, the server maps the query image to a hash tree and traces down a path to a leaf node using a hash function at each level. Since all the hash functions are public, the user can track the path himself(without any server intervention). Say there are n bins in the first level and the query is mapped to the i^{th} bin. We traverse the path

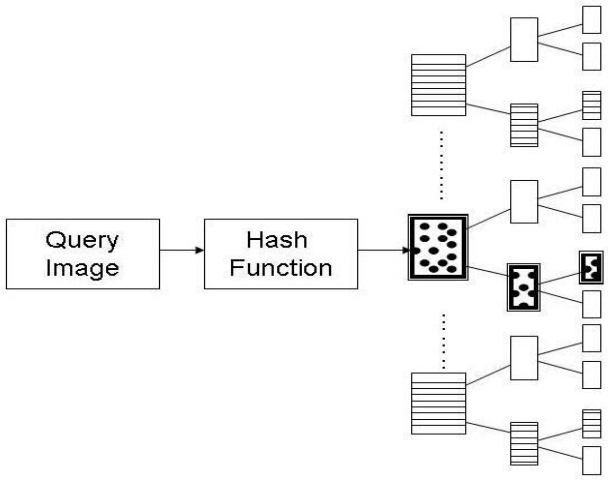

Figure 2. The figure shows a hash tree based indexing scheme. The query image is mapped to a hash bin (or the root node of a hash tree) by the hash function. The path traced by the image in this hash tree is followed in all the other hash trees, confuses the database.

in the hash tree of the i^{th} bin and reach a leaf node which contains the image results for the query. *To keep it private we trace the same path in all the other hash sub trees.* Each sub tree leads to a leaf node thus giving us m leaf nodes. We consider these m leaf nodes to be an array of nodes and apply the private retrieval scheme to extract the information at the i^{th} node. In Figure 2 the query is mapped to dotted bin. The path traversed in this bin is indicated by the dotted bins. The same paths traversed in the other hash sub trees are indicated by the dashed boxes. The database does not know which node the user is accessing thus keeping the privacy intact. Recall that, in traditional PIR, we had to read the complete data in each bin to obtain the result thus making the computation $O(n)$. In this case the server reads only a small part of the data in each bin, thus reducing the computation as well as keeping the database in oblivious of the bin that the user's query is mapped to.

Confusion metric η is defined as the ratio of nodes accessed to the total number of nodes. Hence *confusion metric*

$$\eta = \frac{m}{mk^{\ell}} = 1/k^{\ell}$$

where k is the number of child nodes a node can give rise to in the hash tree and ℓ is the depth of the hash tree. The confusion metric defines how much the server has been confused regarding the query. The confusion metric attains highest confusion when its value is 1 i.e Hash tree of depth 0 (as in Section 3.2). The hash tree technique helps us in trading privacy(or confusion) for the computation complexity.

Algorithm 3 Hash Tree based Private Retrieval

1: The user applies the hash function on the query and determines the bin in to which it falls.(lets say it is i)
2: The user uses the publicly available hash functions to traverse to a leaf of the hash tree of the i^{th} bin.
3: The user traces the same path in all the hash trees. He finally has m leaf nodes/bins.
4: Considering these leaf nodes to be an array of nodes, the user applies the private retrieval scheme explained previously, to obtain the information at the i^{th} node.

Analysis Given an initial hash function which divides the data in to m hash bins, hash tree is constructed where each hash bin has k^ℓ leaf nodes i.e., each node gives rise to k nodes and the height of the tree ℓ. Assuming data is uniformly distributed among the leaf nodes, the amount of data in each node is $O(I/(m * k^\ell))$ (I is the total number of images). Since we are accessing only m leaf nodes the server needs to access only $m \cdot O(I/(m * k^\ell)) = O(I/k^\ell)$ images. In case of a hashing technique or a hierarchical indexing scheme, the computational complexity was $O(I)$ which has been significantly reduced in this case.

However note that while using a hash tree, we are incurring an additional preprocessing cost of $O(k^\ell)$. Counting this, there exists a break even point where the overall gain in time (online -offline) is maximized. The online computation complexity is $O(\frac{I}{k^\ell})$ and offline is $O(k^\ell)$. Therefore overall time is $O(\frac{I}{k^\ell} + k^\ell)$. Maximizing the gain by differentiating with respect to k^ℓ and equating to zero, we obtain the optimal value of k^ℓ as $O(\sqrt{I})$ and the optimum time as $2\sqrt{I}$. One can therefore conclude that as long as $k^\ell \geq \sqrt{I}$, there is a guaranteed enhancement (using our hash-tree technique) in the overall performance that include both online and offline run times. Evidently for all practical purposes, this is true.

4. Experiments and Discussions

In this section, we validate the algorithm and demonstrate the utility in real-life situations. Experiments were conducted to test the performance of the algorithm under different indexing structures and feature descriptors. Feature descriptors used are high dimensional. However descriptors of any length could be used as the length of the feature vector would not substantially affect the performance of our algorithm. Experiments were also conducted to validate the accuracy and study the scalability of the algorithm. The results are attained using GNU C compiler on a 2GHz Intel Pentium M processor with 4 GB of main memory.

Corel Database and K-D Tree The first experiment validates the feasibility of the algorithm when the database is indexed using color features and K-D tree. Both colour features and K-D trees are popular in CBIR literature [15]. The Corel database with 9907 images, scaled to 256×128 resolution, were used to test the algorithm. The images were indexed by K-D tree using color histogram [15] as the feature. Three color channels (R,G,B), each with 256 color values were used to build the feature vector of dimension 768. The private content based image retrieval algorithm was applied over the K-D Tree. Each intermediate node contained the split dimension and the split value. The information at each node was represented in 64 bits (32 bits for each value). The four nearest neighbors were retrieved for a query image. Results for sample queries are shown in Figure 3. The image marked by a black bounding box is the query and the others are the top four retrieved results. The non-private retrieval algorithm on the K-D tree also retrieved the same set of results proving the algorithm to be correct and accurate. With traditional features and indexing schemes, we were able to achieve real-time retrieval with approximately 10K images.

Figure 3. Results of private retrieval over Corel Database indexed by K-D Tree. The first image marked by a black bounding box is the query image, followed by the top four retrieved images.

Visual Words and Vocabulary Tree In the next experiment, we validate the applicability of the algorithm to a state of the art indexing scheme. Vocabulary Tree [12] has been proved to be efficient in handling large datasets offering a much better performance than traditional tree based indexing schemes. The dataset consists of 10,200 images which were used by Nister [12] for vocabulary tree. The database consists of images of several objects taken from different positions and under varying illumination conditions. The database is indexed using SIFT features. Each image yielded an average of 700 features with each feature represented in 128 dimensions.

Vocabulary Tree was constructed using a training set, which consisted of 2000 images from the database. Query images were also selected from the same database, randomly. Sample queries and the obtained results are shown shown in Figure 4. The image with a black bounding box is the query and the other images are the top three retrieved

Figure 4. Results of private retrieval over Vocabulary Tree. The first image marked by a black bounding box is the query image,followed by the top three retrieved images.

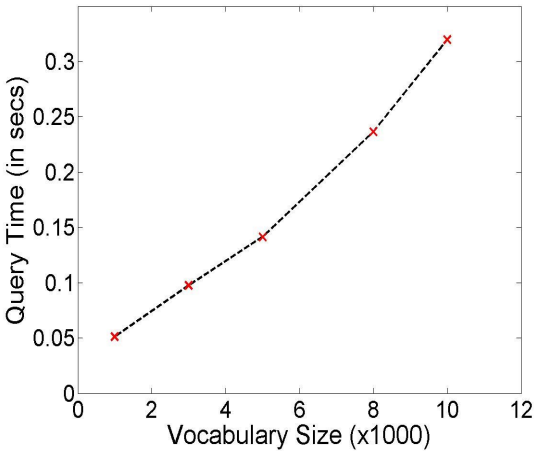

Figure 5. Plot of Vocabulary Size vs Query Time. The graph indicates that the query time increases with the increase in the size of the vocabulary. As the number of visual words increases, the sift features are distributed over a larger vocabulary. Hence each image is indexed under more number of visual words increasing the data in the leaf nodes.

results.

The advantage of using vocabulary tree is that the size of the vocabulary decides the size of the tree. The number of leaf nodes in the tree determines the number of visual words. The retrieval speeds for various sizes of vocabulary are plotted in the graph in the Figure 5. Even for a vocabulary size of 10,000, the retrieval time is in milliseconds. The query time has been averaged over 50 queries, which were processed at a single time using amortization. With the increase in the vocabulary size, the size of the vocabulary tree increases which causes an increase in the retrieval time. Results are presented in Figure 5.

LSH and Image Retrieval Another popular indexing structure for image retrieval is Locality Sensitive Hashing [7]. LSH has been used to index the Corel database consisting of 9907 images. Ninety hash functions, each with 450 bins on an average, were used to index the database. The images in each bin of every hash table were again subdivided using LSH. Effectively a *hash tree* with two level hierarchy was constructed. LSH-based retrieval is found to be more efficient than hierarchical trees, since it accesses only a sub set of the database (see shaded regions in Figure 2)

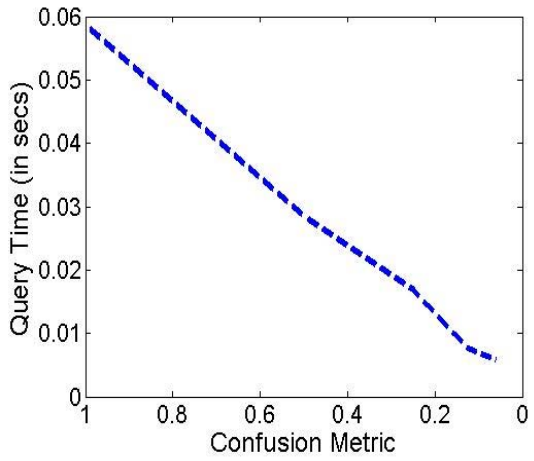

Figure 6. Plot of Confusion Metric vs Query Time. As the confusion metric decreases, lesser number of nodes or lesser amount of data needs to be accessed by the database server thus reducing the computational load on the server side during retrieval. Since lesser number of nodes are being accessed communication is also reduced.

We also studied the efficiency of the retrieval with the confusion metric η. Results are graphically presented in Figure 6. It may be seen that there exists a near-linear region where the trade-off between privacy and efficiency can be controlled directly. With some level of compromise in confusion, efficient retrieval can be achieved.

The query time for each of the three indexing structures have been reported in Table 1. (The average query time for Vocabulary tree was measured on a vocabulary of size 10,000.) All the databases are approximately of 10,000 images. From the query time, we can see that the algorithm works in real time and private retrieval is practical in large image databases. Corel database indexed with LSH reports the least query time since the server accesses only a part of the data in each bin as explained previously. A non private retrieval on a flat file based indexing scheme, employing a large database, takes significantly more time than those reported in Table 1.

Scalability and Applicability Now we demonstrate the feasibility of the private retrieval for very large datasets. The algorithm was tested on data sets of various sizes and

Indexing Algorithm	Database	Query Time(secs)
K-D Tree	*Corel Database*	0.596
Vocabulary Tree	*Nister Dataset*	0.320
LSH	*Corel Database*	0.221

Table 1. Query times for various hierarchical indexing structures. Private retrieval in all the three index structures is practical.

Dataset Size	Query Time(in secs)
2^{10}	0.005832
2^{12}	0.008856
2^{14}	0.012004
2^{16}	0.037602
2^{18}	0.129509
2^{20}	0.261255

Table 2. Query times for varying database size. Table compares the retrieval times when PCBIR is applied on synthetic datasets of various sizes.

the corresponding query time is reported in Table 2. The information at each node was taken to be 64 bits. The communication complexity being an incremental function of the size of database, increases as the number of samples increase. The retrieval time for a data set of size one million is still in milliseconds showing that the algorithm is scalable to hierarchical structures handling large databases. The proposed algorithm has been shown to work on major indexing schemes such as Hierarchical Trees, Vocabulary Trees and Hashing. The numerical figures show that the algorithm is scalable and efficient. Hence it can be used in any large scale applications that ask for the privacy of the user's query. The major bottleneck in PCBIR is the server side computation and not the communication. This is because the server side computation requirement grows linearly whereas the communication grows only sublinearly. While this linear bottleneck may be acceptable for most situations, in the worst case one could use the hash tree based solution to obtain sublinear load on the server.

5. Conclusion

In this paper we have addressed the problem of private retrieval in image databases. The algorithm is shown to be customizable for hierarchical data structures as well as hash-based indexing schemes. Experimental study has proved that the algorithm is accurate, efficient and scalable. We proposed algorithms that are fully private and feasible on reasonably large databases using a variety of state of the art indexing schemes. We have also demonstrated that image databases are amenable to significantly faster private retrieval by exploiting the property of image data. Specifically we demonstrated a near-linear operating region for image databases, where the trade off between privacy and speed is feasible.

References

[1] S. Avidan and M. Butman. Blind vision. In *European Conference on Computer Vision (ECCV)*, pages 1–13, May 2006.

[2] S. Avidan and M. Butman. Efficient methods for privacy preserving classification. In *Neural Information Processing Systems (NIPS)*, June 2006.

[3] C. Bohm, S. Berchtold, and D. A. Keim. Searching in high-dimensional spaces:index structures for improving the performance of multimedia databases. In *ACM Computing Surveys*, pages 322–373, September 2001.

[4] Cachin, Micali, and Stadler. Computationally private information retrieval with polylogarithmic communication. In *18th Annual International Conference on the Theory and Application of Cryptographic Techniques (EUROCRYPT)*, pages 402–414, 1999.

[5] B. Chor and N. Gilboa. Computationally private information retrieval. In *Proc. 29th Annual ACM Symposium on Theory of computing (STOC)*, pages 304–313, 1997.

[6] B. Chor, O. Goldreich, E. Kushilevitz, and M. Sudan. Private information retrieval. In *Proc. 36th IEEE symposium on Foundation of Computer Science (FOCS)*, pages 41–50, 1995.

[7] M. Datar, P. Indyk, N. Immorlica, and V. Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *Proceedings of the Symposium on Computational Geometry*, 2004.

[8] J. J. Foo, J. Zobel, R. Sinha, and S. M. M. Tahaghoghi. Detection of near-duplicate images for web search. In *Proceedings of the 6th ACM international conference on Image and video retrieval*, pages 557–564, 2007.

[9] T. D. G. Shakhnarovich, P. Viola. Fast pose estimation with parameter sensitive hashing. In *International Conference on Computer Vision*, pages 750–757, 2003.

[10] E. Kushilevitz and R. Ostrovsky. Replication is not needed: Single database, computationally-private information retrieval. In *Proc. 38th IEEE symposium on Foundation of Computer Science (FOCS)*, pages 364–373, 1997.

[11] M. S. Lew, N. Sebe, C. Djeraba, and R. Jain. Content-based multimedia information retrieval:state of the art and challenges. In *ACM Transactions on Multimedia Computing, Communications, and Applications*, pages 1–19, February 2006.

[12] D. Nister and H. Stewenius. Scalable recognition with a vocabulary tree. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 2161–2168, 2006.

[13] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. Object retrieval with large vocabularies and fast spatial matching. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2007.

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

[15] M. Swain and D. Ballard. Indexing via color histograms. In *IEEE International Conference on Computer Vision*, pages 390–393. IEEE CS Press, 1990.