Text and Content Based Image Retrieval Via Locality Sensitive Hashing

Nan Zhang, Ka Lok Man, Tianlin Yu and Chi-Un Lei

Abstract—**We present a scalable image retrieval system based jointly on text annotations and visual content. Previous approaches in content based image retrieval often suffer from the semantic gap problem and long retrieving time. The solution that we propose aims at resolving these two issues by indexing and retrieving images using both their text descriptions and visual content, such as features in colour, texture and shape. A query in this system consists of keywords, a sample image and relevant parameters. The retrieving algorithm first selects a subset of images from the whole collection according to a comparison between the keywords and the text descriptions. Visual features extracted from the sample image are then compared with the extracted features of the images in the subset to select the closest. Because the features are represented by high-dimensional vectors, locality sensitive hashing is applied to the visual comparison to speedup the process. Experiments were performed on a collection of 1514 images. The timing results showed the potential of this solution to be scaled up to handle large image collections.**

Index Terms—**Image retrieval, Content based Image retrieval, Locality sensitive hashing.**

I. Introduction

WITH the rapid growth of the numbers of digital images on the Internet and in private or public collections, the need for effectively and efficiently retrieving them has become demanding [1]. Text based retrieval has been widely used where images are indexed by text terms and retrieved by matching terms in a query with those indexed. Due to its simplicity, text based approaches can be easily scaled up to handle billions of image. However, text annotations often carry little information about images' visual features. When users wish to retrieve images of similar visual content, a pure text based approach becomes inadequate.

Content based image retrieval (CBIR), instead of using text annotations as the basis for indexing and searching, uses visual features extracted from images, such as colour, texture, shape and spatial relations of pixels. Unlike text annotations which are subject to human perception, these features make objective representations of images. Since early 1990's, many CBIR systems have been developed [2]–[4]. However, due to the semantic gap and the scaling problem, till now, there is still not a widely used CBIR system like those text based web image search engines.

In the context of CBIR, the semantic gap [5] refers to the problem that the low-level visual features such as various representations of colour, texture and shape often do not carry semantic meanings understood by human beings. For this reason, CBIR approaches often suffer from high retrieval error rate. The scaling problem, on the other hand, prevents a CBIR system from handling large image collections. Times required by the comparison on visual features often grow rapidly as the size of a collection increases. Most representations of visual features are high dimensional vectors of real numbers. The dimensionality can be as high as hundreds or even thousands. It has been demonstrated by earlier research [6] that when facing such high dimensions, the tree-like data structures [7] provide little improvement over a linear searching algorithm.

The solution that we propose aims at addressing two problems in CBIR. The indexing algorithm requires that every image is accompanied by a text description, which contains textual information semantically relevant to the image. A full-text inverted index is created on the description files. Meanwhile, visual features are extracted from the images and are saved with the inverted index. When querying the system, a user needs to provide a text query and a sample image with certain parameters. The text query is used to search through the inverted index to obtain a set of relevant documents, and the set of corresponding images. This set of images are all deemed semantically relevant to the query. The visual features extracted from the sample image are then compared with the features of the images in the subset to further select images of similar visual effects. A locality sensitive hashing [8] is applied to the visual comparison.

The advantages of this approach are threefold. First, the semantic selection using the keywords makes the subsequent visual comparison perform within a set of semantically relevant images. This helps to narrow the semantic gap that exists in a pure CBIR system. Second, because of the semantic selection, the scale of the visual comparison becomes manageable, and so an upper bound can be set for the time required by the visual search. Third, the locality sensitive hashing applied to the visual comparison accelerates the process. Because the hashing functions are computed every time before the search starts, the need is avoided for pre-building dedicated data structures for the extracted visual features.

In this paper, we propose a scalable image retrieval system based jointly on text annotations and visual content. After the discussion of related work in Section II, the indexing and the searching algorithms are presented in Section III. Computational times of the algorithms are analysed in Section IV, followed by the report of the experimental data in Section V. Conclusions are drawn in Section VI, which also contains a discussion about future improvements.

II. Related work

Over the past few years, various techniques have been integrated into CBIR systems to improve the rate of relevant

Manuscript received 3 June 2011.

Nan Zhang, Ka Lok Man and Tianlin Yu are with Department of Computer Science and Software Engineering, Xi'an Jiaotong-Liverpool University, Suzhou, China, 215123. Email: {nan.zhang, ka.man}@xjtlu.edu.cn, tianlin.yu08@student.xjtlu.edu.cn.

Chi-Un Lei is with Department of Electrical and Electronic Engineering, the University of Hong Kong, Hong Kong. Email: culei@eee.hku.hk.

images in the result set. Such techniques include unifying keywords and visual features for indexing and retrieving, using mechanisms of relevance feedback, applying ontology based structures, querying by concept, etc. In the system developed by Zhou and Huang [9], each image was represented by vectors of visual features and text annotations. Keywords were semantically grouped based on user's feedback made during the retrieval process. The system supported joint queries of keywords and example images. Through relevance feedback, retrieving results were further refined.

Zhang and Song [10] implemented a hybrid image retrieval system that was based on keywords and visual contents. Text descriptions of images were stored in a database, on which full-text index catalogues were created. Vectors of visual contents were extracted and saved into a Lucene index. The system was queried jointly by keywords and an example image.

RetrievOnto, an image retrieval system discussed in [11], allied CBIR and semantic techniques, where the image data set was structured by an ontologically controlled term hierarchy extracted from WordNet [12]. The term hierarchy specified conceptual neighbours when similar items were searched. Image sets were directly associated to the leaf terms, and indirectly to all concepts in the hierarchy. The system supported querying by conceptual terms with controlled semantic neighbourhood.

An image retrieval methodology was proposed in [13], where images were divided into regions by a fully unsupervised segmentation algorithm. These regions were indexed by low-level descriptors of colour, position, size and shape, which were associated with appropriate qualitative intermediate-level descriptors that carried high-level concepts. A user could query the system by keywords which carried the high-level concepts. Comparisons were then made with the intermediate-level descriptors and the associated image regions. A relevance feedback mechanism based on support vector machines was employed to rank the obtained image regions that were potentially relevant to produce the results.

A hybrid model of image retrieval was proposed and implemented in [14], where ontology and probabilistic ranking were applied. When the system was queried by a keyword, images annotated by the keyword were selected together with those annotated by keywords conceptually related. The degree of relevance was evaluated by an ontology reasoner whose output were passed to a Bayesian Network to get the rankings.

For large-scale applications of CBIR, linear search over high-dimensional feature vectors must be avoided. Cortina [15], a large-scale image retrieval system for the World Wide Web, was reported to be able to handle over 3 million images. The system had a crawler which collected images and their text descriptions. The text descriptions were stored in a database, where inverted index over the keywords were created. Four MPEG-7 visual features were extracted from the images and stored in another database. To reduce the searching time, the whole dataset was organised in clusters by each descriptor type. When querying the system, a user had to submit a keyword to search through the inverted index to get a set of matching images. The user then had to select one or several images that were visually close to what he/she was looking for. Query vectors from these chosen images were constructed to perform a nearest neighbour search in the spaces of feature descriptors. To avoid a linear search, the visual feature vectors were clustered by the k-means algorithm [16].

The text-based CBIR approaches proposed in [17] were meant to provide quality results within searching times that are acceptable to users who are used to the performance of text search engines. Like Cortina, several MPEG-7 visual descriptors were extracted from the images crawled from the SPIRIT collection [18]. The descriptors were saved as XML documents. An inverted index was created over the terms of the feature vectors. Queries were in the form of example images.

A system architecture for large-scale medical image understanding and retrieval was described in [19], where a hierarchical framework of ontologies was used to form a fusion of low-level visual features and high-level domain knowledge. The implementation was based on the Lucene Image Retrieval Engine (LIRE) and the system supported query by text, by concept and by sample image.

A system for near-duplicate detection and sub-image retrieval was described in [20]. Instead of using global visual features such as colour histograms, the system used a local descriptor, PCA-SIFT [21], to represent distinctive interest points in images. To index the extracted local descriptors they employed locality sensitive hashing [22]. With further optimisation on layout and access to the index data on disk, they could efficiently query indexes containing millions of interest points.

Locality sensitive hashing [22], [23] is an indexing scheme for vectors of high dimensionality where traditional tree-like data structures are unlikely to achieve a better performance than a linear search. Some recent developments are found in [8], [24], [25]. This hashing scheme was proposed to solve the c-approximate R-near neighbour problem, which is defined as given a set of P points and a query point q in a d-dimensional space, and parameters $R > 0$, $\delta > 0$, construct a data structure such that, if there exists an R-near neighbour of q in P, it reports some cR-near neighbour of q in P with probability $1 - \delta$. The successful application of this scheme depends on the existence of families of locality sensitive hash functions that satisfy the condition that the probability of a hash collision for two points that are close to each other is greater than that of a hash collision for two points that are far apart. To amplify the gap between the probabilities the algorithm uses several such functions and concatenate the hashing results. The experimental evidence presented in [23] demonstrated that the scheme of locality sensitive hashing gave significant improvement in runtime over tree-based methods, such as SR-tree, for searching in high-dimensional spaces.

III. Algorithms for Indexing and Retrieving

In [26] (pp. 23), an information retrieval model is defined as a quadruple $(D, Q, \mathcal{F}, R(q_i, d_j))$, where D is a set of logical views for the documents in the collection; Q is a set of logical views for queries; $\mathcal{F}$ is a framework for modelling document representations, queries and their relationships; and $R(q_i, d_j)$ is a ranking function which associates a real number with a query $q_i \in Q$ and a document representation $d_j \in D$.

In our solution, images are jointly represented by their semantic views and visual views. The solution requires that, in the collection, each image is accompanied by a text description. (These descriptions can either be created manually, or be obtained from crawling over the Internet, or over some collections, as in [17].) Let set $I = \{I_1, I_2, \ldots, I_n\}$ denote the collection of images, and set $D = \{d_1, d_2, \ldots, d_n\}$ denote the text descriptions with d_i being the description for I_i. The semantic view of an image I_i is the index term vector $\mathbf{d_i}$ defined in the vector space model of text retrieval.

Definition *1 (Semantic view):* Let m be the number of index terms in the system, and $T = \{t_1, t_2, \ldots, t_m\}$ be the set of all index terms. A weight $w_{t_i d_j} > 0$ is associated with each index term t_i of a description d_j. For a term which is not found in d_j, the associated weight is zero. The semantic view of image I_i is the index term weight vector of d_i, that is, the vector of the weights of terms in d_i. This vector is denoted by $\mathbf{d_i}$. Let $\mathcal{S}(I_i)$ denote the semantic view of image I_i, and we have $\mathcal{S}(I_i) = \mathbf{d_i} = (w_{t_1 d_i}, w_{t_2 d_i}, \ldots, w_{t_m d_i})$. □

To define the visual view of image I_i, we assume the number of visual features used to represent each image is k.

Definition *2 (Visual view):* Let set $\mathcal{V} = \{\mathcal{V}_1, \mathcal{V}_2, \ldots, \mathcal{V}_k\}$ be a family of mappings, and each $\mathcal{V}_j$ maps image I_i to a vector of real numbers. Let $\mathcal{V}_\mathbf{j}(\mathbf{I_i})$ denote the vector. The visual view of image I_i is the set of the vectors obtained by applying the mappings $\mathcal{V}_1, \mathcal{V}_2, \ldots, \mathcal{V}_k$ to the image. We call each of these vectors a visual feature vector. Let $\mathcal{V}(I_i)$ denote the visual view of image I_i, and we have $\mathcal{V}(I_i) = \{\mathcal{V}_\mathbf{1}(\mathbf{I_i}), \mathcal{V}_\mathbf{2}(\mathbf{I_i}), \ldots, \mathcal{V}_\mathbf{k}(\mathbf{I_i})\}$. □

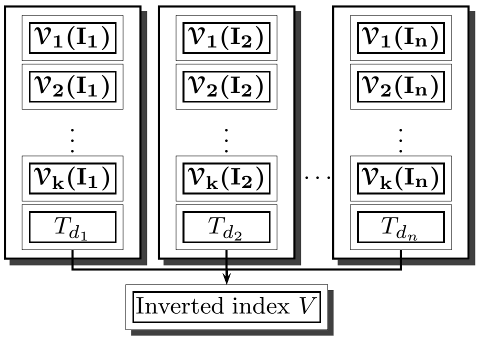
When creating index on image collection I and description set D, for each image-description pair (I_i, d_i) $i \in [1, n]$, description d_i is broken into a set T_{d_i} of index terms $\{t_{d_i 1}, t_{d_i 2}, \ldots, t_{d_i |d_i|}\}$, where $|d_i|$ denotes the number of index terms found in d_i. Image I_i is transformed by all the mappings in $\mathcal{V}$ to get its semantic view $\mathcal{V}(I_i)$. We use $P_{(I_i, d_i)}$ to denote the result obtained by processing image-description pair (I_i, d_i), and $P_{(I_i, d_i)}$ is a tuple which contains T_{d_i} and $\mathcal{V}(I_i)$. So, we have $P_{(I_i, d_i)} = (T_{d_i}, \mathcal{V}(I_i))$, where $T_{d_i} = \{t_{d_i 1}, t_{d_i 2}, \ldots, t_{d_i |d_i|}\}$, and $\mathcal{V}(I_i) = \{\mathcal{V}_\mathbf{1}(\mathbf{I_i}), \mathcal{V}_\mathbf{2}(\mathbf{I_i}), \ldots, \mathcal{V}_\mathbf{k}(\mathbf{I_i})\}$.

The terms in T_{d_i} are then analysed and added to an inverted index V. Following the approach of the vector space model in text document retrieval, the weight $w_{t_i d_i}$ of index term t_i in d_i is calculated by Equation 1,

$$w_{t_i d_i} = \frac{\mathtt{freq}_{t_i d_i}}{\mathtt{max}_l \mathtt{freq}_{t_l d_i}} \times \log \frac{n}{n_{t_i}} \qquad (1)$$

where $\mathtt{freq}_{t_i d_i}$ is the number of times the term t_i appears in the text of d_i, $\mathtt{max}_l \mathtt{freq}_{t_l d_i}$, the maximum occurrence, is computed over all terms in d_i, n is the number of descriptions in the collection, and n_{t_i} is the number of descriptions where t_i appears. (See Fig. 1.)

When querying the system, the user submits a text query T_q, an example image I_q, a weight $w_t \geq 0$ for evaluating the rankings obtained by the semantic similarity comparison, and a set $W_\mathcal{V}$ of weights for evaluating the rankings obtained by the visual similarity comparison. In the vector space model, T_q is represented by the weight vector $\mathbf{T_q}$, and $\mathbf{T_q} = (w_{t_1 T_q}, w_{t_2 T_q}, \ldots, w_{t_m T_q})$. For those terms in the system but do not appear in T_q, the corresponding weights are zeroes.



$\mathcal{V}_1(\mathbf{I_1})$
$\mathcal{V}_2(\mathbf{I_1})$
$\mathcal{V}_\mathbf{k}(\mathbf{I_1})$
T_{d_1}
$\mathcal{V}_1(\mathbf{I_2})$
$\mathcal{V}_2(\mathbf{I_2})$
$\mathcal{V}_\mathbf{k}(\mathbf{I_2})$
T_{d_2}
$\mathcal{V}_1(\mathbf{I_n})$
$\mathcal{V}_2(\mathbf{I_n})$
$\mathcal{V}_\mathbf{k}(\mathbf{I_n})$
T_{d_n}
Inverted index V

Fig. 1: The structure containing the inverted index and the feature vectors.

Algorithm 1: Computing the tuples $P_{(I_i, d_i)}$, $i = 1, 2, \ldots, n$.

```
1 begin
2     for each image – description pair $(I_i, d_i)$ do
3         for each mapping $\mathcal{V}_j$ in $\mathcal{V}$ $j = 1, 2, \ldots, k$ do
4             Compute feature vector $\mathcal{V}_\mathbf{j}(\mathbf{I_i})$.
5             Add $\mathcal{V}_\mathbf{j}(\mathbf{I_i})$ to $P_{(I_i, d_i)}$.
6         Break $d_i$ into a set $T_{d_i}$ of index terms.
7         Add $T_{d_i}$ to $P_{(I_i, d_i)}$.
8         Add terms in $T_{d_i}$ to inverted index $V$.
9 end
```

Now we define the logical view of the query in our solution as follows.

Definition *3 (Query):* The logical view Q of a query is a tuple $(\mathbf{T_q}, \mathcal{V}(I_q), w_t, W_\mathcal{V})$ where

1) $\mathbf{T_q} = (w_{t_1 T_q}, w_{t_2 T_q}, \ldots, w_{t_m T_q})$ is the vector represents weights of index terms in T_q.
2) $\mathcal{V}(I_q) = \{\mathcal{V}_\mathbf{1}(\mathbf{I_q}), \mathcal{V}_\mathbf{2}(\mathbf{I_q}), \ldots, \mathcal{V}_\mathbf{k}(\mathbf{I_q})\}$ is the visual view of the example image I_q.
3) w_t is the weight for evaluating rankings obtained from the semantic comparison, $0 \leq w_t \leq 1$.
4) $W_\mathcal{V} = \{w_{\mathcal{V}_1}, w_{\mathcal{V}_2}, \ldots, w_{\mathcal{V}_k}\}$ is the set of weights for evaluating rankings obtained from the visual comparison. Each weight $w_{\mathcal{V}_j}$ $j \in [1, k]$ is for evaluating rankings obtained from the searching using feature vector $\mathcal{V}_\mathbf{j}(\mathbf{I_q})$, $0 \leq w_{\mathcal{V}_j} \leq 1$.
5) The condition for the weights is $(w_t + \sum_{i=1}^{k} w_{\mathcal{V}_i}) > 0$. □

The entire retrieving process is divided into the stage of semantic similarity comparison and the stage of visual similarity comparison. In the stage of semantic similarity comparison the text query T_q is compared through the inverted index V, and a score $s^{\mathcal{S}}_{(T_q, d_i)}$ is calculated for T_q and each description $d_i \in D$, which measures the degree of relevance that d_i is to T_q. Following the vector space model $s^{\mathcal{S}}_{(T_q, d_i)}$ is calculated by Equation 2

$$s^{\mathcal{S}}_{(T_q, d_i)} = \frac{\mathbf{T_q} \cdot \mathbf{d_i}}{|\mathbf{T_q}| \times |\mathbf{d_i}|} = \frac{\sum_{j=1}^{m} (w_{t_j T_q} \times w_{t_j d_i})}{\sqrt{\sum_{j=1}^{m} w^2_{t_j T_q}} \times \sqrt{\sum_{j=1}^{m} w^2_{t_j d_i}}} \qquad (2)$$

where m is the number of index terms in the system.

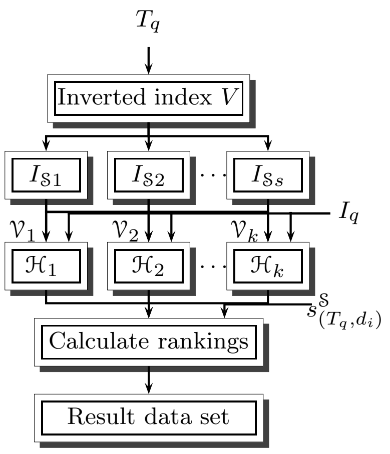


Fig. 2: The searching algorithm ($\mathcal{H}_i$ $i \in [1,k]$ denotes one application of the locality sensitive hashing).

We place a threshold on the scores and only the descriptions whose scores pass the threshold are selected. We use $D_{\mathbb{S}}$ to denote this set of descriptions, and $D_{\mathbb{S}} = \{d_{\mathbb{S}1}, d_{\mathbb{S}2}, \ldots, d_{\mathbb{S}s}\}$, where s denotes the total number of the descriptions in this set. From descriptions in $D_{\mathbb{S}}$, we get the set of corresponding images, which are all deemed semantically relevant to query T_q, and are passed to the stage of visual similarity comparison. We use $I_{\mathbb{S}}$ to denote this set and $I_{\mathbb{S}} = \{I_{\mathbb{S}1}, I_{\mathbb{S}2}, \ldots, I_{\mathbb{S}s}\}$.

In the next stage, locality sensitive hashing is applied k times to the visual views of the images in $I_{\mathbb{S}}$. As in Definition 2 we use $\mathcal{V}_j$ to denote a mapping in $\mathcal{V}$, where $j \in [1,k]$. The set $\mathcal{V}_j(I_{\mathbb{S}})$ of feature vectors of images in $I_{\mathbb{S}}$ under mapping $\mathcal{V}_j$ is $\{\mathcal{V}_\mathbf{j}(\mathbf{I}_{\mathbb{S}\mathbf{1}}), \mathcal{V}_\mathbf{j}(\mathbf{I}_{\mathbb{S}\mathbf{2}}), \ldots, \mathcal{V}_\mathbf{j}(\mathbf{I}_{\mathbb{S}\mathbf{s}})\}$. The feature vector extracted from example image I_q under mapping $\mathcal{V}_j$ is $\mathcal{V}_\mathbf{j}(\mathbf{I_q})$. Locality sensitive hashing is then applied to set $\mathcal{V}_j(I_{\mathbb{S}}) = \{\mathcal{V}_\mathbf{j}(\mathbf{I}_{\mathbb{S}\mathbf{1}}), \mathcal{V}_\mathbf{j}(\mathbf{I}_{\mathbb{S}\mathbf{2}}), \ldots, \mathcal{V}_\mathbf{j}(\mathbf{I}_{\mathbb{S}\mathbf{s}})\}$. Feature vector $\mathcal{V}_\mathbf{j}(\mathbf{I_q})$ is then compared through the hashed structures. A score $s^{\mathcal{V}_j}_{(I_q,I_{\mathbb{S}i})}$ $j \in [1,k]$ and $i \in [1,s]$ is calculated for I_q and each $I_{\mathbb{S}i}$ under mapping $\mathcal{V}_j$, which measures the visual similarity between I_q and $I_{\mathbb{S}i}$ under the particular feature extraction scheme $\mathcal{V}_j$.

By the end of the second stage, for each image $I_{\mathbb{S}i}$ in $I_{\mathbb{S}}$ $i \in [1,s]$, we should already have a score $s^{\mathbb{S}}_{(T_q,d_i)}$ measuring its semantical similarity to the query and its associated weight w_t, and k scores $s^{\mathcal{V}_j}_{(I_q,I_{\mathbb{S}i})}$ $j = 1,2,\ldots,k$ measuring its visual similarities to the query under mappings $\mathcal{V}_j$ $j = 1,2,\ldots,k$ and their associated weights $w_{\mathcal{V}_j}$ $j = 1,2,\ldots,k$. Then a combined score $s_{(I_q,I_{\mathbb{S}i})}$ which measures its similarity both semantically and visually is calculated by Equation 3.

$$s_{(I_q,I_{\mathbb{S}i})} = s^{\mathbb{S}}_{(T_q,d_i)} \times w_t + \sum_{j=1}^{k} (s^{\mathcal{V}_j}_{(I_q,I_{\mathbb{S}i})} \times w_{\mathcal{V}_j}) \tag{3}$$

The maximum is then computed over all $s_{(I_q,I_{\mathbb{S}i})}$ $i = 1,2,\ldots,s$ and is denoted by $\mathtt{max}_l s_{(I_q,I_{\mathbb{S}l})}$ $l \in [1,s]$. The final ranking $R_{(I_q,I_{\mathbb{S}i})}$ for example image I_q and image $I_{\mathbb{S}i} \in I_{\mathbb{S}}$ is

$$R_{(I_q,I_{\mathbb{S}i})} = \frac{s_{(I_q,I_{\mathbb{S}i})}}{\mathtt{max}_l s_{(I_q,I_{\mathbb{S}l})}}. \tag{4}$$

Algorithm 2: Visual comparison and ranking.

```
1 begin
2     Compute visual view $\mathcal{V}(I_q)$ of example image $I_q$,
      $\mathcal{V}(I_q) = \{\mathcal{V}_\mathbf{i}(\mathbf{I_q}), \mathcal{V}_\mathbf{2}(\mathbf{I_q}), \ldots, \mathcal{V}_\mathbf{k}(\mathbf{I_q})\}$.
3     Use $T_q$ to retrieve descriptions from inverted index
      $V$, and obtain set $I_{\mathbb{S}} = \{I_{\mathbb{S}1}, I_{\mathbb{S}2}, \ldots, I_{\mathbb{S}s}\}$.
4     for each mapping $\mathcal{V}_j$ in $\mathcal{V}$ $j = 1,2,\ldots,k$ do
5         Apply locality sensitive hashing to set
          $\{\mathcal{V}_\mathbf{j}(\mathbf{I}_{\mathbb{S}\mathbf{1}}), \mathcal{V}_\mathbf{j}(\mathbf{I}_{\mathbb{S}\mathbf{2}}), \ldots, \mathcal{V}_\mathbf{j}(\mathbf{I}_{\mathbb{S}\mathbf{s}})\}$.
6         Search through the hashed structures by $\mathcal{V}_\mathbf{j}(\mathbf{I_q})$,
          and obtain scores $s^{\mathcal{V}_j}_{(I_q,I_{\mathbb{S}i})}$ $i = 1,2,\ldots,s$.
7     Calculate $R_{(I_q,I_{\mathbb{S}i})}$ for $I_q$ and each $I_{\mathbb{S}i}$ $i = 1,2,\ldots,s$.
8 end
```

IV. Computational time analysis

The indexing program extracts index terms from the descriptions in D and adds them to the inverted index V. Meanwhile, visual features are distilled from the images in I and saved onto disk. If there are N characters in the text of the descriptions in D, in principle, an inverted index can be built in $O(N)$ time. In practice, if we take the amount of available memory installed in a computer into consideration, we may need to swap partial indexes onto disk and later merge them to produce the whole index. If the available memory is M, the cost of the algorithm is $O(N \log \frac{N}{M})$ [26] (pp. 196–198). For the visual feature extraction we have used the MPEG-7 edge histogram [27] and the auto colour correlogram [28]. The edge histogram descriptor is an 80-bin histogram which store numbers of each of the five types of edges, the vertical, the horizontal, the 45 degree, the 135 degree, and the non-directional, detected in each of the 4×4 sub-images of the original one. For an image of H pixel high and W pixel wide, the time for extracting the edge histogram is $O(2HW)$. An $O(HW)$ is needed to convert the image into grey level, and the other $O(HW)$ is needed for calculating the numbers in all the bins. The auto colour correlogram distills the spatial correlation of colours in an image. If the number of colours in an image is N_c after quantisation and the maximum distance considered is K. The auto colour correlogram is a $N_c \times K$ table, where each entry (c,k) $c \in [1,N_c]$ and $k \in [1,K]$ specifies the probability of finding a colour c at a distance k from a pixel of colour c. The dimension of the feature vector of this descriptor is N_cK. In our implementation, we have chosen N_c to be 64 and K to be 4, and the dimension of the feature vector is, therefore, 256. The computation time for an image of height H and width W is $O(HWN_cK)$.

The retrieving process consists of the semantic comparison and the visual comparison. In the semantic comparison, the query text T_q is searched through the inverted index V to obtain the set $D_{\mathbb{S}}$ and accordingly the set $I_{\mathbb{S}}$. The time spent on this text querying depends on the query and the text size in the collection. Single-word queries are easier and faster to solve with inverted indexes than context queries. But it has been demonstrated that the cost of solving queries, even for some complex ones, is sub-linear in the text size [26] (p196). The time complexity is $O(N^{\alpha})$, where N, like before, is the text size and α depends on the query and is very likely in the range $[0.4, 0.8]$.

However, the visual comparison demands more time than the semantic comparison, where locality sensitive hashing is applied k times, each time for the set $\mathcal{V}_j(I_\mathbb{S})$ of feature vectors $\{\mathcal{V}_\mathbf{j}(\mathbf{I}_{\mathbb{S}\mathbf{1}}), \mathcal{V}_\mathbf{j}(\mathbf{I}_{\mathbb{S}\mathbf{2}}), \ldots, \mathcal{V}_\mathbf{j}(\mathbf{I}_{\mathbb{S}\mathbf{s}})\}$ obtained under the mapping $\mathcal{V}_j,\ j \in [1, k]$, k being the number of the visual mappings applied in the system. The locality sensitive hashing we adopted in our implementation is based on the scheme described in [8]. The scheme is directly applicable to vectors in Euclidean space without embeddings, which is an improvement comparing to some earlier works, e.g., [23]. In our implementation, we have used the l_2 norm for measuring distances between vectors.

Each application of the locality sensitive hashing has to go through two phases: preprocessing and retrieving. The phase of preprocessing includes initialising, computing parameters, generating hash functions, building the hash tables and saving the feature vectors into right buckets. The phase of retrieving includes hashing the query vector, finding collided vectors from the hash tables and computing the distance of the query vector to each the collided vectors.

In the original description of the algorithm [8], $\mathcal{H} = \{h : S \to U\}$ is a family of locality sensitive functions, where domain S is a set of vectors, in our case being $\mathcal{V}_j(I_\mathbb{S}) = \{\mathcal{V}_\mathbf{j}(\mathbf{I}_{\mathbb{S}\mathbf{1}}), \mathcal{V}_\mathbf{j}(\mathbf{I}_{\mathbb{S}\mathbf{2}}), \ldots, \mathcal{V}_\mathbf{j}(\mathbf{I}_{\mathbb{S}\mathbf{s}})\},\ j \in [1, k]$, k being the number of the visual mappings, and domain U is a set of integers. Each $h \in \mathcal{H}$ is of the form $\lfloor \frac{\mathbf{a} \cdot \mathcal{V}_\mathbf{j}(\mathbf{I}_{\mathbb{S}\mathbf{i}}) + b}{w} \rfloor$, where $\mathcal{V}_\mathbf{j}(\mathbf{I}_{\mathbb{S}\mathbf{i}}) \in \mathcal{V}_j(I_\mathbb{S})$ is a feature vector, $\mathbf{a}$ is a vector of the same dimension with $\mathcal{V}_\mathbf{j}(\mathbf{I}_{\mathbb{S}\mathbf{i}})$ and whose entries are chosen independently from a p-stable distribution, and b is a random number chosen uniformly from the range $[0, w]$. Then a family $\mathcal{G}$ of hash functions is defined as $\mathcal{G} = \{g : S \to U^k\}$ such that $g(\mathcal{V}_\mathbf{j}(\mathbf{I}_{\mathbb{S}\mathbf{i}})) = (h_1(\mathcal{V}_\mathbf{j}(\mathbf{I}_{\mathbb{S}\mathbf{i}})), h_2(\mathcal{V}_\mathbf{j}(\mathbf{I}_{\mathbb{S}\mathbf{i}})), \ldots, h_k(\mathcal{V}_\mathbf{j}(\mathbf{I}_{\mathbb{S}\mathbf{i}})))$, where $h_i \in \mathcal{H}$, and, here, k is a hashing parameter different from the k that denotes the number of the visual mappings. For hashing the vectors in $\mathcal{V}_j(I_\mathbb{S})$ the algorithm chooses L hash functions $g_1, g_2, \ldots, g_L$ from $\mathcal{G}$ independently and uniformly at random, and stores each $\mathcal{V}_\mathbf{j}(\mathbf{I}_{\mathbb{S}\mathbf{i}}) \in \mathcal{V}_j(I_\mathbb{S})$ in buckets $g_i(\mathcal{V}_\mathbf{j}(\mathbf{I}_{\mathbb{S}\mathbf{i}}))$, for all $i = 1, 2, \ldots, k$, k being the hashing parameter. Given a feature vector $\mathcal{V}_\mathbf{j}(\mathbf{I}_{\mathbb{S}\mathbf{i}})$ of d dimension, it needs $O(dkL)$ time to compute all functions $g_1(\mathcal{V}_\mathbf{j}(\mathbf{I}_{\mathbb{S}\mathbf{i}})), g_2(\mathcal{V}_\mathbf{j}(\mathbf{I}_{\mathbb{S}\mathbf{i}})), \ldots, g_L(\mathcal{V}_\mathbf{j}(\mathbf{I}_{\mathbb{S}\mathbf{i}}))$, where k is the hashing parameter.

To reduce the time required to compute all the $g_i,\ i = 1, 2, \ldots, L$, functions, E^{2}LSH0.1 code modifies the original scheme, where each $h^i_j,\ j \in [1, k]$ in $g_i = (h^i_1, h^i_2, \ldots, h^i_k)$ is chosen independently from the others. Specifically, a family $\mathcal{U} = \{u : S \to U^{k/2}\}$ is defined to reuse some of the functions h^i_j. Suppose k is even and m is a constant. Function $u_i \in \mathcal{U},\ i \in [1, m]$ is defined as $u_i = (h^i_1, h^i_2, \ldots, h^i_{k/2})$, where each h^i_j is drawn uniformly at random from $\mathcal{H}$. Now define functions g_i as $g_i = (u_a, u_b)$, where $1 \le a < b \le m$, and only m functions u_i need to be computed. L is therefore reduced to $\frac{m(m-1)}{2}$ and the time for computing all the hash functions is reduced to $O(dkm)$, that is, $O(dk\sqrt{L})$.

At the beginning of the preprocessing, the algorithm computes the parameters k, m, L for the computations coming after. The E^{2}LSH0.1 code estimates optimal values for k, m, L to minimise the time needed by the retrieving. As L depends on m, which, in turn, depends on k, the task is left to find an optimal value for k. Since the optimal value of k depends on the query, the code experimentally estimates the retrieving time by constructing a sample data structure and runs several queries on that sample data structure, measuring the actual retrieving times. The value for k is then chosen such that the retrieving time is a minimal. The code runs on a number of iterations to calculate the retrieving times. Our experiments later showed that this process of estimation is very time consuming, and, therefore, we disabled it. Instead, we set k to a fixed number manually at the beginning of the preprocessing, and then compute m and L accordingly.

In the rest of the preprocessing, hash functions $g_1, g_2, \ldots, g_L$ are computed. Each vector $\mathcal{V}_\mathbf{j}(\mathbf{I}_{\mathbb{S}\mathbf{i}}),\ i \in [1, s]$, in $\mathcal{V}_j(I_\mathbb{S})$ is added to the buckets $g_1(\mathcal{V}_\mathbf{j}(\mathbf{I}_{\mathbb{S}\mathbf{i}})), g_2(\mathcal{V}_\mathbf{j}(\mathbf{I}_{\mathbb{S}\mathbf{i}})), \ldots, g_L(\mathcal{V}_\mathbf{j}(\mathbf{I}_{\mathbb{S}\mathbf{i}}))$. And L hash tables are then constructed for each $j = 1, 2, \ldots, L$, with the j^{th} hash table containing the vectors in $\mathcal{V}_j(I_\mathbb{S})$ hashed using g_j. It takes $O(\frac{k}{2}dm)$ time to compute all the hash functions, and $O(\frac{k}{2}dsm^2)$ time to construct the tables and to store the vectors into the buckets, where k and m are the hashing parameters as discussed above, d is the dimension of the vectors being hashed, and s is the number of the vectors in the data set.

In the retrieving, it needs $O(dkL)$ time for computing the L functions g_i for the query vector $\mathcal{V}_\mathbf{j}(\mathbf{I_q})$ as well as retrieving the buckets $g_1(\mathcal{V}_\mathbf{j}(\mathbf{I_q})), g_2(\mathcal{V}_\mathbf{j}(\mathbf{I_q})), \ldots, g_L(\mathcal{V}_\mathbf{j}(\mathbf{I_q}))$, and another $O(dC)$ to compute the distances of $\mathcal{V}_\mathbf{j}(\mathbf{I_q})$ to the vectors encountered in the buckets. The quantity C is the number of the collided vectors. The overall computation time, according to [8], is proportional to ds^ρ, where $\rho < 1$, d is the dimension of the vectors and s is the number of the vectors in the data set.

V. Implementation and Experiment

The indexing and retrieving algorithms were implemented in C/C++. Tests were made on an image collection consisting of 1514 pictures of various types of ancient Chinese architectures. In the collection, each image was paired with a text description file. An inverted index on the terms in the text files was created with the help of the CLucene package. The MPEG-7 edge histograms [27] and the auto colour correlograms [28] were extracted from the images and were stored into the same Lucene index built for the terms. The searching program took as input a text query, an example image, and the weights for evaluating scores obtained from the text searching, from comparison on the edge histograms and from that on the auto colour correlograms. Items returned by the searching program were ranked according to their degree of relevance to the query. Additionally, in the query, two radii R_e and R_c had to be specified for the hashing schemes applied on the edge histograms and the auto colour correlograms, respectively. The number of resultant items and the computational time both grew as the radii increased. E^{2}LSH0.1 (available at http://web.mit.edu/andoni/www/LSH) was the locality sensitive hashing package that was used in the retrieving program.

The performance of the implementation was tested on a laptop running Ubuntu Linux 10.10 (64bit). The laptop had 4GB main memory installed but only 1GB was set as the upper bound accessible to the searching program. The CPU was an Intel Core 2 Duo P8600@2.4GHz, although only one

TABLE I: Timing results from the experiments.

Test	Query	T_t	$\lvert I_\mathbb{S}\rvert$	T_e	T_c	T_v	T_{t+v}
1	$Q_1(t_1)$	0.09	21	0.30	0.34	0.64	0.73
2	$Q_2(t_2)$	0.10	27	0.41	0.51	0.91	1.01
3	$Q_3(t_3)$	0.10	40	0.53	0.59	1.12	1.21
4	$Q_4(t_2 \vee t_3)$	0.20	67	0.82	0.92	1.74	1.95
5	$Q_5(t_4)$	0.10	70	0.89	0.96	1.85	1.95
6	$Q_6(t_5)$	0.10	75	0.93	1.03	1.96	2.06
7	$Q_7(t_6)$	0.10	100	1.24	1.40	2.64	2.74
8	$Q_8(t_3 \vee t_5)$	0.20	115	1.47	1.76	3.23	3.43
9	$Q_9(t_5 \vee t_4)$	0.19	145	1.79	2.16	3.95	4.14
10	$Q_{10}(t_4 \vee t_6)$	0.19	170	2.20	2.28	4.49	4.67
11	$Q_{11}(t_5 \vee t_6)$	0.20	175	2.41	2.35	4.76	4.96
12	$Q_{12}(t_3 \vee t_4 \vee t_6)$	0.24	210	2.58	3.01	5.58	5.82
13	$Q_{13}(t_3 \vee t_5 \vee t_6)$	0.24	215	2.60	3.19	5.79	6.03

of the two cores was fully loaded while running the searching program. The code was compiled by Intel icpc 12.0 for Linux with -O3 and -ipo optimisation options. The two radii were set to 1.0. The parameter k was set to 2, m, being dependent on k, to 4, and L, being $\frac{m(m-1)}{2}$, to 6. The testing results are summarised in Table I, which contains the data 13 test cases.

In each of the tests, the query Q_j, $j \in [1, 13]$, contained a text phrase and an example image plus the relevant parameters. In all the tests, the example image, the radii and the weights remained the same, but the text phrases varied in order to give different image sets for the visual similarity comparison. In test cases 1 to 3 and 5 to 7 the text query were all simple phrases, denoted by t_1 to t_6. For the rest, the text queries were connections of simple phrases by logical OR ($\vee$) operator. For example, in test case 10, the text phrase of query $Q_{10}(t_4 \vee t_6)$ was a disjunction expression consisting of terms t_4 and t_6.

In each of the tests we recorded $|I_\mathbb{S}|$, T_t, T_e, T_c, T_v and T_{t+v}, whose meanings are listed below.

- $|I_\mathbb{S}|$, number of images obtained from the text searching.
- T_t, time spent on the searching of the text phrase through the inverted index.
- T_e, time spent on the retrieving through the edge histograms.
- T_c, time spent on the retrieving through the auto colour correlograms.
- T_v, time spent on the visual similarity comparison.
- T_{t+v}, time spent on the text searching and the visual comparison in total.

All the times were measured in millisecond (ms). Roughly speaking, T_v was the sum of T_e and T_c plus extra time spent on constructing data sets for the hashings. T_{t+v} included T_t and T_v and the times spent on calculating the final rankings. T_e and T_c included times spent on i) the initialisation, ii) the computation of hash functions, iii) the construction of hashed buckets, iv) the addition of vectors to the buckets, v) the hashing of the query vector, vi) the retrieving of stored vectors from collided buckets, and vii) the computation of distances of the query vector to each collided vector.

The experiments we did supported the observations we made above. For example, in Table I test 13, the semantic comparison selected 215 images out of the whole 1514 images, and, in the subsequent visual comparison, it took 2.60 milliseconds to finish the similarity comparison on the edge histograms (80-dimensional) and 3.19 milliseconds on the auto colour correlograms (256-dimensional).

VI. Conclusions and discussions

We have presented a hashing based solution to scalable and fast image retrieval. The solution unifies well-established text and content based techniques with the aim of overcoming the semantic gap and long retrieval times in image retrieval systems that are solely content based. Content based approaches evaluate similarities in visual domain, which provides more objective representations for images than text annotations. On the other hand, text annotations often map more directly onto the semantic meanings of images than low-level visual features.

Similarity comparison over a set of high-dimensional vectors is a slow operation and has become an obstacle for scaling up the content based approach to image retrieval. In our solution, we have adopted a scheme of locality sensitive hashing which guarantees a sub-linear searching time on the visual comparison. The image selection by the semantic comparison also helps to reduce the size of the data set for the hashing, which further shortens the time spent on the visual comparison. By such a two-staged retrieving strategy, time spent on the content based comparison can be confined within a user-tolerant range. From the experimental results, the longest query time was about 6 ms (see Table I) for the collection of 1514 images. So we believe that the solution has the potential to be scaled up to suit large image collections. Furthermore, the developed idea can be applied to other identification problems, such as the macromodeling issue in the signal integrity study [29].

However, we are fully aware that some aspects of the work need further improvements. With the wide availability of multi-core processors in the market, there are plenty of opportunities for us to explore the applications of parallel processing in the indexing and the searching stages.

References

[1] R. Perez-Aguila, "Automatic segmentation and classification of computed tomography brain images: An approach using one-dimensional kohonen networks," *IAENG International Journal of Computer Science*, vol. 37, Feb. 2010.

[2] J. R. Smith and S.-F. Chang, "VisualSEEk: A Fully Automated Content-Based Image Query System," in *ACM Multimedia*, Nov. 1996, pp. 87–98.

[3] J. Ashley, M. Flickner, J. Hafner, D. Lee, W. Niblack, and D. Petkovic, "The Query by Image Content (QBIC) System," *ACM SIGMOD Record*, vol. 24, no. 2, p. 475, May 1995.

[4] J. Laaksonen, M. Koskela, and E. Oja, "PicSOM - Self-Organizing Image Retrieval With MPEG-7 Content Descriptors," *IEEE Transactions on Neural Networks*, vol. 13, no. 4, pp. 841–853, Jul 2002.

[5] 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, Dec. 2000.

[6] R. Weber, H.-J. Schek, and S. Blott, "A Quantitative Analysis and Performance Study for Similarity-Search Methods in High-Dimensional Spaces," in *Proceedings of the 24th International Conference on Very Large Data Bases*. Morgan Kaufmann Publishers Inc., 1998, pp. 194–205.

[7] N. Beckmann, H.-P. Kriegel, R. Schneider, and B. Seeger, "The R*-tree: An Efficient and Robust Access Method for Points and Rectangles," *ACM SIGMOD Record*, vol. 19, no. 2, pp. 322–331, Jun. 1990.

[8] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni, "Locality-Sensitive Hashing Scheme Based on p-Stable Distributions," in *Proceedings of the 20th Annual Symposium on Computational Geometry*. ACM, 2004, pp. 253–262.

[9] X. S. Zhou and T. S. Huang, "Unifying Keywords and Visual Contents in Image Retrieval," *IEEE MultiMedia*, vol. 9, no. 2, pp. 23–33, Apr. 2002.

[10] N. Zhang and Y. Song, "An Image Indexing and Searching System Based Both on Keyword and Content," in *Proceedings of the 4th International Conference on Intelligent Computing (ICIC2008)*, ser. LNCS 5226. Springer-Verlag Berlin Heidelberg, Sep. 2008, pp. 1059–1066.

[11] A. Popescu, C. Millet, and P.-A. Mollic, "Ontology Driven Content Based Image Retrieval," in *Proceedings of the 6th ACM International Conference on Image and Video Retrieval*. ACM, Jul. 2007, pp. 387–394.

[12] G. A. Miller, R. Beckwith, C. Fellbaum, D. Gross, and K. J. Miller, "Introduction to WordNet: An On-line Lexical Database," *Intertional Journal of Lexicography*, vol. 3, no. 4, pp. 235–244, Jan. 1990.

[13] V. Mezaris, I. Kompatsiaris, and M. G. Strintzis, "Region-based Image Retrieval using an Object Ontology and Relevance Feedback," *EURASIP Journal on Applied Signal Processing*, vol. 2004, no. 6, pp. 886–901, Jun. 2004.

[14] L. Fan and B. Li, "A Hybrid Model of Image Retrieval Based on Ontology Technology and Probabilistic Ranking," in *Proceedings of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence*. IEEE Computer Society, 2006, pp. 477–480.

[15] T. Quack, U. Mnich, L. Thiele, and B. S. Manjunath, "Cortina: A System for Large-scale, Content-based Web Image Retrieval," in *Proceedings of the 12th Annual ACM International Conference on Multimedia*. ACM, 2004, pp. 508–511.

[16] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification (2nd Edition)*. John Wiley and Sons, Inc., 2000.

[17] P. Wilkins, P. Ferguson, A. F. Smeaton, and C. Gurrin, "Text Based Approaches for Content-Based Image Retrieval on Large Image Collections," in *Proceedings Of the 2nd European Workshop on the Integration of Knowledge, Semantics and Digital Media Technology*, Nov. 2005, pp. 281–288.

[18] H. Joho and M. Sanderson, "The SPIRIT Collection: An Overview of A Large Web Collection," *ACM SIGIR Forum*, vol. 38, no. 2, pp. 57–61, Dec. 2004.

[19] M. Möller and M. Sintek, "A Generic Framework for Semantic Medical Image Retrieval," in *Proceedings of the Knowledge Acquisition from Multimedia Content (KAMC) Workshop, the 2nd International Conference on Semantics and Digital Media Technologies (SAMT)*, Dec. 2007.

[20] Y. Ke, R. Sukthankar, and L. Huston, "An Efficient Parts-based Near-duplicate and Sub-image Retrieval System," in *Proceedings of the 12th Annual ACM International Conference on Multimedia*. ACM, 2004, pp. 869–876.

[21] Y. Ke and R. Sukthankar, "PCA-SIFT: A More Distinctive Representation for Local Image Descriptors," in *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Jun. 2004, pp. 506–513.

[22] P. Indyk and R. Motwani, "Approximate Nearest Neighbors: Towards Removing the Curse of Dimensionality," in *Symposium on Theory of Computing*, 1998.

[23] A. Gionis, P. Indyk, and R. Motwani, "Similarity Search in High Dimensions via Hashing," in *Proceedings of the 25th International Conference on Very Large Data Bases*. Morgan Kaufmann Publishers Inc., 1999, pp. 518–529.

[24] A. Andoni and P. Indyk, "Near-Optimal Hashing Algorithms for Approximate Nearest Neighbor in High Dimensions," in *Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science*. IEEE Computer Society, 2006, pp. 459–468.

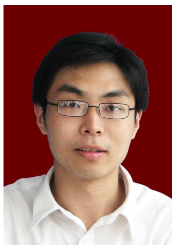
[25] ——, "Near-optimal Hashing Algorithms for Approximate Nearest Neighbor in High Dimensions," *Communications of the ACM*, vol. 51, no. 1, pp. 117–122, 2008.

[26] R. Baeza-Yates and B. Ribeiro-Neto, *Modern Information Retrieval*. Addison Wesley, 1999.

[27] ISO/IEC/JTC1/SC29/WG11, "CD 15938-3 MPEG-7 Multimedia Content Description Interface - Part 3," in *MPEG Document W3703*, 2000.

[28] J. Huang, S. R. Kumar, M. Mitra, W.-J. Zhu, and R. Zabih, "Image Indexing using Color Correlograms," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Jun. 1997, pp. 762–768.

[29] C.-U. Lei, K. Man, and Y. Wu, "VLSI Macromodeling and Signal Integrity Analysis via Digital Signal Processing Techniques," in *Lecture Notes in Engineering and Computer Science: Proceedings of The International MultiConference of Engineers and Computer Scientists 2011, IMECS 2011, 16-18 March, 2011, Hong Kong*, 2011, pp. 1031–1032.



Nan Zhang is a lecturer in the Department of Computer Science and Software Engineering at Xi'an Jiaotong-Liverpool University. He has a Ph.D. and an M.Sc. from the School of Computer Science, the University of Birmingham UK, and a B.Eng. degree in computer science from the University of Shandong China. His research interests focus on high-performance parallel computing and its applications on financial derivative pricing.

Ka Lok Man is currently an academic staff at Xi'an Jiaotong-Liverpool University (China) and a visiting professor at Myongji University (Korea).

Tianlin Yu is a year-three undergraduate student in the Department of Computer Science and Software Engineering at Xi'an Jiaotong-Liverpool University. He is currently working towards a BS.c. degree in computer science.

Chi-Un Lei received B.Eng. (first class honors) and Ph.D. in Electrical and Electronics Engineering from the University of Hong Kong in 2006 and 2011, respectively. He is now a Teaching Assistant for the Common Core Curriculum at the University of Hong Kong. His research interests include VLSI macromodeling, VLSI computer-aided signal integrity analysis and system identification techniques.

He is currently a Co-General Chair in a circuit design (DATICS) workshop series, a reviewer of a few IEEE journals, and a Co-Editor-in-Chief of the "International Journal of Design, Analysis and Tools for Integrated Circuits and Systems" (IJDATICS).

He was awarded with the Best Student Paper Award in IAENG IMECS 2007 and 2010. In 2010, he also received the Best Poster Award in IEEE ASP-DAC Student Forum 2010 and the IEEE ASP-DAC 2010 Student Travel Grant.