KURSOR Journal
Research on Computing and its Aplications

Vol. 7, No. 1, January 2013 **ISSN 0216 – 0544**

NEAR-DUPLICATE REAL-LIFE FACE IMAGE

[a]Intan Yuniar Purbasari, [b]Budi Nugroho
[a,b]Teknik Informatika, Fakultas Teknologi Industri, UPN "Veteran" Jawa Timur, Indonesia
Jl. Raya Rungkut Madya, Gunung Anyar, Surabaya 60294
E-Mail: intan.yuniar@gmail.com

Abstrak

Content-based Image Retrieval (CBIR) merupakan metode temu kembali citra berdasarkan karakteristik numerik pada citra. Pencarian similaritas yang efisien pada ruang dimensi ultra-high telah diajukan menggunakan *two-tier inverted file* dan *Local Derivative Patterns* (LDP) sebagai metode ekstraksi fitur dengan tingkat keakuratan dan kinerja yang tinggi pada data set citra wajah eksperimental. Namun demikian, citra *real-life* memiliki ukuran dan resolusi yang berbeda serta *noise* bawaan. Masih belum diketahui apakah LDP dapat menunjukkan hasil yang sama memuaskan jika diberikan data set citra real-life. Penelitian ini merancang dan membangun search engine citra wajah untuk mencari citra nyaris duplikat pada citra real-life menggunakan metode LDP untuk ekstraksi fitur dan *two-tier inverted file* untuk pengindeks-an multidimensi. Sebuah metode ekpansi state juga diperkenalkan untuk lebih menangkap banyak detil dari histogram citra dengan mempertimbangkan informasi piksel tetangga. Eksperimen ini dilakukan pada 8083 citra wajah real-life dari berbagai ukuran antara 20x20 dan 80x80. Data set berisi kopi duplikat dari citra wajah setelah melalui beberapa proses transformasi. Hasil pencarian mengembalikan 20 citra yang memiliki kemiripan paling tinggi dengan citra query dan memiliki nilai presisi 0.75 atau 75%.

Kata kunci: *Content-Based Image Retrieval, Local Derivative Pattern, Two-tier Inverted File, Real-life Face Image.*

Abstract

Content-based image retrieval (CBIR) is an image retrieval method based on the analysis of numerical characteristics of the image at the absence of text information. An efficient similarity search in ultra-high dimensional space has been proposed using two-tier inverted file and Local Derivative Patterns (LDP) as feature extraction method with high accuracy and high performance on experimental face image data sets. However, real-life images have different size, resolution and a potential noise. It is unknown whether LDP would show the same satisfactory result given real-life image data sets. This research designed and developed a face search engine to find near-duplicate face in real life images using LDP method to extract image features and two-tier inverted file for multidimensional indexing process. A state expansion method was also introduced to capture more detailed description of image histogram by considering neighbor information. The experiment was performed on 8,083 real-life face images of various sizes between 20x20 to 80x80. The data set contained duplicate copies of face images with some transformation processes. The search result returned top 20 images which had the most similarity with the query images and had an average precision rate of 0.75 or 75%.

Keywords: Content-Based Image Retrieval, Local Derivative Pattern, Two-tier Inverted File, Real-life Face Image.

INTRODUCTION

Content-based image retrieval (CBIR) has been gaining a remarkable attention, notably by the number of research papers on the topic, over the last decade [1]. It is an image retrieval method based on the analysis of numerical characteristics of the image at the absence of text information. Face recognition is one area among others enjoying the limelight as there are numerous new algorithms and techniques to build more sophisticated face recognition applications. One important CBIR application in this area is finding duplicate or near-duplicate face for purposes such as identity verification, video surveillance, automated border control, and crime scene footage analysis. In order to achieve a satisfactory recognition result, high dimensional space in images is more preferred.

An efficient similarity search in ultra-high dimensional space has been proposed in [2] using two-tier inverted file as its indexing method and Local Derivative Patterns (LDP) as feature extraction proposed in [3]. The experimental results on 15,488 dimensional histogram features showed high accuracy and high performance on experimental face image data sets with equal size.

However, real-life images have different size, resolution, and a high possibility of various kinds of noises. It is unknown whether LDP will show the same satisfactory result as it has in extracting features if given real-life image data sets. Furthermore, the similarity search method using two-tier inverted file can possibly be extended from finding an exact duplicate image to a near-similarity search, which is finding a near-duplicate image of a query from a dataset in a database.

This research aimed to design and development of a face search engine to find near-duplicate face in real-life images. It used Local Derivative Patterns method to extract features from real-life images and two-tier inverted file as its multidimensional indexing system.

Content-Based Image Retrieval (CBIR) is an image retrieval method based on the analysis of numerical characteristics of the image, where no additional information (i.e. text-based information) on the image is available [4]. Because of this, CBIR relies on analyzing the characteristics of the image's pixels. Feature extraction, multidimensional indexing, and retrieval system design are the three main concern areas in CBIR studies [5].

The term "near-duplicate" itself may have different interpretations. According to [6], in terms of images/videos, a duplicate is not merely an identical copy but more of a transformed copy of the original source files using digital photometric or geometric transformations. Zhang et al in [7] defined Image Near-Duplicate (IND) as *"a pair of images in which one is close to the exact duplicate of the other, but differs slightly due to variations of capturing conditions (camera, camera parameter, view angle, etc), acquisition time, rendering conditions, or editing operations"*. Those definitions are rather general for any kinds of images.

Several investigations have been done to detect near-duplicate images using various combinations of different feature extraction methods and multidimensional indexing methods. Zhang et al use Stochastic Attribute Relational Graphing technique, where Attribute Relational Graphing (ARG) is a graph consisting vertices of regions or interest points in an image. The similarity of two images is done by using a stochastic process on their respective ARGs. Despite the good result it achieves, it suffers from high computational cost.

The use of local descriptor PCA-SIFT to extract features and Locality-Sensitive Hashing to create an index were introduced in [8]. To measure similarity between images, it uses L2 distance and Random Sample Consensus (RANSAC) [9] to further eliminate false-matched features. However, its use of 20 hash tables loaded into memory increases storage space and computation time.

Another approach proposed in [6] by adopting a local-feature-based framework and LSH technique as its indexing structure. It represents an image with a set of local patch and defines a descriptor – Local Difference Pattern – for each local patch. It achieves very good recall and precision rate while still keeping the storage space and computation time low.

CONTENT BASED IMAGE RETRIEVAL

Feature Extraction

Feature extraction underlies all works in a CBIR. Specifically in face recognition, there are numerous methods available and they fall into two main categories: holistic and local methods [10].

Holistic method uses the whole face as input to the system. Eigenface [11] and Fisherface [12] are two examples of holistic method successfully implemented based on Principal Component Analysis (PCA) and Linear Discriminant Analysis (LDA), respectively. However, this approach suffers from variations on illumination and pose.

Local method use local features such as eyes, lips, and nose's position and statistics as inputs. This method is proven to be less sensitive to illumination and various poses. One example of this method is Local Feature Analysis (LFA) [13]. LFA uses set of local-topological fields to extract local features. Another method is Elastic Bunch Graph Matching (EBGM) which creates a topological graph whose nodes consist of Gabor coefficient [14].

Local Binary Pattern (LBP) is another proposed method to describe texture feature. Ahonen et al in [15] has presented an efficient facial image representation based on LBP. LBP general's idea is to divide face image into several regions and extract the "micro-patterns" of the face to be used as face descriptor. In LBP, a pixel is compared to each of its 8 neighbors (called its region) creating a series of binary number generated from the result of the comparison.

In [3], an extended version of LBP, called LDP (Local Derivative Pattern), has been proposed to extract higher order of information, resulting in more detailed information being captured into the feature vector, thus suggests better recognition performance. The first order of LDP is indeed LBP. The second order of LDP compares two derivative directions for two neighboring pixels and so forth. The experiment carried out in several face databases showed that it achieves better recognition rate compared to LBP.

Multi-Dimensional Indexing

After image features have been extracted, the next task is to store the vectors into a database using appropriate indexing method so that the retrieval part in recognition process can be performed efficiently. Efficiency is a big issue here because recognition process needs to be performed in a timely manner given the high dimensional space the image may possess.

Many efforts are based on tree structures which are known can prune the number of feature vectors significantly. R-tree, kd-tree, and M-tree are among others offering faster ways to retrieve high-dimensional data spaces. However, when dimensionality reaches around 10, they are outperformed by a simple sequential search due to overlapping in different branches [16].

Inspired by the inverted file method proposed in [17], which has been an effective solution for indexing large-scale text databases with extremely high dimensionality, [2] developed a two-tier inverted file with state expansion method to index ultra-high dimensional histograms. It indexes data space in two levels: the first is the list of occurring states for each dimension, and the second is the list of occurring images for each state. Before computing the actual distance between a query and candidate images in database, a weighted state-voting scheme is performed. Each candidate is ranked by a score computed based on the sum of value of matched state between a query and a candidate then top-k candidates are returned for the actual histogram computation to find the nearest neighbor to the query. This approach is proven to outperform sequential scan, VA-file, and iDistance performance using experimental data sets.

Local Derivative Pattern (LDP)

Local Derivative Pattern (LDP) is an extended version of LBP and was proposed by Zhang et al in [3]. It claimed to be more superior to LBP in capturing more detailed information of an image. While LBP only captures the first order information, LDP takes a few steps forward by capturing higher order derivative information which contains more discriminative features that LBP cannot capture. It performs satisfactorily in face identification and face verification under various conditions.

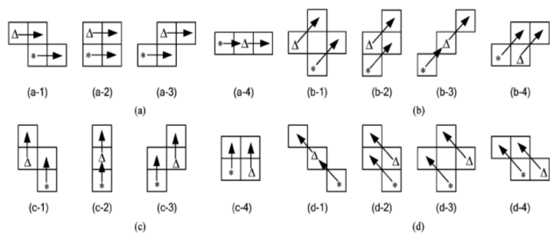


Figure 1. Illustration of LDP templates. (a) Four templates to calculate *f* function of two reference pixels (Δ and *) at 0° (b) Four templates to calculate *f* function of two reference pixels (Δ and *) at 45° (c) Four templates to calculate *f* function of two reference pixels (Δ and *) at 90° (d) Four templates to calculate *f* function of two reference pixels (Δ and *) at 135° (image source: [3]).

Given the same example from Figure 1, *I*(*Z*) denotes an image and $I'_{\alpha}(Z)$ denotes the first order derivatives along α = 0°, 45°, 90°, and 135°, respectively. The first four first order derivatives at $Z = Z_0$ are shown in Equation (1), Equation (2), Equation (3), and Equation (4).

$$I'_{0^0}(Z0) = I(Z_0) - I(Z_4) \quad (1)$$

$$I'_{45^0}(Z0) = I(Z_0) - I(Z_3) \quad (2)$$

$$I'_{90^0}(Z0) = I(Z_0) - I(Z_2) \quad (3)$$

$$I'_{135^0}(Z0) = I(Z_0) - I(Z_1) \quad (4)$$

Figure 1 illustrates the 16 templates of various distinctive spatial relationships in a local region. LDP operator compares two derivative directions at two neighboring pixels and concatenates the results as a 32-bit binary sequence. Each pixel thus is labeled by its 32-bit binary sequence.

LDP is claimed to win over LBP in two aspects: (1) It provides a more detail description for faces by coding the (n-1)th order derivative direction variations, (2) it encodes the various distinctive spatial relationships in a local region, rather than LBP's merely encoding of the relationship between central point and its surrounding neighbors. Thus, LDP can capture more spatial information. Figure 2 shows the comparison between LBP and LDP's visualization of a face image.

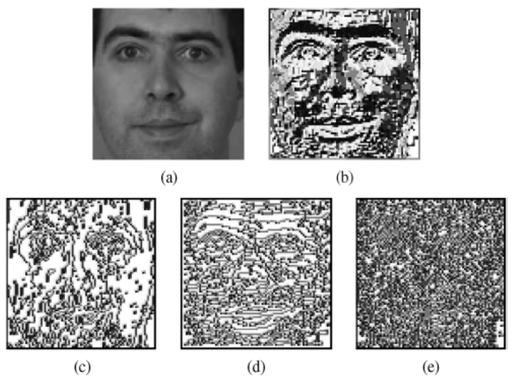


Figure 2. Visualization of LBP and LDP in 0° direction (a) Original face image (b) LBP (c) second order LDP (d) third order LDP (e) fourth order LDP (image source: [3]).

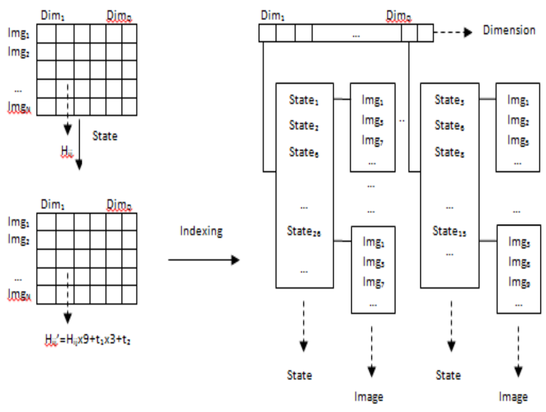
Two-tier Inverted File

The concept was inspired by the widely used inverted file model in text databases. The original model is known for its high efficiency [17] in time and space for very high text dimensionality and very sparse word-document matrix. In [2], the inverted file concept is applied to low-level visual feature databases by exploiting the discrete and finite nature of histograms and constructing a two-tier inverted file structure to search for similarity in ultra-high dimensional space efficiently.

The original text-based inverted file constructs a structure where each word points to a list of documents containing the word. By regarding each dimension as a word and list of images as list of documents and making each dimension to point to a list of images whose values (or states) on the dimension is not zero, the inverted file concept can be applied to image feature files. To avoid a long images list for each dimension and by considering that all values in histograms are distributed in a predetermined state range (from 0 to the maximum number of pixels allowed in a bin), a second level of inverted file is created. For each dimension, a list of non-zero states is generated, with each state pointing to a list of images having the same state for that dimension. An additional state expansion method is introduced to improve the discriminative power of inverted file, based on the fact that the number of possible states is much smaller compared to the number of images and it is needed to balance the state list size and the image list size for better performance.

State Expansion

To preserve the original state information, the expansion method tries to consider the local neighbor information. By looking into the relationship between the states of ith dimension with its neighbor dimensions, i.e. its left and right neighbors, there are three possible relationships, which are "less than (<)", "larger than (>)", and "equal (=)". With two neighbor dimensions, a single i^{th} dimension's state can be expanded into 3 x 3 possible states.



Figrure 3. Construction of two-tier inverted file indexing structure

Index Construction

For *N* images represented by *N* histograms, with *D* dimensionality, the general process of constructing two-tier inverted file is illustrated in figure 3.

Given an image histogram, $H = (h_1, h_2, \ldots, h_D)$, it is first transformed using state expansion to $H' = (h_1', h_2', \ldots, h_D')$. In H', each dimension of an image is associated with a new state value, generated by considering the relationships with its left and right neighbor dimensions.

This research is mainly focused on developing a database application to organize high dimensional real-life images with a feature of finding near-duplicate ones. It is an extension of the work in [2] where the dataset used is an experimental dataset.

EXPERIMENTAL METHODS

Pre-processing

A collection of 10,000 real-life images of various sizes range from 20 x 20 to 88 x 88 has been compiled into the dataset. They are color images in JPG format. Since LDP feature extractor application only receives images in size of 88 x 88 and bitmap image (BMP) file format, it is necessary to pre-process the images into the size and format that can be accepted by LDP generator application.

There are two steps in pre-processing part: resize images and restoring images. Firstly, to resize images, larger images were simply cropped images or stretched smaller images to 88 x 88 pixels, the size used in LDP feature extraction.

Secondly, to restore images a simple interpolation method, Lanczos algorithm, was applied. Restore process was needed to fill empty pixels resulted from the resize process.

Image Transformation

A pair of images can be categorized as near-duplicate images when they differ slightly due to editing operations, among other causes. To create duplicate images from our dataset, around 1,000 distinct images were selected to undergo seven transformation processes: blur, contrast up, contrast down, brightness up, brightness down, rotation, and text addition. Together with the original images (as the true duplicate ones), there were a total of 8,083 images in the smaller data set.

LDP Feature Extraction

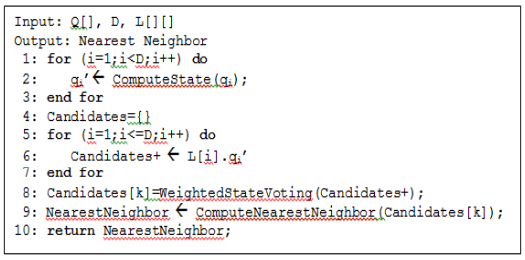
After images have been resized and restored, the next process was to extract texture feature using LDP. This part of work was already completed in another research [3]. The only required work to do here was to supply the images into the application to generate LDP histogram feature of each image.

Indexing

This step followed the methodology steps of the already completed work in [2].

Two-tier inverted file with state expansion method was used to index all images in dataset. The index was created by first expanding states on each block. The first tier stored the list of occurring states for each dimension of an image, while the second tier stored the list of occurring images for each state.

Following the result of previous research [2] which selected the value of ε = 5% as the default setting, in this research, the value of ε was set to 5% of the total images in dataset. It means that only states with image lists' size less than 5%*8,083=405 were kept in the two-tier inverted file.



```
Input: Q[], D, L[][]
Output: Nearest Neighbor
 1: for (i=1;i<D;i++) do
 2:     qi'← ComputeState(qi);
 3: end for
 4: Candidates={}
 5: for (i=1;i<=D;i++) do
 6:     Candidates+ ← L[i].qi'
 7: end for
 8: Candidates[k]=WeightedStateVoting(Candidates+);
 9: NearestNeighbor ← ComputeNearestNeighbor(Candidates[k]);
10: return NearestNeighbor;
```

Figure 4. Query Processing Algorithm.

Query Processing

This step also followed that of in [2]. The algorithm for query processing is listed in figure 4.

The algorithm started by performing state expansion method to the query file (1-3) in all dimensions D. Line 5-7 searched into two-tier inverted file for list of states matching the states of the query file in each dimension and retrieved list of candidate images which shared at least one common state with the query image. Matched states with higher values were ranked higher and contributed more to the final similarity. In that case, state values needed to be appraised in ranking the candidates. In addition, if the state value of a candidate image matched with Q in more than one dimension, then it is possible to have more than one copy of that candidate image in the list of candidate images. Here the `WeightedStateVoting` function was applied to rank all candidates (line 8). To compute the function, the expanded states of each matching image must be transformed back to its original states, according to equation 11. The ranking score for each candidate is computed by summing the matched state value between a candidate and Q ($\sum \bar{q}_i$, where $\bar{q}_i$ is the value of matched between Q and a candidate). Top-k candidates were returned as the final candidates whose then actual histogram intersection would be computed to obtain the nearest neighbor to Q (line 9). To compute the multiplication and cumulative summation, an external piece of code was used [18]. It made use of Run-Length Encoding method to compress data. In this research, the value of k was set to 20, which was regarded as "the reasonable default value for both precision and efficiency consideration" in [2].

Measuring Similarity

After the final candidates were obtained, the L_1 (1-norm) distance measure as shown in Equation (5) was used to measure similarity between the final candidate images and the query image.

$$L_1(H,S) = \sum_{i=1}^{D} |H_i - S_i| \qquad (5)$$

where *H* and *S* are two LDP histograms having the same dimension, *D*. If *H* and *S* are the exact same images, the distance is 0, thus they have the highest similarity. The top-*k* candidate images were then ranked based on their similarity with the query image.

RESULT AND DISCUSSION

To start searching for near-duplicate images, user selects an image from a predefined dataset and then clicks Search button. A list of true duplicate images of the selected image can also be displayed to check whether all near-duplicate images were retrieved successfully or not.

The average precision percentage was calculated based on the Equation (6).

$$Average\_Precision = \frac{\sum_{k=1}^{n} (P(k) rel(k))}{number\_of\_relevant\_images} \qquad (6)$$

Where k is the rank of in the sequence of the retrieved images, n is the number of retrieved images, *P(k)* is the precision at the cut-off *k* in the list, and *rel(k)* is a function equals to 1 if item at rank *k* is a relevant image, or equals to 0 otherwise.

The top-20 near-duplicate images of a query image were listed in the result panel starting from the top left corner in a column-wise order. From one example in figure 5, it could be noticed that the first six retrieved images were all duplicate copies of the original query image. This means that for the first six images, the precision percentage is 100%. Using the assumption that all its duplicate images were those retrieved in figure 6, the recall percentage for that query was 6 out of 8 and equals to 0.75 or 75%.

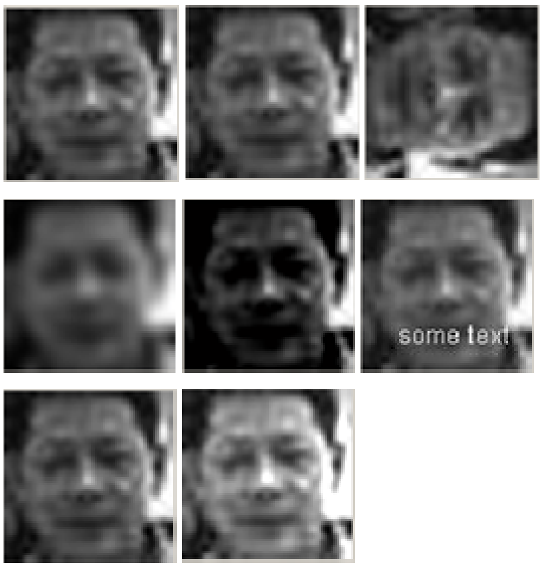


Figure 5. List of Duplicate Images of Image ID 0041 Based on Their Same Partial name.

Most of the 20 queries gave 75% recall, while a few gave higher and lower values. In a closer look, the ones which gave lower than 75% recall were transformed duplicate images (not original ones). This is very reasonable considering transformed images were mostly of lower quality than the original images. Thus, the LDP feature extractor application was unable to capture more detail features of those images.

Among all eight copies of an image (a full set), the ones which did not get retrieved mostly were the blurred and the rotated ones. Despite the fact that there were very few query images have successfully retrieved their blurred copies (for example, image ID 0068) or the rotated ones (image ID 4450), most other queries were unable to do that. However, this experiment was still at its earliest stage with limited variations in images transformation steps to say that LDP feature extraction is unable to extract features from blurred and rotated images despite the claim that LDP is a robust face descriptor that is insensitive to image rotation, translation, and scaling. A further investigation is required to unveil the exact reason behind this phenomenon.

As a comparison with LDP, the face search engine application also included LBP face descriptor as an option to perform the search.

Table 1 lists the comparison of average recall percentage and average elapsed time ran over 20 random queries for LDP and LBP face descriptors.

From table 1, it is clear that LBP performed faster than LDP in all 20 sample queries, which was already expected. The average precision, however, gave a rather surprising result since [3] claimed that LDP outperforms LBP in recognition rate using standard face image datasets. Table 1 shows that LBP had higher average precision values in almost all sample queries and thus had higher mean average precision compared to that of LDP. This result would also need a further investigation using more samples of real-life images to give a solid support to be able to say that LDP does not perform better than LBP for real-life images.

Despite the 75% average performance of this search engine application, it has demonstrated satisfactory results in retrieving near-duplicate real-life images. Six out of eight duplicate images were retrieved in the top six results. In few other queries, blurred and rotated ones were also retrieved in the top ten results and so did several other near-duplicate images (judged by human eyes, for example with query image ID 0041 in figure 7, two other near-duplicate images were retrieved in rank 11 and 12). These other near-duplicate images do not share the same partial name as the query one, so they were not counted in the average precision calculation. Based on this reason, the real percentage of average precision can be higher than what is displayed.

Another more extensive experiment needs to be performed to investigate the cause of the problem where LDP has not been very successful in retrieving blurred and rotated images. Varying parameter settings on each transformation method would be very useful to determine which setting values give high recognition rate. This work might be considered to be challenging since real-life images come in different settings in contrast and brightness levels that defining a single value for contrast and brightness are not sufficient enough.

Table 1. Comparison of LDP and LBP in Query Result over 20 Trials.

No	Image ID	LDP		LBP	
		Average Precision	**Elapsed Time**	**Average Precision**	**Elapsed Time**
1	0041	0.75	2.8	0.8375	1.7845
2	0001	0.75	3.3139	0.875	2.0204
3	4986	0.75	3.1596	0.82292	2.0595
4	0067	0.8125	3.0388	0.875	2.1353
5	4450	0.80833	3.2504	0.75	2.2618
6	4458	0.875	3.1163	0.875	2.0394
7	7434	0.75	3.2138	0.75	2.3028
8	6594	0.75	3.0331	0.875	2.0322
9	2913	0.84722	3.0783	0.875	1.9933
10	0137	0.75	3.018	0.875	2.1169
11	0265	0.75	3.1408	0.875	1.981
12	6986	0.82955	3.4861	0.75	2.0249
13	6634	0.75	3.1965	0.75	2.0929
14	7714	0.75	3.115	0.75	2.0027
15	6018	0.75	3.1884	0.875	1.9799
16	8035	0.66667	3.1232	0.8333	2.0096
17	8071	0.66667	3.0831	0.8333	2.0035
18	0114	0.51176	3.1553	0.40451	2.02525
19	7236	0.75	3.213	0.75	1.9095
20	5346	0.75	3.0916	0.875	2.0251
Average		0.750885	3.14076	0.8053265	2.0400225

CONCLUSION

A face search engine application to find near-duplicate images has been developed using Local Derivative Pattern (LDP) as its feature extractor and two-tier inverted file as its indexing system. The search engine was able to retrieve an average of 75% near-duplicate images, based on controlled dataset within a reasonable response time. However, higher percentage is very likely to be achieved since there is no exact method to determine how near-duplicate some images are, besides using human's judgment.

The main contributions of this research are:

1) A database system to organize a large scale of real-life face images has been developed.
2) Image transformation procedure has been performed for experiment purpose in finding near-duplicate images.
3) A search engine to find near-duplicate real-life face images has been developed using LDP and LBP feature extraction methods and two-tier inverted file indexing structure.
4) A performance study on different feature extraction methods has been conducted with a result that LDP does not perform better than LBP as it has been claimed using the dataset in this experiment.

FUTURE WORK

Some possible future work as the continuation of the research might be to increase the number of images in the dataset with more variety on parameter settings in the image transformation part to give a solid base for stating that LDP indeed does not perform better than LBP for real-life images.

Since the learning in this experiment was categorized as unsupervised learning, another possible future work is to repeat this experiment in a supervised learning. All images in the dataset could be assigned a class where it belongs to using a weighted class voting scheme in the hope that the real precision and recall rate can be achieved.

ACKNOWLEDGEMENT

This work was completed at The University of Queensland, Australia. It is benefited by Helen Huang and Jiajun Liu from Data and Knowledge Engineering (DKE) group of School of Information Technology and Electrical Engineering (ITEE).

REFERENCES

[1] R. Datta, D. Joshi, J. Li, and J. Z. Wang, "Image retrieval: Ideas, influences, and trends of the new age, " *ACM Computing Surveys*, vol. 40, no.2, pp. 5.1-5.60, 2008.

[2] J. Liu, Z. Huang, H. T. Shen, and X. Zhou, "Efficient Histogram-based Similarity Search in Ultra-high Dimensional Space", In *Proceedings of 16th International Conferences on Databases Systems for Advanced Systems (DASFAA)*, Hongkong, 2011.

[3] Y. Gao, B. Zhang, S. Zhao, and J. Liu, "Local Derivative Pattern Versus Local Binary Pattern: Face Recognition With High-Order Local Pattern Descriptor," *IEEE Transactions on Image Processing*, vol. 19, no.2, pp. 533-544, 2010.

[4] N. S. Vassilieva, "Content-based image retrieval methods," *Programming and Computer Software*, vol. 35, no.3, pp. 158-180, 2009.

[5] Y. Rui, T. S. Huang, and S.-F. Chang, "Image Retrieval: Past, Present, and Future," *Journal of Visual Communication and Image Representation*, vol. 10, pp. 1-23, 1997.

[6] X. Yang, Q. Zhu, and K.-T. Cheng, "Near-duplicate detection for images and videos," in *Proceedings of the First ACM workshop on Large-scale multimedia retrieval and mining*, Beijing, 2009.

[7] D.-Q. Zhang and S.-F. Chang, "Detecting image near-duplicate by stochastic attributed relational graph matching with learning," in *Proceedings of the 12th annual ACM international conference on Multimedia*, New York, 2004.

[8] 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*, New York, 2004.

[9] [9] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM*, vol. 24, no.6, pp. 381-395, 1981.

[10] R. Chellappa, A. Rosenfeld, W. Zhao, and P. J. Phillips, "Face recognition: A literature survey," *ACM Computing Surveys*, vol. 35, no.4, pp. 399-459, 2003.

[11] M. Turk and A. Pentland, "Eigenfaces For Recognition," *Journal Of Cognitive Neuroscience*, vol. 3, no. 1, pp. 71-86, 1991.

[12] J. P. Hespanha, P. N. Belhumeur, and D. J. Kriegman, "Eigenfaces vs. Fisherfaces: recognition using class specific linear projection," *IEEE Transactions On Pattern Analysis And Machine Intelligence*, vol. 19, pp. 711-720, 1997.

[13] J. Atick and P. Penev, "Local feature analysis: a general statistical theory for object representation," *Network: Computation in Neural Systems*, vol. 7, no.3, pp. 477-500, 1996.

[14] L. Wiskott, J. M. Fellous, C. von der Malsburg, and N. Kuiger, "Face recognition by elastic bunch graph matching," *IEEE Transactions On Pattern Analysis And Machine Intelligence*, vol. 19, no.7, pp. 775-779, 1997.

[15] T. Ahonen, M. Pietikainen, and A. Hadid, "Face description with local binary patterns: Application to face recognition," *IEEE Transactions On Pattern Analysis And Machine Intelligence*, vol. 28, no.12, pp. 2037-2041, 2006.

[16] 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 24rd International Conference on Very Large Data Bases*,New York, 1998.

[17] A. Moffat and J. Zobel, "Inverted files for text search engines," *ACM Computing Surveys*, vol. 38, no.2, 2006.

[18] US, *"rude: a pedestrian run-length decoder-encoder"*. Matlab Central. [Online]. Available: http://www.mathworks.com/matlabcentral/fileexchange/6436-rude-a-pedestrian-run-length-decoder-encoder/content/html/rudedemo.html. [Accessed: Feb 10, 2012].