International Journal of Computer Science and Software Engineering (IJCSSE), Volume 8, Issue 10, October 2019
ISSN (Online): 2409-4285 www.IJCSSE.org Page: 224-233

Robust Design of Architecture of Multimedia Database System

Chi-Wai Wong[1], Professor H.C Man[2] and Professor Kenneth Lam[3]

[1, 2, 3] Department of Industrial and Systems Engineering, Hong Kong Polytechnic University, The Hong Kong Polytechnic University, Hung Hom, Kowloon, Hong Kong

[1]hc.man@polyu.edu.hk, [2]kin.man.lam@polyu.edu.hk, [3]roy.cw.wong@polyu.edu.hk

ABSTRACT

With the rapid development of deep learning, the accuracy of object detection is dramatically increased. Image annotation becomes more important in Content-based Image Retrieval. In addition, different kinds of image hashing are developed. The image can be represented by a series of numbers, also known as hash. The hash can be used to obtain the similarity for image retrieval. The traditional visual feature vectors, such as shape descriptor, can also be used in image retrieval. This kind of information can be stored in a system as indexing for fast retrieval. Similar to relational database system, indexing plays an important role to speed up data retrieval. Therefore, database system needs to apply indexing in order to achieve fast retrieval. Storing multimedia, such as image or PDF, is different from relational database system. This article advocates a robust approach that there is no limitation to apply some of algorithms for indexing. The proposed architecture can be extended to add any algorithm in order to achieve fast retrieval. In addition, the system design is considered to be used in internet environment. It can be easily integrated with the internet-based systems.

Keywords: *CBIR, Multimedia Database, Image Hashing, Image annotation, Architecture.*

1. INTRODUCTION

Many famous algorithms for Content-Based Image Retrieval (CBIR) were developed in the past ten years. Most of the researchers demonstrated the success story to apply the algorithms to retrieve the corresponding image from the image dataset. The most commonly used approach is to compare similarity between the query image and the images in the dataset. The visual feature is the critical candidate for the comparison. It includes color histogram [1,2], texture [3], shape [4] and visual bag of features [5]. Some researchers applied algorithm to generate hash string from the image for image retrieval [6]. With the recent development of the deep learning, some of the researchers applied deep neural network to generate deep feature as a hash for image retrieval [7]. Regardless of traditional computer vision algorithm such as spatial pyramid [8] and deep neural network such as convolutional neural network and VGGNet [9], the accuracy of object detection from the image is very high. Therefore, the keywords can be obtained from object detection frameworks. The visual feature vectors and hash are eventually an array of numbers or a sequence of digits. These kinds of data can be used for indexing. Similarly, the keywords can also be used for indexing. The query, therefore, can be an image with some textual information. The image in the dataset can be based on the combination of visual feature and textual information to be retrieved from the dataset with the aid of indexing of visual feature and keywords. However, it is rare to find out the research for the architecture of multimedia database. With the aid of indexing, the image stored in the multimedia database can be retrieved effectively.

Nowadays, most information is posted on internet. The behavior of people is changing from reading books to browsing internet. The web pages from internet contains a lot of pictures besides textual information. In the past, the web spiders for internet search engine only need to scrape the textual content from the web pages for ranking in order to list the most relevant web sites within top position. However, in order to enhance the performance of search engine, web spiders need to scrape the visual content also. The visual content needs to be interpreted as some form of information for ranking or it is needed to be transformed into another representation format for indexing so that the corresponding image can be correctly retrieved. Although the images scraped can be categorized and stored in different directory, it is not efficient for management. If the number of directories and files are huge, there is no effective and efficient way to conduct backup and recovery as it is very difficult to verify whether the files can be backed up and recovered successfully. In addition, it needs to set up a system or software to record the location of the file associated with keywords or hash string. No one can guarantee that the file can be obtained even if the record in the system or software can be retrieved since the system is isolated with the storage of the files.

In industrial application, the multimedia database system is also necessary. Owing to advocate of industry 4.0, artificial intelligence is one of the elements to build up smart manufacturing. Most of the application is

migrating from traditional sensors to computer vision. Image processing is one of the important techniques to be applied in industrial application. Instead of using sensor, the application recognizes the object by an image captured by a camera. Unlike surveillance application, it is unnecessary to store in video format. A still photo is captured based on a pre-defined time interval. The images are required to be stored. With a multimedia database system, the images can be stored effectively. The query sent to the database system can be an image associated with textual information. If the content of the query is matched, the image can be retrieved and displayed correctly. Supposing that the design of the architecture is appropriate, the multimedia database system can achieve the key features of blockchain technology. Same as the core concept in Blockchain Technology, the image, stored in the multimedia database, cannot be deleted. If the database is destroyed or hacked, the data will be synchronized from another decentralized database to build up again. The details of the architecture will be discussed in Section 4.

2. LITERATURE REVIEW

The algorithms for machine learning can be categorized as traditional Computer Vision Algorithms and Deep Neural Network in this paper in order to explain the details more easily. It does not imply this is a common approach in the world of machine learning to categorize the algorithms. The major difference between traditional computer vision algorithm and deep neural network is the feature extraction. In traditional computer vision algorithm, the feature extraction is handcraft. The feature extraction (the kernels) is learnt from the training dataset. Through the architecture of the deep network, the critical features for classification will be learnt. The features extracted may be edge, shape, texture… etc. This section will briefly describe some of the common algorithms in Computer Vision and Deep Learning.

2.1 Traditional Computer Vision Algorithms

The traditional computer vision algorithms mostly depend on the handcraft feature extraction. The most commonly used algorithms are Color Histogram (color), Local Binary Pattern (texture), Fourier Descriptor (shape) and Visual Bag of Words (keypoints).

Color is an important characteristic of the image to distinguish the objects. The common color representation models include RGB, CMY and HSV. RGB model is based on three channels. They are Red, Green and Blue channel. Each represents Red, Green and Blue color respectively. Swain and Ballard advocated the use of opponent color representations [10]. It makes a substantial improvement to the RGB color representation. The newly employed opponent color axes (R-G, 2B-R-G and R+G+B) successfully isolate the brightness information on the third axes. The first two axes can be down-sampled since humans are more sensitive to brightness rather than chroma [10]. The CYM color model is based on Cyan, Magenta and Yellow channel. It is also known as subtractive model since they are obtained by subtracting light from white. It is commonly used in color printing. However, in printing industry, the model is improved by adding channel of Black (Dark). Although black color can be obtained by using CMY, the generated color is quite soft. Therefore, black is added. The model applied in printing industry becomes CMYK. The HSV model depends on Hue, Saturation and Value representation. Gonzalez and Woods advocated to apply this model because of its invariant properties [11].

Color Histogram is the most commonly used feature representation for color characteristics. It is relatively easy to be computed. If an interest of region (ROI) is circular, the histogram of ROI is invariant to rotation since the proportion of pixels which have particular color intensity levels does not change. It is invariant to scaling if the region of interest is selected properly. The comparison between query image and test image is normally based on mean square error (MSE). The smaller the error is, the most similar the images looks. H(Q) denotes the histogram of query image and H(T) denotes the histogram of the test image.

$$\text{MSE} = \sqrt[2]{\sum_{i=0}^{n}(H(Q_i) - H(T_i))^2} \quad (1)$$

In Equation (1), *i* represents the index of the histogram bin. The similarity can also be computed by using dot product as illustrated in Equation (2).

$$\text{Similarity} = \sum_{i=0}^{n}(H(Q_i)\,.\,H(T_i)) \quad (2)$$

Texture matching is another widespread approach applied in Content-Based Image Retrieval. Local Binary Pattern (LBP) [12] is the most recent texture descriptor. Its computation is quite simple, it is very effective to describe the visual texture pattern. Each pixel at the center of a patch such as 3x3 is compared with its eight neighbors. The neighbors having smaller value than that of the central pixel will have the bit zero, whereas the neighbors having bigger or equal value than that of the central pixel will have the bit one [12]. LBP algorithm generates a binary number that is obtained by concatenating all binary bits in anti-clockwise manner, which starts from its top-left neighbors. Figure 1 illustrates the calculation of Local Binary Pattern. The normalized LBP histograms of the query image is computed. The Chi-square distance metric is applied to

compare the images. The lower the Chi-Squared distance, the better is the match.

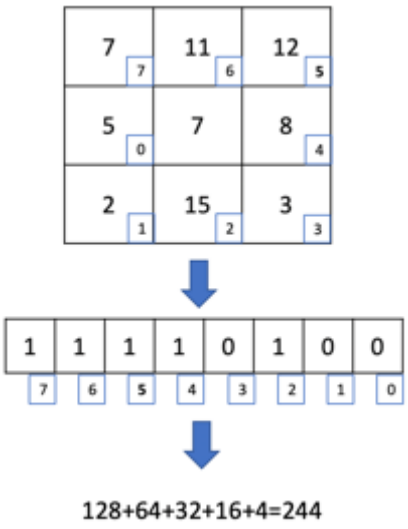

Fig. 1. Generation of Local Binary Pattern.

Shape matching is another widespread approach applied in CBIR. The most popular method to describe the internal region of a shape is moment invariants. The moments of a region R are defined as:

$$M_{pq} = \sum_{(x,y)\in R} x^p y^q f(x,y) \quad (3)$$

where f(x,y) is the image and (p+q) indicates the order of moment. Another important shape descriptor is Fourier Descriptor. The descriptor can be obtained by applying Discrete Fourier Transform (DFT) on the boundary curve from the centroid of the object. It is basically defined as:

$$F(k) = \frac{1}{N}\sum_{n=0}^{n-1} z(n) e^{-j2\pi nk/N} \quad (4)$$

Another method is autoregressive (AR) method. It is based on the stochastic modeling of a 1D function obtained from the shape. A linear AR model expresses a value of a function as a linear combination of a certain number of preceding values [13]. The drawback of AR method is that a small number of AR parameters is not sufficient to describe the shape if the boundaries are very complicated.

David Lowe, in 1999, published an algorithm named as Scale-invariant feature transform (SIFT) to detect and describe local features in images [14,15,16]. Inspired by the idea of bag of words in document retrieval, visual bag of features/words is applied in CBIR. Besides comparing the feature vectors, image annotation plays an important role in image retrieval through object detection. With object detection from the keypoints in the image, the objects can be recognized. A feature vector with the keywords such as aero-plane, sky, … etc, can be generated. The keywords can be stored in a database for indexing so that the image in the dataset can be retrieved efficiently. Spatial pyramid [17] is one of the effective algorithms applied in object detection. The keypoints of the trained dataset are computed. Visual Vocabulary is generated based on the trained dataset with k-means clustering and the image is divided into several sub-image for spatial characteristics.

The features, including the visual content descriptor or the keywords generated by the object detection framework, obtained by the traditional computer vision algorithms can be used for indexing in the multimedia database system.

2.2 Deep Neural Network

Le Cun et al [18], in 1999, developed a convolutional neural network (CNN) by using learnable convolutional filter in convolutional layer and maximum pooling/average kernel in sub-sampling to down-sample the output value from the previous activation layer. With this architecture, the number of trainable parameters is decreased dramatically. Therefore, the dilation of error in the back-propagation learning algorithm is decreased. The weights in each neuron will not become zero after several ecophs. Figure 2 illustrates the architecture of LeNet-5 [18].

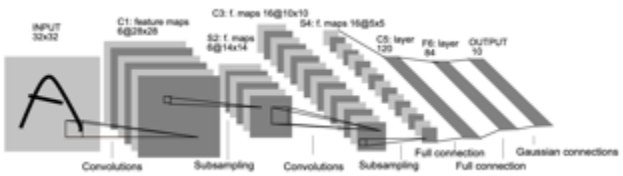

Fig. 2. Architecture of LeNet-5 [18].

Four famous CNN architectures, AlexNet, VGGNet, ResNet and Inception, were developed for object detection and outperform a lot of algorithms. The CNN architectures provide a high accuracy in object detection. Therefore, it is applied in image annotation to generate a feature vector contains the keywords by object recognition in the images.

AlexNet [19] is much larger than the original convolutional neural network (CNN). It has 60 million parameters and 650,000 neurons. Overfitting is a problem for a neural network with 60 million parameters. The researchers experimented with other ways to reduce overfitting. The technique is called dropout. In dropout, a neuron is dropped from the network with a probability of 0.5. When a neuron is dropped, it will not contribute to either forward or backward propagation. Figure 3 illustrates the architecture of AlexNet.

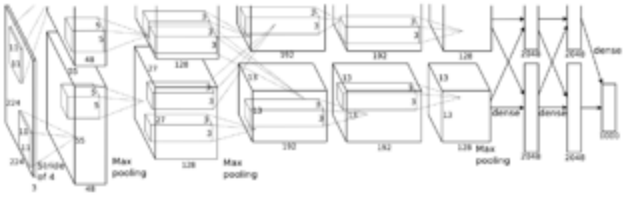

Fig. 3. Architecture of AlexNet [19].

VGGNet was introduced by Simonyan and Zisserman [9]. VGGNet demonstrated that the architecture with very small (3x3) filters can be trained to increasingly higher depths around 16~19 layers. It outperforms a lot of algorithms to obtain state-of-the-art classification on the challenging ImageNet classification challenge. In CNN, the first layer usually is 7x7 or 11x11 filter. And then, the filter sizes are progressively reduced to 5x5. Finally, the deepest layers of the network use 3x3 filter. However, VGGnet uses 3x3 kernels throughout the entire architecture. Figure 4 illustrates the architecture of VGGNet. The image with 224x224x3 dimensions is inputted into the network. 3x3 convolutional filters are then applied with more convolutions on top of each max pooling as shown in Figure 4.

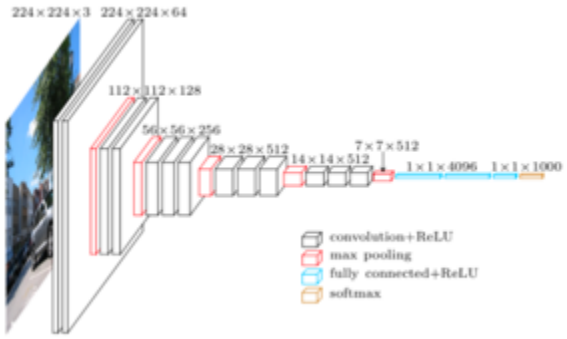

Fig.4. Architecture of VGGNet [9].

Basically, increasing the depth of the deep neural network means the accuracy is increased, but at the same time, overfitting is raised. In addition, vanishing gradient will appear as the earlier layers are most negligible since the weight learned becomes zero. By constructing the network through modules, it allows training of deep neural network without impact of vanishing gradient. The modules are called residual models. The network is called residual network. Kaiming et.al [20] developed a new neural network, named as ResNet. Figure 5 illustrates the residual model.

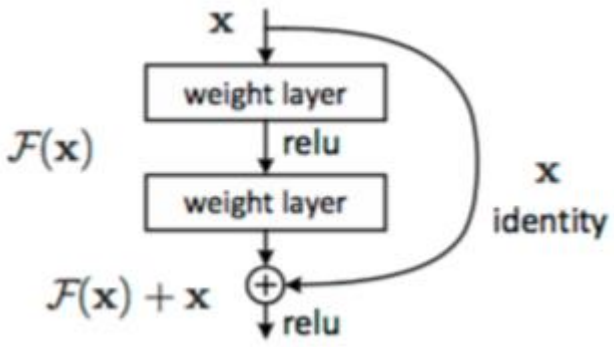

Fig. 5. Residual learning: a building block [20].

In 2014, Szegedy et al. [21] introduced the inception module and developed Inception architecture as illustrated in Figure 6. The goal of inception module is to act as "multi-level" feature extractor by computing 1x1, 3x3, and 5x5 convolutions within the same module of the network, the output of these filters are then stacked along the channel dimension before being fed into the next layer in the network [21].The first branch in the Inception module simply learns a series of 1x1 local features from the input. The second branch first applies 1x1 convolution, not only as form of learning local features, but instead as dimensionality reduction [21]. The branch then applies 3x3 convolutional filter. The third branch applies the same logic, but 5x5 filter replace 3x3 filter. The final branch performs 3x3 max pooling with a stride of 1x1.

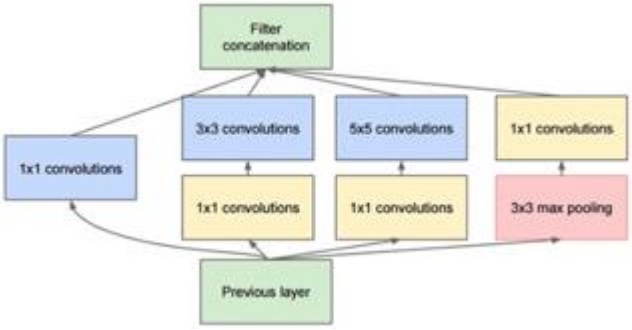

Fig. 6. The inception module. [21]

Nikolaos and Anastasios combined the bag of features and deep neural network to generate a new feature vector named as neural bag of feature which can be applied as image hashing [7]. Figure 7 shows the Retrieval-oriented Neural Bag-of-Feature (RN-BoF).

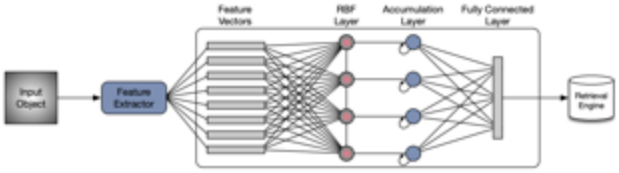

Fig. 7. Architecture of Retrieval-oriented Neural Bag-of-Feature (RN-BoF) [7].

Similarly, the deep features computed by the deep neural network or the keywords obtained by object detection can be used in indexing for image retrieval. However, it is necessary to have a robust database architecture so that the database system can have different kind of indexing approach to handle different kind of the feature vectors. The indexing component in the database system cannot be isolated. Otherwise, the retrieval cannot be optimized. It is ideal for the system to allow users to retrieve images from the database system with different hints for the indexing approach. This motivates a robust approach for the architecture of multimedia database system.

3. ARCHITECTURE OF MULTIMEDIA DATABASE SYSTEM

The core components in the multimedia database system are Database Engine and Indexing Engine. The two engines must seamlessly integrate in order to provide fast retrieval. Two components can be analogous to kernel of operating system. A shell is needed so that the external party can communicate with the kernel of the database system. In addition, the database system should facilitate users for some database manipulation, and the database system should be internet-ready so that the other system can communicate with the database system through the industry-proven protocol. The proposed architecture of Multimedia Database System is shown in Figure 8.

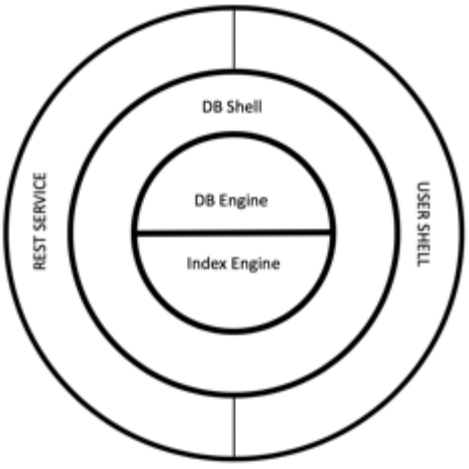

Fig. 8. Architecture of Multimedia Database System

The Multimedia Database System consists of DB Engine, Index Engine, DB Shell, User Shell and REST Service.

3.1 DB Engine

The primary roles of DB Engine are to read the content from the designed database format and to write the content in the designed database format. The proposed format of the multimedia database is illustrated in Figure 9.

Starter Byte	Starter Byte	File Name (6 bytes)
Reserved Bytes (8 bytes)		
Status Byte	Timestamp of data (7 bytes)	
Next Offset Byte (8 bytes)		
Data		
Status Byte	Timestamp of data (7 bytes)	
Next Offset Byte (8 bytes)		
Data		
Status Byte	Timestamp of data (7 bytes)	
Next Offset Byte (8 bytes)		
Data		

Fig. 9. Data Format of Multimedia Database.

The two starter bytes are used to check if the data file is operated by our system. The first starter byte is 1111000 and the second starter byte is 10101010. The file name consists of one byte for identification such as machine id, two bytes for year, one byte for month and the remaining byte for day. Therefore, the format of the file name saved following two starter bytes is machine_id + year + month + day. The reserved byte is for future use. Figure 10 demonstrates the header of the data file.

Starter Byte	Starter Byte	File Name (6 bytes)
Reserved Bytes (8 bytes)		

Fig. 10. Header of the data file.

The data block includes status byte, timestamp bytes, next offset bytes and the data object. The status byte can reflect the status of the data object. The timestamp bytes are to record the datetime of the object stored or the timestamp of data object to be stored in the database. The timestamp can be defined based on different application for the multimedia database. The next offset bytes point to the next data block. The bytes facilitate for the sequential reading and separating each data block for statistics. Data is to store the multimedia file such as JPEG file. Figure 11 shows the data block.

Status Byte	Timestamp of data (7 bytes)
Next Offset Byte (8 bytes)	
Data	

Fig. 11. Data Block.

The data objects, such as JPEG files, are stored in the data file sequentially. Since the next offset bytes always point to the next data block, the last data block will point to the end of the file. Therefore, the DB Engine is easy to determine whether the reading reaches end of files. Depending on the application of multimedia database, two end bytes can be added to the end of file whereas the last data block points to the first of end byte. The DB Engine can base on the value of the two bytes to determine whether end of file is reached. In normal situation, the proposed data format shown in Figure 9 is most suitable for most of the applications.

3.2 Index Engine

For each data write requested to DB Engine, it will return the pointer of the file to the caller. The pointer is eventually a long number. The caller can save it to another file for fast retrieval. Combined with object recognition framework, the number associated with the keywords generated is suggested to be stored in MongoDB, which is a document-oriented database, or NoSQL database. MongoDB does not limit the database creator to create the database schema before saving. The schema can be changed dynamically. In addition, each document, analog to a record in relational database, can have different schema. For example, the data file has two records, named as data block 1 and data block 2. The information stored in MongoDB for data block 1 can be {'name':'sea.jpg', 'ts':'201909271530','offset':16308,'keywords': ['sea', 'sky']}. The information stored in MongoDB for data block 2 can be {'name':'beach.jpg', 'ts':'201909281429', 'offset':38976, 'keywords': ['sea', 'tree', 'sand'], 'hash': 61339203679389783293442356 63}. The schema of each document can be different. It is not required to create before saving. With the well-defined JSON data passed to MongoDB, it will be stored in the database. MongoDB is flexible and easy to save semi-structured data. Fundamentally, indexes in MongoDB are similar to indexes in other database systems. MongoDB defines indexes at the collection level and supports indexes on any field or sub-field of the documents in a MongoDB collection [23].

Besides MongoDB, the image hash can be stored in K-Dimensional Tree (K-D Tree). K-D tree is a binary tree. In order to illustrate how K-D tree operates, let us take an example of 2-D tree. 2-D tree has two planes. The root would have an x-aligned plane, whereas the root's children would all have y-aligned planes. Similarly, the root's grandchildren would all have x-aligned planes, whereas the root's great-grandchildren would all have y-aligned planes and so on. For two-dimensional tree, the plane can be numbered as 0 and 1. It means x-plane and y-plane. For K-dimensional tree, the plane can be numbered as 0,1,2, …(K-1). The plane number can be obtained by the following formula.

Plane number = D mod K

where D is the depth of a node and K is the dimension. Let us consider the points (4,7), (18,16), (14,16), (7,13), (10,2), (3,8), (11,20). The 2-D tree is illustrated in Figure 12.

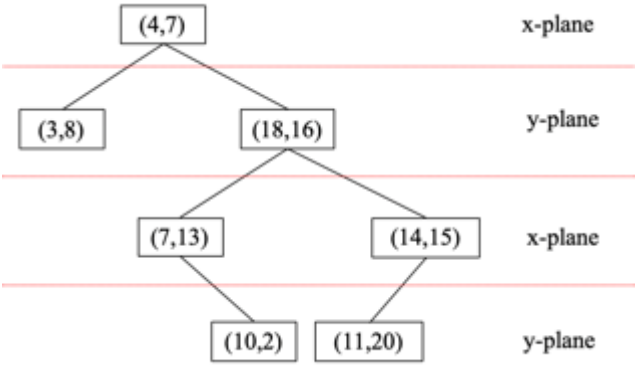

Fig. 12. 2-D Tree.

Since tree is empty, (4,7) becomes a root. The node (3,8) compares the value of x-plane of the root (4,7) since the root is x-aligned. (3,8) lies on the left subtree of root since 3 is smaller than 4. Similarly, (18,16) compares the value of x-plane of the root (4,7). The node (18,16) lies on the right subtree of root since 18 is greater than 7. Node (7,13) compares the value of x-plane with the root. It then goes to the right subtree. Since node (18,16) is y-aligned, (7,13) compares the value of y-plane with (18,16). Owing to 13 is smaller than 16, (7,13) lies on the left subtree of (18,16). Nodes (14,15), (10,2) and (11,20) works similarly as described above. The data structure of K-D tree can enhance fast retrieval.

The index engine does not limit the algorithm or software applied. Instead of K-D tree, VP tree can be integrated with index engine in the Multimedia database. The query sent to the database can provide the hint for the system to search from the specified index. For example, the command in the user shell can be "get-image –keyword ['sea','sky'] –hash 61339203679389783293442356 63". User shell will be discussed later. The index engine can use MongoDB to store the keywords and K-D Tree to store the hash. This is transparent to users. Database Designer can use all types of optimized algorithm or software to provide indexing for users in order to achieve fast retrieval.

3.3 DB Shell

The previous two sections discussed the core components of the database. DB Engine is responsible for reading and writing the data from/to the database. Index Engine

is responsible for providing fast retrieval feature through the optimized algorithm and software. DB shell is responsible for the commands received to communicate with the kernel of the database. This is similar to the operating system as users normally will not communicate with kernel directly.

DB Shell is designed to provide API calls or functions for external parties to use database. For example, the command ("get-image –keyword ['sea','sky'] –hash 61339203679389783293442356 63") is typed in user shell. User shell will trigger the corresponding API call to DB shell with arguments of keywords and hash. The result will be passed back to the user shell for further processing. Since the design of DB shell is separated from user shell, it can be more flexible and concentrated on what features the database should be provided with this approach of separation of concern.

3.4 User Shell

User shell is the human machine interface. It reads all the commands typed by the users. If the syntax of the command is wrong, it will prompt the meaningful error to users. With the separation of concern, a set of commands can be designed in the user shell without any impact to the kernel of the database. All results returned from the DB shell can be transformed to be a more man-kind readable message. Since it is interactive with human, the image returned from the database can be displayed by user shell. DB shell does not need to cater any coding for user interaction. This simplified the coding and enhance the bug-free feature.

User shell may not be necessary in some application. However, in most situation, user shell is necessary as it is a bridge for the users to communicate with the kernel of the database system. For example, the statistics or health check can be provided from the DB shell. User shell can interpret the results returned to provide a more meaningful report for users. Therefore, with user shell, the database can be easily operated and administrated.

3.5 REST Service

Unlike user shell, REST Service is the interface between machines. It is a machine-to-machine interface. Another system may call the REST services provided by the database system. This enhances the automation without any user interactions. Simply, REST contains four verbs: GET, POST, PUT and DELETE. Analog to data manipulation from relational database, GET corresponds to SELECT statement of Standard Query Language (SQL). POST corresponds to INSERT statement of SQL. PUT corresponds to UPDATE statement of SQL and DELETE corresponds to DELETE statement of SQL. REST leverage http/https to send request of one of four verbs with the data in JSON format. The data in JSON can be data and/or the arguments of the function call. The arguments in GET REST service call are passed through the URL, for example, http://192.168.10.1:14430/get-image/873901. The URL specifies the IP address of the REST service provider with the port number. The function name, get-image, is also embedded in the URL with the argument, 873901.

Nowadays, REST web service is a de facto standard of remote function call. Most of computer languages, such as Python, can be easily developed REST web services through the bundled library or framework. For example, Django and Flask are the famous Python framework for building RESTful API call.

Similar to user shell, some applications may not need REST services. However, in most situation, remote function call is necessary, especially for automation or integration with other systems. REST is a well-known and industry proven technology. Therefore, it is recommended to include REST services layer in the database system. In addition, REST services will communicate with DB shell directly. If DB shell provides any administrative features, REST services can facilitate the remote administration. This is a normal practice in IT industry.

4. EXPERIMENTS

The Multimedia Database based on Figure 8 is developed by using Python. With the elegant framework of the database system, the system can also run in ARM-based device such as Raspberry Pi and other Linux systems which support Python. In our experiment, the system runs on Ubuntu 18.04 Desktop.

4.1 Data Write

The image can be saved into the database. In order to demonstrate what data written into the database, the system will print out the message. Normally, this will be stored in log file, rather than displaying on screen.

Figure 13 shows the image file ('Sydney.jpg') is saved in the database. If the database does not exist, the system will create a new data file with the corresponding data header as illustrated in Figure 13.

```
ubuntu1804@MMDBSvr:~/MMDB$ python3
Python 3.6.8 (default, Jan 14 2019, 11:02:34)
[GCC 8.0.1 20180414 (experimental) [trunk revision 259383]] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> from MMDB import DBShell as shell
>>> shell.save('Sydney.jpg','11:29')
DBNotFound.
Write Header - First Start Byte: 1 byte(s)
Write Header - Second Start Byte: 1 byte(s)
Write Header - FileName(MachineID): 2 byte(s)
Write Header - FileName(Year): 2 byte(s)
Write Header - FileName(month): 1 byte(s)
Write Header - FileName(day): 1 byte(s)
Write Header - Reserved Byte: 8 byte(s)
Write Data Header - Status Byte: 1 byte(s)
Write Data Header - Timestamp: 5 byte(s)
Debug: current position of file is 16
Debug: object size is 78353
Write Data Header [Next Offset] is 78383
Write Data Header - NextOffset: 8 byte(s)
Write Data Object - File Write: 78353 byte(s)
The offset returned is [16]
Sydney.jpg is saved at 11:29
```

Fig. 13. Save the image ('Syndey.jpg') into the database.

Since the data file is not found, the database system will create a new data file. Figure 13 clearly shows what the header of the data file and the header of the data block are written in the database followed by the data object, the image file of Sydney.jpg.

```
ubuntu1804@MMDBSvr:~/MMDB$ python3
Python 3.6.8 (default, Jan 14 2019, 11:02:34)
[GCC 8.0.1 20180414 (experimental) [trunk revision 259383]] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> from MMDB import DBShell as shell
>>> shell.save('Sea.jpg','11:38')
DBFound.
DB Exists: No need to write header.
Write Data Header - Status Byte: 1 byte(s)
Write Data Header - Timestamp: 5 byte(s)
Debug: current position of file is 78383
Debug: object size is 89682
Write Data Header [Next Offset] is 168079
Write Data Header - NextOffset: 8 byte(s)
Write Data Object - File Write: 89682 byte(s)
The offset returned is [78383]
Sea.jpg is saved at 11:38
```

Fig. 14. Save another image ('Sea.jpg')

Figure 14 shows that the database system will not create a new data file if the data file found. Depending on the application, DB Shell can provide the feature for users to specify the filename. The file of Sea.jpg is stored in the database following the data of Sydney.jpg. The images can be sequentially saved in the database.

Unlike online transactional processing, the transaction is based on the pattern of CRUD – (Create, Read, Update and Delete). Although some of the applications may require deleting the image in multimedia database system, the image stored normally will not be deleted. Imagining the surveillance system, the images recorded in the system will not be deleted. The most suitable application of the database system is to record the image for object recognition, such as warehouse management system by using computer vision. The image stored will not be asked to delete.

4.2 Data Read

The images stored in the database can be retrieved by keywords, hash and/or the position. This depends on the database designer how to design the system. With the aid of some other python package such as matplotlib and scipy, the data string returned from the database system can display on the screen easily. Figure 15 shows the function call from the user shell to retrieve the first data object and display the image on the screen.

```
Python 3.6.8 (default, Jan 14 2019, 11:02:34)
[GCC 8.0.1 20180414 (experimental) [trunk revision 259383]] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> from MMDB import UserShell as shell
>>> shell.displayImagebyPosition(1)
DBFound.
```

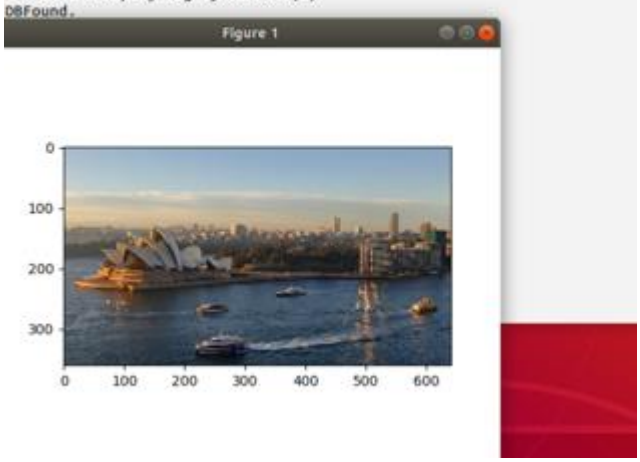

Fig. 15. Display the image stored in the first position.

Since the system is capable of obtaining an array object in Python, the stored image can be directly passed to external Python application for deep learning or computer vision without any conversions. This is an advantage that it can integrate with any other computer vision Python-based application. It is very important for research as the keywords can also be saved through the object recognition framework. Figure 16 shows the result of array returned from the database when it is asked to retrieve the first data object.

```
Python 3.6.8 (default, Jan 14 2019, 11:02:34)
[GCC 8.0.1 20180414 (experimental) [trunk revision 259383]] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> from MMDB import DBShell as shell
>>> img = shell.returnNumArray(1)
DBFound.
>>> img
array([[[170, 187, 203],
        [170, 187, 203],
        [170, 187, 203],
        ...,
        [141, 165, 193],
        [141, 165, 193],
        [141, 165, 193]],

       [[170, 187, 203],
        [170, 187, 203],
        [170, 187, 203],
        ...,
        [141, 165, 193],
        [141, 165, 193],
        [141, 165, 193]],
```

Fig. 16. Obtain the array for further processing.

5. FURTHER DISCUSSIONS

Image hashing is a hot topic in research area. A hash can be generated by an algorithm such as ITQ [23] from an image. Hashing is one of the important components in Blockchain technology. The hash generated from the image can be combined with the hash from previous data block to produce a new hash. Therefore, the current hash is dependent on the previous hash to form a chain as the basic concept of Block chain. It is not easy for a hacker to change the data block in the middle. With the design of decentralized database system, the corrupted or hacked data block can be recovered from the decentralized database. Therefore, with the multimedia database, Block chain technology can be easily applied. Furthermore, the database system does not limit to store images. It can store word documents and/or PDF. The status byte may be used to classify the data object. Figure 17 shows the concept of Blockchain technology applied in the database system.

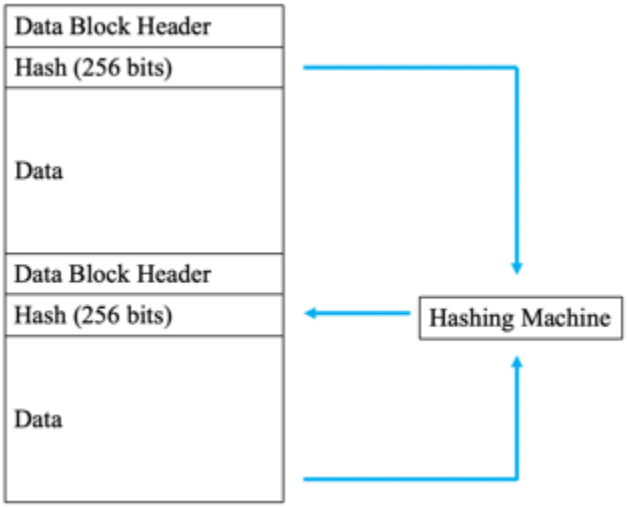

Fig. 17. Concept of hashing applied in multimedia database

6. CONCLUSIONS

The database system can be developed by using Python. Therefore, it can be run on any Linux systems which support Python. Instead of installing a large number of Python package, no additional package is needed for DB Engine. This means that the application can be run on ARM-based device, such as Raspberry Pi. In addition, there is no limitation to algorithm to be applied in Index Engine. Index Engine can use any optimized algorithm or software which can be integrated by using Python. The design is robust to apply any algorithm. If the system needs to display an image, only matplotlib and scipy package is needed to install. The REST service layer can facilitate automation and the returned result can be an array which can be passed to another Python application such as deep learning application without any conversion. In addition, with image hashing, the database can be designed to have features of Blockchain which can prevent any unattended modification. The multimedia database is especially useful to apply in industrial area to store the image for object recognition.

REFERENCES

[1] Shapiro, Linda G. and Stockman, George C., Computer Vision, Prentice Hall, 2003
[2] Xiang-Yang Wang, Jun-Feng Wu1 and Hong-Ying Yang "Robust image retrieval based on color histogram of local feature regions", Multimedia Tools and Applications, Vol 49, Issue 2, 2010, pp. 323-345.
[3] Zhao Hai-ying, Xu Zheng-guang and Peng Hong, "A Texture Feature Extraction on Two Fractal Dimensions for Content Based Image Retrieval", in WRI World Congress on Computer Science and Information Engineering, 2009, pp. 117-121.
[4] A. EI-ghazal, O. Basir, and S. Belkasim, "A New Shape Signature for Fourier Descriptors.", in IEEE International Conference on Image Processing, 2007, pp. 161-164.
[5] G Csurka, C Dance, L Fan, j Willamowski, C Bray, "Visual Categorization with bags of keypoints", in Workshop on statistical learning in computer vision, 2004, ECCV 1:22.
[6] Meenalochini. M, Saranya. K, G.V. Rajkumar and Akash Mahto, "Perceptual Hashing for Content Based Image Retrieval", in 3rd International Conference on Communication and Electronics Systems (ICCES), 2018, pp. 235-238.
[7] Nikolaos Passalis, Anastasios Tefas, "Learning Neural Bag-of-Features for Large-Scale Image Retrieval", IEEE Transactions on Systems, Man, and Cybernetics: Systems, Volume 47, Issue 10, 2017, p.2641-2652.
[8] S. Lazebnik, C. Schmid and J. Ponce, Beyond Bags of Features: Spatial Pyramid Matching for Recognizing Natural Scene Categories, In IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2006, Vol. 2, pp. 2169-2178.
[9] Karen Simonyan and Andrew Zisserman. "Very Deep Convolutional Networks for Large-Scale Image Recognition". In: CoRR abs/1409.1556 (2014). URL: http://arxiv.org/abs/1409.1556
[10] Swain, M. J., and Ballard, D. H., "Color indexing.", International Journal of Computer Vision, 1991, Vol 7, 11–32.
[11] Gonzalez, R. C., and Woods, R. E., Digital image processing. Prentice Hall Press, 2002.
[12] K. Meena and A. Suruliandi, "Local binary patterns and its variants for face recognition," in International Conference on Recent Trends in Information Technology (ICRTIT), 2011, pp. 782-786.
[13] Kauppinen H, Seppanen T, Pietikainen M, "An experimental comparsion of autoregressive and Fourier-

based descriptors in 2DS shape classification.", IEEE Trans PAMI, 1995, Vol. 17, No. 2, pp. 201-207.
[14] Lowe, David G., "Object recognition from local scale-invariant features", in Proceedings of the International Conference on Computer Vision, 1999, Vol. 2, pp. 1150-1157.
[15] Lowe, David G., "Distinctive Image Features from Scale-Invariant Keypoints.", International Journal of Computer Vision, Vol. 60, No. 2, 2004, pp. 91-110.
[16] H. Bay, A. Ess, T. Tuytelaars and L. Van Gool, "Speeded-Up Robust Features (SURF)", Computer Vision and Image Understanding, VOL 110, 2008, pp. 346-359.
[17] S. Lazebnik, C. Schmid and J. Ponce, "Beyond Bags of Features: Spatial Pyramid Matching for Recognizing Natural Scene Categories", In IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2006, Vol. 2, pp. 2169-2178.
[18] Yann Lecun, L´eon Bottou, Yoshua Bengio, and Patrick Haffner, "Gradient-Based Learning Applied to Document Recognition", In Proceedings of the IEEE, Vol. 86, No. 11, 1998, pp. 2278-2324.
[19] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton., "ImageNet Classification with Deep Convolutional Neural Networks", In Advances in Neural Information Processing Systems, Vol. 25, 2012, pp. 1097–1105.
[20] Kaiming He, Xiangyu Zhang, Shaoqing Ren and Jian Sun, "Deep Residual Learning for Image Recognition". In: CoRR abs/1512.03385 (2015). URL: http://arxiv.org/abs/1512.03385
[21] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke and Andrew Rabinovich, "Going deeper with convolution". 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, 2015, pp. 1-9.
[22] Y. Gong, S. Lazebnik, A. Gordo and F. Perronnin., "Iterative Quantization: A Procrustean Approach to Learning Binary Codes fro Large-Scale Image Retrieval", In IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol 35, No. 12, 2013, pp. 2916 – 2929.
[23] https://docs.mongodb.com/manual/indexes/# id2
[24] J. Burton Browing and Marty Alchin, Pro Python, Third Edition, Apress, 2019.