The Similarity Sort of Single Objects on Monotonous Background

Jau-Ji Shen and Yung-Chen Chou

Department of Information Management, Chaoyang University of Technology, 168 Gifeng E. Rd., Wufeng, Taichung County, TAIWAN 413, R.O.C.

Email: jjshen@mail.cyut.edu.tw Fax: 886-4-23742337

Abstract: In this paper, a simple but efficient similarity sorting technique is proposed by which specifically to classify color images with monotonous background into several similar groups. There is only the color attribute to be considered during similarity retrieval. In order to solved the representing inadequacy of the color attribute to spatial characteristic. The key idea is to partition the images and to find major sub-image from each image by a designated majority function, and finally, using major sub-image as a key with threshold to proceeded image similarity retrieval or classification.

Keywords: classification, color attribute, image index database, similarity retrieval, threshold

1. Introduction

In recent years, due to the wide spread usage of the internet, there has been an increasing need for dada transformations via the net, instead of the data in mere text only of the early days, but has now gradually to include those multimedia data, e.g. video, voice, animation and etc [7][11]. Additionally, the development of the E-commerce has made digital product catalogs such as images with those apparent topical subjects in monotonous background on the Web. The techniques of images

retrieval are used to sort images by features such as the meta-data of the image (such as location, author, time ...et al.), color, shape, texture and so on. [2][3][4][5][6][12].

Owing to diversified natures of the user requirement, if sorting images by merely its meta-data, the user requirements may not be fulfilled fully. Besides, by using image's meta-data would take up more storage spaces by the necessary features. Conversely, if the features were not taken enough, it could fall into the dilemma of not representing the images adequately. Therefore, it would become important to find a way to retrieve features from the image itself.

Color is not only a straightforward feature but also a natural one. However, in the early days of limited computer capability, this feature was not taken seriously as the retrieval of color informations would require a vast amount of costly calculation power. Nevertheless, the vast progress of information technology, the retrieval of color features has become more common than before. Color attribute can be represented by color distribution in an image [13]. This is done so by abstracting and summing up an image's pixel values of the three primaries color Red, Green and Blue (RGB), and to find the average, which can be used as a reference for image similarity retrieval. Although this technique could find similar images, it would, at the same time, get some images that were different at all. The cause is, when averaging the RGB, the features of color distribution would be weakened, and thus picking some images with same average in RGB values but totally different RGB distributions on images. To counter this problem a new classification technique is proposed upon color features. This technique can improve the representative inadequacy of color feature to spatial characteristic, and thus to enhance the accuracy of image similarity retrieval. What is more, the technique sorts and reaches the effect of images classification by finding major sub-images through majority function.

In the beginning of Section 2, some initial works would be introduced, including

the usage of RGB, image partition, and definition of similarity metric. After that, in Section 3, it would be partition technique for image data, using majority function to find out major sub-images, and with the aid of threshold to proceed with sorting. Within Section 4, there is experiment through some actual images, and to discuss the effectiveness of our technique. Within Section 5, it would brief discuss about the affections of image process. Lastly, a brief conclusion would be found in Section 6.

2. Preposition

The most instinctive feature gathering of a color image's information is to analysis the composition of the color. In [8] [9], there is a simple way, which is to get the average of each pixel's RGB, and use the average to proceed searching and comparison, and get the result of sorting similar images. In order to measure the level of similarity of two images, similarity metric [1][10] would normally need to be defined. For simplicity purposes, we would define similarity metric by Euclidian Distance, as shown in the following.

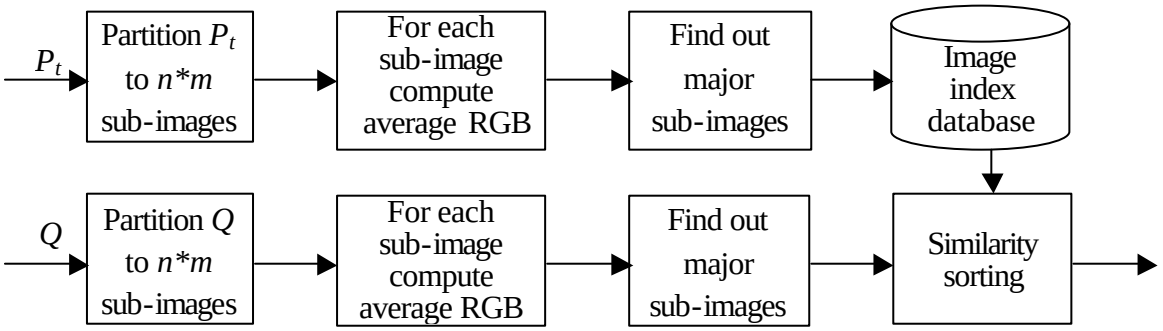
Distance function:

$$D(Q,P_t)=\sqrt{(R_1-R_2)^2+(G_1-G_2)^2+(B_1-B_2)^2} \quad (1)$$

Where Q and P_t are two images with RGB values as (R_1, G_1, B_1) and (R_2, G_2, B_2) respectively. Through this definition, the similarity between two images can be calculated.

Fig. 1 is the conceptual illustration of image similarity retrieval process, when the color images are going to be categorized, we would partition the image would be partitioned into pieces at first, and then determine each image's feature, and save into image index database. When enquiring for images, the query image Q would be

partitioned, and gathered features, then sorting features of images in the image index database according to similarity metric. After such sorting, those features representative the query image most would appear in the front.



- P_t: the archive image
- Q: query image
- $n*m$: number of the sub-images

Figure 1 Concept of similar image retrieval

Although it is a fast and instinctive way of similarity search by using the average values of RGB but sometimes may go wrong. The main reason is that the average of the color would deteriorate the difference of the content of the images, as shown in the Fig. 2. Image A and B are totally different images. Yet the result of RGB average value clearly shows that they are identical. In other words, this technique could find two totally "different" images. To solve this problem, the images should be partitioned into smaller pieces. The averaging RGB value of the divided images after partitioning image A and B, it is apparent that the value of the each sub-image of A and B is vastly different. Therefore, by basing on the RGB average value of the sub-images, A and B are two different images.

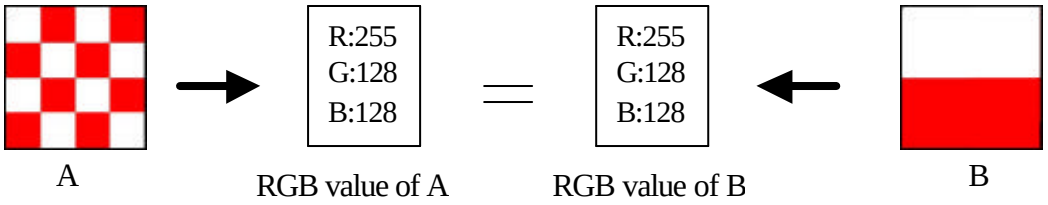


Figure 2 before partition image average RGB value

3. Our Technique

Considering $P_1,P_2,\ldots,P_k$ (k is the total number of images in the image index database) represent all the images in a database, and each of image $P_t(t=1,2,\ldots,k)$'s partitioned sub-image P_{t_i} would have one majority value $M(P_{t_i})$, whereas majority function is defined as follow:

$$M(P_{t_i}) = \sum_{j=1}^{n*m} D(P_{t_i}, P_{t_j}) \tag{2}$$

Where i denote the i-th sub-image of image P_t. Within $D(P_{t_i}, P_{t_j})$ is distance between two sub-images P_{t_i} and P_{t_j}. Each image has its own sub-images' similarity sequence $S_{P_t} = (i_1, i_2, ..., i_{n*m})$, meaning to use sub-image numbering sequence of P_t presented as S_{P_t} and such that $M(P_{t_{i1}}) \le M(P_{t_{i2}}) \le ... \le M(P_{t_{i(n*m)}})$.

The key concept is illustrated, as Fig. 3, and Block A resembles the action of classification of images. First, to prepare for partition of the input image P_t, and to calculate the RGB average values of every sub-image of P_t, and then using the value $M(P_{t_i})$ to the sub-images' similarity sequence of $P_{t_{i1}}$ in order to find the sub-image $P_{t_{i(n*m)}}$ which has the greatest majority value, the sub-image $P_{t_{i1}}$ which has the least majority value and the sub-image $P_{t_{i(mid)}}$ which has the nearest average majority value (*mid* means the closest to $P_{t_{i(n*m)}}$ and $P_{t_{i1}}$'s RGB average value's sub-image), and save RGB average value into image index database individually. Block B illustrates when a query image Q enters, Q will be partitioned like others, calculated each of Q's sub-images RGB average value, and the sequence Q's sub-images by order of majority function value to get S_Q, and then computing the similarity metric

between every images in database, represented by the sub-images $P_{t_{i(n*m)}}$, $P_{t_{i1}}$, $P_{t_{i(mid)}}$ and Q, represented by the sub-images $Q_{i(n*m)}$, Q_{i1}, $Q_{i(mid)}$, i.e. $D(Q_{i(n*m)}, P_{t_{i(n*m)}})$, $D(Q_{i1}, P_{t_{i1}})$, $D(Q_{i(mid)}, P_{t_{i(mid)}})$, and taking the increasing order of images P_1, P_2,..., P_k with respect to those three distances, the frontier the more similar to Q.

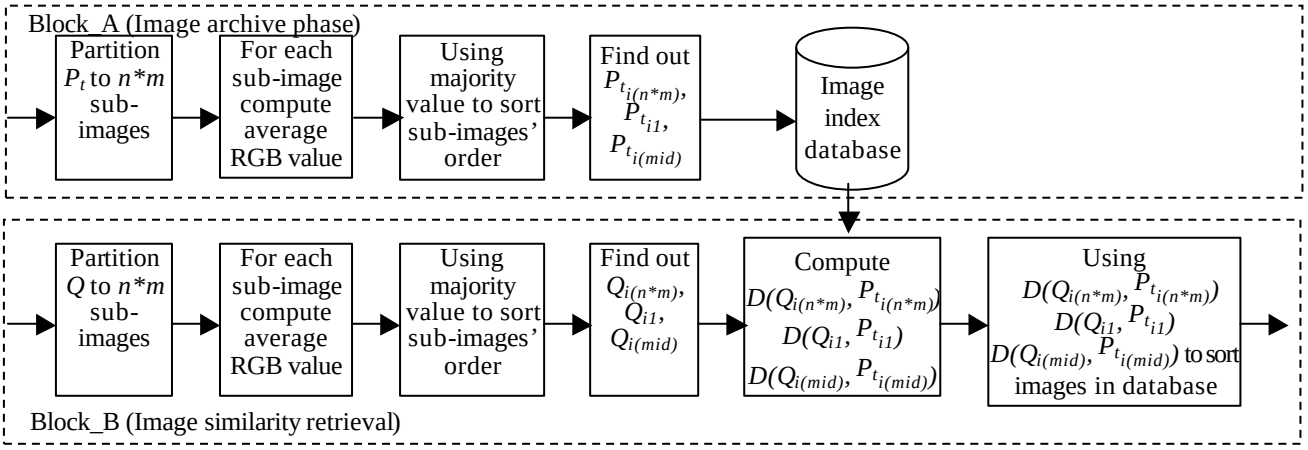


Figure 3 Illustration for similar image retrieval

3.1 Image archive phase

Algorithm A (for image archive phase):

Input: P_t (P_t is the archive image)

Function: $R(image)$ is the calculated *Red* average value in the image

Function: $G(image)$ is the calculated *Green* average value in the image

Function: $B(image)$ is the calculated *Blue* average value in the image

Output: save the P_t's sub-images feature into image index database

Step 1: partitioning P_t to $n*m$ sub-images

Step 2: For each sub-image compute RGB average value

Step 3: Compute majority value from each of sub-image P_{t_i} by Equation (2)

Step 4: Sorting $M(P_{t_i})$ by in increasing order to get P_t's sub-images and obtain the sequence $S_{P_t} = (i_1, i_2, ...i_{n*m})$

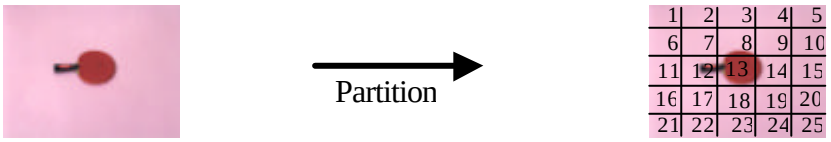
Step 5: Find out the RGB average values of $P_{t_{i(n*m)}}$, $P_{t_{i1}}$, $P_{t_{i(mid)}}$

Step 6: Store the P_t's features to image index database

Example 3.1 illustrates clearly the process of Algorithm A.

Example 3.1 Image archive to image index database

Let image P_1 (Fig. 4 (a)) and partitioned by 5*5 (as show in Fig. 4 (b)), calculating every sub-image's RGB average value as in Table 1. Table 2 is the calculated distance between each sub-image (i.e. $D(P_{t_i}, P_{t_j})$).



(a) (b)

Figure 4 Example for image partition

Table 1 Average RGB value for P_1 partitioned (Figure 4 (b))

sub-images	Average RGB value
1	(241,188,212)
2	(243,190,213)
3	(244,191,216)
4	(240,188,212)
5	(230,177,202)
6	(249,195,219)
7	(247,192,216)
8	(231,166,185)
9	(241,187,209)
10	(232,178,201)
11	(251,198,221)
12	(218,168,189)
13	(169,75,79)

sub-images	Average RGB value
14	(226,167,187)
15	(231,178,202)
16	(251,198,222)
17	(250,197,221)
18	(242,185,211)
19	(235,180,205)
20	(230,176,201)
21	(246,194,220)
22	(244,192,217)
23	(236,183,209)
24	(231,177,203)
25	(227,174,200)

Lastly, sorting $M(P_i)$ by increase order, and get sub-images sequence S_{P_1} as {*i*1=9, 1, 23, 4,18, 19, 2, 15, 24, 10, 3, 5, 22, 20, 7, 21, 25, 6, 17, 11, 16, 14, *i(mid)*= 8,

12, $i25$=13}, and use S_{P_1} to find out the index of image P_1's sub-image with the greatest majority value (i.e. $P_{1_{i25}}$=13) and find such sub-image's RGB value as (169,75,79); the index of sub-image with the least majority value (i.e. $P_{1_{i1}}$=9) and get the sub-image's RGB value as (241,18,209); the index of sub-image with the nearest average majority value (i.e. $P_{1_{i(mid)}}$=8) and find out the sub-image's RGB value as (231,166,185).

Table 2 Distance between each sub-image $D(P_{t_i}, P_{t_j})$ of image P_1

sub-image / distance / sub-image	…	8	9	…	12	13	14	…	23	24	25
1	…	36	3	…	38	189	36	…	8	17	23
2	…	39	5	…	41	191	39	…	11	20	26
3	…	42	9	…	44	195	42	…	13	23	29
:	:	:	:	…	:	:	:	…	:	:	:
12	…	52	19	…	0	152	8	…	31	21	15
13	…	14	36	…	152	0	153	…	182	172	167
14	…	153	186	…	8	153	0	…	29	20	15
:	:	:	:	…	:	:	:	…	:	:	:
23	…	43	10	…	31	182	29	…	0	10	16
24	…	30	6	…	21	172	20	…	10	0	6
25	…	21	15	…	15	167	15	…	16	6	0
$M(P_i)$, i=1,2,…,25	…	904	536	…	937	4383	885	…	539	581	662

3.2 Image similarity retrieval phase

When a user given a query image *Q*. Through actions like image partitioned, and sub-image sorting, and find the relegated information which will be calculated, and single out every image in the image index database and find query image *Q*'s

$D(Q_{i(n*m)}, P_{t_{i(n*m)}})$, $D(Q_{i1}, P_{t_{i1}})$, $D(Q_{i(mid)}, P_{t_{i(mid)}})$, with the aid of threshold r to make sequence correction.

When a representative sub-image of each image is selected, such as $P_{t_{i(n*m)}}$, as an attribute to proceed image ordered sequence, meanwhile the threshold r such that if $D(Q_{i(n*m)}, P_{t_{i(n*m)}}) < r$ then the difference between P_t and Q may not be distinguished properly. In other words, as to each image P_t of $D(Q_{i(n*m)}, P_{t_{i(n*m)}}) < r$, we think to sort sequence by using the greatest majority value, cannot determine the similarity with Q. Therefore, those images would alternatively select other representative sub-image, i.e. $P_{t_{i1}}$, to adjust the order of the images by $D(Q_{i1}, P_{t_{i1}})$. Like wisely, to obtain the final result of sequencing by using sub-images' $D(Q_{i(mid)}, P_{t_{i(mid)}})$ of $P_{t_{i(mid)}}$. The reason why use the sub-image with the greatest majority value is because it is the most representative single object of the images; and the sub-image with the least majority value is because it is the most related to the background. And, to use the sub-image with the nearest average majority value is because it locates in between background and the object itself. The process is illustrated as Fig. 5.

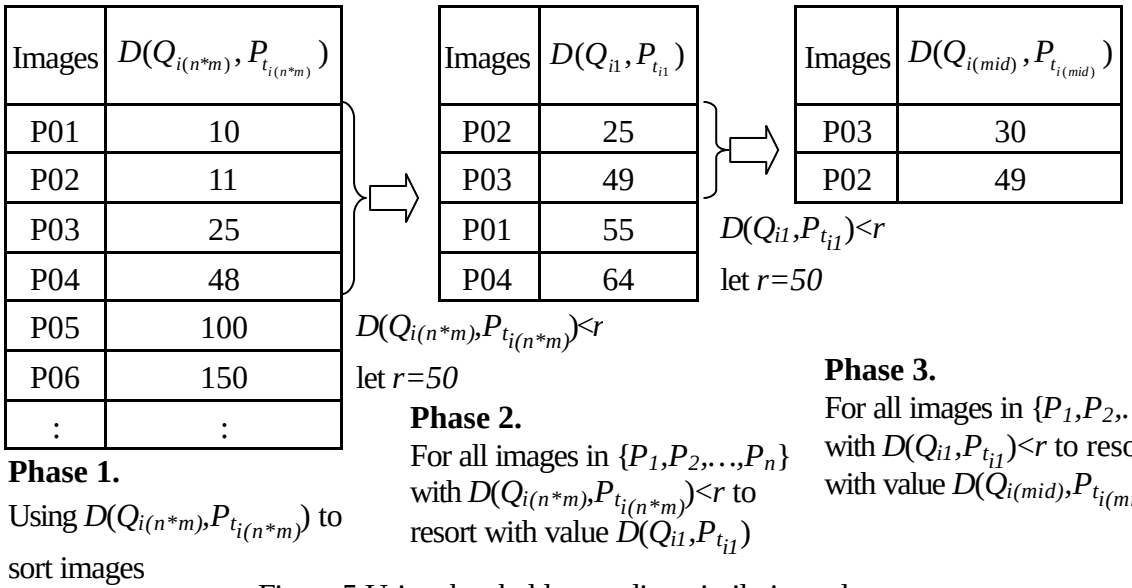


Figure 5 Using threshold r to adjust similarity order

As shown in Fig. 5, the sorting process has three phases. Phase 2 sequencing is conducted by using those images with $D(Q_{i(n*m)}, P_{t_{i(n*m)}})$ smaller than threshold r=50 found from Phase 1 sequencing. Similarly, the sequence in Phase 3 is obtained by using the images with $D(Q_{i1}, P_{t_{i1}})$ smaller than threshold r=50. After the three phases of sequencing, we can find more accurate and precise similarity sequence. The whole process is illustrated as follows.

Algorithm B (for image similarity retrieval phase):

Input: Q (Q as query image from user' s enquiry)

Output: Shows the images order after sequencing

Step 1: Performing Algorithm A for Q

Step 2: Calculate individually the following related distances of each sub-images,

$$D(Q_{i(n*m)}, P_{t_{i(n*m)}}),\ D(Q_{i1}, P_{t_{i1}}), D(Q_{i(mid)}, P_{t_{i(mid)}})$$

Step 3: Image Sequencing

Step 3.1: Sort $\{P_1,P_2,\ldots,P_k\}$ in accordance with value $D(Q_{i(n*m)}, P_{t_{i(n*m)}})$ of each P_t by increasing order, $\{t=1,2,...,k\}$.

Step 3.2: For all images in $\{P_1,P_2,\ldots,P_k\}$ with $D(Q_{i(n*m)}, P_{t_{i(n*m)}}) < r$ resort with value $D(Q_{i1}, P_{t_{i1}})$ by increasing order.

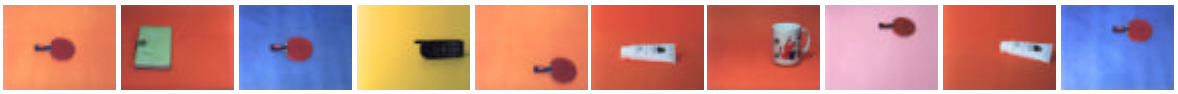
Step 3.3: For all images in $\{P_1,P_2,\ldots,P_k\}$ with $D(Q_{i1}, P_{t_{i1}}) < r$ resort with value $D(Q_{i(mid)}, P_{t_{i(mid)}})$ by increasing order.

Step 4: Output similar images after sequencing.

Example 3.2 illustrates clearly the process of Algorithm B.

Example 3.2 A simple image index database to explain Algorithm A and B

Let there be 10 pieces of images in the database.



P01 P02 P03 P04 P05 P06 P07 P08 P09 P10

Figure 6 Random similarity images

According to Algorithm A, we found RGB average value of $P_{t_{i25}}$, $P_{t_{i1}}$, $P_{t_{i13}}$ as shown in the following table.

Table 3:RGB average value of every sub-image

RGB value \ sub-images	$P_{t_{i25}}$ (the sub-image with the greatest majority value)	$P_{t_{i1}}$ (the sub-image with the least majority value)	$P_{t_{i(mid)}}$ (the sub-image with the nearest average majority value)
P01	(150,63,71)	(241,138,94)	(216,127,96)
P02	(120,169,133)	(195,78,63)	(156,90,77)
P03	(136,62,77)	(80,115,197)	(100,111,180)
P04	(35,32,31)	(211,178,77)	(144,128,64)
P05	(157,99,77)	(249,147,98)	(200,103,82)
P06	(183,154,160)	(183,154,160)	(190,81,61)
P07	(166,174,179)	(186,79,65)	(185,84,70)
P08	(145,79,85)	(230,174,201)	(203,152,174)
P09	(184,177,189)	(197,81,62)	(183,85,69)
P10	(141,57,70)	(93,132,228)	(90,128,238)

Set query image as Q (Fig. 4 (a)), within which Q_{i25}'s RGB average value as (169,75,79), Q_{i1}'s RGB average value as (241,187,209), and $Q_{i(mid)}$'s RGB average value as (231,166,185). The calculated distance, and sequenced (if r = 50) as shown in Fig. 7.

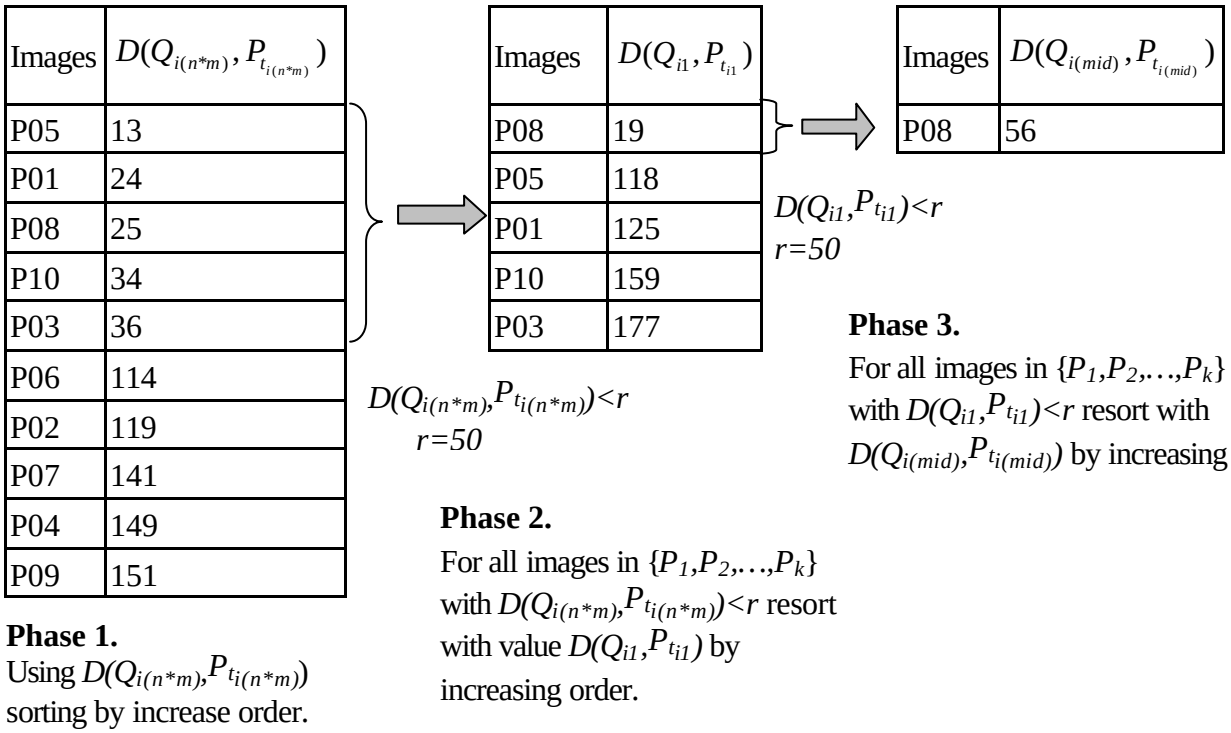
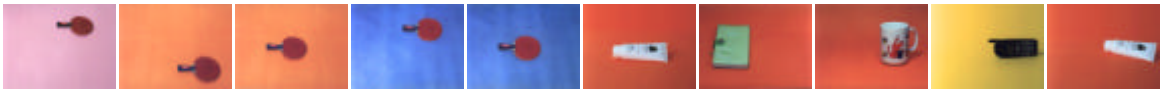


Figure 7 The three phases of sorting process of Example 3.2

Fig. 7 illustrates the usage of threshold r to proceeds sequencing correction's action and the variance. The sequenced result as followings:



P08 P05 P01 P10 P03 P06 P02 P07 P04 P09

Fig. 8: Order of the sequenced images

After the above mentioned processed, we could smoothly sequence images in the image index database, and output the result of similarity sequence.

4. Experimentation and result

In order to testify the conductibility of the proposed similarity sequencing technique, 36 color images with monotone background were used as experiment samples. Scanned and saved as 800*600 JPG files, the images were divided in five categories: Ping-Pong rackets, facial cleansers, cellular phones, notebooks, and mugs. Querying after the feature extraction that were saved into image index database. We

would measure the effectiveness of our experiment by identify rate and classification capability, while the Identification rate is defined as:

$$\text{Identification rate} = \frac{\text{Actual number of pieces at the front}}{\text{The number of pieces that should be at the}}$$

Whereas Classification capability is defined as:

$$\text{Classification capability} = \frac{\text{The number that belongs to such category}}{\text{First piece image to the last of classified images}}$$

The technique' s conductibility is testified by those two measurements. The result as follow:

Q1 Q2 Q3 Q4 Q5

Figure 9 Query images in the experiment

As shown from the lines in Fig. 10, when the partition as 5*5, and *r*=50, better result would be conducted. Whereas, average speaking, partitioned as 5*5, the Identification rate would reach 0.86, better than 3*3' s 0.65, 4*4' s 0.45, and 6*6' s 0.76, respectively.

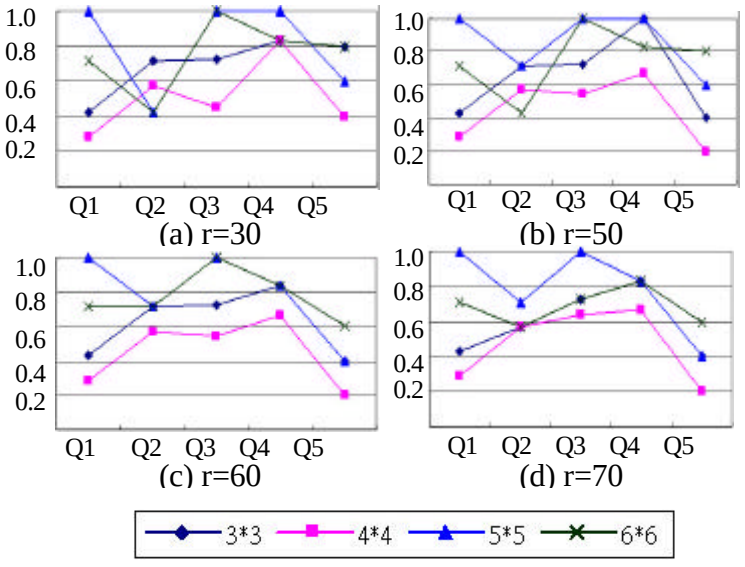


Figure 10. Identification rate hologram (a cross axle is queries; a vertical axle is Identify Rate)

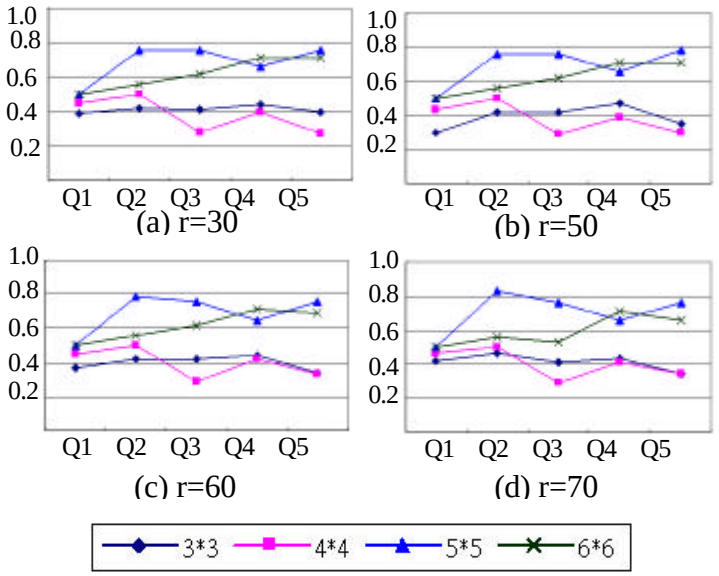


Figure 11. Classification capability hologram
(A cross axle is queries; a vertical axle is Classification capability)

The graph in Fig. 11 shows that, regardless of threshold *r*, partitioned at 5*5, the effect of Classification capability would be better than 3*3, 4*4, and 6*6. Average speaking, partitioned at 5*5, the Classification capability could reach as high as 0.69, much better then partition of 3*3' s 0.4, 4*4' s 0.39, and 6*6' s 0.61.

5. Discussions

Object rotation, shift and scaling are used to happen in image processing. In this section, some discussion about the impacts of our proposed method are made when object rotation, shift or scaling is considered.

Fig. 12 and 13 are examples of image rotation and shift as shown from (a) to (b). It is only that rotation shall make no difference in finding the major sub-images. But for the shift case, if image partition just put the object almost at one sub-image then affection is weaken. Other wisely, if object is put on many sub-images then the proposed method will be unable to find out the same major sub-images.

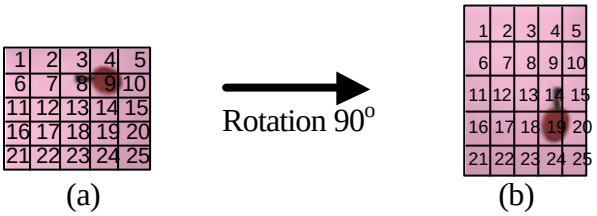


Figure 12. The example for image rotation 90°

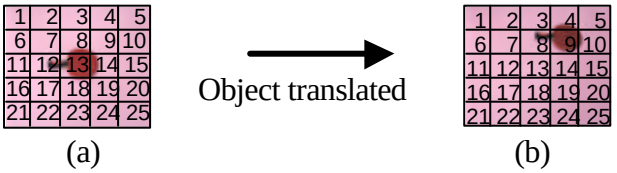


Figure 13. The example for image object translation

For the image scaling case, the scaled object on image must be put on many sub-images as the example show in fig. 14. Our method may fail to handle the object-scaling problem of an image. But if it is a scaling of whale image in spite of only the object in side it then the proposed method is still work on this case.

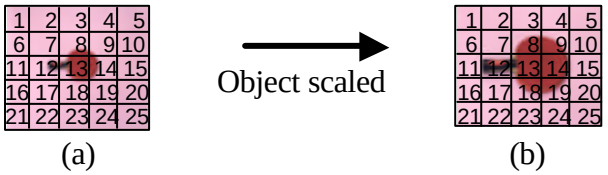


Figure 14. The example for image object translation

6. Conclusion

The usual color feature techniques lack of spatial resemblances. Thus, we refine this problem by image partition, and make color feature extraction much sensible. In order to find the identifiable feature metric, a designed majority function is defined to find the major sub-image, through this concept to distinct the similarity between images. Also, this technique makes classification of these images to be possibly by considering only the color attribute.

Moreover, threshold is used in coordination, in order to have a better and more precise sequence. The experiment shows, when an image is partitioned at 5*5 and the

threshold *r*=50, the Identify rate could reach 0.86. And when partitioned at 3*3, 4*4, 6*6, the rate results at 0.65, 0.45, and 0.76 respectively. It, therefore, asserts that the Identify rate can be comprised when the partitioned quantities are either too big, or too small. As to Classification capability, it also shows that partition at 5*5 would result an average value of 0.69, better than partition at 3*3, 4*4, and 6*6' s 0.4, 0.39, and 0.61 respectively. Nevertheless, due to some values appeared to be far to out of range, the average value of Classification capability was declining considerably. Yet, if exclude those fringe values in the calculation of the capability value, partition at 5*5 could reach 0.76 plus at Classification capability.

Reference:

[1] **Tolga Bozkaya and Meral Ozsoyoglu,** Indexing Large Metric Spaces for Similarity Search Queries, *ACM Transactions on Database Systems*, 24(3), 1999, pp. 361-404

[2] **Surajit Chaudhuri and Luis Gravano,** Optimizing Queries over Multimedia Repositories, *In 16th ACM Symposium on Principles of Database Systems*, 1997, pp. 91-102

[3] **M. Carey, L. Haas, P. Schwarz, M. Arya, W. Cody, R. Fagin, M. Flickner, A. Luniewski, W. Niblack, D. Petkovic, J. Thomas, J. Williams and E. Wimmers,** Towards heterogeneous multimedia information systems: The GARLIC approach. In 5th Intern. *Ws. On Research Issues in Data Engineering: Distr. Object Management*, 1995, pp. 124-131

[4] **G. Costagliola, F. Ferrucci, G. Tortora and M. Tucci,** Correspondences Non-redundant 2D strings, *IEEE Trans. Knowledge Data Engrg.* 7(2), 1995, pp. 345-350

[5] **Ulrich Guntzer, Wolf-Tilo Balke and Werner KieBling,** Optimizing Multi-Feature Queries for Image Databases, *Proceedings of the 26th VLDB Conference, Cairo, Egypt*, 2000, pp.419-428

[6] **D.J. Guan, Chun-Yen Chou and Chiou-Wei Chen,** Computational complexity of similarity retrieval in a pictorial database, *Information Processing Letters*, 75, 2000 pp.113-117

[7] **Ju-Hong Lee, Deok-Hwan Kim, Seok-Lyong Lee, Chin-Wan Chung and Guang-Ho Cha,** Distributed similarity search algorithm in distributed

heterogeneous multimedia databases, *Information Processing Letters*, 75, 2000 pp. 35-42

[8] **Chih-Chin Liu, Alien J.L. Hsu and Arbee L.P. Chen,** Efficient Near Neighbor Searching Using Multi-Indexes for Content-Based Multimedia Data Retrieval, *Multimedia Tools and Applications*, 13(3), 2001, pp. 235-254

[9] **Babu M. Mehtre, Mohan S. Kankanhalli, A. Desai Narasimhalu and Guo Chang Man,** Color Matching for image retrieval, *Pattern Recognition Letters*, 16, 1995, pp. 325-331

[10] **M. Ortega, K. Chakrababarti, K. Porkaew and S. Mehrotra,** Supporting Ranked Boolean Similarity Queries in MARS, *IEEE Trans. Knowledge Data Engrg*., 10(6), 1998, pp. 926-946

[11] **T. Seidl and H. Kriegel,** Optimal Multi-Step *k*-Nearest Neighbor Search, in: *Proc. ACM SIGMOD Internet. Conf. On Management of Data*, 1998, pp. 154-165

[12] **M. Tucci, G. Costagliola and S. K. Chang,** A remark on NP-completeness of picture matching, *Information Processing Letters*, 39, 1991, pp. 241-243.

[13] **V. V. Vinod and Hiroshi Murase,** Focused Color Intersection with Efficient Searching for Object Extraction, *pattern Recognition*, 30(10), 1997, pp. 1787-1797