A Color Ratio based Image Retrieval for e-Catalog Image Databases

Byoungkyu Lee and Yunmook Nah

Department of Computer Engineering, Dankook University
Seoul 140-714, Korea
{bklee, ymnah}@dankook.ac.kr

ABSTRACT

Most of the content-based image retrieval systems are based on RGB color space. Using the average amount of red, green, and blue is appropriate for natural images. But, for the case of e-catalog images, the number of colors for a given catalog image is only a few and the average color of an image is meaningless to customers. This paper presents a color comparison scheme based on the HSI color ratio to improve the accuracy of retrieval on e-catalog images. We have divided hue by 30 degrees, resulting in 12 colors. By considering saturation and intensity, and eliminating some duplicate combinations, we further divided each hue into 15 categories, thus resulting in 186 representative colors which is quite smaller than 1.7 million colors of 24 bit RGB case. The resulting 186-element HSI histogram is represented using presence bitmap vector(186 bit) and ratio vector(93 bytes). For the e-catalog images, most of the presence vector bits are 0 since there are only a few colors. We have implemented the prototype retrieval system and showed the usefulness of the proposed system by comparing measures such as precision and search speed with RGB histogram based schemes.

Keywords: image databases, color ratio, content-based retrieval, HSI histogram

1. INTRODUCTION

The amount of multimedia information on the internet is ever increasing rapidly. Most of the internet search engines are supporting only keyword based retrievals and do not support image content based retrievals directly. We are currently developing the agent-based 2D/3D image search engine which supports the content-based retrieval on the web images[1]. Some traditional image retrieval systems are based on descriptive text keywords, and others are based on the image features, such as color, texture, and shapes[3,8]. The color based retrieval systems usually use the average RGB color or RGB histograms[5]. For the case of e-catalog images, the average color of an image is meaningless to customers. The 256-element RGB histogram based retrieval is quite slow because of the dimensionality curses and the correctness degree is relatively low since there are many cases where similar colors are assigned quite different values especially for the boundary colors[7]. We have experimented the color content-based retrieval on e-catalog images using RGB based engines and found that the colors of the retrieved images were quite different from what users wanted.

This paper is an effort to improve the correctness of the content based retrievals on image colors, especially for the e-catalog images on the web. The major contribution of this paper is to show how to utilize HSI(Hue, Saturation, and Intensity) based web colors to realize more advanced web image search engines. The purpose of this paper is to show how to extract and represent image color features based on the 186-element HSI histograms and how effectively web catalog images can be retrieved.

The remainder of this paper is organized as follows. Section 2 describes basic concepts. Section 3 proposes the procedure to extract color ratio information and to represent the 186-element HSI histogram efficiently. In section 4, we explain the prototype implementation and compare our system with RGB based schemes using performance measures, such as precision and search speed. Finally, section 5 summarizes the paper.

2. BASIC CONCEPTS

2.1 Color class

HSI color space describes colors in terms of hue, saturation, and intensity(lightness) [4]. The Color Picker of Adobe Photoshop supports Only Web Colors, which are most stable colors on the web browsers. The Color Picker divides hue by 12 degrees, resulting in some duplicate colors. We divided hue by 30 degrees to eliminate duplicate colors without loss of original colors, thus resulting in 12 color angles. Each color angle is further divided into 15 categories, by considering saturation and intensity. There are 6 common colors(black, white, and 4 gray colors) to each 12 color angles. Therefore, there are $12 \times 15 + 6 = 186$ representative colors in total. In this paper, we call these 186 representative colors as color classes. Figure 1 shows the representative colors for one of 12 color angles. The white boxes represent 15 color categories to the given color angle. The 6 shaded boxes represent 6 common colors.

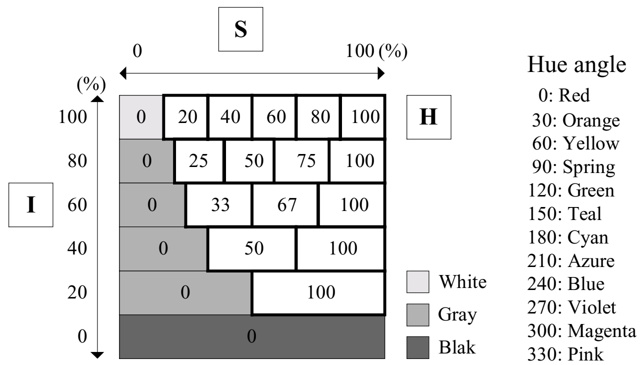


Figure 1. Representative colors for one of 12 color angles

We have assigned a unique number from 1 to 186 to each color classes. Figure 2 shows these numbers. The number of white color is 0, the number of black color is 1, and the number of color having HSI combination (330, 100, 20) is 186. These numbers are used to build HSI histograms.

1	2	3	4	5	6
white	black	gray1	gray2	gray3	gray4

Intensity : Saturation :	100 100	100 80	100 60	100 40	100 20	80 100	80 75	80 50	80 25	60 100	60 67	60 33	40 100	40 50	20 100
Hue (0) :	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
Hue (30) :	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36
Hue (60) :	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51
Hue (90) :	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66
Hue (120) :	67	68	69	70	71	72	73	74	75	76	77	78	79	80	81
Hue (150) :	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96
Hue (180) :	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111
Hue (210) :	112	113	114	115	116	117	118	119	120	121	122	123	124	125	126
Hue (240) :	127	128	129	130	131	132	133	134	135	136	137	138	139	140	141
Hue (270) :	142	143	144	145	146	147	148	149	150	151	152	153	154	155	156
Hue (300) :	157	158	159	160	161	162	163	164	165	166	167	168	169	170	171
Hue (330) :	172	173	174	175	176	177	178	179	180	181	182	183	184	185	186

Figure 2. Color numbers assigned to 186 color classes

2.2 Decision tree to decide color class

To build HSI histogram, it is necessary to decide the color class for a given pixel represented in HSI value (H, S, I). For this purpose, the decision tree is used (Figure 3).

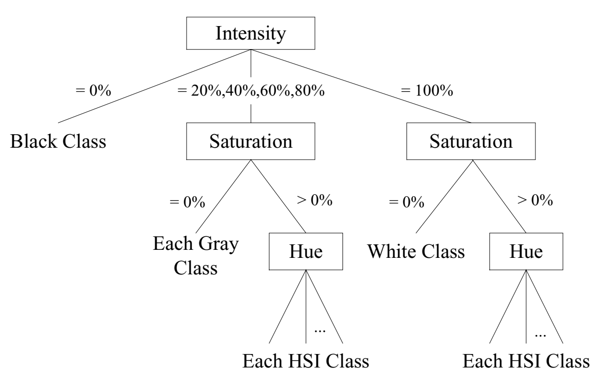


Figure 3. Decision Tree

3. COLOR RATIO BASED IMAGE RETRIEVAL

3.1 Overall procedure to extract color ratios

To extract color ratios from a given image, we have to extract the image information, analyze the image, and build the HSI histogram. The image information consists of image size information and color information. During the analysis process, each RGB pixel is converted into HSI pixel. After that, the color class for each HSI pixel is determined using the decision tree. The frequency number of each representative color is accumulated to derive HSI histogram. The detail steps are as follows.

Step 1. Convert source image files (in JPEG or GIF) into PPM raw images.
Step 2. Decide the background color for the given image. In our method, the RGB value of the first pixel from the raw image is considered as background color.
Step 3. For the next pixel having different RGB value with the background color, convert it into HSI pixel.
Step 4. Decide the appropriate color class using the decision tree and increment the frequency value of the color class.
Step 5. Repeat step 3 and 4 until there are no more RGB pixels.
Step 6. Compute the color ratio for each color class and store the result in the ratio vector. Indicate non-zero ratio as 1 in the presence bitmap vector.
Step 7. Classify the given image using the highest color ratio (most frequent color).

Figure 4 summarizes this overall procedures. The major algorithms are explained in the following sections.

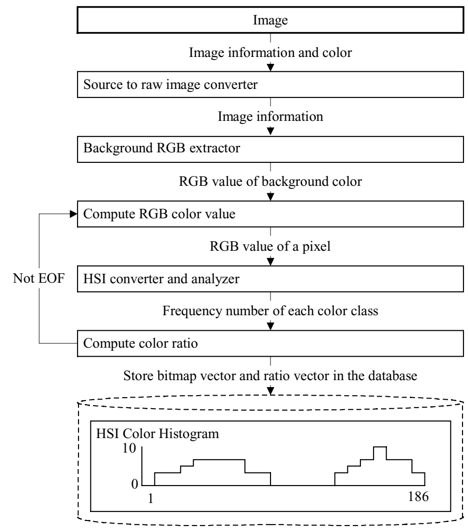


Figure 4. Overall procedure to extract color ratios

3.2 Conversion of RGB color into HSI color

Once the source image file is converted into raw image file, the image information is extracted using the algorithm of Figure 5.

```
For i=0 To 4
    For j=0 To 4
        ch=a character from input file
        If enter character or blank character Then
            go to outer loop
        Else
            str[i][j]=ch //store character in the array
        End If
    Next
Next
Image information   = str[0]
Image width         = str[1]
Image height        = str[2]
Image color         = str[3]
```

Figure 5. Extraction of image information

The e-catalog images usually have background colors and these background colors are meaningless for the retrieval purpose. If we add these background colors to the color ratio, the correctness degree of retrieval becomes low, thus these colors must be eliminated. Almost of all the e-catalog images have single background colors, and the first pixel of an image always belongs to the background color (Figure 6). The first byte of the first pixel is the background R value, the second byte is the background G value, and the third byte is the background B value.

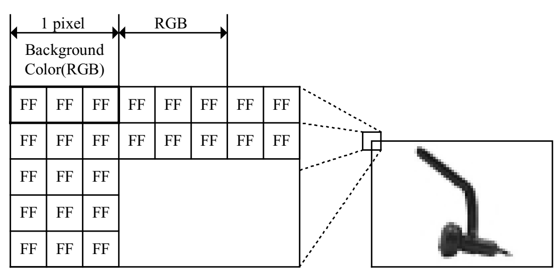


Figure 6. The location of image background pixel

Once the background color is determined, we can proceed to step 3 and step 4. Figure 7 shows the algorithm for these steps. The formula to convert RGB values into HSI values are so well known, thus these are not explained here.

```
For ch ≠ EOF
        r = R of file,    g = G of file,    b = B of file
        Increment total pixel count
        If background RGB ≠ next    RGB    Then
                Increment pixel count
                Convert to HSI
                Find the HSI color class
        End if
Next
```

Figure 7. Main procedure with background elimination

3.3 Deciding the color class

After the HSI values are determined, these values must be mapped into the representative color values. First of all, we find one of 12 color angles for the input H value (Figure 8a). The tolerance for hue angle is ±15°. If the H value is greater than 360, it is assumed as 0. After the color angle is determined, we find one of 6 intensity values(0%, 20%, 40%, 60%, 80%, 100%) for the input I value (Figure 8b). The tolerance for intensity is ±10%. Then, we find the appropriate saturation value, whose number is different according to the intensity value (Figure 8c). For example, when intensity is 100%, there are 6 classes with tolerance ±10. When intensity is 80%, there are 5 classes with tolerance ±12. Finally, we find the color class using the decision tree(Figure 3) and accumulate the frequency number of the color class. Figure 8d shows algorithmic version of the decision tree.

```
If (H+15) > 360    Then    // H greater than 360° is considered as 0°
        h=0
Else
        For c=1 To 12      // there are 12 color angles
                // classH holds color angle information
                If classH[c] ≤ (H+15)    Then // tolerance ±15°
                        h=classH[c]
                End If
        Next
End If
```

(a) Finding the color angle for the H value

```
For c=1 To 6 // there are 6 intensity values
    If classI[c] ≤ (I+10)    Then    // tolerance ±10%
        i=classI[c]
    End If
Next
```

(b) Finding the representative intensity for the I value

```
switch (i)    // number of saturation values is dependent on the intensity value
case 100 :    // there are 6 classes when intensity = 100%
        For c=0 To 5
                If type[c] ≤ (S+10)    Then // tolerance ±10%
                        s=type[c]
                End If
        Next
case 80   :   // there are 5 classes when intensity = 80%
        For c=6 To 10
                If type[c] ≤ (S+12)    Then // tolerance ±12%
                        s=type[c]
                End If
        Next
case 60   :   // there are 4 classes when intensity = 60%
        For c=11 To 14
                If type[c] ≤ (S+16)    Then // tolerance ±16%
                        s=type[c]
                End If
        Next
case 40   :   // there are 3 classes when intensity = 40%
        For c=15 To 17
                If type[c] ≤ (S+25)    Then // tolerance ±25%
                        s=type[c]
                End If
        Next
case 20   :   // there are 2 classes when intensity = 20%
        For c=18 To 19
                If type[c] ≤ (S+50)    Then // tolerance ±50%
                        s=type[c]
                End If
        Next
default   :   // when intensity = 0%
        s=type[20]
```

(c) Finding the representative saturation for the S value

```
If i = 0    Then // black if intensity is 0
            Increment the frequency number of black color
Else If i = 100    Then      // if intensity is 100, there is a possibility of white
            If s = 0    Then    // white if saturation is 0
                        Increment the frequency number of white color
            Else
                        Increment the frequency number of appropriate color class
            End If
Else    // saturation depends on intensity values
            If s = 0    Then    // one of 4 gray colors
                        switch (i)
                        case 80 :     increment the number of gray 1
                        case 60 :     increment the number of gray 2
                        case 40 :     increment the number of gray 3
                        default :     increment the number of gray 4
            Else
                        Increment the frequency number of appropriate color class
            End If
End If
```

(d) Accumulating the frequency number of the color class using the decision tree

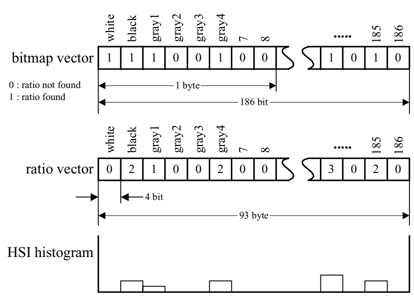
Figure 8. Decision of color class for HSI values

3.4 Building the ratio vector and presence bitmap vector

After all the frequency numbers of each color classes are determined for the given image, we compute the color ratio for each color class and store the result in the ratio vector and the presence bitmap vector. Building the ratio vector is trivial. The individual bit of the presence bitmap vector is set to 1 if the value of the corresponding position of the ratio vector is non-zero (Figure 9).

```
ratio_vector[i] = ratio of each color class
If ratio_vector[i] Then
            bitmap_vector[i]=1
Else
            bitmap_vector[i]=0
End If
```

Figure 9. Presence bitmap vector generation



The value of color ratio is between 0 and 100. To represent these values, we need 7 bits for each color ratio values, thus resulting in 186 × 7 = 1,302 bits = 163 bytes. If we represent 100% as 10 and 94% as 9, it is possible to use 4 bits for each ratio values, thus resulting in 186 × 4 = 744 bits = 93 bytes. Of course, the correctness degree has to be sacrificed to save the disk space. The final HSI histogram consists of one presence bitmap vector(186 bits) and one ratio vector(93 or 163 bytes). Figure 10 shows the color ratio based HSI histogram.

Figure 10. Color ratio based HSI histogram

3.5 Database schema

After the completion of all the processing steps, all the results are stored in the database and used for the color ratio based image retrieval. Figure 11 shows the corresponding database schema. For a given image, the unique identifier, the classification class (the hue value of the highest ratio color), the color class number of the most frequent color, the bitmap vector, and the ratio vector are stored in the table.

Column name	Data type	Length	Content
Image_id	Varchar	10	
Classification	Int	2	(0, 30, …, 330)
Class_number	Char	1	0 ~ 186
Bitmap	Char	24	186 bit
Ratio	Char	93	744 bit

Figure 11. Database schema

3.6 Illustrative example

Figure 12 shows an example showing all the intermediate results of the proposed procedure. We can easily observe that almost all the valid colors are located near the most frequent color. The number of total pixels was 28,000 and the number of valid pixels after background elimination reduced to 17,226. We can provide higher retrieval correctness with less pixels. The final result(Figure 12f) shows that the dominant color of this image is red(hue 0; the value of the classification class) and the HSI value of the dominant color is HSI 0, 100, 20(the color class number 21). The classification class values, bitmap vectors and ratio vectors are used for color ratio based retrieval operations. The classification class values are also utilized for image classification and user access pattern mining in the agent based image search engine[1].

(a) Sample image

Image information:	a002.jpg Image						[width : 200 * length : 140]								
Intensity:	100	100	100	100	100	80	80	80	80	60	60	60	40	40	20
Saturation:	100	80	60	40	20	100	75	50	25	100	67	33	100	50	100
Hue (0: Red):	0	0	0	0	0	0	0	0	58	0	14	674	4572	956	6668
Hue (30: Orange):	0	0	0	0	1	0	0	0	14	0	0	27	0	30	11
Hue (60: Yellow):	0	0	0	0	1	0	0	0	5	0	0	12	0	13	3
Hue (90: Spring):	0	0	0	0	0	0	0	0	0	0	0	0	0	4	4
Hue (120: Green):	0	0	0	0	0	0	0	0	0	0	0	3	0	7	7
Hue (150: Teal):	0	0	0	0	0	0	0	0	1	0	0	0	0	0	14
Hue (180: Cyan):	0	0	0	0	6	0	0	0	11	0	2	35	2	71	55
Hue (210: Azure):	0	0	0	0	0	0	0	0	3	0	0	2	0	3	7
Hue (240: Blue):	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Hue (270: Violet):	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Hue (300: Magenta):	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Hue (330: Pink):	0	0	0	0	0	0	0	0	1	0	0	2	127	6	88
Other color:	White: 1956 black: 478 gray1: 274 gray2: 219 gray3: 234 gray4: 545														
Report:	Number of pixel without background: 17226								number of sum pixel: 28000						

(b) Frequency values of each pixel (result of step 3, 4, and 5)

Intensity:	100	100	100	100	100	80	80	80	80	60	60	60	40	40	20
Saturation:	100	80	60	40	20	100	75	50	25	100	67	33	100	50	100
Hue (0: Red):	0	0	0	0	0	0	0	0	0	0	0	4	27	6	39
Hue (30: Orange):	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
Hue (60: Yellow):	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
Hue (90: Spring):	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Hue (120: Green):	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Hue (150: Teal):	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Hue (180: Cyan):	0	0	0	0	6	0	0	0	0	0	0	0	0	0	0
Hue (210: Azure):	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Hue (240: Blue):	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Hue (270: Violet):	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Hue (300: Magenta):	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Hue (330: Pink):	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1
Other color:	White: 11 black: 2 gray1: 1 gray2: 1 gray3: 1 gray4: 3														

(c) Ratio vector (163 bytes)

Intensity:	100	100	100	100	100	80	80	80	80	60	60	60	40	40	20
Saturation:	100	80	60	40	20	100	75	50	25	100	67	33	100	50	100
Hue (0: Red):	0	0	0	0	0	0	0	0	0	0	0	0	3	1	4
Hue (30: Orange):	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
Hue (60: Yellow):	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
Hue (90: Spring):	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Hue (120: Green):	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Hue (150: Teal):	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Hue (180: Cyan):	0	0	0	0	6	0	0	0	0	0	0	0	0	0	0
Hue (210: Azure):	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Hue (240: Blue):	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Hue (270: Violet):	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Hue (300: Magenta):	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Hue (330: Pink):	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

==

Other color: White: 1 black: 2 gray1: 0 gray2: 0 gray3: 0 gray4: 0

==

(d) Ratio vector (93 bytes)

```
1000000000000000000111000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000
```

(e) Bitmap vector (186 bits)

```
      Image_ID:   A002
Classification:   0
  Class_number:   21
        Bitmap:   80003800000000000000000000000000000000000000000
         Ratio:   00000000000000000314000000000000000000000000000000000000000000000000
                  00000000000000000000000000000000000000000000000000000000000000000000
                  00000000000000000000000000000000000000000000000000
```

(g) Final results in the database

Figure 12. An example

3.7 Color ratio based retrieval

There are two query interfaces for the color ratio based image retrieval. One is using RGB ratio input form and another is using query images. Queries such as 'find images with 60% red and 0% yellow' can be presented through the RGB ratio input form. Also, users can upload the query image directly. The query processing steps are as follows.

Step 1. Analyze the input RGB ratio values or the query image
Step 2. Extract the dominant color having the highest color ratio
Step 3. For the database images belonging to the same classification class with the query image, compare the presence bitmap vectors.
Step 4. For the matched images in step 3, compare the ratio vectors.

In step 2 and step 3, we utilize the dominant color to reduce search ranges. Only for the images within the same classification class, we do apply detail comparison processes based on presence bitmap vectors and ratio vectors.

4. EXPERIMENTAL RESULTS

The prototype system was developed on the Pentium-II 300, Windows 2000 platform using SQL Server 7.0. The search module was implemented by CGI programming using C language. We have used 400 e-catalog images. We have experimented the color content-based retrieval on e-catalog images using the average RGB color and found that the colors of the retrieved images were quite different from what users wanted. Figure 13 shows sample query results for a given query image by using the proposed method and the average RGB color based method.

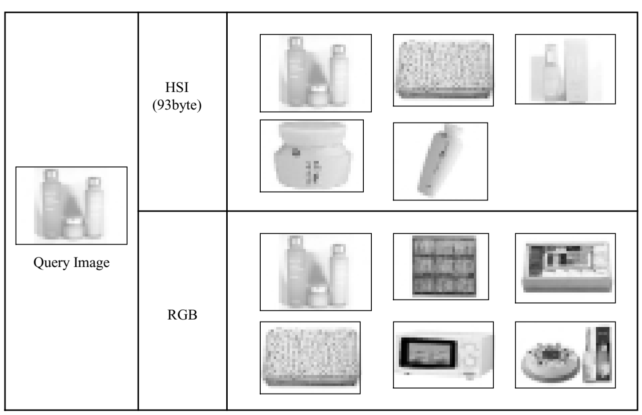
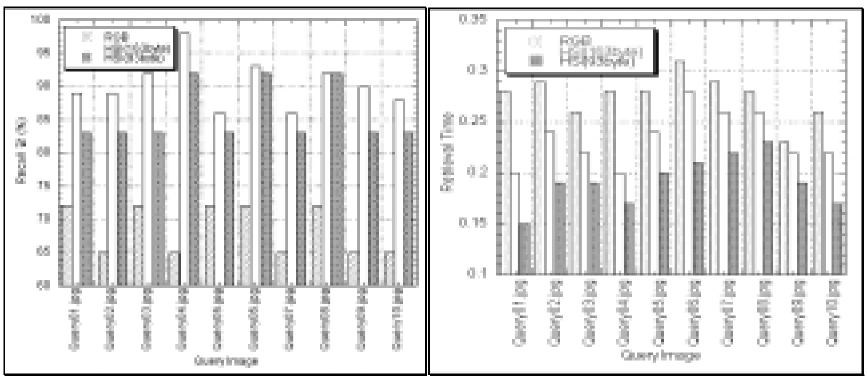


Figure 13. Sample query results for a given query image

Figure 14 shows the performance comparison results between RGB based schemes and HSI based schemes. The HSI method with 163 bytes ratio vector showed the best performance with more storage overhead. Also, the performance of HSI method with 93 bytes ratio vector was quite better than the RGB method in recall and retrieval time.

Query image	Recall			Retrieval Time		
	RGB	HSI (163byte)	HSI (93byte)	RGB	HSI (163byte)	HSI (93byte)
Query01.jpg	72%	83%	83%	0.28ms	0.20ms	0.15ms
Query02.jpg	65%	89%	83%	0.29ms	0.24ms	0.19ms
Query03.jpg	72%	92%	83%	0.26ms	0.22ms	0.19ms
Query04.jpg	65%	98%	92%	0.28ms	0.20ms	0.17ms
Query05.jpg	72%	86%	83%	0.28ms	0.24ms	0.20ms
Query06.jpg	72%	93%	92%	0.31ms	0.28ms	0.21ms
Query07.jpg	65%	86%	83%	0.29ms	0.26ms	0.22ms
Query08.jpg	72%	92%	92%	0.28ms	0.26ms	0.23ms
Query09.jpg	65%	90%	83%	0.23ms	0.22ms	0.19ms
Query10.jpg	65%	88%	83%	0.26ms	0.22ms	0.17ms
The average of Recall and Time(10Queries)	68.5%	90.3%	85.7%	0.28ms	0.23ms	0.19ms

(a) Performance comparison on 10 sample images



(b) Comparison of recall (c) Comparison of retrieval time

Figure 14. Performance comparison of RGB and HSI

5. CONCLUSIONS

In this paper, we presented a color ratio based retrieval scheme on web e-catalog images. We utilized HSI color space which is used to represent web colors in the Color Picker of Adobe Photoshop. We divided hue by 30 degrees, resulting in 12 colors. By considering saturation and intensity, and eliminating some duplicate combinations, we further divided each hue into 15 categories, thus resulting in 186 representative colors.

We proposed an algorithm to build 186-element HSI histograms. The resulting HSI histogram is represented using presence bitmap vector(186 bit) and ratio vector(93 bytes or 163 bytes). For the e-catalog images, most of the presence vector bits are 0 since there are only a few colors. We also proposed how to utilize these vectors to search web images. We implemented the prototype system using hundreds of e-catalog sample images. The proposed system will contribute to the development of next generation internet search engines, which support content based retrievals on web images.

There should be further research on the compression schemes of presence bitmap vectors and ratio vectors and more intelligent classification methods of web images.

ACKNOWLEDGEMENTS

This work was funded by KOSEF(Korea Science and Engineering Foundation) under contract no. 98-0102-06-01-3.

REFERENCES

1. Myungsun Park, Sukho Lee, Yunmook Nah, and Soochan Hwang, “A Development of the Agent-based 2D/3D Image Search Engine supporting Intelligent Content-based Retrieval”, Database Research, KISS SIGDB, Sept. 1999, pp. 21-27.
2. Myungsun Park, *A Design and Implementation of WWW Image Search Engine*, Master Thesis, Seoul National University, 1997.
3. Alberto Del Bimbo, *Visual Information Retrieval*, Mogan Kaufmann, 1999, pp.81-113.
4. Rafael C. Gonzalez, Richard E. Woods, *Digital Image Processing*, Addison Wesley Longman, 1998, pp.226-252.
5. M. Flickner, et al., "Query By Image and Video Content : The QBIC System," IEEE Computer, Vol. 28, No. 9, IEEE, Sept. 1995, pp.23-32.
6. Niblack, W., R. Barber, W. Equitz, M. Flickner, E. Glassman, D. Petkovic, and P. Yanker, "The QBIC project: Querying images by content using colour, texture and shape," *SPIE 1908: Storage and Retrieval for Image and Video Databases*, February 1993.
7. Hafner, J., H.S. Sawhney, W. Equit, M. Flicker, and W. Niblack, "Efficient colour histogram indexing for quadratic form distance functions," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 17(7), 1995, pp.729-736.
8. Smith, J.R. and S.F. Chang, *Automated Image Retrieval Using Colour and Texture*, Technical Report No. 4-95-20, Columbia University, July 1995.
9. Corridoni, J.M., A. Del Bimbo, and P. Pala, "Image retrieval by colour semantics," *ACM Multimedia System Journal,* 1998.
10. Pass, G., R.Zabih and J. Miller, "Comparing images using colour coherence vectors," in *Proc. ACM Conf. on Multimedia*, Boston, 1996, pp.65-73.
11. Jain, A.K, *Fundamentals of Digital Image Processing*, Prentice-Hall, 1989.
12. Vellaikal, A. and C.C.J. Kuo, "Content-based retrieval using multiresolution histogram representation," *Digital Image Store Archiving Systems*, 2602, October 1995, pp.312-323.
13. Funt, B.V. and G.D. Finlayson, "Colour constant colour indexing," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 17(5), 1995.
14. Corridoni, J.M., A. Del Bimbo and S. De Magistris, "Querying and retrieving pictorial data using segmentics induced by colour quality and arrangement," in *Proc. IEEE Int. Conf. On Multimedia and Computing Systems ICMCS'96*, Hiroshima, June 1996, pp.219-222.
15. V. E. Ogle and M. Stonebraker, "Chabot: Retrieval from a Relational Database of Images," *IEEE Computer*, Vol.28, No.9, Sept. 1955, pp. 40-48.