CSIT 691A Independent Project

Content-Based Image Retrieval

Student: XIE Ning
E-mail: nxie@ust.hk
Supervisor: Prof. David Rossiter

Table of Contents

1 Introduction

To focus on content-based image retrieval, this independent project involves the development of a software system. There are several aspects to be considered. The first step is feature extraction. A user provides one exact image, and the system analyses it, comparing either the colour or the shape. Then, the system measures similarity. Finally, the system searches through the database to find similar answers. When a user uploads an image to search the similar ones instead of offering some words, the system returns similar images based on edge or colour, which involves two methods, RGB and HSV. Through comparing the results among these three methods, it is easy to observe in which circumstances people can best get what they want. Furthermore, the system is also able to combine the three methods so when retrieving an image the users could choose which percentage of each method to use.

This report introduces the basic methods to analyse an image. Section 2 mentions how to use the RGB method, which is easy to handle by computer. Section 3 introduces the HSV method, which is designed according to customers' concept on images. Section 4 involves another aspect of analysing an image, which is the Edge method. In section 5, we explain how the three methods are combined. Section 6 shows the software system design, and how it works. Section 7 provides a conclusion based on results, through comparing the four methods, that is, RGB, HSV, Edge, and a combination of the three methods.

2 The RGB method

RGB adheres to the concept that any colour consists of red, green, and blue components ([1]). So we define a colour with three dimensional vector (a, b, c), where a, b, and c are positive integers, and 0 <= a, b, c <= 255. In the software system, we simplify all colours, about 256*256*256, to 27 kinds of colour (see in table 1). Figure 1 shows the relationship between RGB and some colours used in the system.

Figure 1 Example of the RGB concept

i	RGB
1	(0, 0, 0) (black)
2	(0, 0, 127)
3	(0, 0, 255) (blue)
4	(0, 127, 0)
5	(0, 127, 127)
6	(0, 127, 255)
7	(0, 255, 0) (green)
8	(0, 255, 127)
9	(0, 255, 255) (cyan)
10	(127, 0, 0)
11	(127, 0, 127)
12	(127, 0, 255)
13	(127, 127, 0)
14	(127, 127, 127)
15	(127, 127, 255)
16	(127, 255, 0)
17	(127, 255, 127)
18	(127, 255, 255)
19	(255, 0, 0) (red)
20	(255, 0, 127)
21	(255, 0, 255) (magenta)
22	(255, 127, 0)
23	(255, 127, 127)
24	(255, 127, 255)
25	(255, 255, 0) (yellow)
26	(255, 255, 127)
27	(255, 255, 255) (white)

Table 1 The 27 kinds of colour used in the system

When the system gets the RGB of one pixel from an image, it calculates the distances between this RGB (R_k, G_k, B_k) and the RGB_i (R_i, G_i, B_i) above, through using the following equation. $Distance[i] = (R_k-R_i)^2 + (G_k-G_i)^2 + (B_k-B_i)^2$, for every i in the range [1, 27]. If distance[i] is the smallest one, we take this RGB as RGB_i. The system counts the times that RGB_i occurs by adding 1. Then we calculate the percentage of each RGB taken in the whole image. So every value is in the scale [0, 100], and will be recorded.

When the system starts, it calculates the percentage of the 27 kinds of colour in each image in the database and records them in a file. After the user uploads an image, the system does the algorithm,

shown in table 2, to calculate the similarity between the uploaded image and each image in the database.

Assume Pcur stores a percentage of each of 27 kinds of colour in the image uploaded, and P stores a percentage of each of 27 kinds of colour in each image in the database.

for i *from* 0 *to* the number of images in the database *by* i++
  for j *from* 0 *to* the number of kinds of colour *by* j++
  similarity[i] += |Pcur[i]-P[i][j]|;
  similarity[i] /= the number of kinds of colour;

Table 2 Pseudo-code for the RGB algorithm

Through the above steps, the system could get all similarities. It could then reorder the images in ascending order, for the user to see the most similar images from the database.

3 The HSV method

Even though RGB is easy for a computer to understand and calculate by algorithm, it does not efficiently express human perception of the real colour world. To improve that, HSV, which stands for hue, saturation, and value, considers what colour is like in the real world ([2]). So, in the cylinder of HSV, the angle around the central vertical axis corresponds to "hue" (H), the distance from the centre of the axis corresponds to "saturation" (S), and the distance along the other axis corresponds to "value" (V). This is shown in figure 2.

According to the concept of HSV, it is easy to convert RGB to HSV, as follows. Assume M = max (R, G, B), m = min (R, G, B), and delta = M-m.

$$H' = \begin{cases} \text{undefined}, & \text{if delta} = 0 \text{ or } M = 0; \\ (|G-B| / \text{delta}) \bmod 6, & \text{if } M = R; \\ |B-R| / \text{delta} + 2, & \text{if } M = G; \\ |R-G| / \text{delta} + 4, & \text{if } M = B. \end{cases}$$

$$H = 60 * H'.$$

$$S = \begin{cases} 0 & \text{if delta} = 0; \\ \text{delta} / M & \text{otherwise.} \end{cases}$$

$$V = M / 255.$$

In the HSV method, the system needs to analyse an image through three different parts. First, to calculate the hue of an image, the system simplifies the value of "hue" into 36 kinds of colour. This is done through the way that the system separates a 360^{o} circle into ten parts. Then we count the times each hue occurs, and convert it to the percentage of the whole image. Second, for each hue value, we also simplify the value of "saturation" from 0 to 10 (they are all positive integers), and count the times each value happens. In addition, convert each number into the percentage of each hue value. Third, for each hue value, we do the same steps to calculate the value of "value". All this information will be recorded in a file. Notice that the values of "hue", "saturation", and "value" are all in the scale [0, 100].

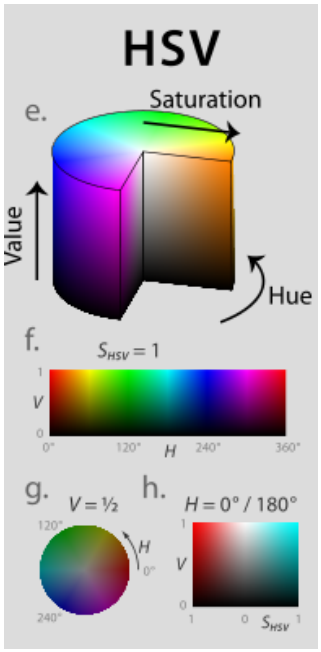

Figure 2 The concept of HSV

When the system starts, it calculates the information mentioned above for each image in the database, and records it in a file. After a user uploads an image, the system compares HSV between this image and each image in the database. The pseudo-code this method uses is described in table 3.

Assume that Hcur, Scur, and Vcur are to store the H, S, and V values of the image uploaded, and H, S, V are to store the H, S, and V values of images in the database.

for i *from* 0 *to* the number of images in the database *by* i++
  for j *from* 0 *to* the number of hue value *by* j++
  Hsum[j] = |Hcur[j] – H[i][j]|;
  for k *from* 0 *to* the number of S value *by* k++
  Ssum[j] += |Scur[j][k] – S[j][k]|;
  Ssum[j] /= the number of S value;
  for k *from* 0 *to* the number of S value *by* k++
  Vsum[j] += |Vcur[j][k] – V[j][k]|;
  Vsum /= the number of V value;
  similarity[i] += Hsum[j] + Ssum[j] + Vsum[j];
  similarity[i] /= 3;

Table 3 Pseudo-code for the HSV algorithm

After the above steps, the system can obtain similarities between the uploaded image and each image in the database. Then the system will reorder the images in ascending order, for the user to see the most similar images from the database.

4 The Edge method

The two methods mentioned above are both based on the analysis of colour to retrieve images. The edge method does not consider what the colour is. This method is based on the Sobel operator,

which is described below in figure 3 ([3]).

$$\mathbf{G}_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix} * \mathbf{A} \quad \text{and} \quad \mathbf{G}_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} * \mathbf{A} \qquad \mathbf{G} = \sqrt{\mathbf{G}_x^{\,2} + \mathbf{G}_y^{\,2}}$$

where * here denotes the 2-dimensional convolution operation.

Figure 3 Computation of Sobel operators

Hence, for every pixel in the picture, the system calculates its G_y and G_x. Notice that A shown in figure 3 is the related RGB matrix. According to the computation,

$G_y = -1*RGB_{(x-1, y-1)} - 2*RGB_{(x, y-1)} - 1*RGB_{(x+1, y-1)} + 1*RGB_{(x-1, y+1)} + 2*RGB_{(x, y+1)} + 1*RGB_{(x+1, y+1)}$;

$G_x = -1*RGB_{(x-1, y-1)} - 2*RGB_{(x-1, y)} - 1*RGB_{(x-1, y+1)} + 1*RGB_{(x+1, y-1)} + 2*RGB_{(x+1, y)} + 1*RGB_{(x+1, y+1)}$,

where $RGB_{(x, y)}$ means the RGB of one pixel at (x, y). These three equations, G_x, G_y, and hence G, are calculated three times for red, green, and blue components separately. We choose the average G of the red's G, the green's G, and the blue's G as the result for one pixel. Because the system divides the whole image into 16 parts, for each part, the system counts the total sum of G, by adding every pixel's average G. Then convert each total sum into the percentage of the whole image's G.

When the system starts, it will calculate each part's total sum of G in a whole image (notice that this will happen 16 times because we take the whole image as 16 parts), and record the information of every image in the database in a file. After a user uploads an image, the system does the following algorithm in table 4. Notice that the system calculates the percentage. It is in the scale [0, 100].

Assume that Changecur is changes that the image uploaded occurs, and Change is changes that each image in the database happens.

for i *from* 0 *to* the number of images in the database *by* i++
  for j *from* 0 *to* the number of parts divides the whole image *by* j++
  similarity[i] += |Changecur[j] – Change[i][j]|;
  similarity[i] /= the number of parts divides the whole image;

Table 4 Pseudo-code for the Edge algorithm

Using the above algorithm, the system also gets similarities between the uploaded image and each image in the database. Then the system will reorder the images in ascending order, for the user to see the most similar images from the database.

5 A combination of the three methods

So far, the system has three different methods to analyse an image. As a result, the system could use a combination of these three methods. The system takes them as three vertexes of a triangle. A user could input each weight related to each method in order to measure the similarity by using one number. This is shown in figure 4.

When the system starts, it calculates each image's information mentioned above with these three methods, and records them separately in three different files. A user uploads an image first, and then this user sets the weights of each of the three methods. For instance, the user wishes that the HSV method takes the main weight, and the Edge method takes a little weight. Then the user would set weight (HSV) = 90, weight (Edge) = 10, and weight (RGB) = 0. The system uses weight (HSV) *

the number obtained by HSV method + weight (Edge) * the number obtained by Edge method + weight (RGB) * the number obtained by RGB method for each image in the database. Notice that the number obtained by each method is the similarity between the uploaded image and each image in the database.

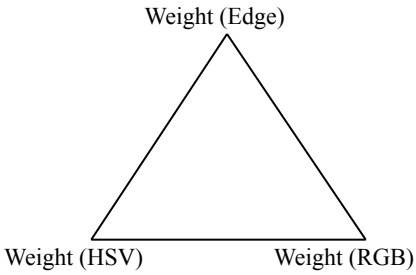

Figure 4 A triangle combined of the three methods

Hence, the system gets similarities, by using the equation mentioned above, between the uploaded image and each image in the database. Then the system will reorder the images in ascending order, for the user to see the most similar images from the database.

6 Implementation and example usage

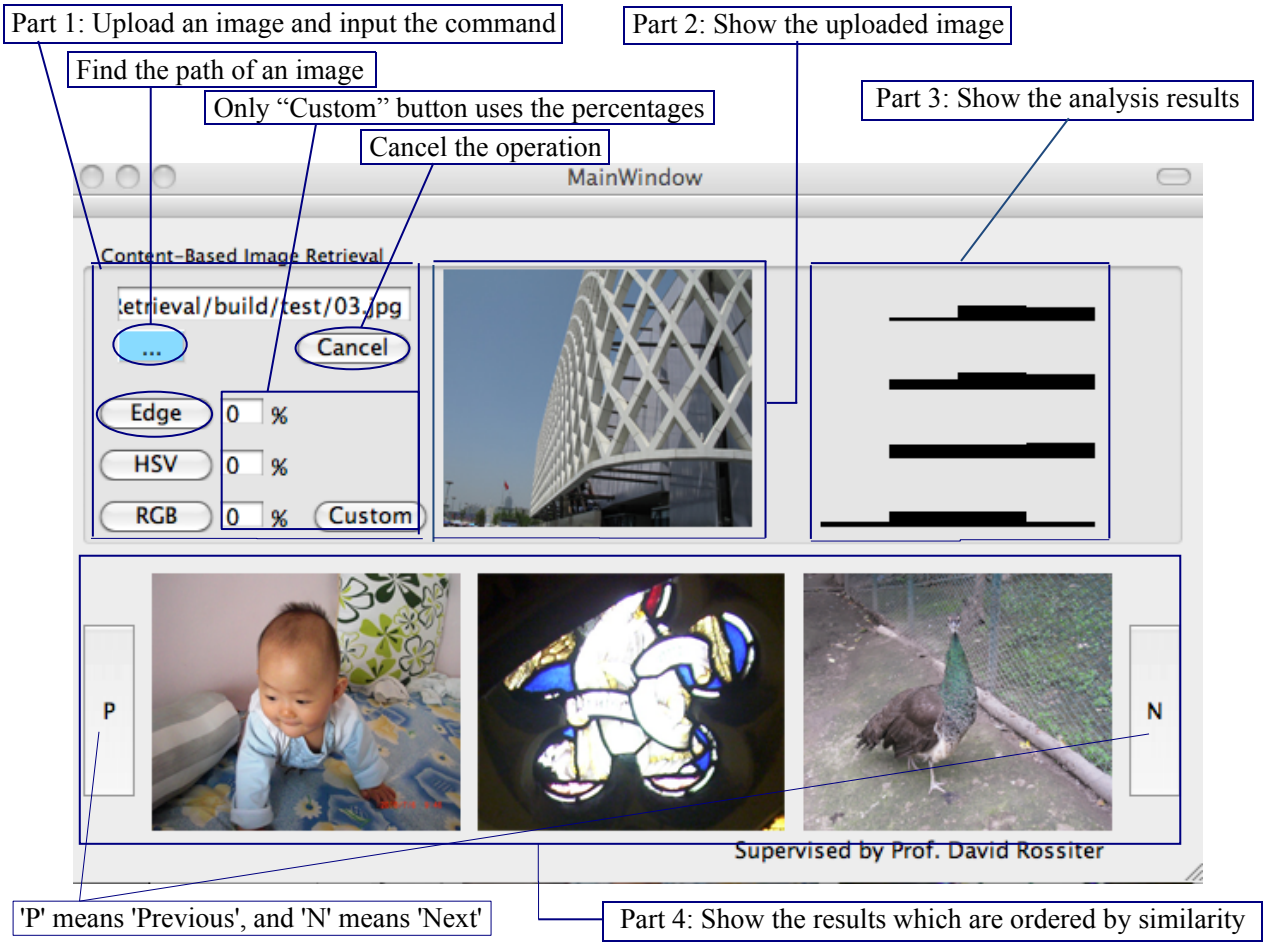

Figure 5 Example of the system using the Edge method

The system is coded using C++ and Qt4 ([4]), and the development environment is the Macintosh 10.6.4. Because Qt is a cross-platform application framework, the system still could work on any PC with Windows or Linux. However, it needs the qt environment.

The system is developed into four parts, and shown in figure 5. If a user does not want to use the "Custom" button, which is the combination of the three methods, the user does not need to set each percentage. In part 4 of the GUI, the system uses the number 20 as a threshold, and the system only returns the images whose similarities are lower than the threshold.

When the user wants to operate the system, this user follows some steps. First, the user clicks the button "..." or input the path of a image, in part 1of the GUI , to upload an image, and this image will show in part 2 of the GUI . Then the user chooses the method, which is listed in part 1 of the GUI, that he or she wants to use to search similar images. There are four methods provided, including the three mentioned above, and one combination of the three methods. After the user clicks a button which is related to the method, the analysis results will show in part 3 of the GUI, and, at the same time, the search results show in part 4 of the GUI.

Here we give an example, using one image. It shows how the system works.

Use RGB, and notice that part 3 of the GUI shows the percentages of 27 colours in the uploaded image. If there is no colour showed in the related area, it means this colour is not in this image. This is shown in figure 6.

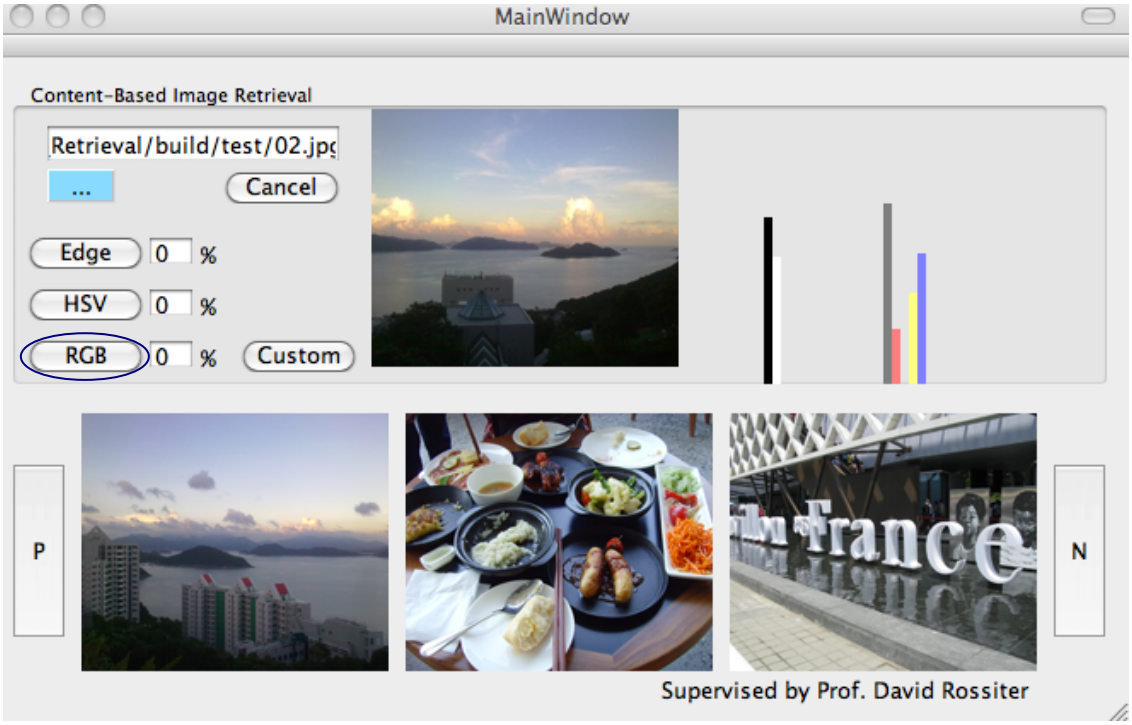

Figure 6 Example using RGB

Use HSV, and notice that part 3 of the GUI simply shows that values of H, S, and V cover. This is shown in figure 7.

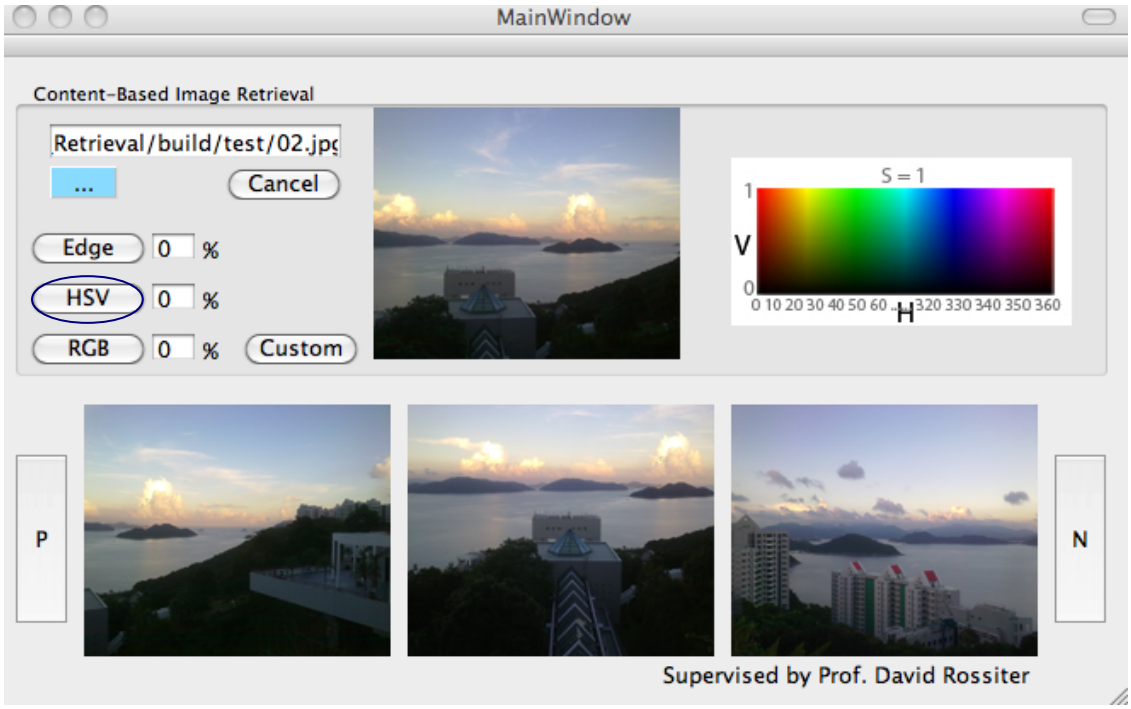

Figure 7 Example using HSV

Use Edge, and notice that part 3 of the GUI shows the percentages of change in each divided part of the whole uploaded image. This is shown in figure 8.

Figure 8 Example using Edge

Use Custom, and notice that the user chooses 10 percent of Edge, 80 percent of HSV and 10 percent of RGB. Part 3 of the GUI shows three methods take how much percentage in the triangle. This is shown in figure 9.

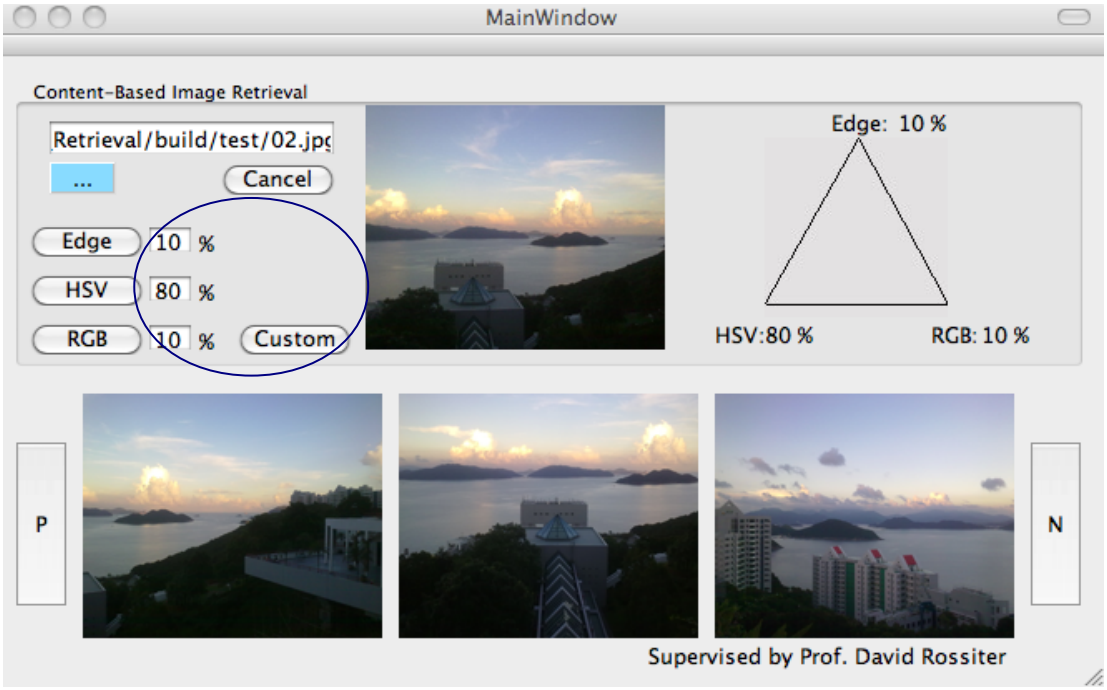

Figure 9 Example using Custom

Use Cancel. The system cancels all operations and goes back to the initial state, and part 4 of the GUI shows all images from database. This is shown in figure 10.

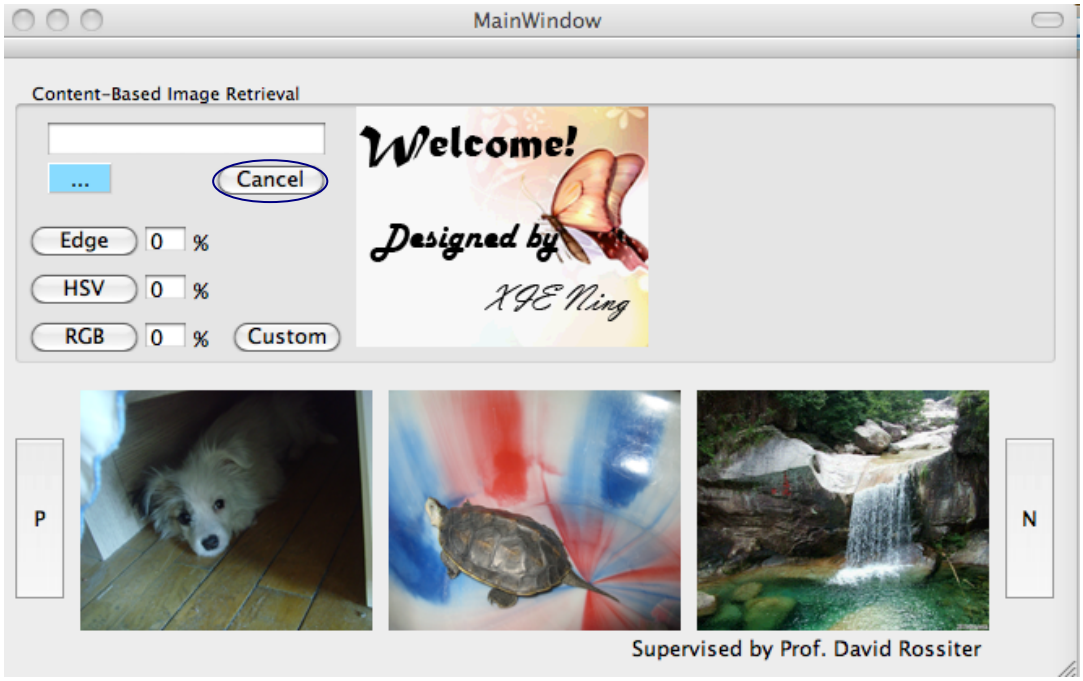

Table 10 Example using Cancel

7 Conclusions

Through trying a number of images in the system, HSV gave better results than RGB in many cases. Because the Edge method does not pay attention to what the colour is, it focuses on the change only so that the results it provides may not have the same visual content as the uploaded image. However, results have a similar structure as the uploaded image. With the combination of these three methods, the system could always get results with highest similarity from the database.

To sum up, the system, Content-Based Image Retrieval, provides three basic methods, which are RGB, HSV, and Edge, to search for similar images, and one method that combines of those three methods. Users can use the system to search for the most similar images according to the content of an image, rather than by using any text.

8 References

[1] WIKIPEDIA. http://en.wikipedia.org/wiki/RGB_color_model. Retrieved on 2 November 2010.
[2] WIKIPEDIA. http://en.wikipedia.org/wiki/HSL_and_HSV. Retrieved on 2 November 2010.
[3] WIKIPEDIA. http://en.wikipedia.org/wiki/Sobel_operator. Retrieved on 2 November 2010.
[4] Qt Class Reference. http://doc.trolltech.com/3.3/index.html. Retrieved on 2 November 2010.