

# Digital Image Processing - Final Project Report

梁展瑞      10389084  
张艺帆      13941213  
李金洋      13941212  
邓达生      10389019

January 11, 2014

## Contents

<b>Subject of Project</b>	<b>1</b>
<b>Workload distribution</b>	<b>1</b>
<b>Structure of Our Work</b>	<b>2</b>
<b>The Histogram Intersection Method</b>	<b>2</b>
<b>Implemented Methods</b>	<b>3</b>
RGB based . . . . .	3
CIELab based . . . . .	3
HSV based . . . . .	3
<b>Results</b>	<b>3</b>
<b>Conclusion</b>	<b>4</b>

## Subject of Project

In this project we need to implement a image searching system. This system should be able to search image from database by query image, and calculate the correct rate according to the give criteria.

## Workload distribution

Name	work	workload
梁展瑞	Lab HSV and Tuned-HSV method	28 %
张艺帆	Web page design and implemtation. The auto testing system	27 %
李金洋	Django Deploying and configuration. RGB method	24 %
邓达生	C++/SWIG algorithm implemtation.	21 %

## Structure of Our Work

Our system is web based. Users will be able to send query image using web browsers. So users don't need to setup any extra softwares on their computer.

On the server side, we use Python with Django and deploy it on Linux. Since native Python is a interpreted language, it's a bit slow when dealing with the algorithm part. We have tried different methods to solve this problem:

- Use SWIG to generate wrapper from C++ code. As for data communication, we use IPC shared memory.
- Use Cython to add static typing support for Python. Then we can “compile” part of performance sensitive Python code.
- Use Numpy with native Python and avoid hand-rolled loops. This can avoid the heavy data communication cost.

Finally, after playing with different git branches, we found the most suitable method is using Numpy. This is more easy to write and can be more flexible. The performance is quite well since we could cache the images in memory.

## The Histogram Intersection Method

We implemented several methods, all of which are histogram based. Now we will have a brief introduction for the histogram intersection method [1].

$$HI = \sum_{c \in P} \min(h_T(c), h_M(c))$$

Here  $h_T$  is the histogram of the target image(image to query), and  $h_M$  is the histogram of the model image(image in database).  $h(c)$  is the normalized count value of color  $c$ .  $P$  is the merged color palette [2] of the two image. Here our merge method is by quantization, which is a simplified version of merge method described in [2].

It is actually a measure of the difference between the two image, since the difference:

$$\begin{aligned} D &= \sum_{c \in P} |h_T(c) - h_M(c)| \\ &= \sum_{c \in P} (\max(h_T(c), h_M(c)) - \min(h_T(c), h_M(c))) \\ &= \sum_{c \in P} (h_T(c) + h_M(c) - 2 * \min(h_T(c), h_M(c))) \\ &= \sum_{c \in P} h_T(c) + \sum_{c \in P} h_M(c) - 2 * \sum_{c \in P} \min(h_T(c), h_M(c)) \\ &= 2 * (1 - \sum_{c \in P} \min(h_T(c), h_M(c))) \\ &= 2 * (1 - HI) \end{aligned}$$

Here we can know that the larger  $D$  is ,  $HI$  will get smaller value, vice versa. We choose the  $HI$  to calculate since it needs only one operation while calculation  $D$  needs two(subtract then abs).

Since an colored image normally has 3 channels, we perform the HI method to each channel and than sum them up. For example, in RGB color space, we have three channels  $r, g, b$ .

$$HI = HI_r + HI_g + HI_b$$

Before we performing HI, we must calculate the color pallette. We use quantization to do this. More specifically, for a channel  $i$ , denote its color range as  $R_i = [a_i, b_i)$  and use  $n$  histogram bins, then the quantized color value for color  $c$  is:

$$c_i = \left\lfloor \frac{(c - a_i)n}{b_i - a_i} \right\rfloor$$

Note that now  $c_i$  is an integer, denoting the bin that it fall over.

Another note is that the histogram is normalized, that is

$$\sum_{c \in P} h(c) = 1$$

## Implemented Methods

We implemented several methods, in different color space.

### RGB based

This is the most naïve one.

### CIELab based

The CIELab color space is a visually uniformed space, so it will gain better result when comparing the difference of two images.

### HSV based

The HSV(Hue, Saturation, Value) color space, compared to the RBG color space, is more close the human perception of colors.

Since the H axis is actually an angle value which will “wrap” around, we need the threat it specially. We first calculate the normal histogram  $HI$  and then a “wrapped” histogram  $HI_w$ . Suppose we quantized and rescale the color values to  $[0, 1)$ .

$$HI_w(h_T, h_M) = \sum_{c \in (0,1)} \min(h_T((0.5 + c)\%1), h_M((0.5 + c)\%1))$$

Here  $\%$  is the generalized modulo operator.  $3.2\%1 == 0.2$ , for example.

Finally we take  $(HI + HI_w)/2$  as this channel’s HI value.

## Results

Figures at the end of this report shows the results of our methods. For each method we take 20 images from each group, and compare each one to the whole database to get a correct rate. And then take the average rate.

The naïve HSV method seems to be the best one. Though theoretically the Tuned-HSV method described above should be better. We guess this is due to the correct rate criteria, which groups non-visually similar images into one group. For example, Figure 1 show a overview of group-6, which contains different types of images. However, if we want to get high correct rate according to the criteria, we should make the algorithm think they are “similar”, which is not correct.

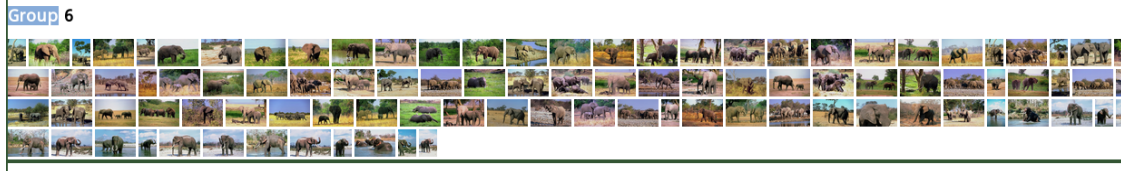


Figure 1: Overview of group 6

## Conclusion

## References

- [1] Swain, Michael J., and Dana H. Ballard. “Indexing via color histograms.” *Active Perception and Robot Vision. Springer Berlin Heidelberg, 1992. 261-273.*
- [2] Wong, Ka-Man, Chun-Ho Cheung, and Lai-Man Po. “Merged-color histogram for color image retrieval.” *Image Processing. 2002. Proceedings. 2002 International Conference on. Vol. 3. IEEE, 2002.*
- [3] Jia, Wenjing, et al. “A comparison on histogram based image matching methods.” *Video and Signal Based Surveillance, 2006. AVSS’06. IEEE International Conference on. IEEE, 2006.*

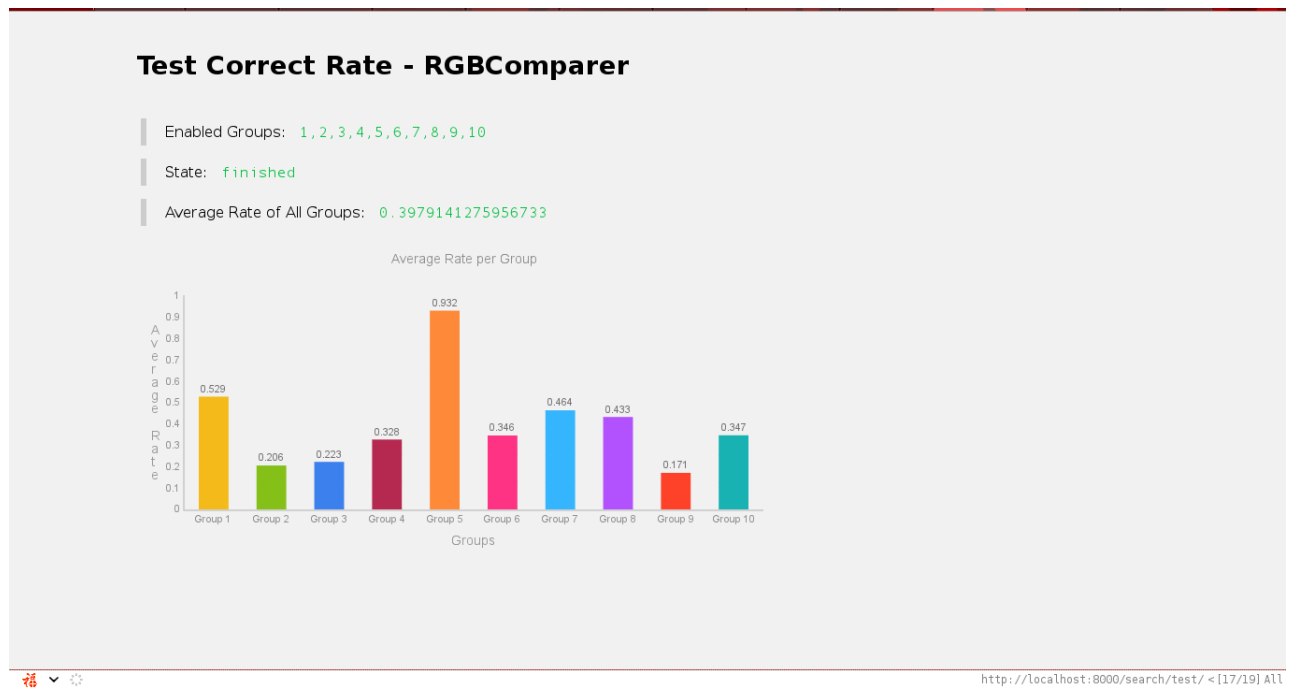


Figure 2: RGB method result

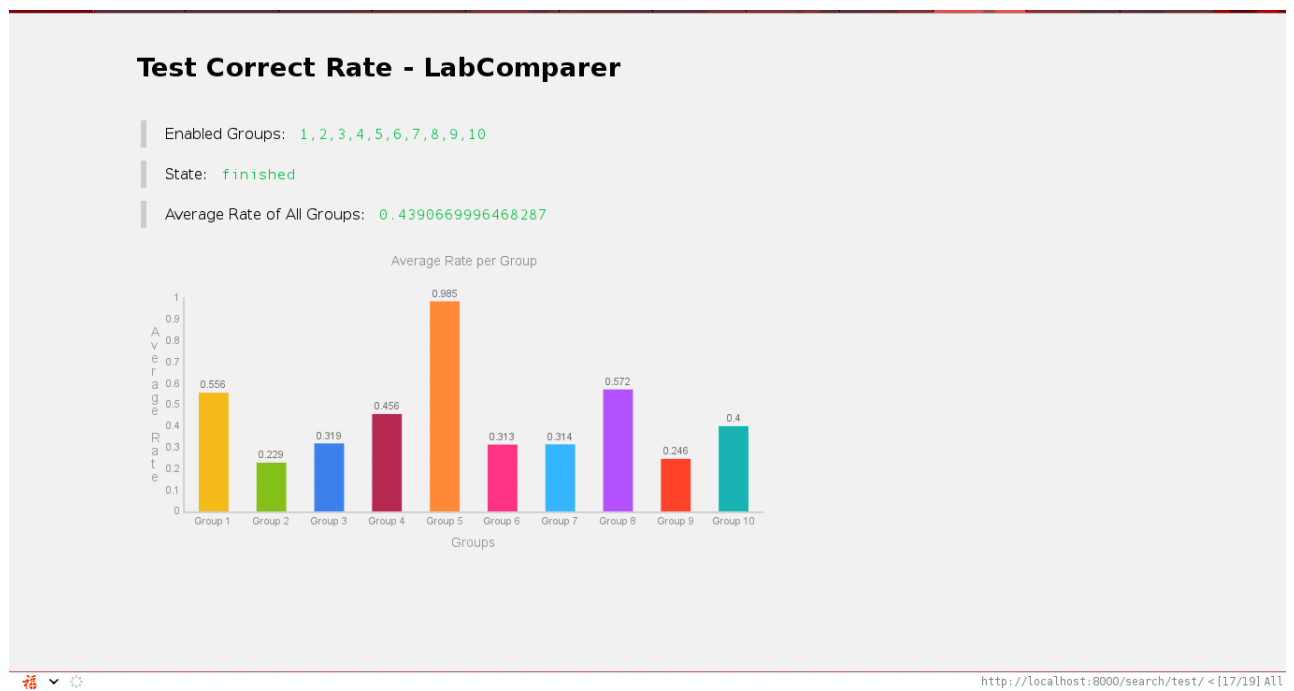


Figure 3: Lab method result

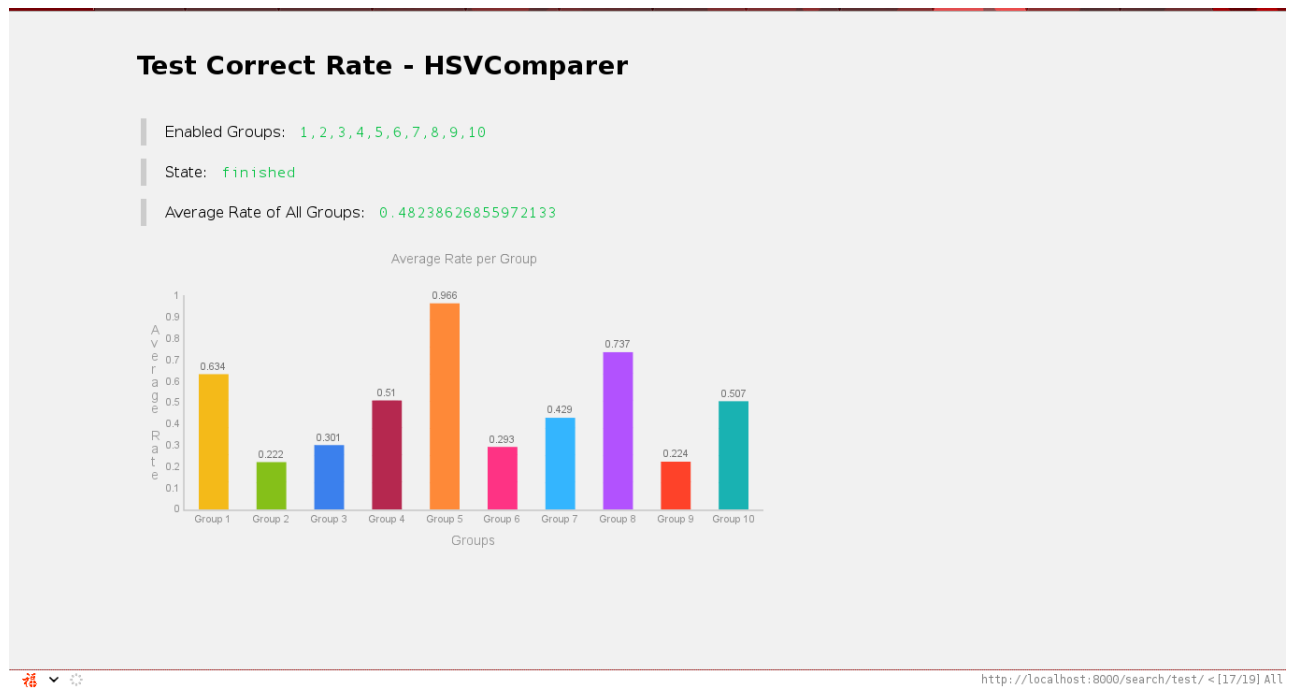


Figure 4: HSV method result

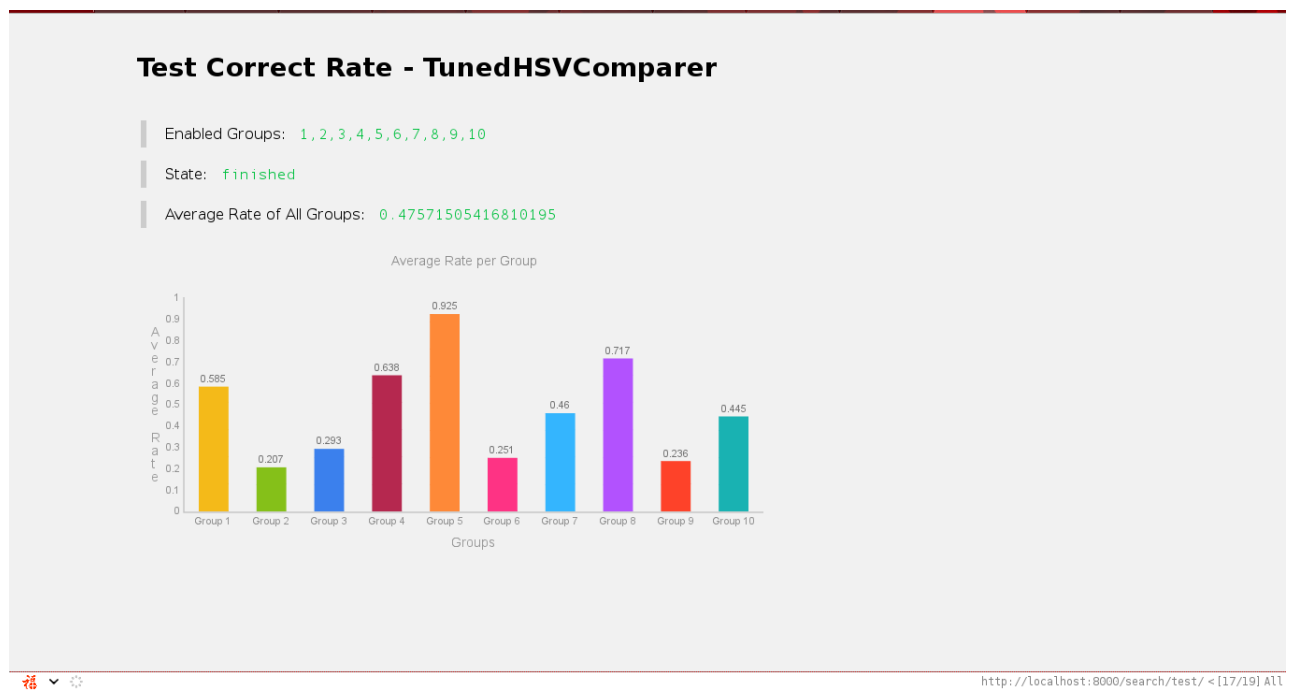


Figure 5: TunedHSV method result

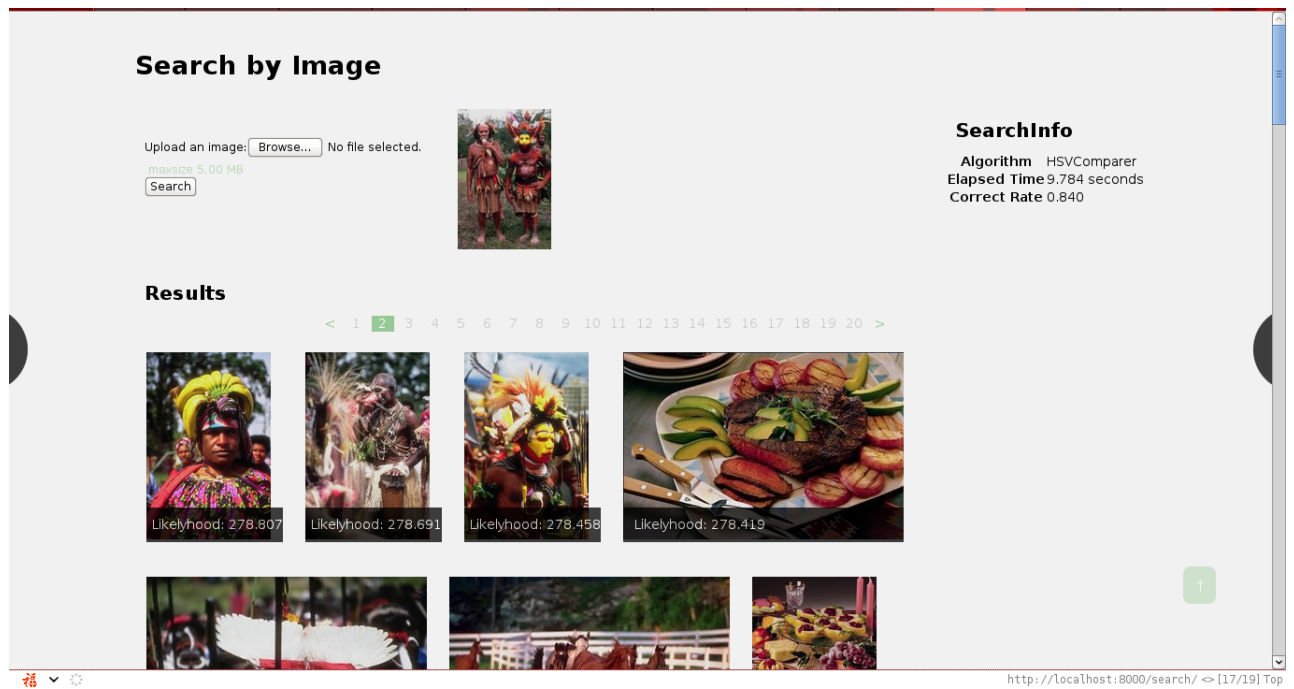


Figure 6: Search page

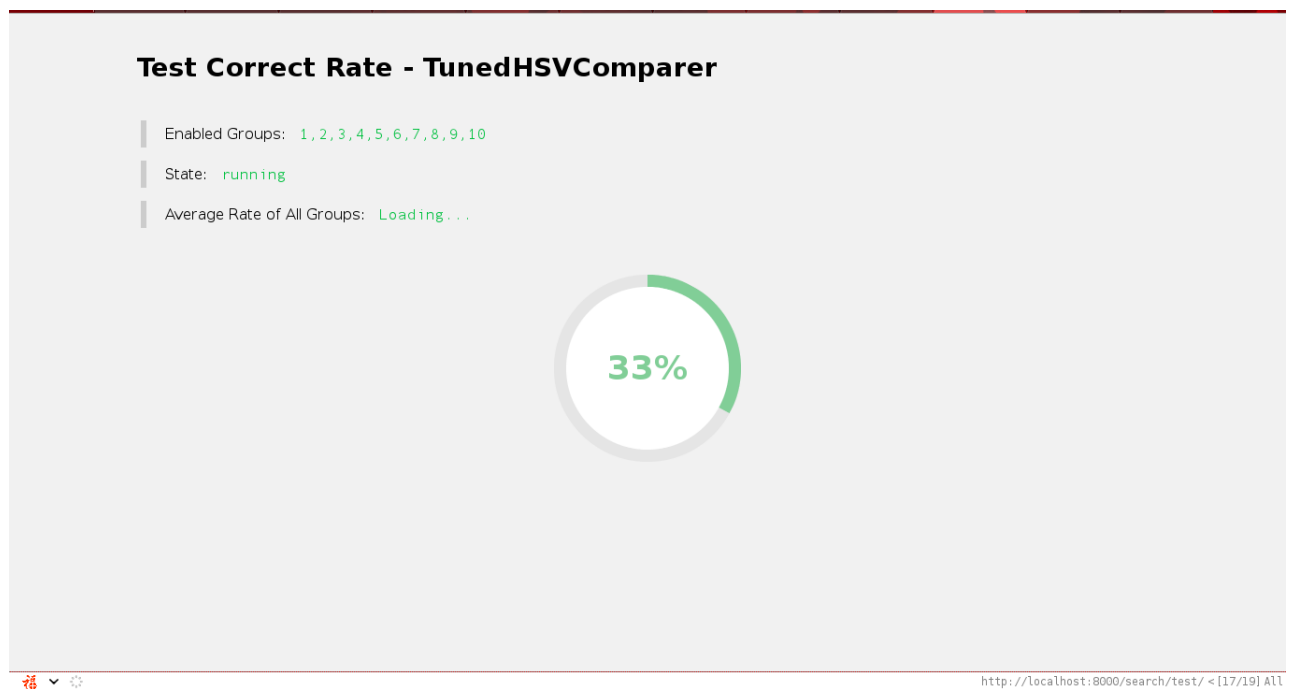


Figure 7: Batch correct rate testing.

### Test Correct Rate - TunedHSVComparer

#### Configure Parameters

Group: 1 Number: 100	Group: 2 Number: 100	Group: 3 Number: 100
Group: 4 Number: 100	Group: 5 Number: 100	Group: 6 Number: 100
Group: 7 Number: 100	Group: 8 Number: 100	Group: 9 Number: 100
Group: 10 Number: 100		

Samples per Group

Start Test



  <http://localhost:8000/search/test/> <=> [17/19] All

Figure 8: Batch correct rate testing, parameter configure.