TONGUE IMAGE DIFFERENCE DETECTION

A Thesis

presented to

the Faculty of the Graduate School

at the University of Missouri-Columbia

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

---

by

HAO CHANG

Ye Duan, Thesis Supervisor

MAY 2014

The undersigned, appointed by the dean of the Graduate School, have examined the thesis entitled

TONGUE IMAGE DIFFERENCE DETECTION

presented by Hao Chang,

candidate for the degree of master of science

and hereby certify that, in their opinion, it is worthy of acceptance.

DR. YE DUAN

DR. DONG XU

DR. JIANLIN CHENG

ACKNOWLEDGMENTS

Foremost, I would like to express my sincere gratitude to my advisor Prof. Ye Duan, for his excellent guidance, caring and patience. I attribute the level of my master's degree to his encouragement and effort. This project and my graduate study would not have been completed without his support.

Besides my advisor, I would like to thank Prof. Dong Xu for providing me with an excellent atmosphere to work on the iTongue project. Thanks for his professional suggestions in every version of my segmentation algorithm.

My sincere thanks also goes to my committee member, Prof. Jianlin Cheng, for reviewing and giving suggestions regarding my project and for the encouragement he gave when I faced the biggest difficulty in my graduate study.

Special thanks goes to Prof. Jiatuo Xu and Wenchuan Qi. Prof. Jiatuo Xu offered lots of help with my thesis research. A lot of tongue image data were confirmed and organized by Prof. Jiatuo Xu. Thanks to the help from Wenchaun Qi about the tongue image segmentation algorithm.

Finally, I would like to thank Dr. Jingqing Jiang and my labmates: Dahai Liu, Zheyu Lu, Meng Zhang, Xu Wang and Minxin Cheng. Thanks for your encouragement and help.

Table of Contents

List Of Figures

List of Tables

TONGUE IMAGE DIFFERENCE DETECTION

Hao Chang

Dr. Ye Duan, Thesis Supervisor

ABSTRACT

Based on tradition Chinese medicine (TCM) theories, all the import human organs have connections with the tongue. So the tongue can be a very important indication of the health status. For this reason, the changing of the tongue surface can be very important health information.

This thesis proposes a tongue changing detection solution that can keep track of the changing of the tongue. Based on the tongue images this method can detect the changing areas on the tongue within a few seconds of time. Also, this thesis introduces the system architecture of the iTongue system.

Chapter 1. INTRODUCTION

Traditional Chinese Medicine (TCM) originated in China and developed in Eastern Asia for thousands of years. Being different from medical theory originating in western countries, which is based on anatomy, TCM comes from doctors’ empirical summaries. Although TCM has proven to be efficacious, most of its standards of diagnosis have not been quantized. So TCM diagnosis relies heavily on the doctors’ observations and their practical experience. Thus, the difficulty in utilizing TCM is that only experienced doctors can guarantee an accurate and precise diagnosis. Some of the hard to recognize and less distinctive features of the tongue can lead to different diagnoses; hence, TCM also has disadvantages in that it lacks objectivity and repeatability. These disadvantages have already become a very big challenge to further development of TCM. So it is necessary to bring in a scientific and quantized research method into the TCM diagnostics for its further development [1].

Researchers using CAD with TCM are quantizing TCM standards based on a huge number of proven diagnostics attained by machine learning. These attempts are not only successful in widening the scope of CAD research in this field of study, but also contribute to a database that will hopefully become a resource for practitioners worldwide thereby adding to the development and documentation of TCM.

The development of electronics has made CAD universally accessible. Even for those who are not computer scientists, a diagnosis at home has become feasible through personal electronics and the Internet. Portable devices have finally given people a chance to monitor their health anytime anywhere. By giving people ways to check their own vitals and investigate potential problems online as well as at their doctor's offices, diseases can be prevented before they progress or before outbreaks can occur, which improves quality of life and reduces the cost of health care for the individual and society.

According to the TCM theory, the whole human body system is an organically inter-connected system, and the appearance of the tongue is a very important indicator of health status. As Figure 1-1 shows, almost all of the important organs of human body such as liver, stomach, gallbladder, spleen, lung and heart, which have a strong relationship with the tongue. So, the changing of the tongue if correctly interpreted can tell us much about the health of the body's important organs.

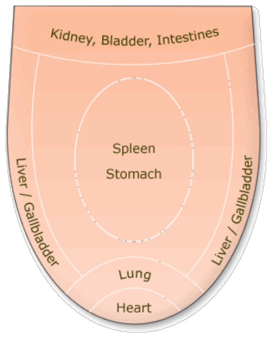


Figure 1-1. The relation between tongue and human organs. (Tongue and diagnosis n.d.)

Also, changes of the tongue can be a quantitative standard whereby a practitioner can assess the effect of the patient's treatment. Figure 1-2 is the tongue image of a gastritis patient before and after treatment. The human eye can easily see the difference of the tongue before and after treatment. In this thesis, we propose a method that can quantitatively detect the change on the tongue

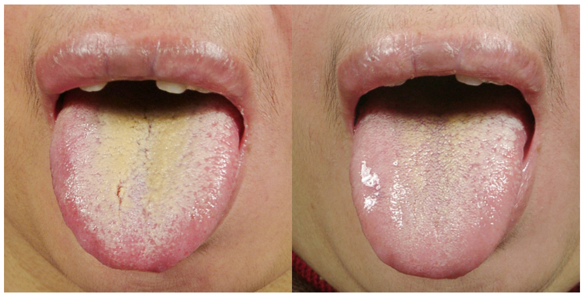


Figure 1-2. The tongue difference before and after treatment

In addition to the traditional methods for tongue diagnosis, there is now research advancing new ways of tongue diagnosis application. According to research by Lo, Cheng, Chiang, and Damdinsuren (2013), the tongue diagnosis can also be used in the diagnosis of breast cancer [2].

Differences in tongue images during detection process create several difficulties. First, tongues are not rigid bodies, so it is difficult to register the tongue images due to changes in position (movement). Second, light conditions can cause tongue images to vary from each other causing the color to change and leading to an inaccuracy in the tongue difference detection algorithm. Third, since the first step of registering tongue images is so difficult, the detection algorithm relies on tongue image segmentation results. In this thesis, we will talk about solutions to these difficulties.

Also, in this thesis, we will introduce the iTongue diagnosis system. The iTongue system is a CAD system set up to provide TCM diagnosis. One important feature of the iTongue system is an insistence on letting the user keep track of the changes in his/her tongue.

Chapter 2. BACKGROUND

2.1. Color Model

A color model is an abstract mathematical model describing the way colors can be represented as tuples of numbers, typically as three or four values or color components [3]. Each color space has its own advantages and disadvantages.

The disadvantage of the RGB color model is that since human eyes have different levels of sensitivity to different colors, the colorimetric distances between the individual colors do not correspond to perceived color differences. For example, a difference between green and greenish-yellow is relatively large, whereas the distance distinguishing blue and red is quite small. In order to avoid the disadvantages of single color spaces, in this thesis, we applied the multiple color spaces including RGB, Lab and Hsv for the tongue image difference detection.

2.2. RGB color space

The RGB color space is one of the most common color models in computer science. The idea of the RGB color space is that combination of red blue and green colors can represent every color. Like the example in Figure 2-1; we can get yellow color by adding red and green; also we can get purple by adding red and blue. In the RGB color space, every color is represented in three components R G and B that correspond to the red green and blue component. In computer science, the R G and B color values are represented by an 8 bit integer that has value from 0–255. In the rest of the thesis, the R G and B color value will also be 0–255.

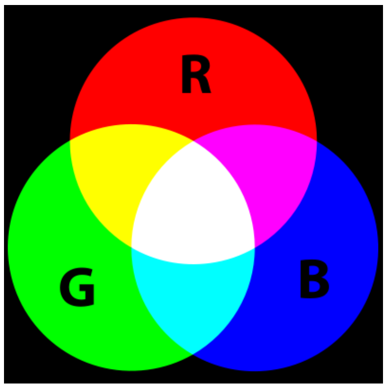


Figure 2-1. RGB color space (RGB color space n.d.)

The RGB color space is very simple for calculation and easy to understand, but it is not enough for the tongue image difference detection because it has the following disadvantages. (1) The RGB color space cannot represent all the colors that the human eye can see. (2) As mentioned in Section 2.1, in the RGB color space, the colorimetric distance between the individual colors do not correspond to human perceived color differences.

2.3. Lab Color Space

A lab color space is a color-opponent space with dimension L for lightness and a, b for the color-opponent dimensions, based on nonlinearly compressed CIE XYZ color space coordinates [4]. The range of the component L in Lab color space changes from 0 to 100, with 0 representing minimum light and 100 representing maximum light. The a component ranges from -120 to +120; furthermore, this component represents the colors between red and green; +120 of the a component represents red, and -120 of the a component represents green. The b component represents the color from yellow to blue. The range of b is also from -120 to +120, with +120 yellow and -120 blue.

To convert an image from RGB color space into Lab color space, we have to first convert from RGB color space into XYZ color space. The equation representing this conversion from RGB to XYZ color space follows:

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 0.412453 & 0.357580 & 0.180423 \\ 0.212671 & 0.715160 & 0.072169 \\ 0.019334 & 0.119193 & 0.950227 \end{bmatrix} * \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

After converting the image from RGB color space into the XYZ color space, the equation to convert from the XYZ color space into the Lab color space will be applied to the image:

$$L^* = 116 * f\left(\frac{Y}{Y_n}\right) - 16$$
$$a^* = 500 * [f\left(\frac{X}{X_n}\right) - f(\frac{Y}{Y_n})]$$
$$b^* = 200 * [f\left(\frac{Y}{Y_n}\right) - f(\frac{Z}{Z_n})]$$

In the equation above, X_n Y_n and Z_n are the tristimulus values of the reference while (X_n= 0.3127, Y_n= 0.3290, and Z_n= 0.3583). And the formula of function f is:

$$f(t) = \begin{cases} t^{\frac{1}{3}}, & t > \frac{6}{29}^{3} \\ \frac{1}{3} * \left(\frac{29}{6}\right)^2 * t + \frac{4}{29}, & Otherwise \end{cases}$$

Lab color space is designed to approximate human vision. It aspires to perceptual uniformity, and its L component closely matches the human perception of lightness.

Another advantage of the Lab color space against RGB color space is that it cannot only represent all the colors that RGB can represent, but it can also represent colors that the RGB color space doesn’t contain.

2.4. HSV Color Space

HSV is one of the most common cylindrical-coordinate representations of points in an RGB color model. The two representations rearrange the geometry of RGB in an attempt to be more intuitive and perceptually more relevant than the Cartesian (cube) representation. Developed in the 1970s for computer graphics applications. The three components in HSV represent Hue, Saturation and Value. In the HSV cylinder, the angle around the central vertical axis corresponds to the “Hue,” the distance from the axis corresponds to “Saturation,” and the distance along the axis corresponds to the “Value.

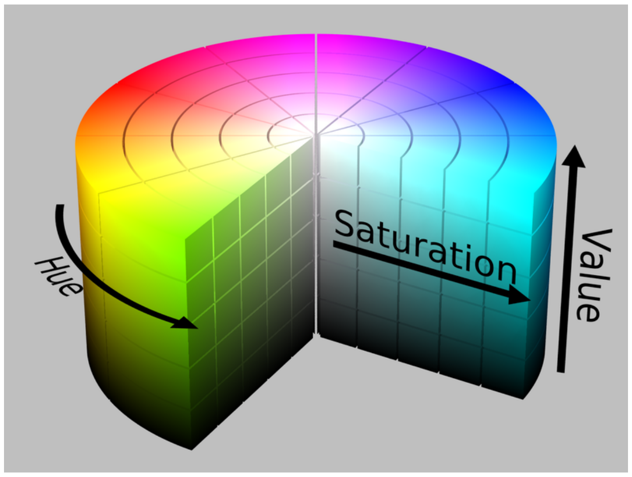


Figure 2-2. The Hsv color space (SharkD 2010)

To convert an image from RGB color space into Hsv color space, the first step is to normalize R, G and B value to 0~1. Because the original values of R G and B are 0~255, so we do the normalization by dividing 255.

$$R' = R/255$$

$$G' = G/255$$

$$B' = B/255$$

Then for each pixel on the image, we calculate the values C_{max}, $C_{min}$and Δ using the following equations.

$$C_{max} = \max(R', G', B')$$

$$C_{min} = \min(R', G', B')$$

$$\Delta = C_{max} - C_{min}$$

After getting the three values, we can use them to calculate the H, s and v values using the following equations.

$$H = \begin{cases} 60^o * \left(\frac{G' - B'}{\Delta} mod6\right) & , C_{max} = R' \\ 60^o * \left(\frac{B' - R'}{\Delta} + 2\right) & , C_{max} = G' \\ 60^o * \left(\frac{R' - G'}{\Delta} + 4\right) & , C_{max} = B' \end{cases}$$

$$s = \begin{cases} 0 & , \Delta = 0 \\ \frac{\Delta}{C_{max}} & , \Delta <> 0 \end{cases}$$

$$v = C_{max}$$

Hsv is a very useful color space for detecting color changes. It’s different from RGB and Lab color space because there is only channel that represents the color, so it is also easier for us the use this color space to tell us to tell how the color has changed.

Chapter 3. METHODOLOGY

3.1. Introduction to Tongue Image Difference Detection

In order to improve the results of the diagnosis, we track the changes of the user's tongue. By applying the image-differencing algorithm on the tongue image, we can accurately tell which part of the user's tongue has changed. This information not only helps achieve a better diagnosis but also provides an alert when the tongue image has significantly changed.

Image differencing [5] is an image processing technique used to determine changes between images. The difference between two images is calculated by finding the differences between each pixel in each image and then generating an image based on the result. For this technique to work, the two images must first be aligned so that corresponding points coincide, and their photometric values must be made compatible, either by careful calibration or by post-processing. The complexity of this type of image preprocessing varies with the type of image.

The image difference detection of the tongue image is a challenging problem for three reasons. 1. The tongue is not a rigid body; hence, two images of the same tongue can be very different. 2. The tongue differencing analysis is based on the accuracy of the tongue image segmentation. 3. The illumination condition under which the tongue image was taken can disturb the image differencing analysis.

In order to overcome these challenges, the following assumptions will be established. 1. The input images for the tongue difference detection algorithm is from the same person and the way he/she sticks out his/her tongue must be almost the same. 2. The image segmentation algorithm works well that the whole tongue can be segmented from a single image. 3. The illumination condition under which the tongue image was taken must be exactly the same. We found that the second assumption can fail; however, the segmentation results had enough quality to allow the difference detection algorithm to provide reasonable results.

Figure 3-1 shows the tongue image difference detection flow chart; we will introduce each process in the following chapters.

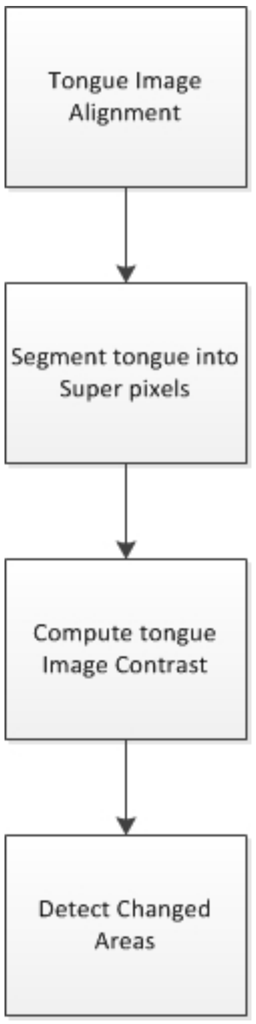

Tongue Image Alignment
Segment tongue into Super pixels
Compute tongue Image Contrast
Detect Changed Areas

Figure 3-1. Tongue Image Difference Detection Flow Chart

3.2. Tongue Image Alignment

As the first step of the tongue image differencing analysis, the tongue of the two images must be aligned. According to Zitova and Flusser (2003) of the majority of the image registration methods consist of the following four processes: 1) Feature detection, 2) feature matching, 3) transformation model estimation, and 4) image resampling and transformation [6]. This kind of method cannot be applied to the tongue image because unlike other objects, we can hardly find feature points of the tongue. Also, since the tongue images are always changing, feature detection and feature matching methods are very difficult to obtain when trying to capture the tongue image. In this thesis, we use another method for the tongue alignment.

The whole tongue image alignment process is illustrated in the flow chart of Figure 3-2.

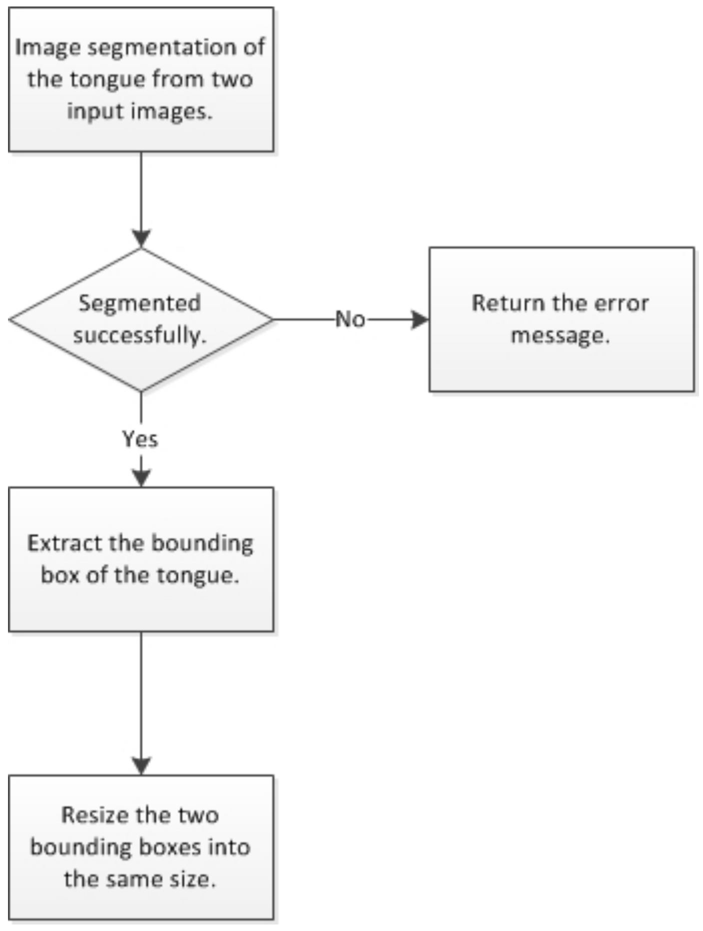
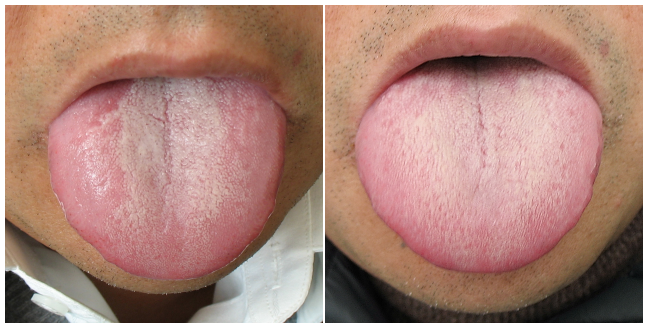


Image segmentation of the tongue from two input images.
Segmented successfully.
No
Return the error message.
Yes
Extract the bounding box of the tongue.
Resize the two bounding boxes into the same size.

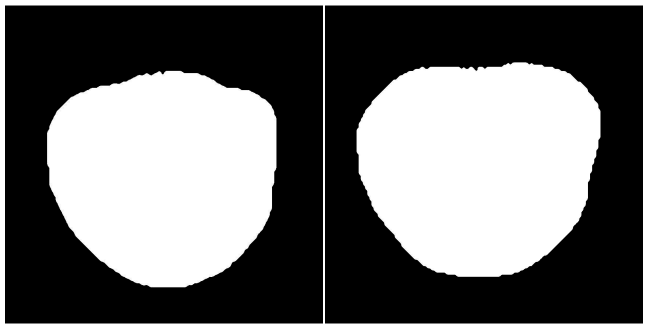
Figure 3-2. Tongue Image Alignment Flow Chart

The first step of the alignment process is to segment the tongue from two images. In this thesis, the tongue image segmentation using GrowCut [7] is applied. This segmentation algorithm is an optimized version of the GrowCut algorithm for tongue image segmentation problem. After running the tongue segmentation, we can get the output in the form of binary masks as shown in Figure 3-3(c) and Figure 3-3(d) shows. The white part of the mask represents the tongue area in the input image.

In the second, step, we use the algorithm to check if the segmentation is successful. Performing this check is necessary because errors can occur as shown in Figure 3-4. These kinds of errors are caused by over-segmentation. In order to prevent this situation, the segmentation algorithm performs over-segmentation detection to measure the segmentation quality. In the alignment process, we previously performed manual checks looking for errors in segmentation. In the current version of the algorithm, once we find any segmentation quality is not good enough, a wrong segmentation message will be returned.

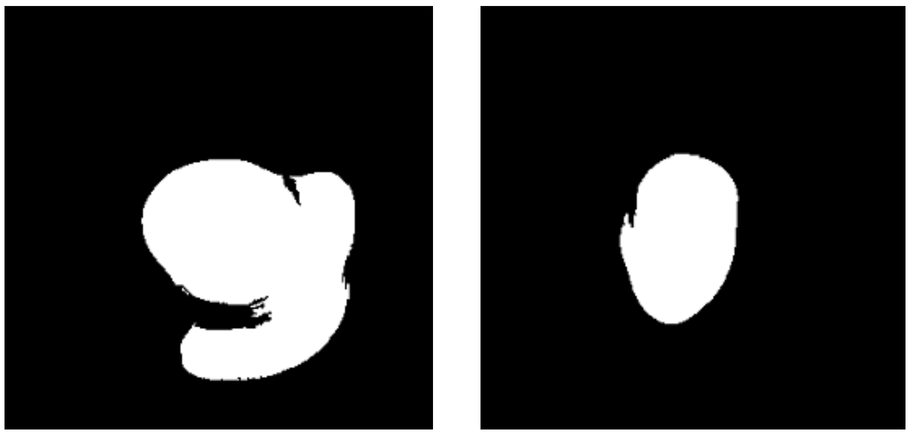


(a) (b)



(c) (d)

Figure 3-3. The Tongue Segment Results. (a) The first input image. (b) The second input image. (c) The segmentation result of the image (a). (d) The segmentation result of image (b)



Serious over segmentation Small over segmentation

Figure 3-4. Examples of unsuccessful segmentation (Wenchuan Qi, 2013)

After the segmentation step, two input images and two binary masks of the same sizes are added to the segmentation file along with the original images. The third step of the image alignment process is to extract the bounding box of the tongue from the image. There are four steps to get the bounding box based on the binary mask:

1. To get the upper edge of the bounding box, scan the binary mask from the top-down once there is a mask pixel row with one or more “1” value; then, return to the row number and set it as the upper edge.

2. To get the bottom edge of the bounding box, scan the binary mask from the bottom-up once there a mask pixel row with one or more “1” value; then, return the row number and set it as the bottom edge.

3. To get the left edge of the bounding box, scan the binary mask from left to right once there is a mask pixel in one column with one or more “1” value; then, return the column number and set it as the left edge.

4. To get the right edge of the bounding box, scan the binary mask from right to left once there is a mask pixel in one column with one or more “1” value, then return the column number and set it as the right edge.

The whole bounding box extraction process is.

```
FOR I = 1: Number of rows of the binary mask
    IF the Ith row of the mask has any element that equals to “1”
        UP-Edge = I;
    END IF
END FOR

FOR I = Number of rows of the binary mask: 1
    IF the Ith row of the mask has any element that equals to “1”
        BOTTOM-Edge = I;
    END IF
END FOR

FOR I = 1: Number of columns of the binary mask
    IF the Ith row of the mask has any element that equals to “1”
        LEFT-Edge = I;
    END IF
END FOR

FOR I = Number of rows of the binary mask: 1
    IF the Ith row of the mask has any element that equals to “1”
        RIGHT-Edge = I;
    END IF
END FOR
```

Code 3-1. Bounding Box Extraction

Figure 3-5 is two examples of the bounding box extraction result.

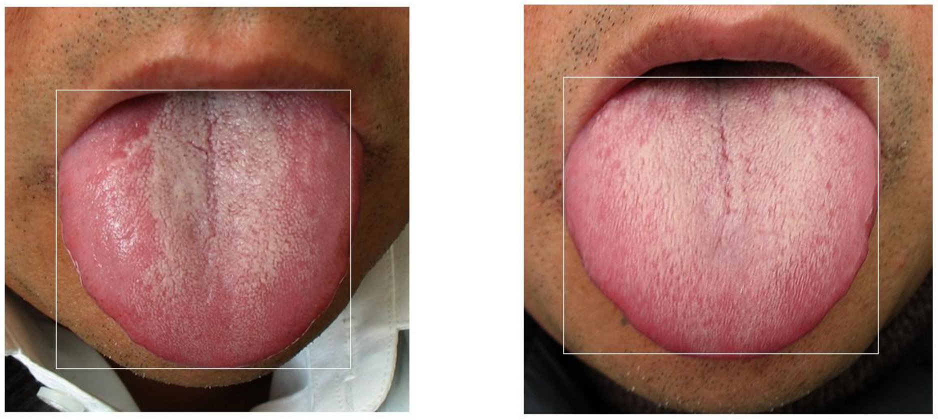

(a) (b)

Figure 3-5. Bounding Box Extraction Results.

After extracting the bounding box for the tongues in the two images, the final step is resizing the bounding box into the same size so that we can align the two tongues from the images. The first step for the alignment process is to resize the first bounding box into the width of 400 pixels, after which the length of this bounding box will be proportionally resized according to the width. The reason why all the tongue images are resized into the width of 400 pixels is because the width of the human tongue doesn't vary that much. Also, because after resizing of the tongue into the width of 400 pixels, the granules on the tongue are still perfectly visible. They do not merge together, which means that resizing does not cause a loss of information.

After resizing the first bounding box, the second bounding box will be resized to exactly the same size as the first one. After this step, the alignment process is finished and the tongues captured by both images will be aligned into the same bounding box.

Figure 3-6 shows one sample result of the alignment process. From the result we can see how corresponding areas have been aligned into the same relative position in the images. But after looking more closely, we find the shapes of the two tongues are not exactly the same. For example, the tip of the tongue in the first result is a little bit concave and the second result is convex. The experiment results have shown that these edge pixels won't affect the tongue difference detection results. This difference is due to edge pixels, which will be discussed later in the thesis.

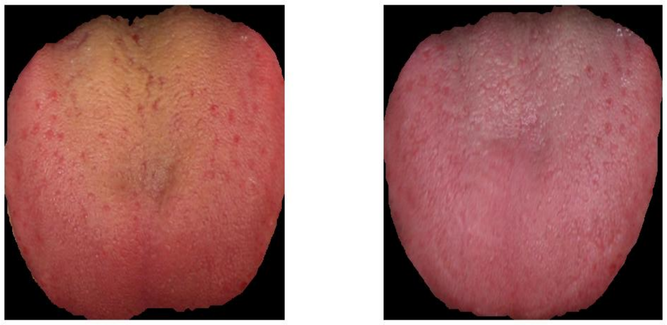


Figure 3-6. The alignment result for two tongues

3.3. Tongue Difference Detection

Now that the tongue bounding boxes are aligned, we can now apply the tongue difference detection algorithm to detect tongue difference. Before the current version of the tongue difference detection method was activated, other trial versions were tested. The lack of definition (pixel alignment), which created tongue image differences was a major problem. The final version of the tongue difference detection algorithm is based on the difference detection method using regular patches. Instead of detecting the tongue changes from single pixel values, in the final version of the algorithm, we merged the pixels into regular patches so that we could handle the imperfect alignment in a much better way. We will discuss these two methods in Sections 3.3.1 and 3.3.2.

3.3.1. Pixel-wise Tongue Difference Detection

Pixel-wise tongue difference detection was the first task for the tongue difference detection algorithm. The idea we focused on considered the time after the alignment process discussed in Section 3.2, where the change in each pair of corresponding pixels in the bounding boxes were calculated, and saved as the difference values were fit into another 2D array which was the same size and achieved good alignment in the two-tongue bounding boxes.

The pseudo code for the pixel wise tongue difference detection is in Code 3-2:

```
FOR I = 1: # Of Rows Of the Bounding Box
    FOR J = 1: # Of Cols Of the Bounding Box
        IF (MASK1 (I, J) == 1 && MASK2 (I, J) == 1)
            RESULT (I, J) = IMAGE1 (I, J) – IMAGE2 (I, J)
        END IF
    END FOR
END FOR

RETURN RESULT
```

Code 3-2. Pixel-wise Tongue Difference Detection

Inside the inner layer of the loop in the pseudo code, we can see that the algorithm must detect if both of the masks have the value 1. The reason why we do that is because after the alignment process, the two tongues are not perfectly aligned. So this leads to the problem that for the edge pixels of the tongue, it might happen that in one of the bounding boxes, there are some particular pixels that are a part of the tongue and in the other bounding box, these corresponding pixels are not shown as being a part of the tongue. When this situation happens, if we do not perform this check, the algorithm will calculate the difference from tongue pixels with zero value pixels. Figure 3-7 is one example of this situation. The red part in the image is where the two tongues are not perfectly aligned.

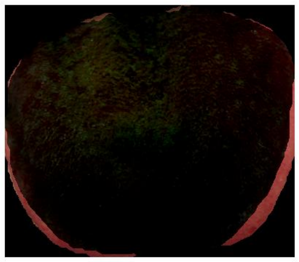
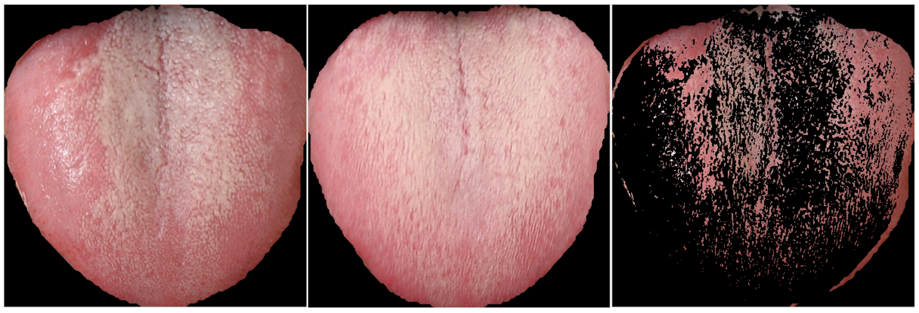


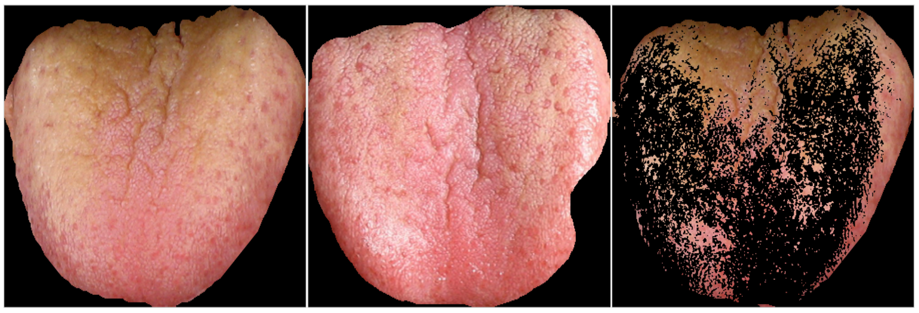
Figure 3-7. One tongue in the image is bigger than the other. The larger part is indicated in red color.

Another thing to notice from Code 3-1 is that this process was applied to 2D images. Since the RGB, Lab and Hsv images are 3D matrixes. There are two ways to apply the code for these kinds of images. First, we can convert the 3D image into a 2D image that reflects the color information. Changing a RGB to a gray scale image is another technique. Second, we can apply code to the 3D images 3 times, each time for single color channels. Then, we can use the 3 result matrixes to detect the changing area.

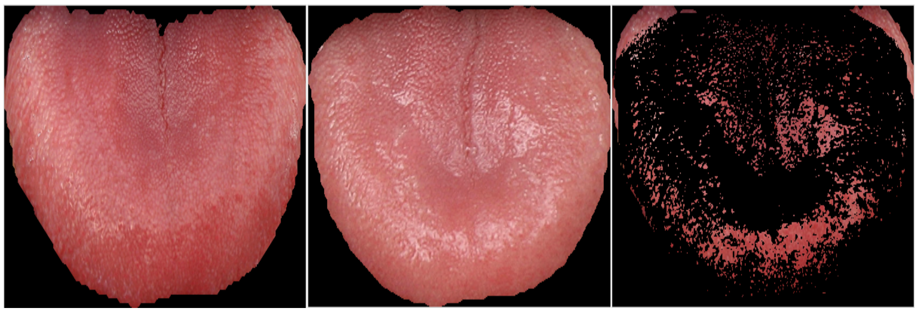
After dealing with the problem for the edge pixels of the tongue, we continued to the next step and calculated the difference of the tongue based on Code 3-2.



(a)



(b)



(c)

Figure 3-8. Sample results using pixel wise difference detection:

(a) Gastritis patient tongue image. (b) Urehra infection patient tongue image. (c) Dyspepsia patient tongue image.

Figure 3-8 represents sample test results of three real patients' tongue images using the pixel-wise difference detection algorithm. For each row of images, the first two pictures are the original tongue images taken before and after treatment of the patients, and the third picture is the result of the difference detection result. For the result pictures, the black areas indicate the tongue areas that haven't changed and the color areas indicate the tongue areas that have changed. In order to get the result, we apply the process described in Code 3-2 three times under R G and B color channels. The end result gave us three 2D matrixes. Then for each 2D matrix, we examined each element in the matrix, and if any element was smaller than the threshold, it was set to zero. After repeating this process for all the three matrixes, we calculated the "OR" result of the three matrixes. Then we used this matrix as the mask for all the detected changing pixels in the original image. For the results in Figure 3-8, the RGB threshold was 45 for each image, indicating that if the corresponding pixels had a value change larger than 45, it had changed; otherwise it had not. However, in the current version of the algorithm, 45 is not the threshold. Further study revealed a need for a new methodology which will be explained in Section 4.5 where the final threshold for difference value is discussed.

From the results, we can see that the pixel-wise difference detection can detect most of the coating changes. Like the first and second results shown in in Figure 3-8, the algorithm did detect a coating change from gray to white and from yellow to white. This algorithm can also detect differences in tongue color as shown in the third result column of Figure 3-8.

Applying the algorithm under Lab and Hsv color are very similar to applying RGB color space. We only had to choose another set of thresholds for the color channels in the color spaces. Also, in Lab color space, we were able to use only a and b channels for the color difference detection because as introduced in Section 2.3, the L channel represents lightness that doesn't relate to our result. Also, for the same reason, in Hsv space, H and s channels are enough for detecting the color change because v channel in Hsv color space also indicates lightness.

From the results in Figure 3-8, we determined that there were many noise points scattering all over the result image, even on the areas of the tongues where the two images did not differ. This happened because the algorithm could not align the two tongue images with pixel-wise accuracy. In order to retain the main parts of the results, we applied a median filter to the result to remove the noise. These results are shown in Figure 3.9.

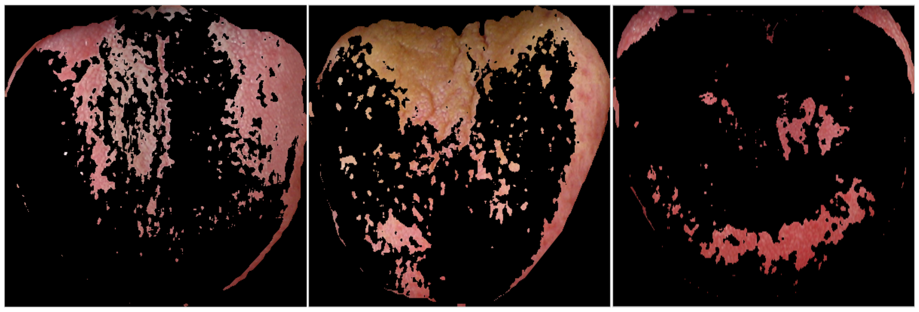


Figure 3-9. The results after the noise remove.

The pixel-wise tongue image difference detection was the first attempt for the tongue image difference detection. It has the advantage of a faster running speed, and implementation is relatively simple. The disadvantage of this method is that it requires a high quality image alignment; furthermore, it is very sensitive to noise. Also for this algorithm, in most cases, instead of comparing the corresponding pixels on the tongue; we compared pixels that are very near in the same area of the tongue.

The difficulty of tongue image alignment has to do with the unavoidable range of movement in corresponding pixels; hence, we compared correspond areas on the tongue for the difference detection. This method will be introduced in the following section.

3.3.2. Tongue Difference Detection Using Regular Patch

Since the pixel wise tongue difference detection method requires very high alignment accuracy, another choice is to merge the single pixels into regular patches and then compare the differences within the regular patches to detect the changes on the tongue surface.

Comparing the method using pixel-wise difference detection, using regular patches is advantageous because the regular patch difference detection method doesn't require perfect alignment of the two tongues. Instead of comparing the same exact corresponding pixels from the two tongues, we can compare corresponding areas of the tongue.

The whole process of the regular patch difference detection is as the following flowchart.

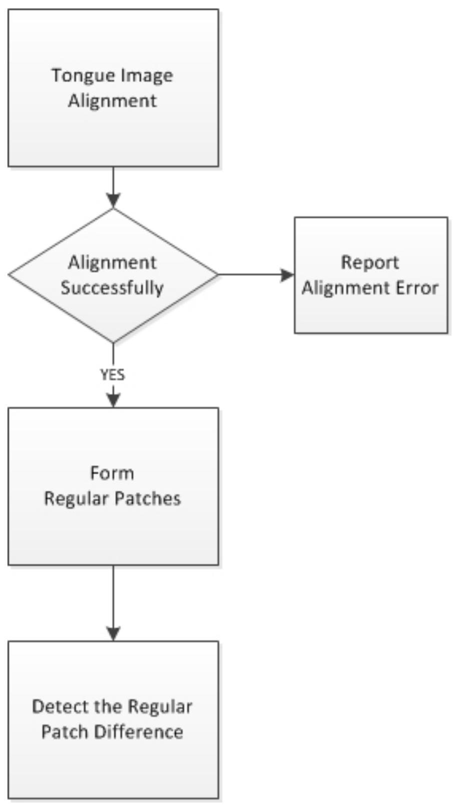
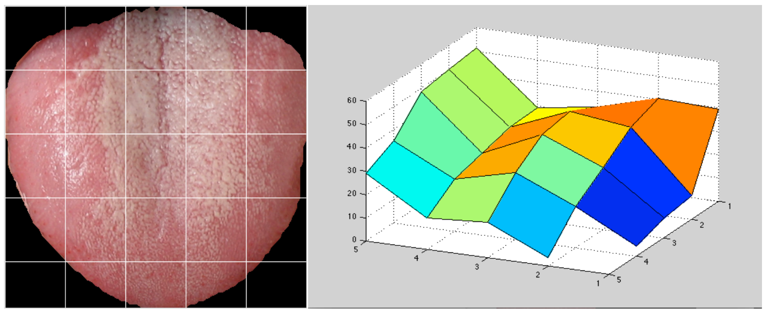
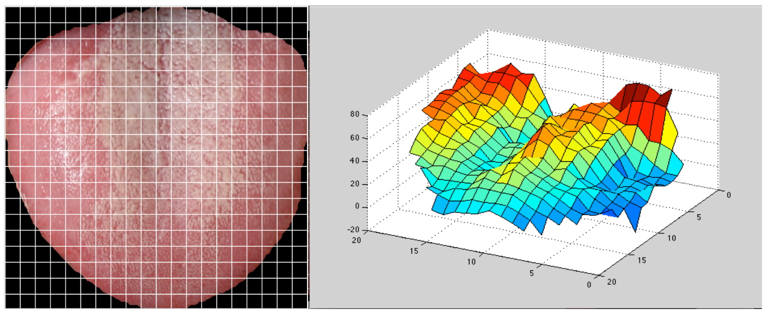


Figure 3-10. Flow chart of the regular patch difference detection

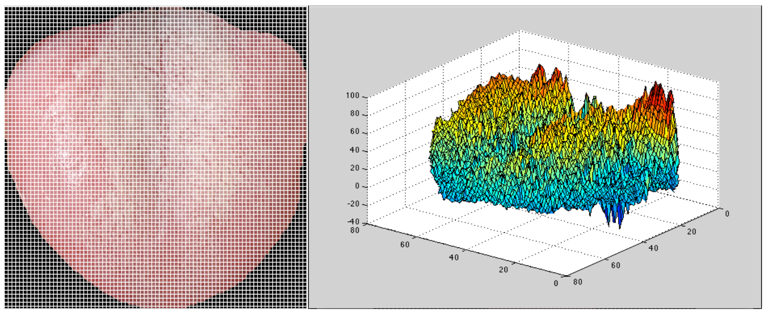
One more thing we should pay attention to is the regular patch size. Regular patches cannot be too small or too large. If we choose a regular patch size that is too small, then, as described in Section 3.3.1, the segmentation method will require high quality alignment. If we choose a regular patch size that is too large, it will cause a loss of information. Choosing the wrong patch size prevents detection of the specific position on the tongue that has changed. Figure 3-11 shows three sample results using different regular patch sizes. Comparing these results, we can see that using a smaller patch size will provide us with much more detailed information about the position that has changed. After some experimentation, the best regular patch width for the current version of the algorithm is 5 pixels wide and 5 pixels high. And since the bounding box of the tongue is already normalized to a width of 400 pixels, the regular patch images have 80 columns.



(a)



(b)



(c)

Figure 3-11. (a) Regular patch 80*80.   (b) Regular patch size 40*40.   (c) Regular patch size 5*5

After deciding the regular patch size, another problem is to deal with the patch edges. In the pixel-wise difference detection method, a single pixel is treated as a valid pixel only when the pixels from the two images in the same position are valid, which means their value in the mask is equal to “1”, as Code 3-2 shows. But in the regular patch difference detection algorithm, the situation is more complicated. Because the regular patches consist of multiple single pixels, in the regular patches, we cannot simply check the mask to tell if a regular patch is valid because of the patches’ edges. As noted in Figure 3-3, some of the single pixels in the regular patches are valid and some are not. So here in our current version of the algorithm, the rule is that a regular patch is valid if and only if there is more than one valid single pixel in the regular patch. For example, in Figure 3-12(a), the edges and side area of regular patches, showing blue and yellow squares, respectively, represent valid areas within regular patches because there are valid single pixels in those areas. The red squares do not represent valid areas because there are no pixels in them.

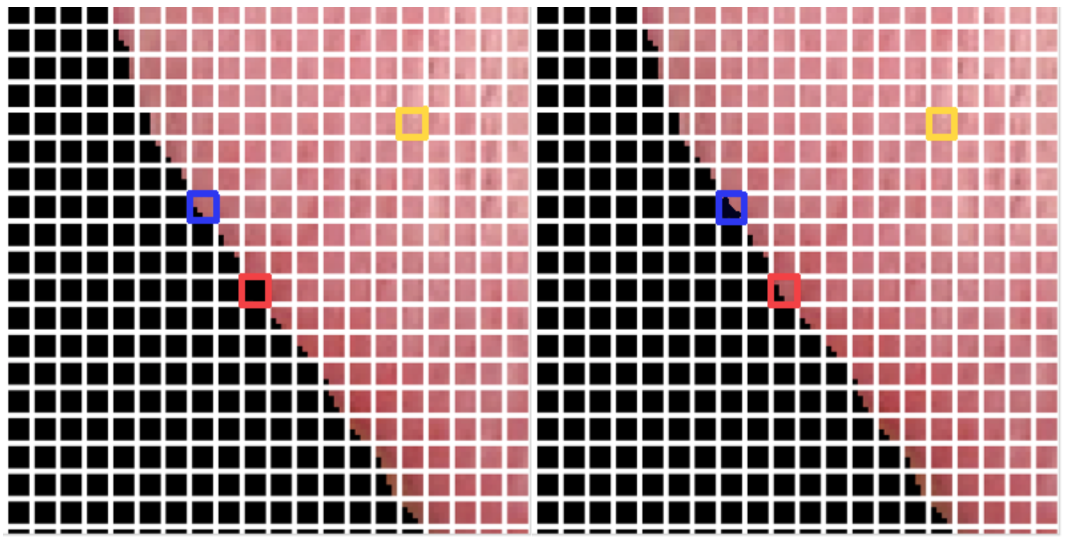

(a) (b)

Figure 3-12. Regular patch edge area

After figuring out a way to decide whether a regular patch is valid or not, we can get a regular patch mask just like the single pixel mask in Figure 3-3. The regular patch mask is a 2D matrix that has a row number equal to the number of rows of the regular patch image, and a column number equal to the number of columns of the regular patch image. We used the regular patch mask to label the valid regular patches with value "1" and nonvalid regular patches with value "0". So, for the example in figure 3-12 (a), the mask value for the blue and yellow squares shown on the regular patches is "1" and the mask value for the red circled regular patch value is "0". For the same reason, for Figure 3-12 (b), the mask value for red, blue and yellow squares within regular patches are all "1". After getting the two regular patch masks, we calculated the "AND" result of the two regular patch masks. This final result is what we call "Regular Patch Mask" in Code 3-3. The reason why we used "AND" calculation is because in the two input images, if one of the regular patches is nonvalid, then calculating the difference between a valid and nonvalid regular patch would make no sense. So in Figure 3-12, the value of the "Regular Patch Mask" for the blue and yellow squares in the regular patch is "1" and the value for the red square in the regular patch is "0" because the left red patch area is nonvalid.

```
FOR I = 1: Number Of Rows Of Regular Patch Image
    FOR J = 1: Number Of Cols Of the Regular Patch Image
        IF (REGULAR PATCH MASK1 (I, J) == 1
            RESULT (I, J) = IMAGE1 (I, J) – IMAGE2 (I, J)
        END IF
    END FOR
END FOR

RETURN RESULT
```

Code 3-3. Tongue Difference Detection Using Regular Patch

After finishing the process described above, the final step is to calculate the color value difference of the regular patches. This process is very similar to the pixel-wise tongue difference detection process described in Code 3-2. The difference is that we used the color value of the regular patches instead of the color values of single pixels. To calculate the color values of a regular patch, we counted the number of valid single pixels in the regular patch and then used the average color values of these single pixels as the color value of the regular patch.

3.3.3. Whole Tongue Difference Detection

In Sections 3.2.1 and 3.2.2, we introduced two methods for the tongue differencing detection problem. Both of the methods were applied to detect the different areas of two tongue images. In this chapter, we will introduce another method for tongue difference detection. This method is different from the two methods in Sections 3.2.1 and 3.2.2 because this method is used to quantize the changing of the whole tongue. For example, Figure 3-11 shows the two tongue images of a gastritis patient before and after treatment. Figure 3-12 shows the two tongue images of healthy people. Comparing Figure 3-13 and Figure 3-14, the human eye can easily see that the two tongue images in Figure 3-13 are very different from the two tongue images in Figure 3-14. In this chapter, we propose a method to quantize the changing for the whole tongue.

After this chapter, we will use the term "contrast" to refer to the value that reflects the difference between the two tongue images. If a big contrast value exists between two tongue images, that means the two tongue images are very different, and a small contrast values indicates that the two tongue images are less different.

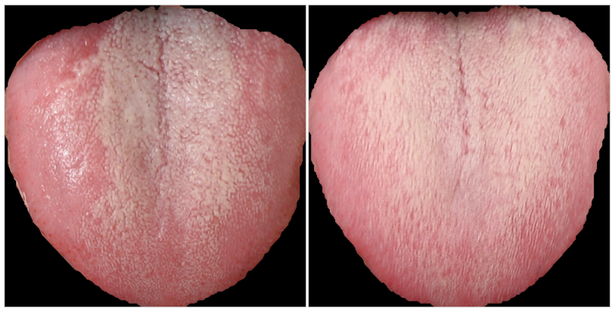


Figure 3-13. Tongue images before and after treatment

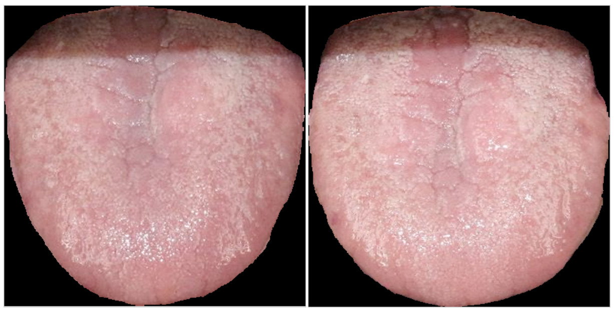


Figure 3-14. Tongue images from health people whose tongue hasn't changed.

To calculate the contrast value between two tongue images, the first step is to get the histogram as shown in Figure 3-15. This histogram describes the difference level of the two images. In the histogram, the x coordinate is the difference value of the pairs of regular patches. And the y coordinate of the histogram is the number of pairs of regular patches that have a difference value of x. For example, in Figure 3-15, the histogram indicates there are 130 pairs of regular patches that have a difference value of 0.

To get the histogram, we have to finish the image segmentation and the tongue bounding box extraction as described in Sections 3.1 and 3.2. Then we should apply the method described in Section 3.3 to cut the whole image into regular patches. After finishing these steps, we can apply Code 3-4 to get the histogram.

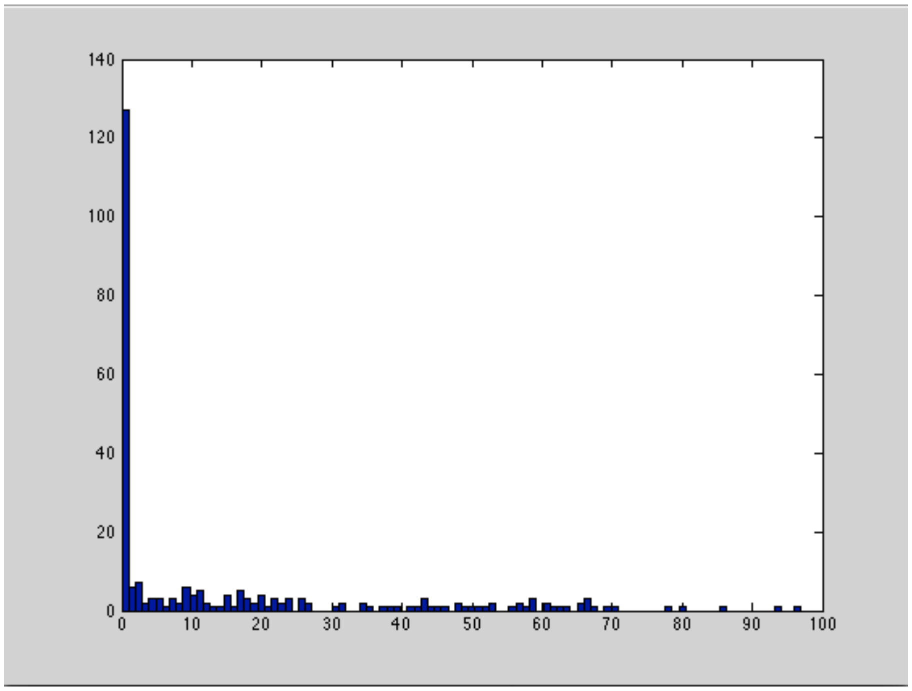


Figure 3-15. One sample result of the histogram

```
// INITIALIZE THE HISTOGRAM P
P [NUM OF POSSIBLE VALUES OF SINGLE PIXEL] = 0
// GET THE VALUES FOR P
FOR I = 1: NUM OF ROWS OF REGULAR PATCH IMAGE
    FOR J = 1: NUM OF COLS OF REGULAR PATCH IMAGE
            IF IMAGE1 [I, J] IS VALID && IMAGE2 [I, J] IS VALID
                P [IMAGE2 [I, J] – IMAGE1 [I, J]] ++
            END IF
    END FOR
END FOR
```

-

Code 3-4Contrast Histogram

In line 6 of the code, notice we should first check if the regular patch is valid or not. The way to differentiate between a valid and nonvalid regular patch is described in Section 3.3.2.

After the process of Code 4-1 is finished, the histogram will be like Figure 3-15. The reason why the histogram has the large value when the x-coordinate is small is because the color values of the tongue are similar to each other, so a large amount of regular patch values doesn’t fall far from each other.

After getting the histogram, we still cannot measure the exact difference of the two tongue images. So we proposed a method that can convert the histogram into a single value that can reflect the level of difference for two tongue images; this value is called contrast. To calculate contrast, we apply the following equation (in the notation of the equation, i denotes the x coordinate and $P(\Delta i)$ denotes the y coordinate in the histogram):

$$CONTRAST = \sum_i i^2 * P(\Delta i)$$

Chapter 4. RESULT ANALYSIS FOR THE TONGUE IMAGE DIFFERENCE DETECTION

After the introduction of the tongue difference detection algorithm, in this chapter we will analyze the algorithm results and find the advantage and disadvantage of these methods. We will discuss contrast value results in Section 4.1 and will then devote the rest of this chapter to a discussion of the results of tongue difference detection under different color spaces.

4.1. Contrast

Table 4-1 and Table 4-2 show sample results for image contrast. All of the tongue images were collected from real patients. For the test group in Table 4-1, the tongue pictures were collected from the patients before and after his/her treatment. For the test group in Table 4-2, the tongue pictures were collected from healthy people whose tongues did not show any big change after their two tongue images were taken.

By computing the contrast between two tongue images, we were able to quantitatively measure how much the whole tongue changed. Using results from Table 4-1 and Table 4-2, we computed the tongue image contrast under RGB channels. By comparing results of these two tables, we were able to find the specific channel to apply for the whole tongue changing.

Table 4-1. Patient Tongue Image Contrast in RGB Color Space

Patient Group	Contrast in R channel	Contrast in G channel	Contrast in B channel
1	832.2848	1157.9684	1086.3287
2	1053.8	731.1870	1237.1053
3	216.9108	396.3721	1923.3984
4	479.3511	1447.5875	59.1128
5	249.2467	252.0866	130.0897
6	269.8313	564.3483	757.0637
7	365.7141	161.9006	136.6870
Average	495.3056	673.0644	761.3979
Standard deviation	324.3986	477.0193	702.8703

Table 4-2. Healthy People Tongue Image Contrast in RGB Color Space

Health Group	Contrast in R channel	Contrast in G channel	Contrast in B channel
1	52.8380	96.5928	90.2699
2	195.0853	268.8607	281.9967
3	123.7797	181.1067	154.6873
4	80.0930	186.2128	157.9837
5	62.3408	261.8106	182.7334
6	93.2111	196.5217	166.8707
7	145.5666	234.1515	206.6120
Average	107.5592	203.6081	177.3077
Standard deviation	50.6094	59.0053	58.3346

From the results in these tables, we can see that the contrast for the patient group is obviously larger than the health group. To detect if the whole tongue has changed, in the current version of the tongue difference detection, we use the threshold 200 for the contrast under the R channel. Using this threshold, we can accurately detect the change of the whole tongue.

One interesting thing we noticed is that in the patient group, the standard deviations are much larger than the healthy group. The reason for this is that there is no universal model for the tongue changing in tongue analysis. For patients with different diseases, their tongues change in different ways, so that's why the standard deviation for the patient group is very large. However, healthy people do not normally show significant changes after tongue analysis so the contrast for the healthy group is relatively stable.

4.2. Tongue Difference Detection in the RGB Color Space

The RGB color space is the most common color space of computer science. Because the R channel is the most significant channel in tongue imaging, the RGB color space is a very suitable color space for tongue difference analysis.

In the RGB color space, the tongue difference detection algorithm detects the difference of tongue features including differences in the R, G, and B channels. The algorithm also calculates the Euclidean distance of the color from two corresponding pixels using the following equation.

$$Distance = \sqrt[2]{\Delta R^2 + \Delta G^2 + \Delta B^2}$$

Also, as introduced in Section 2.2, the Euclidean distance in the RGB color space does not correspond to the color difference that the human eye can see, because human eyes are most sensitive to green color, and least sensitive to blue. To better simulate the color distance that human eyes can perceive, we also tried to use weighted distance to see if it could provide better results. The parameters are the same one as the ones to calculate luma in CCIR 601 [8].

$$Weighted\ Distance = \sqrt[2]{(\Delta R * 0.299)^2 + (\Delta G * 0.587)^2 + (\Delta B * 0.11)^2}$$

The following figures are the sample results of the tongue image differencing algorithms. The process of how we obtained the results is described in Code 3-3. To better display the results, the difference value matrix was displayed in the form of a height map. Also, the values were color-coded; the red value stands for large difference values and blue stands for small difference values. All the results in the following sections of this chapter are also displayed in the same way.

These following sample results are for a gastritis patient.

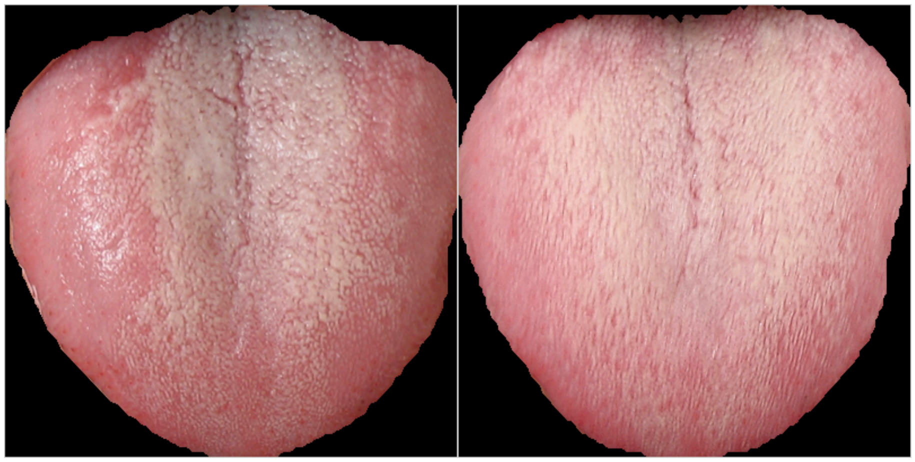


Figure 4-1. The input tongue images for the tongue difference detection (From gastritis patient)

Experiment results have shown that the tongue difference detection algorithm provides very similar results using single channels in RGB color space. As shown in Figures 4-2 to 4-4, all R, G and B channels can detect the changing of the white coating.

Even though R, G and B channel can all detect the changing area of the tongue, we found that color channels are sensitive to different kinds of tongue changings. For example, the blue channel is not sensitive to the yellow coating changing. Because of this reason, in the current version of the algorithm, when we want to detect tongue difference in RGB color space, we use all of R, G and B channels instead of only one of them.

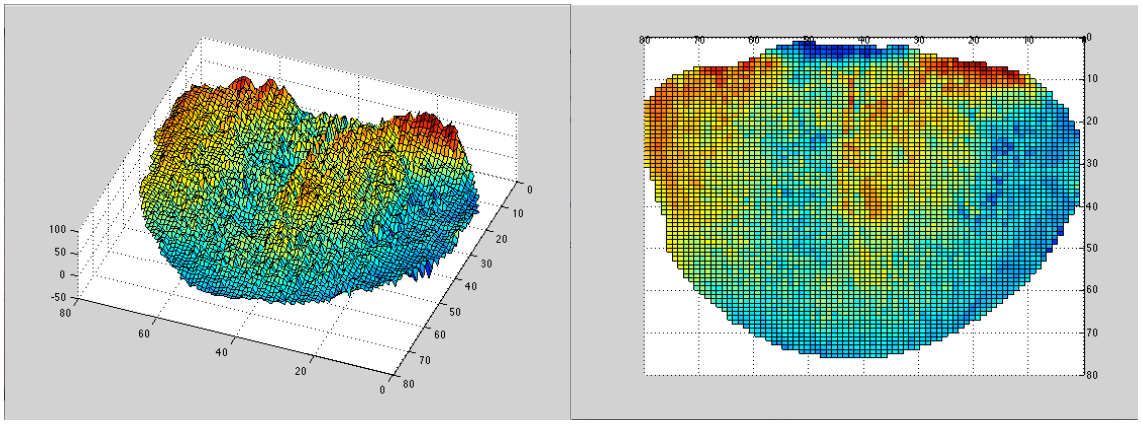


Figure 4-2. Tongue changing detection result in the R channel

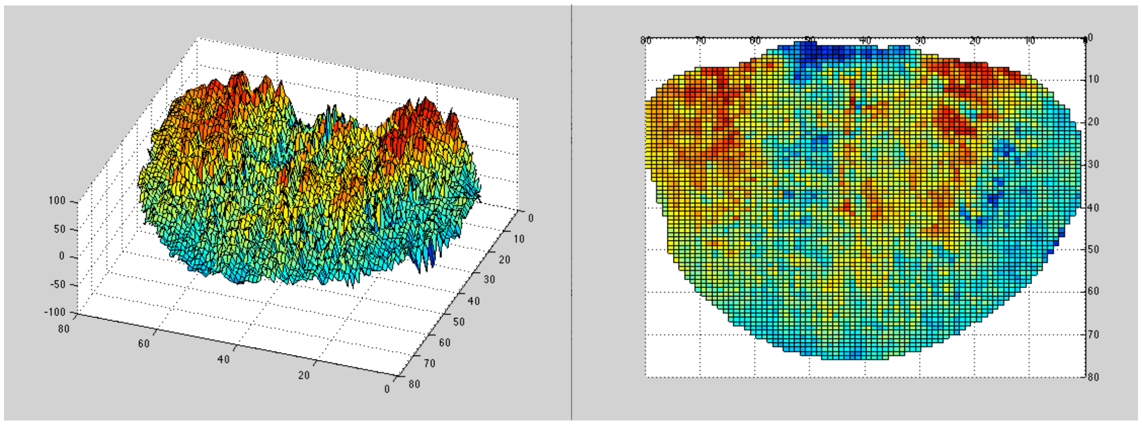


Figure 4-3. Tongue changing detection result in the G channel

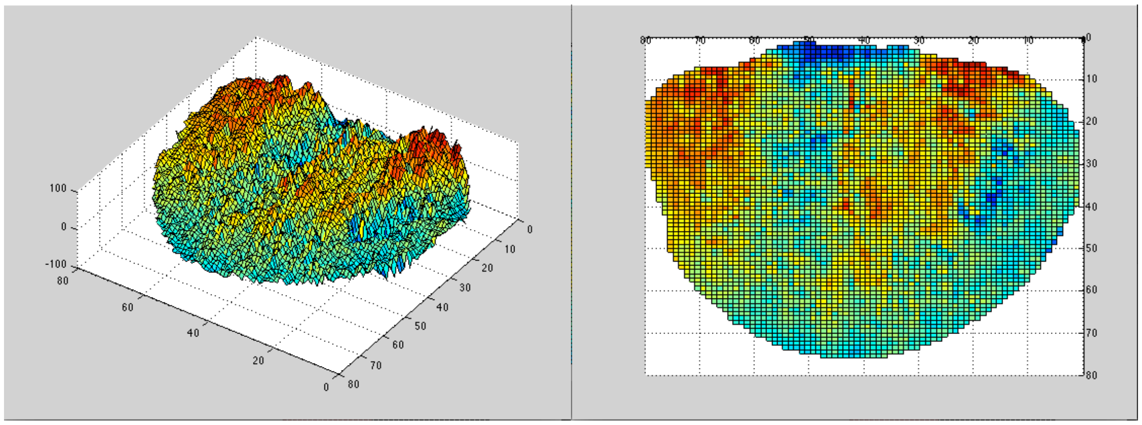


Figure 4-4. Tongue changing detection result in the B channel

Using Euclidean distance can also provide very similar results to those shown when applying single channels of the RGB color space to tongue changing detection. Figure 4-5 shows how using the Euclidean distance can help detect the change of the white coating on tongue. The result of using the weighted Euclidean distance and Euclidean distance are very similar as illustrated by Figure 4-5 and Figure 4-6.The disadvantage of applying Euclidean distance and weighted Euclidean distance is that we can only see how much the color changed, but we cannot see the direction of the color change.

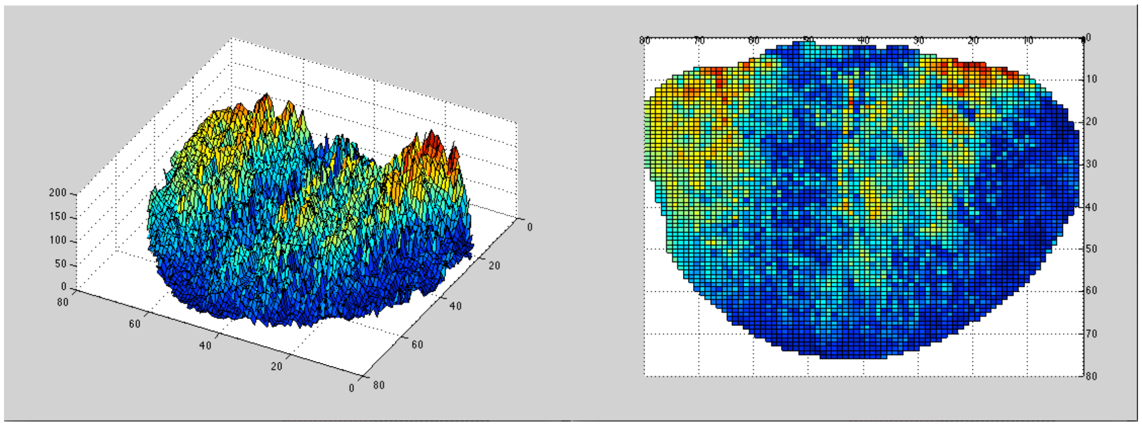


Figure 4-5. Difference detection result using Euclidean Distance

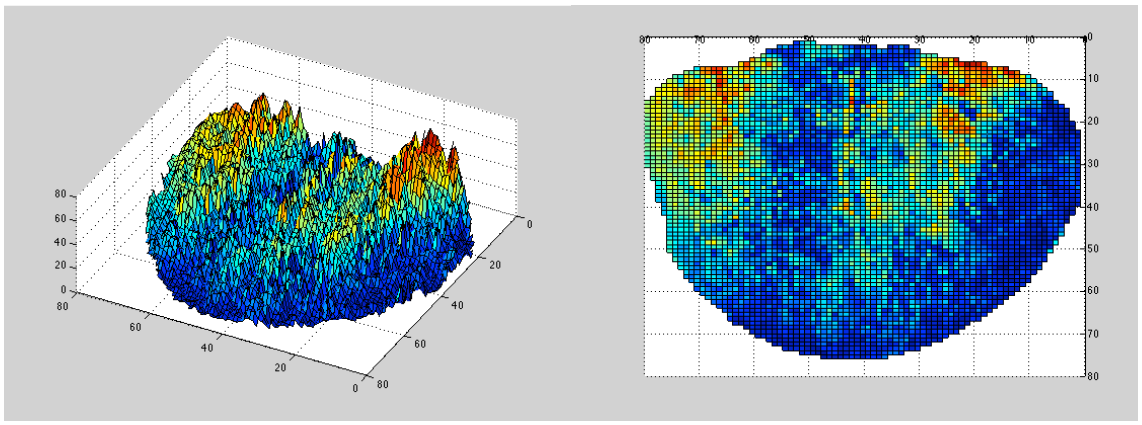


Figure 4-6. Weighted Difference detection result using Euclidean Distance

4.3. Tongue Difference Detecting in the Lab Color Space

In the Lab color space, the tongue changing differences as shown in the detection algorithm uses the L, a and b channel for the difference detection.

As introduced in Section 2-3, the L channel in Lab color space stands for Lightness, and the a, b channel stands for color-opponent dimensions, which means we only use single channels to detect the color changes on the tongue. The tongue difference detection algorithm also detects the Euclidean distance for color dimensions:

$$Color\ Distance = \sqrt[2]{\Delta a^2 + \Delta b^2}$$

The following figures show sample results of the tongue difference detection in Lab color space. Figure 4-7 shows the tongue images of a gastritis patient before and after treatment.

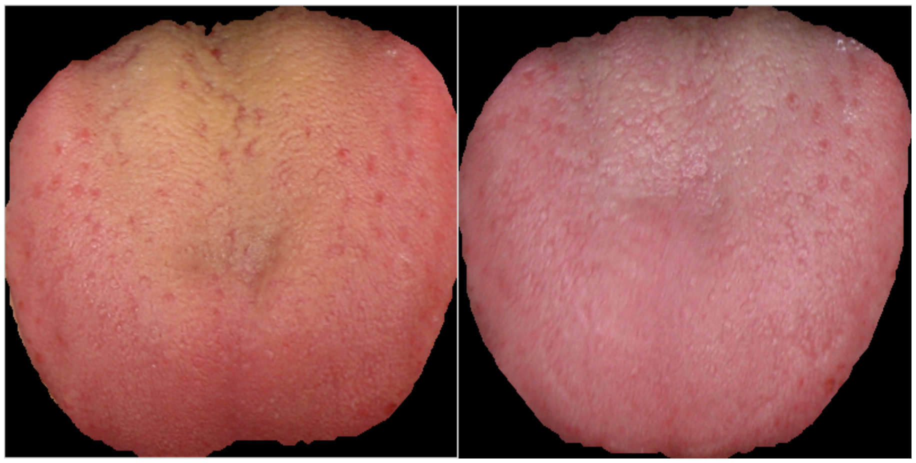


Figure 4-7. The input tongue images for the tongue difference detection (from gastritis patient)

As introduced in Section 4.2, in RGB color space, the tongue difference detection detects nearly the same positions that have changed based on R, G and B channels. But, the situation is different in Lab color space. Since L, a and b have different meanings, these three channels will give different predictions about the position on the tongue that has changed. For example, Figure 4-8 shows how using the tongue difference detection algorithm under the L channel will help us to see the position on the tongue where lightness has changed; hence, the tip of the tongue has a large value in the difference detection algorithm; this means that in this area, the second tongue has more lightness than the first one. For the same reason, the blue area on Figure 4-8 means the second tongue has less lightness than the first one. Since this discussion of the tongue difference detection problem is restricted to color change, we do not discuss the L channel's use in tongue difference detection.

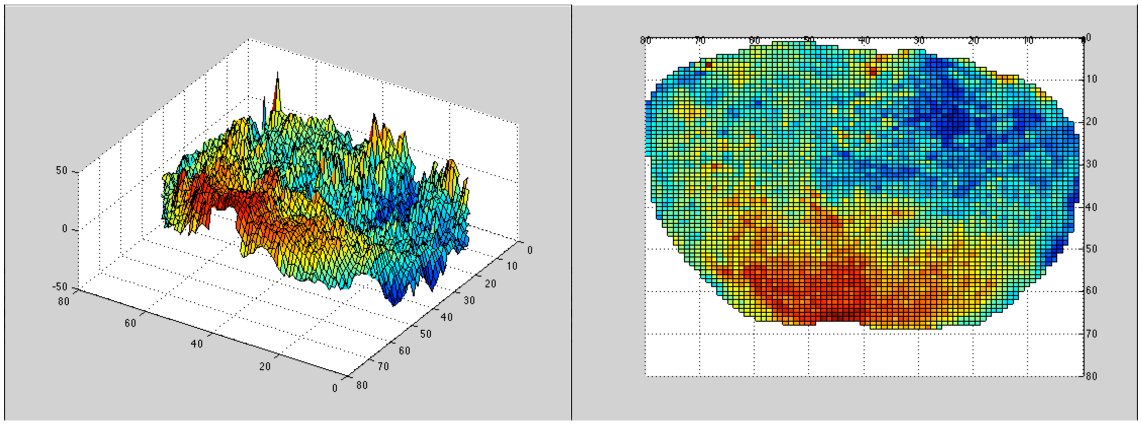


Figure 4-8. Difference detection result for the L channel

Figure 4-9 shows the sample results for the *a* channel. From the introduction in Section 2.3 we know that the *a* channel reflects a color change from green to red. So the positive values in this result means the in the corresponding area, the color of second image is closer to red than in the first image. And for the negative values in the corresponding area, the color of the first image is closer to the red color than the second image. The experiment results have shown that using this color channel, we can detect the changes from a healthy tongue color to other colors. For example, in Figure 4-9, we can see the area in the middle where positive *a* values in the corresponding area indicates that the second image is redder than the first image. Because of this *a* channel property, by applying detection in a channel for the patient, we can monitor which areas on the tongue are recovering well and which areas are becoming worse.

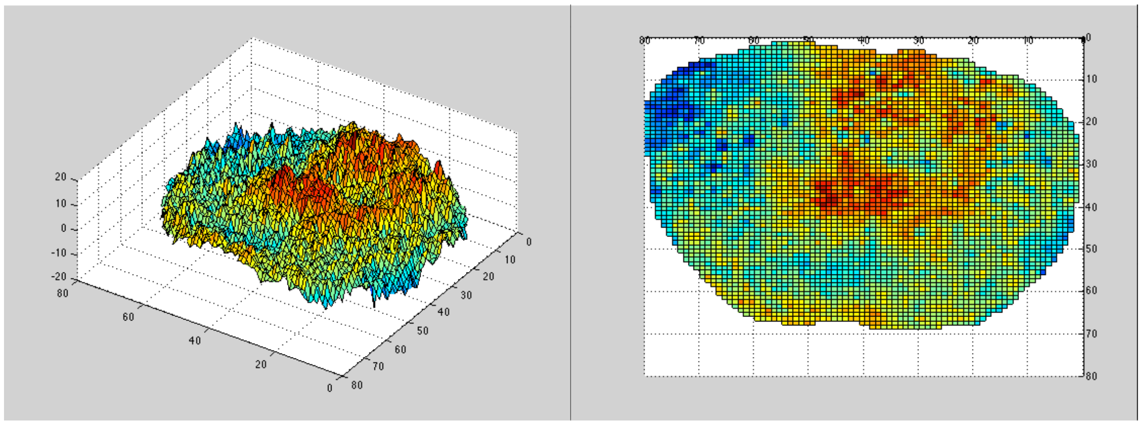


Figure 4-9. Difference detection result for the a channel

Figure 4-10 shows the sample result for the *b* channel. From the introduction in Section 2.3 we know the *b* channel reflects the color change from blue to yellow. So the positive values in the result means that in the corresponding area, the color of second image is closer to yellow than the first image. And the negative values in the corresponding area indicate that the color of the second image is a lighter yellow than the first image. Because of this *b* color channel property, it is more suitable for detecting the yellow coating on the tongue. Figure 4-10 shows that in the middle of the tongue, there is an area of negative values, which indicates that the second tongue is less yellow than the first tongue in the corresponding area. Thus, Figure 4-7 conforms to our observation results.

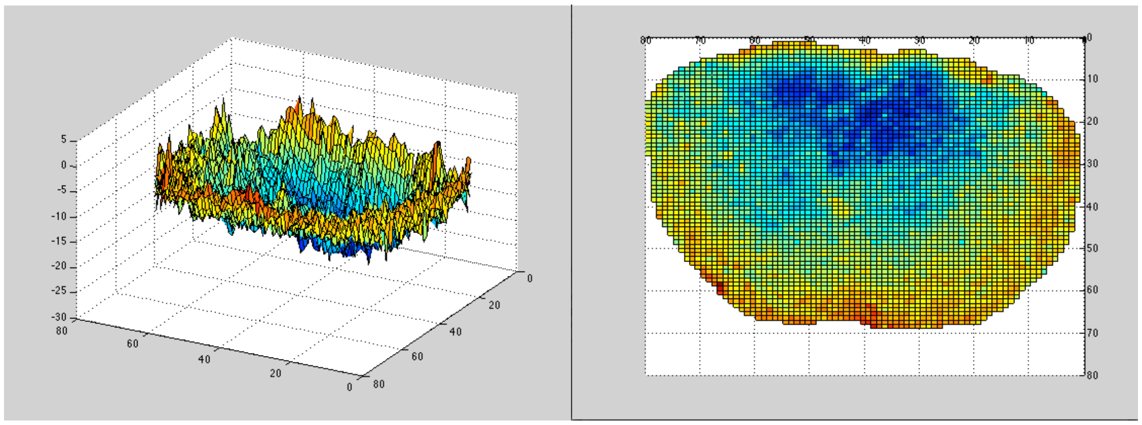


Figure 4-10. Difference detection result for the b channel

Figure 4-11 represents the Euclidean distance of the two color channels. Unlike the result using single channels *a* and *b* that can tell how the color on the tongue has changed, this feature simply tells whether the different colors negative or positive as they relate to zero in the corresponding area. Hence, in Figure 4-11, the red color represents a large positive value, and the blue color represents values that are close to zero. Hence, using this feature allows detection of the area that has changed most—which is in the middle of the tongue, while the area that changed the least is the tip of the tongue.

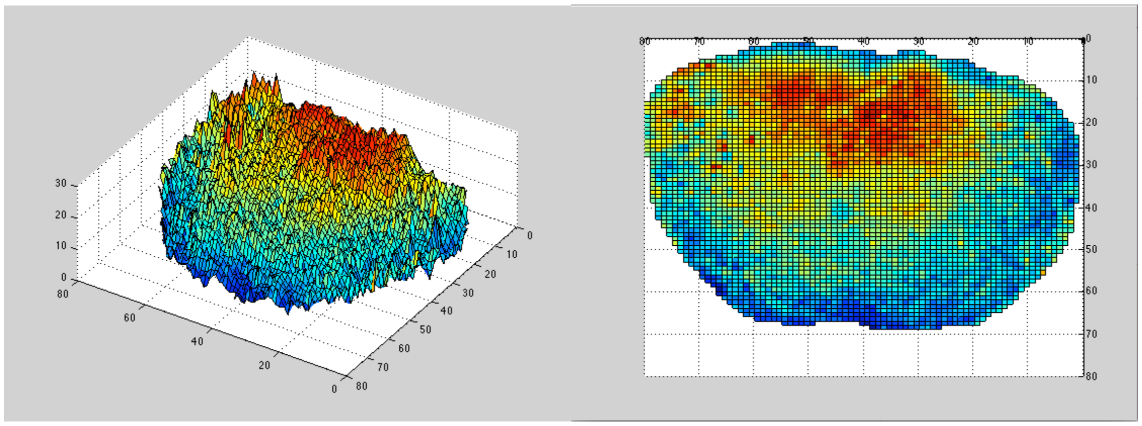
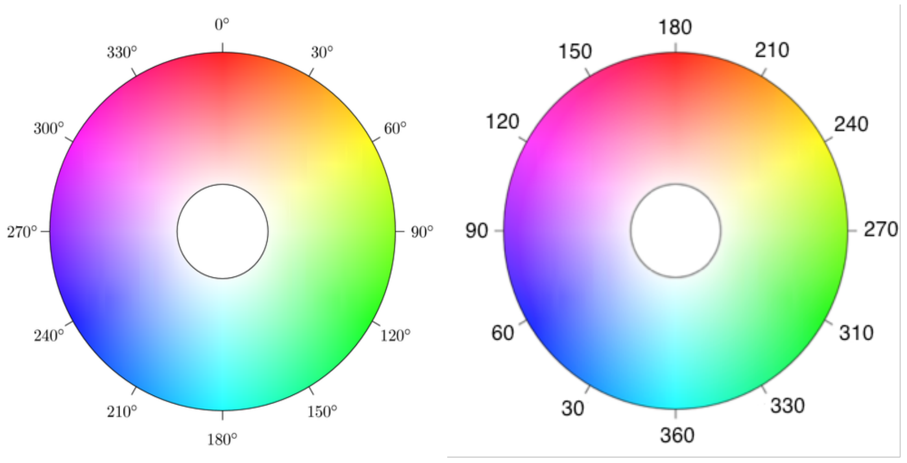


Figure 4-11. Euclidean distance $\sqrt[2]{\Delta a^2 + \Delta b^2}$

4.4. Tongue Difference Detection in Hsv Color Space



(a) (b)

Figure 4-12. The Hue value in the Hsv color space (Hue in Hsv)

Before applying the tongue difference algorithm under Hsv color space, some preprocessing is necessary because of the specificity of the Hsv color space. As described in Section 2.4, the H channel in the Hsv color space represents the hue value. As Figure 4-12 shows, the hue value around 0 or 360 degrees represents the red color, which is the most significant color in tongue imaging. The experiment results have shown that for a tongue image, all the hue value of the pixels fall into the range from 240–360 and 0–120. Direct use of the original Hsv color values will give poor results in the tongue differencing algorithm. For example, for two tongue images where we want show the differences in color, especially when there is a great difference, one pixel on the tongue might have the hue value of 1 degree while the corresponding pixel on the other image has a hue value of 359 degrees. These two values are both red and they are just 2 degrees from each other indicating location means very little. But when calculating the hue value difference between the two pixels using direct subtraction, we obtain the difference value of 358 degrees, which is a very large difference. In order to handle this situation, we rotate the axis of the hue value by 180 degrees. In the new axis, the hue value of the red colors now correspond to other red values at around 180 degrees. Also because the vast majority of the pixels of the tongue are very close to red, we won't run into the same trouble because there are no values around 0 degree in the new axis.

After the preprocessing step described above, we can use Hsv color space for our tongue image difference detection.

The following figures show the sample results for the tongue difference detection algorithm using Hsv color space. Figure 4-13 shows the tongue images of a dyspepsia patient before and after treatment.

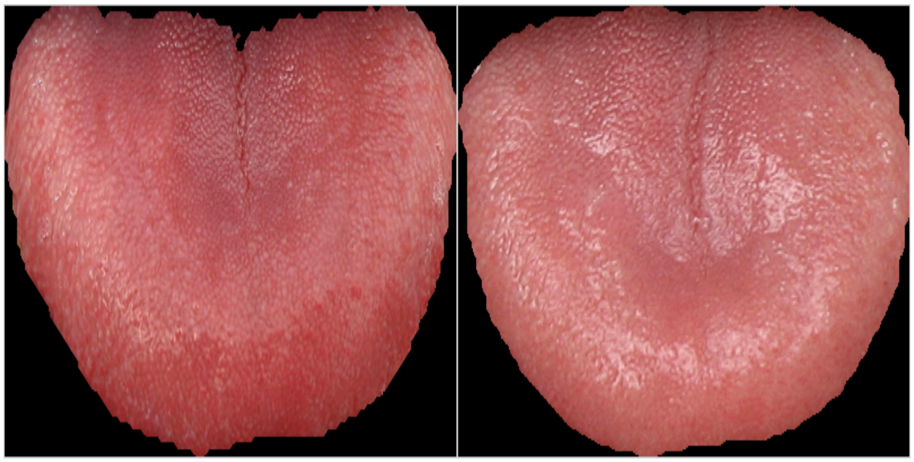


Figure 4-13. The input tongue images for the tongue difference detection (from dyspepsia patient)

Since only H represents the color, the change of H can accurately detect the color change on the tongue. Figure 4-14 shows how the tongue difference detection algorithm can accurately detect the changes on the tongue.

Concerning the H channel, since *s* and *v* are not related to color, *s* and *v* channels won't be used for color difference detection.

Figure 4-14 shows how using the difference detection algorithm with the H channel can detect areas where the surface of the tongue changed from yellow coating to red. Like the *a* and *b* channels in Lab color space, positive or negative values indicate in what direction the color has changed. In this example, on the tip of the tongue, the second image has smaller H values than the first image. According to Figure 4-12 (b), we know that in the corresponding area, the tongue has become less yellow after the treatment.

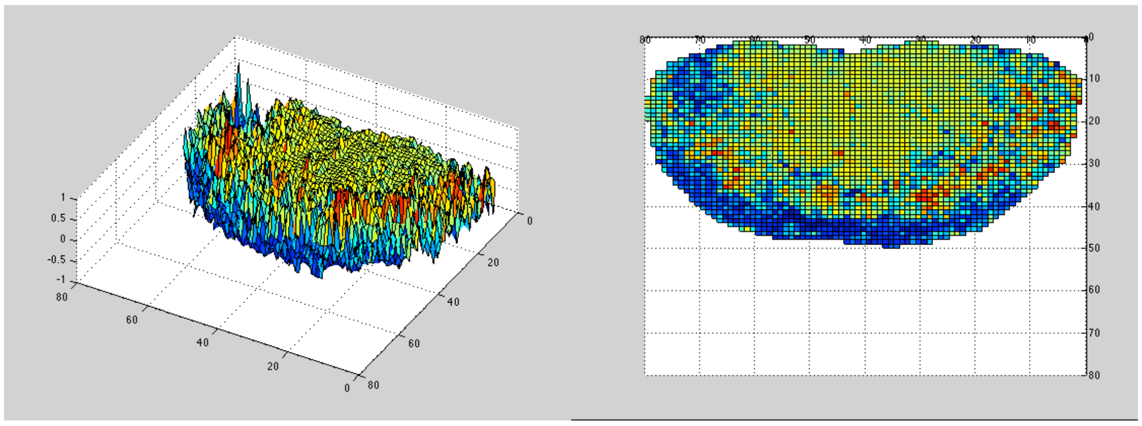


Figure 4-14. Difference detection result for the H channel

Since the *s* and *v* channel are not related to the color, the tongue difference detection algorithm under *s* and *v* channels cannot detect color changes on the tongue surface. From Figure 4-15 and 4-16, we can see that the results using *s* and *v* channels are relatively flat. But one thing to notice about the result is that we can detect the lightness change on the tongue. This could hopefully be a useful feature to enable correction of the lightness condition in future work.

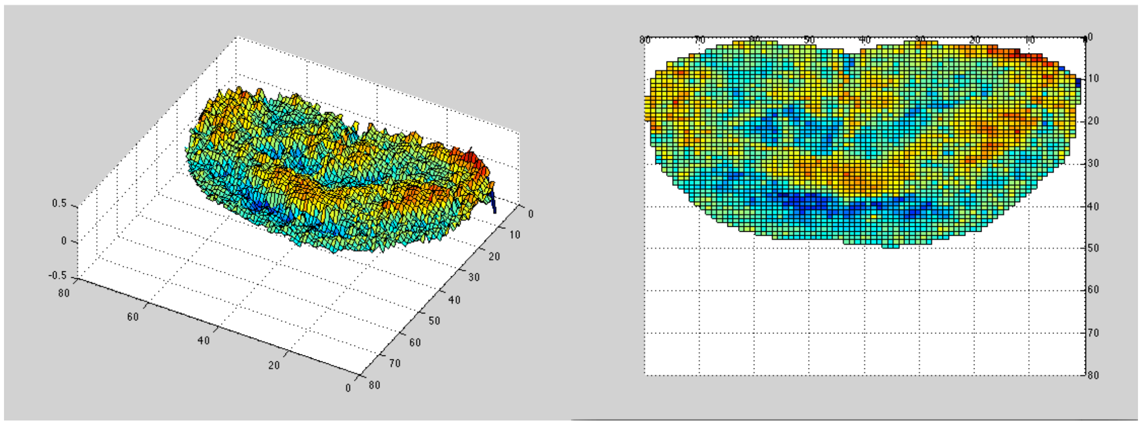


Figure 4-15. Difference detection result for the s channel

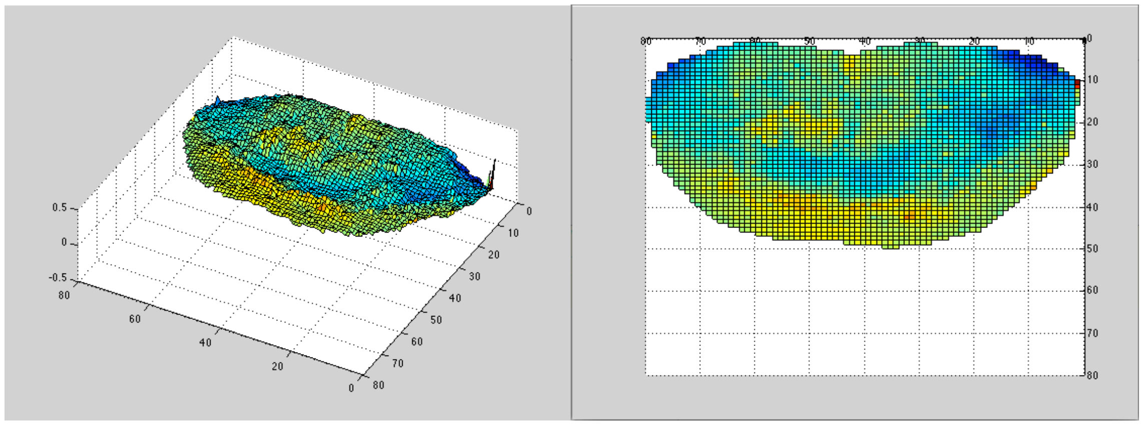


Figure 4-16. Difference detection result for the v channel

4.5. Threshold For Difference Area Detection

The results in Sections 4.2 to 4.4 have shown the height map of the tongue difference detection. However, the height map alone is not enough, so the next step is to find the threshold to tell if the tongue area has changed or if it is not based on the height map. To get the threshold for image changing detection, according to Rosin, we should have either the noise or the colored noise signals modeled, and the model covers either the spatial or intensity distribution characteristics [9]. But currently we don't have a universal model for the tongue image, so in this thesis, a statistic method was applied to get the threshold. We used the data in the following tables to help us find the threshold. We obtained the data in Table 4-3 by following three steps: First, we manually picked 30 pairs of pixels on the tongue images that had changed. Second, we calculated the difference value of these pixels in R, G, B, *a*, *b*, and H channels. Except for the difference value in single channels, we also calculated the color distance in RGB, weighted color distance in RGB, and color distance in Lab (the equations for the color distance are introduced in Sections 4.2 and 4.3). Third, we calculate the mean and standard deviation of these data under the channels. In order to get the data in Table 4-4, we manually picked 30 pairs of pixels on the tongue images that had not changed, and then repeated the same process as for Table 4-3.

Table 4-3. Tongue Difference Value for changed pixels

	R	G	B	*a*	*b*	H
Mean	66.3832	64.6933	82.1923	14.4527	19.5652	40.6719
Standard Deviation	11.0026	11.4837	12.5621	2.7665	2.8114	8.0504

	RGB Distance	RGB Weighted Distance	Lab Distance
Mean	117.4836	53.9428	23.9249
Standard Deviation	32.4725	6.2305	2.3264

Table 4-4. Tongue Difference Value for non-changed pixels

	R	G	B	*a*	*b*	H
Mean	32.4547	27.3077	25.4853	6.3840	5.9257	8.7159
Standard Deviation	9.8645	10.6716	14.4364	2.9404	2.7485	8.0461

	RGB Distance	RGB Weighted Distance	Lab Distance
Mean	38.4873	21.0019	12.7146
Standard Deviation	13.3255	6.8801	4.4303

According to the data in Tables 4-3 and 4-4, in the current version of the tongue difference detection algorithm, we chose the following threshold listed in Table 4-5.

Table 4-5. Threshold values for the tongue difference detection

	R	G	B	RGB color Distance	RGB weighted color distance
Threshold	49.4190	46.0005	53.8388	77.9855	37.4723

	a	*b*	Lab color distance	H
Threshold	10.4184	12.7454	18.3197	24.6939

One thing we should pay attention to is that the results in Table 4-3, 4-4 and 4-5 are related to light conditions and the camera. Once the light condition or camera is changed, we have to collect the data again to find the right threshold for different area detections.

After having the thresholds listed in Table 4-5, we can now use the output matrix in Code 3-3 to tell which parts of the tongue have changed.

```
FOR I = 1: Number Of Rows Of Regular Patch Image
    FOR J = 1: Number Of Cols Of the Regular Patch Image
        IF (RESULT(I,J) > THRESHOLD) == 1
            CHANGE_AREA(I,J) = 1
        END IF
   END FOR
END FOR

RETURN RESULT
```

Code 4-1. Detect changing areas on the tongue

After running the Code 4-1, the result is saved in the matrix called "CHANGE_AREA" in the code. In the code, all the elements that are equal to "1" labels the areas on the tongue that are changed.

Chapter 5. THE ITONGUE SYSTEM ARCHITECTURE

The "iTongue" project is a CAD technique used to diagnose human health automatically. This system is able to let users obtain their health status within a few minutes. Users need only to use their ios device to take a picture of his/her tongue and the iTongue system will be able to diagnose his/her health status and send back the information. This system consists of the client side and server side of the program. The client side of the system is an ios app that can run on any ios device with cameras. The system architecture is as Figure 5-1 shows.

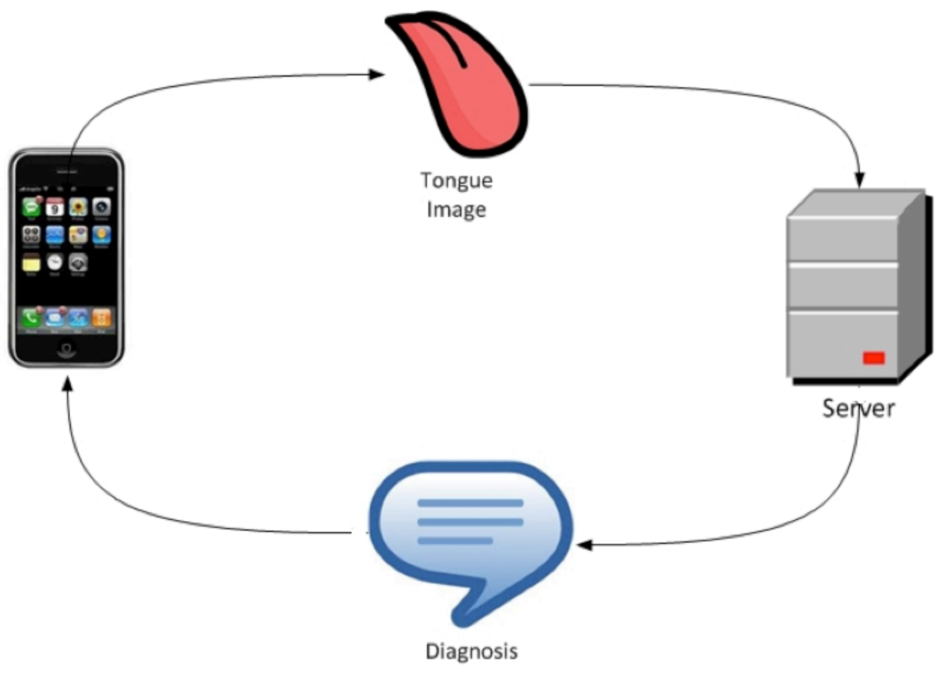


Figure 5-1. iTongue system architecture

5.1. iTongue Client app

The main functions of the app include taking photo, uploading photo to the server and keeping the medical advice that the iTongue system provides to the user.

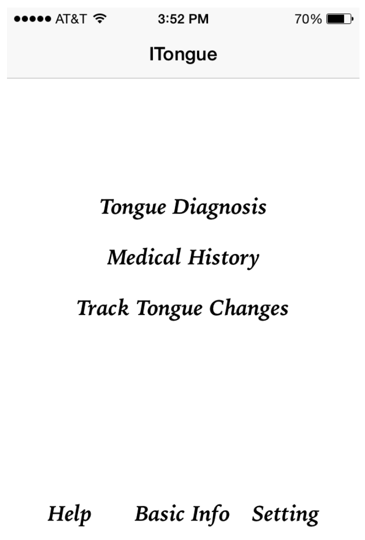
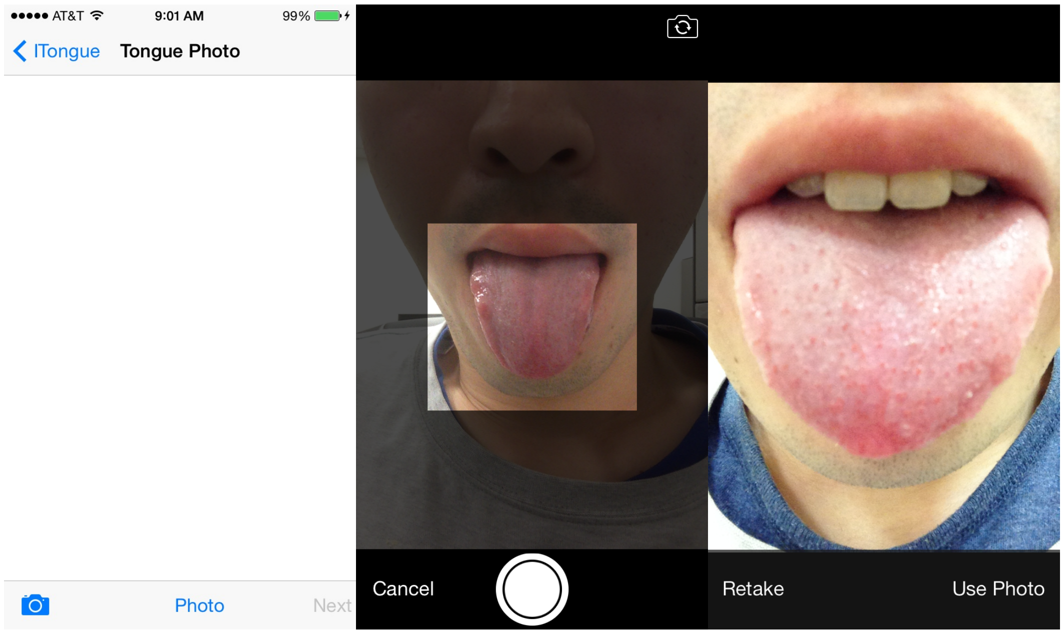


Figure 5-2. Home screen of the iTongue app

From the home screen in Figure 5-2, we can see the main functions of the iTongue system. The app allows the user to get diagnosis from their tongue image, keep track of their medical history and see their tongue differencing result.

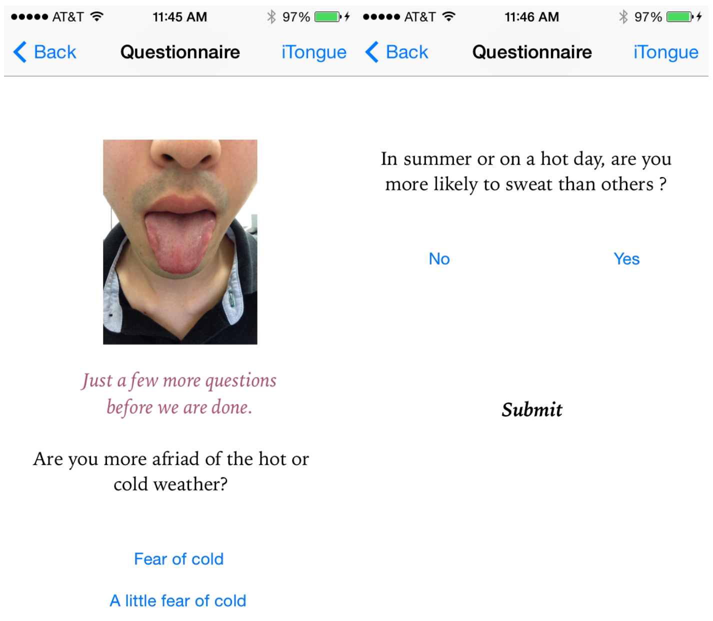
The process used to get the diagnosis result is very simple: The user only needs to click on the “Tongue Diagnosis” button, and then the interface on Figure 5-3 (a) will show up. On this interface, the user can decide if he/she wants to shoot a new photo or use the older photos taken previously.



(a) (b) (c)

Figure 5-3. User Interfaces to get new Diagnosis

Then after choosing the tongue image or taking the new tongue image as shown in Figure 5-3 (b), there is a questionnaire shown in Figure 5-4 that includes eight questions the user can fill out.



Just a few more questions before we are done.

Are you more afriad of the hot or cold weather?

In summer or on a hot day, are you more likely to sweat than others ?

Figure 5-4. Questionnaire

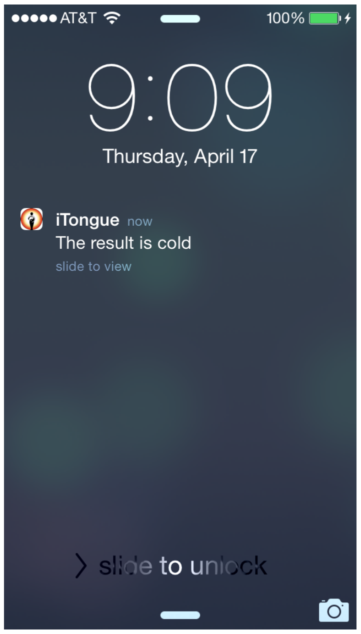
The questionnaire is optional, but we recommend the user to fill it out because it can improve the accuracy of the diagnosis result, the questionnaire includes the following questions:

1. *Are you more afraid of the hot or cold weather?*
2. *Do you feel your hands and feet warm or cold?*
3. *Do you prefer hot or cold food and drink on daily basis?*
4. *Typical stool texture*
5. *Do you often get diarrhea after eating raw or cold food?*
6. *D you often flush and feel thirsty?*
7. *In cold day, do you need to wear more clothes than others?*
8. *In summer or on a hot day, are you more likely to sweat than others?*

If the user answers all the questions, we will get a diagnosis in the form of “hot, cold and normal” based on the answers. Then we can combine the diagnosis of the questionnaire with the diagnosis based on the tongue image using the following strategy.

1. **If the two diagnoses are the same, return the diagnosis.**
2. **If one of the diagnoses is normal and the other one is not, return the diagnosis that is not normal.**
3. **If two of the diagnoses are “hot and cold,” we return the diagnosis based on the tongue image.**

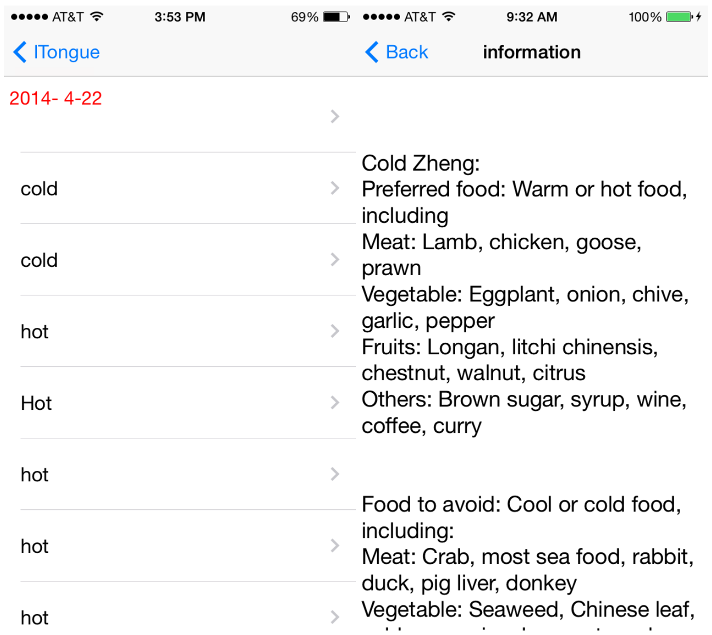
After pushing the submit button at the bottom of the questionnaire, the photo will be sent to the server. In order to send the photo faster and occupy less bandwidth, in the current version of the iTongue app, before sending the photo, we use the JPEG algorithm to compress it. Then after the server receives the photo, the server will run the whole diagnosis process to get the result for the user. Then the diagnosis will be sent back in the form of a push notification as shown in Figure 5-5.



The result is cold

Figure 5-5. The push notification result

In order to help the user to keep track of their diagnosis history easier, we added the "Medical history" interface for the user. By pushing the "Medical History" button on the home screen, the user will be able to see the user interface as shown in Figure 5-6 (a). In order to let the user keep of his/her medical history easier, we categorize the entries into different dates. If the user is interested in any special entry, he/she can simply push on it and another user interface as shown in Figure 5-6 (b) where details will show up. In this user interface, we provide advice that improves the user's health based on his/her information.


●●●●● AT&T 3:53 PM 69%

ITongue

2014- 4-22

cold

cold

hot

Hot

hot

hot

hot

●●●●● AT&T 9:32 AM 100%

Back information

Cold Zheng:
Preferred food: Warm or hot food, including
Meat: Lamb, chicken, goose, prawn
Vegetable: Eggplant, onion, chive, garlic, pepper
Fruits: Longan, litchi chinensis, chestnut, walnut, citrus
Others: Brown sugar, syrup, wine, coffee, curry

Food to avoid: Cool or cold food, including:
Meat: Crab, most sea food, rabbit, duck, pig liver, donkey
Vegetable: Seaweed, Chinese leaf,

(a) (b)

Figure 5-6. Medical history user interface

Notice that diagnosis results are sent back to the user as “cold, hot and normal”. These terms here don’t mean temperature as it relates our daily use. The use of “cold hot and normal” here corresponds to the “yin” and “yang” concepts in the Chinese medical theory. Chinese ancient medical scientists use “Yin Yang Wu Xing” Theory extensively in the traditional treatment to explain the origin of life, human body, pathological changes, clinical diagnosis and prevention. It has become an important part of the Traditional Chinese Medicine [10]. The idea is that cold means the body status of the person is inactive, so he/she needs more exercise and eat more “hot” food like lamb and eggplants to bring his/her body to balance. The term hot means the body status of the person is to active, so he/she needs to reduce workload and eat food like crab and seaweed to get balance. Below are two-detailed list of food recommendation we provide to the user for their “hot and cold” diagnosis.

Preferred food:
Meat: Crab, most seafood, duck.
Vegetable: Seaweed, Chinese leaf, cabbage, spinach, carrots, celery, cucumber, wax gourd, gourd, lotus root, tomatoes, green been.
Fruits: Pears, watermelon, melon, persimmon, banana, pineapple
Others: Sugar, ice sugar, tofu.
Food to avoid:
Meat: Lamb, chicken, goose, prawn
Vegetable: Eggplant, onion, chive, garlic, pepper
Fruits: Chestnut, walnut, citrus
Others: Brown sugar, syrup, wine, coffee, curry

Food Recommendation 1: For users who get hot diagnosis

Preferred food:
Meat: Crab, most seafood, duck.
Vegetable: Seaweed, Chinese leaf, cabbage, spinach, carrots, celery, cucumber, wax gourd, gourd, lotus root, tomatoes, green been.
Fruits: Pears, watermelon, melon, persimmon, banana, pineapple
Others: Sugar, ice sugar, tofu.
Food to avoid:
Meat: Lamb, chicken, goose, prawn
Vegetable: Eggplant, onion, chive, garlic, pepper
Fruits: Chestnut, walnut, citrus
Others: Brown sugar, syrup, wine, coffee, curry

Food Recomendataion 2: For users who get cold diagnosis

Another function of the iTongue app is to keep track of the tongue changes. To detect the tongue changes, we use the tongue difference detection algorithm described in this thesis. To open this function, the user can click the “Track Tongue Changes” button on the home screen. After entering the new screen for this function like figure 5-7, the user can see his/her tongue image difference detection result. Then the user can slide down to see his/her normalized tongue image and tongue difference detection result with the normalized tongue. We get the normalized tongue image by taking the average value of all the previous tongue images for the user. By getting the normalized tongue image, we can see the base line of this user's tongue surface.

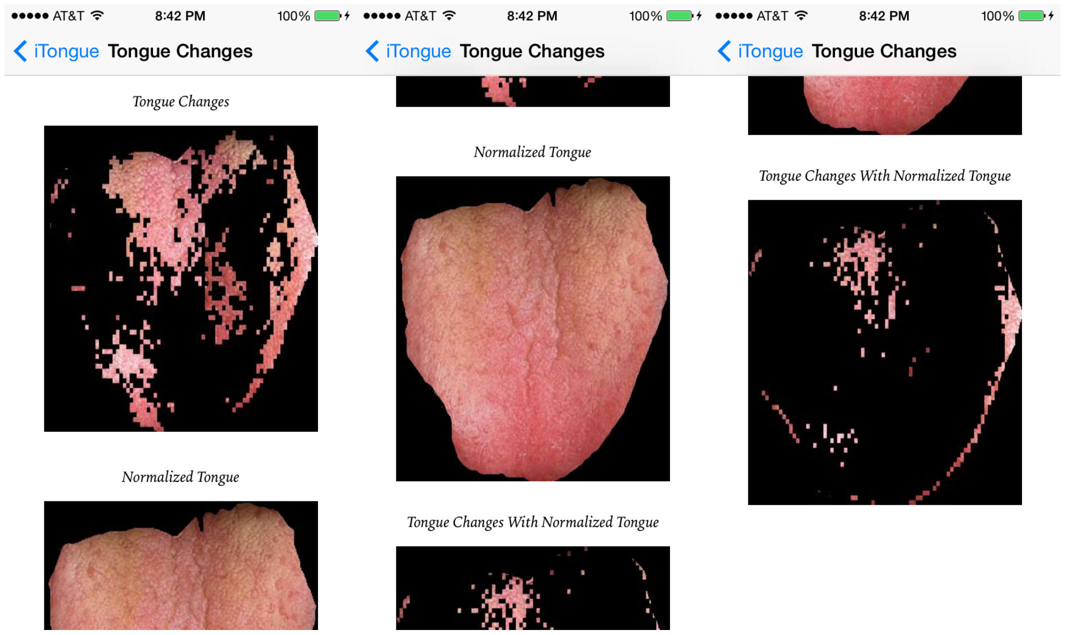


Figure 5-7. Tongue difference detection result on the iTongue app

5.2. iTongue Server

The server side program is realized in PHP script. The server program has five functions. (1). When user uploads his/her tongue image, the server will save it in his/her personal image folder and then save the new image information into the MySQL database. (2).

The server runs the MATLAB (matrix laboratory) program to do the tongue image segmentation. (3). The server runs the MATLAB program to detect the difference area on the tongue. (4). The server uses LIBSVM (A Library for Support Vector Machines) to get the diagnosis for the patient. (5). The server sends the diagnosis back to the server by push notification.

The flow chart below in Figure 5-8 is the working process of the server program after it receives a user-uploaded photo.

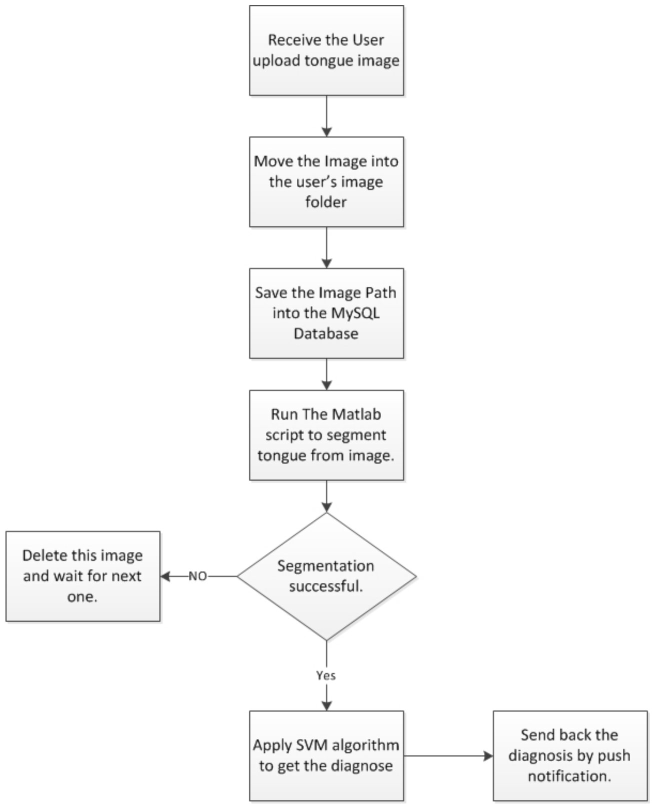

Receive the User upload tongue image
Move the Image into the user's image folder
Save the Image Path into the MySQL Database
Run The Matlab script to segment tongue from image.
Segmentation successful.
NO
Delete this image and wait for next one.
Yes
Apply SVM algorithm to get the diagnose
Send back the diagnosis by push notification.

Figure 5-8. Working process of the server side of iTongue system

In the previous chapters in this thesis, we've already introduced the tongue difference detection algorithm. Here, we'll briefly introduce the tongue segmentation algorithm and SVM algorithm for the diagnosis in Sections 5.2.1 and 5.2.2.

5.2.1. Tongue Segmentation Algorithm Using GrowCut

The most detailed information of the segmentation algorithm can be found in Wenchuan Qi's thesis submitted to University of Missouri 2013 in [6]. Here, we'll just briefly introduce this algorithm.

GrowCut is a segmentation algorithm based on cellular automaton. It labels pixels as seeds with different values in the background area and the object area needed. The labeled pixels in the different areas are considered as good and bad cells. The unlabeled pixels are considered as free space to be filled by good or bad cells. Both kinds of cells keep growing and fight with different cells (enemies) to seize larger area and conquer other's positions. When the growing in the whole image stops, the final distribution of different cells' area is the segmentation result [6].

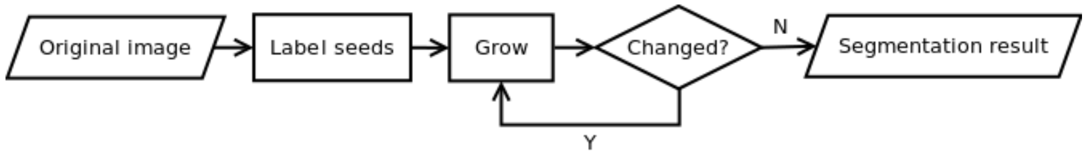


Figure 5-9. GrowCut process flow (Wenchuan Qi 2013)

The labeling process of the segmentation of the iTongue project is simplified. As shown in Figure 5-10 in the label map, the square seed is labeled with -1 and a circle seed (white circle in Fig. 3-5) is labeled with 1, all other areas (unlabeled gray area in Fig. 3-4) is gray with value 0 [6].

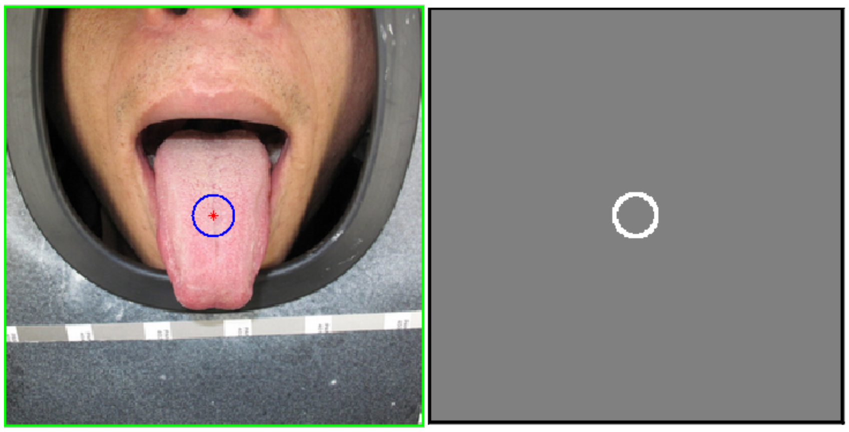


Figure 5-10. iTongue seed position and initial seeds in label map (Wenchuan Qi 2013)

After the labeling for inside and outside areas of the tongue, then the growing process will try to grow toward the inside area and outside area of the tongue until the growing process ends.

After the growing process, as introduced in Section 3.2, the algorithm will be checked for over segmentation. The first step is to use the following equation to get the C value.

$$C = \frac{Perimeter^2}{Area}$$

Triangle is the polygon with the highest C value (C_t). Circle is the polygon with the lowest C value (C_c). If the tongue is segmented correctly, its C value should be bounded to (C_c, C_t). Otherwise, it may be over segmented [6].

If you would like to know any more detailed information about the segmentation algorithm, please see Wenchuan Qi's thesis submitted to University of Missouri 2013.

5.2.2. SVM Algorithm for Diagnosis

After the segmentation step, we will have the original image and binary mask as figure 5-11 shows.

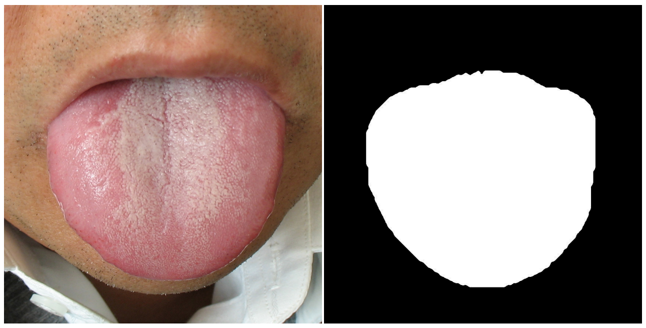


Figure 5-11. Original image and binary mask

Then using the original image and mask, we can compute the average tongue color under R, G and B channel using Code 5-1.

```
#_OF_VALID_PIXELS = 0
FOR I = 1: NUMBER OF ROWS Of ORIGINAL IMAGE
    FOR J = 1: NUMBER Of COLS Of ORIGINAL IMAGE
        IF (MASK (I, J)) == 1
            R = R + ORIGINAL IMAGE (I, J, R);
            G = G + ORIGINAL IMAGE (I, J, G);
            B = B + ORIGINAL IMAGE (I, J, B);

            #_OF_VALID_PIXELS ++;
        END IF
   END FOR
END FOR
R = R/(#_OF_VALID_PIXELS);
G = G/(#_OF_VALID_PIXELS);
B = B/(#_OF_VALID_PIXELS);
RETURN (R, G, B)
```

Code 5-1. Get the feature vector in RGB color space

After computing the feature vector in RGB color space using Code 5-1, we repeat the same process under Hsv, YIQ, Y Cb Cr, XYZ, Lab, luv and CMYK color space and finally we will get a 25-dimensional feature vector for this tongue.

After the feature extraction, the next step of the algorithm is to use the 25-dimensional feature vector to get the diagnosis for the patient. Our SVM algorithm is realized by Libsvm and trained by 396 labeled tongue images. In the 396 labeled images, 132 of them are hot, 132 of them are normal and 132 of them are cold. Currently, our algorithm has the accuracy of more than 75%.

After the classification, the server will send back the diagnosis and advises back to the users so they can see on the client side of their iTongue app.

Chapter 6. CONCLUSION AND FUTURE WORK

In this thesis, a tongue image difference detection method with high accuracy and speed was proposed and developed. The method shows good performance in tongue image difference detection. Compared to other image-differencing algorithms, the algorithm in this thesis is more optimized for the tongue difference detection.

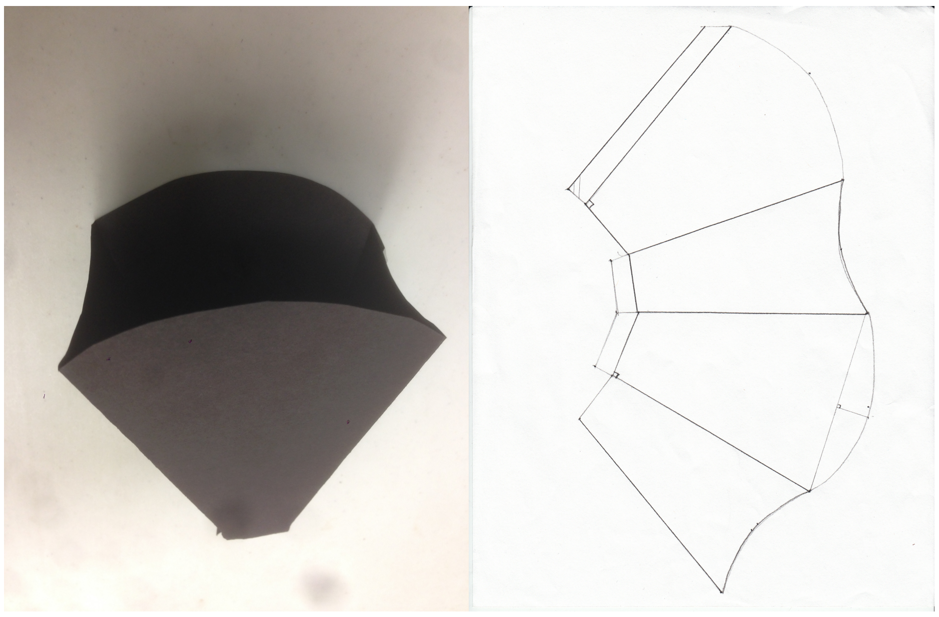


Figure 6-1. Tongue Cover for cell phone

The improvement of this algorithm will be focused on overcoming the two difficulties mentioned in Section 3-1 which are: 1) Tongue image alignment and 2). Light condition control. To overcome the second problems, a few attempts have been tried, one of which involves the help of external hardware. Figure 6-1 shows one of the tongue covers we designed to regulate the position of the tongue and restrict the light source of the picture to the flashlight from the cell phone. But this solution had the following disadvantages: 1. When using the tongue cover, the only light source is the flashlight, which makes it difficult to focus the camera because before taking the photo, there is no light source. 2. Because the flashlight is pointing directly on the tongue, the tongue reflects the light making the photo overexposed. Another possible solution for this problem is the use of software for color correction. According to the research of Xu and Mulligan, there are many different automatic color correction approaches, which have been proposed by different research communities [11]. The automatic color correction method can help us get rid of additional hardware. Also, it can improve the quality of the diagnosis.

According to tongue coating segmentation results from Zhang, Qin and. Zeng [12], another possible future work would be to separately detect the changes for the tongue and tongue coating.

BIBLOGRAPHY

1. Wenshu Li, "*Research on the several key technologies of Image Analysis for Tongue Diagnosis in Traditional Chinese Medicine*", Dissertation Submitted to Zhejiang University. 2005

2. Lo LC, Cheng TL, Chiang JY, Damdinsuren N, "*Breast cancer index: a perspective on tongue diagnosis in traditional Chinese medicine*", J Tradit Complement Med, 2013 Jul; 3(3):194-203. doi: 10.4103/2225-4110.114901.

3. Wikipedia, "Color model",
< http://en.wikipedia.org/wiki/Color_model> (retrieved April 15 2014)

4. Wikipedia, "Lab color space",
< http://en.wikipedia.org/wiki/Lab_color_space> (retrieved April 15 2014)

5. Wikipedia, "*Image differencing*",
< http://en.wikipedia.org/wiki/Image_differencing > (retrieved April 12 2014)

6. Barbara Zitova, Jan Flusser, "*Image registration methods: a survey*", Image and Vision Computing 21(2003) 977-1000

7. Wenchuan Qi, "*Tongue Image segmentation using growcut*", Thesis submitted to University of Missouri. 2013

8. Wikipedia, "Luma",
<http://en.wikipedia.org/wiki/Luma_%28video%29> (retrieved April 15 2014)

9. Paul L. Rosin, "*Thresholding for Change Detection*", Computer Vision and Image Understanding Volume 86, Issue 2, May 2002, Pages 79-85

10. Yingshan Zhang, "*Mathematical Reasoning of Treatment Principle Based on Yin Yang Wu Xing Theory in Traditional Chinese Medicine*", Chinese Medicine, Vol. 2 No. 1, 2011, pp. 6-15. doi: 10.4236/cm.2011.21002.

11. Wei Xu, Jane Mulligan, "*Performance Evaluation of Color Correction Approaches for Automatic Multi-view Image and Video Stitching*" Computer Vision and Pattern Recognition, 2010 IEEE Conference on 13-18 June 2010.

12. L. Zhang, J. Qin, Y.J. Zeng. 2011. Tongue-coating image segmentation based on combination of morphological gradient and watershed algorithms. Imaging Science Journal. Vol. 59 Issue 6, p311-316. 2011.

IMAGE REFERENCE

1. *Tongue and diagnosis n.d.* photograph, viewed April 10 2014, <http://www.holistichealthpc.com/wp/wp-content/uploads/2011/12/Tongue+Diagnosis.jpg>

2. *RGB color space n.d*. photograph, viewed April 12 2014, < https://bpiinc.files.wordpress.com/2011/11/rgb.png>

3. SharkD 2010, *The HSV color model mapped to a cylinder*, photograph,viewed April 12 2014, < http://en.wikipedia.org/wiki/HSL_and_HSV>

4. Wenchuan Qi, "*Tongue Image segmentation using growcut*", Thesis submitted to University of Missouri. 2013

5. Hue in Hsv n.d. photograph, viewed April 18, 2014, < http://stackoverflow.com/questions/13266735/draw-hsv-circle-at-runtime>