ABSTRACT

Tongue diagnosis is one of noninvasive methods to diagnose the condition of a patient's internal organs in Traditional Chinese Medicine (TCM). Since this way is noninvasive, it encourages self-diagnosis at home. One of the essential aspects of this diagnosis is the tongue color which is prone to being influenced by the lighting environment and different sensor sensitivity. It brings ambiguity and problems to get a consistent diagnosis result in self-diagnosis. In facing this problem, many researchers have found great solutions to make a reliable automated tongue diagnosis (ATD) system which potentially can solve the problem for a smartphone. However, the system can still be improved to minimize the error which is caused by the different sensor sensitivity and unoptimized parameters. For improving it, this paper suggests an alternative framework for tongue color classification system (TCCS) as the part of ATD system in getting more consistent color before going to the disease prediction in ATD system. The improvisation is done by implementing the kernel partial least square regression (K-PLSR) optimized algorithm in color correction of framework and other reliable algorithms. After doing an experiment by implementing this framework in several smartphones, this framework can show an improvement and can be used in more than one sensor and environment. As a result, it gets a more consistent and objective tongue color classification result.

Keywords: *Tongue color classification system, image processing, machine learning, sensor-independent, self-service healthcare, K-PLSR*

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 Background and Motivation

TCM has a long history that was around 3000 years ago. Many researchers had done researches to see how effective TCM in the medical field. As a result, it had been proved that TCM is adequate for handling many diseases and improving quality of life [1, 2]. In the process of development, TCM found four specific methods in diagnosing its patients. Those are inspection which is using visual inspection for observing abnormalities in the tongue and another vital body, listening and smelling which is detecting abnormalities inpatient voice and odor, inquiring which is inquiring about observable symptoms and relevant medical history, and palpation which is palpating corresponding body parts to predict pathological changed.

In the inspection method, the tongue becomes one of the things that is being observed because the tongue becomes a link between the internal organs and the outside world. Therefore, the tongue becomes an indicator to judge whether it's abnormal or not. If there is an abnormality, it means there is something wrong with the internal organs. There are several aspects to consider whether tongue can be said as abnormal or not, such as a body of tongue and coating of the tongue. In Joyce et al. [3], I found that when judging body and coating, also there are some aspects, such as the shape of the body, the color of coating and body, the thickness of the coating, moisture, movement, etc. However, color is easy to get influence by the light environment, age of the doctor, experience, etc. Thus, researchers researched to make a system to get a consistent result, that is ATD. ATD system is needed to make a consistent result in TCM treatment which can handle the subjectivity of doctors in judging color [4].

In nowadays, the self-diagnosis and health monitoring demands have been increased. It encourages the implementation of ATD in the smartphone for a public online used case. So that, it provides a media to doctor and patient for diagnosing tongue. But in the public online case, the diversity of sources will make a higher chance to get a bias in the result such as different sensor sensitivity and lighting conditions. In Hu et al. [5], their framework implements SVM to estimate lighting conditions and polynomial color correction (PCC) to do color correction. The result shows it can be implemented well in one smartphone under a controlled condition in the black box. However, the actual implementation will require more various conditions. So that, the framework that can be adapted in the various condition is needed to support ATD in the smartphone used, especially in public used condition.

1.2 Problem Addressed

In a public online used case, a smartphone will be used as the device for capturing the image. But smartphones available in public are various, which means the used sensors are different from each other. Every sensor has different sensitivity through the light and color, which is easy to be influenced by lighting conditions in the captured image. So that it will cause a bias in the result and needs dataset of each sensors in big amount. However, big amount dataset for all sensors is not available now.

Although some researchers have tried to propose color correction algorithms like K-PLSR for solving small amount of dataset problem, I found that every sensor can be optimized more by choosing the best parameter and kernel. So, it still can be improved for getting a better result to handle the various environmental problem and different sensor sensitivity. The improvement of it still not implemented yet in current ATD system.

In the current ATD system, the steps before disease prediction are usually image acquisition, color correction, image segmentation, and feature extraction. The feature extraction step has many variations which is one of them is color classification. Since this thesis content is about proposed framework for TCCS, the feature extraction step is color classification. TCCS is part of ATD system. However, as has been said before, current ATD system has not implemented yet improved color correction algorithms in TCCS.

Because of that, current TCCS system needs an alternative framework to get more consistent color of tongue image which is not affected by lighting condition and sensor condition. For getting the consistent color of tongue image, this thesis proposes a framework as a part of ATD system before predicting the disease. So that the proposed framework for TCCS will correct the color of tongue, segmenting the corrected tongue image, and classify the color of tongue to certain reference for becoming the input data for ATD system to predict a disease. However, the predicting disease is outside of this thesis scope.

1.3 Content and Innovation

This thesis proposes an alternative framework that has four general steps, those are image acquisition, color correction, image segmentation, and color classification. This thesis will prove that the proposed framework can get consistent color in tongue images under various conditions. The primary role in handling various condition problems is located in the color correction step. So, this framework content is mainly focused on the proving implementation of recently improved algorithms which has not been implemented yet in one unity ATD system for smartphone used.

The color correction algorithm that is used in this framework is K-PLSR optimization. K-PLSR optimization is more reliable in choosing the best parameters of K-PLSR and handling linearity problems, which often happens in the PLSR and polynomial regression. So, this algorithm is more robust to handle various conditions. Besides that, the algorithm that will be used in the image segmentation is U-Net and the algorithm that will be used in the classification is euclidean distance.

1.4 Thesis Content Structure

The thesis content structure is divided into six chapters. For more detail about the explanation of each chapter can be shown as follows:

1. **Introduction**
   This chapter discusses the background of this thesis, the problem that wants to be solved, and the innovation that is offered by this thesis.
2. **Related Works**
   This chapter discusses the current research status of ATD system and the problem that can be improved in this thesis.
3. **Methodology**
   This chapter discusses the methodology that will be used in the proposed framework including the definition of the methodologies, related works, and the reason for choosing the algorithms.
4. **Design of Framework**
   This chapter discusses the detail of framework design including the flows and the implementation steps.
5. **Experimental Result and Analysis**
   This chapter discusses the result of the experiment by using the proposed framework including the setting of the experiment, result, and the discussion.
6. **Closing**
   This chapter discusses the final conclusion, the limitation that is existed in the current proposed framework, and suggestion for future works.

CHAPTER 2

RELATED WORKS

2.1 Color Correction

One of the most important steps to keep the color consistent in ATD is color correction. Color is easy to get influence by the light environment [6] and different sensor sensitivity. Because of that, many researchers try to improve the algorithm in color correction. Nowadays, there are three common algorithms to do color correction [7], those are polynomial regression, support vector regression (SVR), and deep neural network (DNN).

Recently, DNN is very popular to become a research object to solve image problems because of the reliability in imitating human visual systems [8]. So that, researchers also try to implement DNN to do color correction for ATD. In Yunxi et al. [9], they implement DNN to correct tongue image. To perform it, they use tongue pictures in different lighting conditions for the dataset. As a result, they can get better performance rather than other algorithms such as polynomial regression and SVR. This algorithm is superior when we have a big dataset [10], because the more dataset we have, will improve the performance. Moreover, they don't need an X-Rite color checker as a reference for predicting it because this algorithm will learn to improve the weights from the comparison of input and ground-truth. This algorithm is indeed the best for the dependent-sensor case and if we have a big dataset. But unfortunately, in the self-diagnosis case, DNN is not appropriate because of the complexity of the architecture and sensor-dependent problem. The complexity of the deep layer leads to the expensive computational cost when training and predicting. Besides that, the complexity can cause a long training time and long computational speed which cannot be implemented in the embedded device or online public server for self-diagnosis. Also, The need for a big dataset also cannot be fulfilled in a self-diagnosis case. So, DNN cannot be used for reaching the purpose of this paper.

SVR is not as popular as polynomial regression to be adopted in ATD. So, not many researchers use this as their main research content. In Zhang et al. [11], they proposed a novel approach to do color correction. In their work, they use SVR as their main method and make a new color checker which has 24x4 colors. To correct the color, they convert their image to CIELAB color space for calculating the difference between the reference image and input image. CIELAB is an absolute color space which does not depend on the device because this color space is intended to mimic the nonlinear response of the eye [12], not like RGB which can show the different result in a different device. As a result, they show the effectiveness of their new approach to get a better result rather than polynomial regression. Moreover, SVR doesn't need to have many datasets as many as DNN which can fulfill one of our requirements. However, the method that they use is suitable in the medical environment such as a hospital or under expert observation which cannot be used too for our self-diagnosis purpose because they use a 24x4 colors color checker which is a novel approach. So, it's not available in the public which makes it cannot fit in our self-diagnosis ATD system.

In Min Chun et al. [5], SVM and polynomial regression is used to estimate lighting condition and predict color correction matrix depends on the captured images with or without flash for tongue color. In their work, they need to train support vector machine (SVM) by using captured images with or without flash which is performed in the big black box environment. Then the estimated lighting condition will be input for PCC. As a result, their experiment can be performed without dependency on a color checker and can predict a fairly accurate result. However, their experiment is still in a limited dataset which leads to the environment-dependent and sensor-dependent. In the daily environment, lighting condition and sensor sensitivity which is produced by the different company are varied and uncontrolled which will lead the captured image with some illumination and different color saturation. So that, a new alternative way is needed to overcome this problem.

PCC is also can be used to overcome the expensive computational cost and long computation speed of DNN. Not only fast, but this method is also reliable and used widely in different fields for color correction [13, 14]. In Wang et al. [15], they use PCC as their main algorithm to perform color correction. In their work, I found that they categorize color space becomes two groups, those are device-independent and device-dependent. To overcome a sensor-dependent problem, they do the same as the work in [11]. Before doing that, as said in their paper, the standard illumination for tongue inspection is sunshine and it should be performed in an open area at 9 A.M. So, for fulfilling that requirement, they render the images to the standard D65 (color temperature is 6500K) illuminant to simulate the daylight sunshine. Their work shows that the implementation of their method can get a constant result in different sensors and different lighting conditions. However, the number of the sample that they use should not be too small because it may cause a bias in the result and cause overfitting. Besides that, the performance of PCC can be improved more in handling illumination problems because PCC is easily influenced by the lighting environment [16]. Although so, PCC can reduce mapping error by more than 50% [17]. So, the other researchers try to improve this part by implementing root polynomial regression in a color correction which is usually called root polynomial color correction (RPCC).

In Jiayun et al. [16], they use RPCC to improve PCC in ATD. In their work, they also use CIELAB to handle the sensor-dependent problem. Their experiment which is done in different lighting conditions can prove how RPCC performance can succeed in that condition. RPCC is known not sensitive to lighting source [18], not like PCC which is easy to get influence. It is because RPCC can eliminate nonlinearity changes that are caused by illumination. Because of that, RPCC is fit for different illumination conditions in color correction. The experiment also shows how competitive time cost of RPCC with PCC. However, their experiment doesn't render the images to the standard D65 illuminant to simulate the daylight sunshine which can affect the result.

In Wei [19], he implements partial least squares regression (PLSR) in tongue color correction to overcome too few samples problem which often happens in common PCC. In their work, he uses RGB as his color space. PLSR is a robust statistical analysis method because it can solve a problem that happens in multiple regression such as multiple correlations between variables and a small number of samples. Their result can show how it can improve. But unfortunately, he uses RGB color space. RGB color space is a device-dependent color space [20] because it will show the different results in different devices and it cannot describe the human visual accurately. Because of that, this color space is not fit to be implemented in ATD. On the other side, if device-independent color space like CIELAB is used in PLSR, this algorithm cannot perform so well. So, an improvement of PLSR is needed which is known as K-PLSR.

In Zhuo et al. [21], K-PLSR is used in ATD to overcome the RGB color space problem. Not only has PLSR characteristics, because K-PLSR uses the nonlinear kernel, so the independent variable space can be mapped to high dimensional feature space, then PLSR can be performed in this feature space. Because of that advantage, K-PLSR can handle the linearity in the PLSR become nonlinear which makes this algorithm can improve the precision of fitting and prediction. Their experiment is performed in CIELAB color space and renders it to the standard illuminant. In their experiment result, they show how the implementation of K-PLSR can improve the performance in different lighting conditions rather than use PCC or SVR. Although so, their experiment doesn't prove the improvement of K-PLSR in a captured image by a different sensor which cannot be known how powerful this algorithm. So, further research of K-PLSR is needed to prove how effective this algorithm.

In this thesis, from the fact that the parameter of K-PLSR will give the optimum result according to the input, then for searching the best parameter, an iterative process will be used to search the best parameter. An iterative process is used to optimized K-PLSR by searching the appropriate parameter, such as gamma value and kernel that will be used. I intend to do further research in the implementation of this K-PLSR optimization to perform color correction in my ATD system. The reason is because of the reliability to fulfill my requirements such as handling the sensor-independent, handling lighting environment problem, can choose better parameters of K-PLSR, and competitive computational speed and cost.

2.2 ATD System and TCCS System

In this technological development, many researchers have proposed new frameworks and new solutions for the ATD system. As a result, there are a wide variety of frameworks available as well as their ways. Each framework implements different technologies and methods depending on the era. Several years ago, around 2002, Cai [22] proposed a novel system for tongue inspection. The proposed system is made by a new framework at that time. The framework contains image acquisition, color correction, image segmentation, feature extraction, and classification functions. In the image acquisition step, the framework doesn't use a specific device but using a normal commercial camera with a resolution of 640 x 480 pixels, the color correction is done by using Munsell ColorChecker to calibrate the color, the image segmentation is done by using snake method, the feature extraction is done by converting RGB to CIELAB color space, and the classification is done by using K-mean. This proposed system is a good starting point for giving a general overview of the tongue diagnosis framework, moreover, it's already a complete framework for classifying the color of the tongue. However, the resolution used was only 640 x 480 pixels because technological advances at that time were different from now. Low resolution will cause certain features to be lost during classification thereby reducing the accuracy of the results. Moreover, for advanced diagnosis, detection of fur and detection of cracks in the tongue will add more accuracy, but if the resolution is small it will be difficult to detect small features such as crack and fur. In addition, the segmentation method used is the snake. Although snake can segment an image well even though it is quite simple, snake still relies too much on the color threshold where the color of the lips and tongue are quite identical.

In nowadays, there is an intended device that can be used for capturing tongue images with high resolution. These intended devices usually are equipped with color calibration such as used in Zhang et al. [23]. The use of an intended device for tongue diagnosis is quite effective because it can use the framework with classification only without segmentation and color calibration. If implementing the intended device in the hospital for the examination stage, it is not a problem because it is handled by experts and at an adequate cost. but for individual use, an intended device is not a suitable solution. In this day and age, self-diagnosis is needed so that the implementation of tongue diagnosis on smartphones is quite popular. Labor hours, cost, and simplification are some of the reasons behind this. Diagnosis for the patient on the bed will also be made easier. In addition, early detection using the tongue as input is very helpful. In [24], I found that "The characteristics of the tongue and tongue coating in the cancers indicated that the tongue diagnosis may provide potential screening and early diagnosis method to cancer". It indicates the potential to diagnose the disease early by diagnosing it. Lo et al. [6] created a system to detect early-stage breast cancer by diagnosing the tongue. The framework used includes color calibration, tongue segmentation, feature detection, and statistical analysis. This system detects several special features such as tongue color, tongue shape, saliva, tongue fur, tongue quality, tongue fissure, and ecchymosis as well as several other areas of the tongue that are related to the internal part of the tongue. Logistic regression was used to show the relationship between breast cancer data and non-breast cancer. The final results show that early detection is possible. Then Zhang et al. [7] also perform a diagnosis of diabetes with an image of the tongue. This system segments the tongue coating and tongue body so that it is colored. In this paper it is said that tongue coating has yellow, gray, black, and white colors, so apart from that color is a tongue body. Then the classification of diabetes and non-diabetes is carried out using a certain algorithm. There are four algorithms being compared, they are k-NN, Naive Bayes, BP-NN, and GA-SVM. GA-SVM achieved the highest accuracy of 79.72%. The evidence in this paper indicates it is possible to detect diabetes from images of the tongue.

Evidence of the early-stage diagnosis or diagnosis of disease of the tongue carries a good potential for implementing self-diagnosis in the smartphone. Ming et al. [25] created a system for detecting tongue fur on smartphones. To implement it on a smartphone, environmental diversity is something that needs attention, because in different environments it will produce different light. In this work, they wanted to try that white fur could be detected in a variety of different light environments. Pictures are taken in two ways, with flash and without flash. After that, a Munsell ColorChecker-based color correction will be performed as a reference using the PCC method. Then the two types of corrected images will be used to train a SVM. This SVM will be used to estimate light. Then the data also needs to be segmented into the white fur section, then labeled and trained into the SVM classification model. As a final result, this work can detect white fur if the overlap rate of corrected tongue images exceeds 60%.

Another work tries to implement a similar framework on smartphones. Min-Chun et al. [5] capture an image of the tongue by using a smartphone in different light conditions (flash and without flash) too. In this work, they wanted to implement the framework to prove that tongue diagnosis is related to liver disease. This framework starts from training the color correction matrix based on color checker. The experiment was being done in the black box and by using the Samsung Galaxy S3. After that, the result of the image's light is estimated by using SVR-based light condition. Finally, the result shows that there is a possibility for early diagnosis of liver disease by using a smartphone. Even though the method is only implemented in one smartphone, it can indicate the possibility of using a smartphone to detect disease of the tongue and self-diagnosis.

CHAPTER 3

METHODOLOGY

3.1 Color Correction for TCCS System

3.1.1 Introduction

Color correction is a method for correcting captured image color. This method is often applied in photography, television, and cinematography to get a correct image color that is influenced by the lighting environment. Since this thesis innovation is about making a consistent result, this method is very important to be applied.

In Nowadays, there are three common algorithms to do color correction [7], those are PCC [26], SVR [11], and DNN [27]. Those algorithms are often compared in making an improvement in the color correction [21]. However, their best implementation depends on the case that they need. Besides those three algorithms, there is an algorithm that is often used for searching the relation between input and output, that is PLSR. In color correction, the main purpose is to find the relation between uncorrected images and corrected images by calculating the mapping variable. Usually, the relation is represented in equation (3.1).

$$Y = A.X \tag{3.1}$$

Where the uncorrected image is X, the corrected image is Y, and the mapping variable is A. To find A variable is the problem that is tried to solve here by those algorithms.

3.1.2 Color Theory

In the ATD system, two common color spaces which are often used are RGB and CIELAB. They are usually used for the color correction step and evaluation step. Their explanation can be seen below.

1. RGB

RGB is a color model that contains three colors R(red), G(green), and B(blue). Those colors will be combined to reproduce the other colors. Those three colors will represent 256 colors in each component. Each of them has a range from 0 until 255. If the color is lower and close to 0, it will produce a color that is close to black. Otherwise, it will produce near to that color. So, if all three colors are 255, it will produce an array [255,255,255] which means white color.

RGB is usually used for displaying an image on an electronic device and is also used in conventional photography. Because of that, typical RGB input devices are a digital camera, image scanner, etc., whereas typical RGB output devices are TV, computer, smartphone display, etc. However, RGB

Figure 3.1: RGB illumination.
Source: https://www.wikipedia.com

Table 3.1: Tongue ten colors reference.

Color	R	G	B
Cyan	188	188	185
Red	189	99	91
Purple	226	142	214
Deep Red	136	72	49
Light Red	227	150	147
Light Purple	225	173	207
Black	107	86	56
Gray	163	146	143
White	200	167	160
Yellow	166	129	93

color space is a device-dependent color space as said in [20]. The different devices will produce different colors depends on the manufacturer and the other factors. So, this color space is needed to be calibrated for getting a constant result in the different sensors.

For classifying tongue color, it will need a reference. In Zhang et al. [23], they use twelve colors reference for classifying tongue such as shown in table (3.1). So, that table will be used as a reference in the classification step in ATD.

2. CIELAB

CIELAB is a color model that contains three components, those are L* for perceptual lightness, a*, and b* for the unique colors of human vision: red, green, yellow, and blue. The L* has a range from 0 until 100 which defines 0 as black and 100 as white. The a* and b* have ranged from -128 until 127, which a* is relative to green-red (negative value toward green and positive value toward red) and b* is relative to blue-yellow (negative value toward blue and positive value toward yellow).

This color space is shown in a three-dimensional axis as shown in figure (3.2) for showing the non-linear relation of human eye responses of color [8]. Because of that, it is known as device-independent color space. Since this color space is made for that purpose, this color space is often used for converting from one color model to the other color model for keeping the color constancy and for measuring how big the difference value of two perceived colors. So, This color space will be important in this thesis for evaluating how reliable our proposed framework in getting the result.

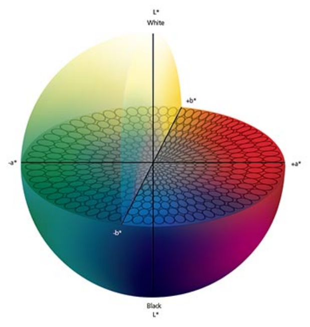


Figure 3.2: CIELAB three-dimensional axis.
Source: https://www.xrite.com

3.1.3 Existing Algorithm

In the introduction, there are three common algorithms that are often used in color correction: DNN, SVR, and PCC. But, besides those three algorithms, there are other algorithms that are also powerful, such as RPCC and K-PLSR. Their explanation can be seen below.

1. DNN

DNN works by using the weight at each node for figuring out the relation between input and output. The weight of each node will represent the relation of input and output which the weight can be represented as a mapping variable in the equation (3.1). The gradient descent (GD) formula which is shown in equation (3.2) is used for training the weight.

$$\omega' \leftarrow \omega - \alpha.\Delta\omega \sum_{i=1}^{N} L_N(\omega) \tag{3.2}$$

Where ω' is the new updated weight, ω is the previous weight, α is the learning rate, $\Delta\omega$ is the gradient of the weight, N is the index of weight, and $L(\omega)$ is the loss function. In Lu et al. [9], they implement DNN to correct tongue image. As a result, they can get better performance rather than PCC and SVR. Moreover, it doesn't need a color checker. This algorithm is the best for the dependent-sensor and big dataset case [10]. But the convergence of GD which is represented as $\Delta\omega$ in the equation (3.2) is going with small steps which means will need a big dataset for figuring out the relation of input and output. Moreover, because the layer of a DNN is deep, it means it will need a big computational cost for calculating the output.

2. SVR

SVR works by searching the hyperplane and margin which limit the range in predicted value for corrected image. The hyperplane and the margin will show the relation of input and output. The margin can define constraint as shown in the equation (3.3).

$$|y_i - w_i x_i| \leq \varepsilon \tag{3.3}$$

Where y is the predicted value, w is the weight, x is the input value, i is the index, and ε is the limit of margin tolerance. In Zhang et al. [11], they proposed a novel approach in the color correction by using SVR and new color checker $24x4$ colors. As a result, they show their new approach can get a sensor-independent result. Moreover, SVR doesn't need to have many datasets as many as DNN which can fulfill one of these case requirements. However, this method is limited with the new color checker. If it's done by using the usual color checker, the ε will limit the regression when the case is outside of the trained data which will cause overfitting.

3. PCC

PCC is usually used for regression analysis and can be used as an alternative of DNN [28]. Not only fast, but this method is also reliable and used widely in a different field for color correction [13, 14]. This algorithm works by searching the mapping variable which represents the relation between input and output. The common equation of PCC is as shown on equation (3.1). In Wang et al. [15], they use PCC as their main algorithm to perform color correction. Their work shows that the implementation of their method can get a constant result in different sensors and different lighting conditions. Moreover, the PCC method is the best choice for online color correction since this algorithm has a short training time and high precision [16].

The concept of PCC in the color correction is: assume N is the length of the patch color in color checker. So each patch is shown as $P = (R_i, G_i, B_i)$ where i = 1, 2, 3,..., N. Supposed the input is $X = [R, G, B, 1]$ and A is matrix with element a, and the output $Y = [YR, YG, YB]$, then the model can be transformed to model (3.4).

$$\begin{cases} YR_i = a_{11}R_i + a_{12}Gi + a_{13}Bi + a_{14} \\ YG_i = a_{21}R_i + a_{22}Gi + a_{23}Bi + a_{24} \quad (i = 1,2,3,.....,N) \\ YB_i = a_{31}R_i + a_{32}Gi + a_{33}Bi + a_{34} \end{cases} \tag{3.4}$$

This model also can be represented as in equation (3.5).

$$Y = A^T.X \tag{3.5}$$

Suppose Y is the reference X-Rite color checker, X is the captured X-Rite color checker, and A is the color correction matrix (CCM). Then for finding the CCM, we can transform equation (3.5) becomes equation (3.6).

$$A = (X^TX)^{-1}X^TY \tag{3.6}$$

After A has been found, we can apply it to the equation (3.5) by replacing X with the real tongue image, then we will get the Y as the result of the corrected tongue image.

However PCC is sensitive with different lighting environment which will affect the result. Different lighting environment usually will cause nonlinear problem in the final result which makes this algorithm cannot behave well.

4. RPCC

RPCC is also usually used for regression analysis. The implementation of this algorithm in the color correction is for improving the result of PCC. This algorithm is reliable for handling the nonlinear problem which happens as the result of PCC. Basically, this algorithm works in the same way as PCC, but the difference is this algorithm takes the root of each element according to the degree of it. In Sui et al. [16], they use RPCC to prove how this algorithm can give an improvement. As it is expected, the result can improve and handle nonlinear problem quite well.

Basically, the difference in this algorithm is in the concept of each degree. If the degrees in the PCC are shown as below.

$X_{1.3} = [R,G,B]$
$X_{2.3} = [R,G,B,R^2,G^2,B^2,RG,GB,RB]$
$X_{3.3} = [R,G,B,R^2,G^2,B^2,RG,GB,RB,R^3,G^3,B^3,R^2G,R^2B,RB^2,RG^2,B^2G,BG^2,RGB]$

Then the RPCC will show the degree-root as below.

$\bar{X_{1.3}} = [R,G,B]$
$\bar{X_{2.3}} = [R,G,B,\sqrt{RG},\sqrt{GB},\sqrt{RB}]$
$\bar{X_{3.3}} = [R,G,B,\sqrt{RG},\sqrt{GB},\sqrt{RB},\sqrt[3]{R^2G},\sqrt[3]{R^2B},\sqrt[3]{RB^2},\sqrt[3]{RG^2},\sqrt[3]{B^2G},\sqrt[3]{BG^2},\sqrt[3]{RGB}]$

Usually, when R, G, and B are affected by external illumination, the result will be changed nonlinearly. Assume scalar k is the external illuminant factor, then R, RG, RGB will be changed become kR, k^2R, and k^3RGB. But, by applying RPCC, it will become $(kR)^{\frac{1}{1}} = kR$, $(k^2RG)^{\frac{1}{2}} = k(RG)^{\frac{1}{2}}$, and $(k^3RGB)^{\frac{1}{3}} = k(RGB)^{\frac{1}{3}}$. It can be seen that it will eliminate the nonlinear problem.

However, the performance of RPCC will be decreased if a small sample dataset is used. It will affect the result and lead to the overfitting problem. So, improvisation is still needed to handle this problem.

5. K-PLSR

Basically, K-PLSR is the improvisation algorithm of PLSR for solving a nonlinear problem that is caused by illumination. As we know, the relation between the corrected and uncorrected images is nonlinear. So, K-PLSR can solve it by mapping the independent variable which is the uncorrected image in dimensional F to the higher dimensional space in R. But for mapping it, we need to transform the independent variable by using nonlinear transformation. Then, after the transformation is done, PLSR will be performed in R dimensional space.

The transformed space is usually called feature space and the nonlinear transformation is called mapping. For doing mapping, kernel function $K(x,y)$ will be used. According to the Mercer theorem, if K is a continuous symmetric kernel of a positive integral operator in Hilbert space, K can be expressed as equation (3.7).

$$K(x,y) = (\phi(x).\phi(y)) = \phi(x)^T \phi(y) \tag{3.7}$$

Where $K(x,y)$ represents the kernel Gram matrix of the dot product. By using this equation, we can transform the original space to the possibly infinite space by building mapping ϕ. In order to build the kernel, the commonly used kernel function is like PCC, sigmoid kernel, Radian Basis Function (RBF) kernel, etc. But in this thesis, we will use the RBF kernel for the comparison with the other algorithms. RBF kernel is usually written in equation (3.8).

$$K(x,y) = exp(-\gamma \|x-y\|^2) \tag{3.8}$$

Where x and y is the input vector that to be mapped and γ is the parameter that defines how far the influence of a single training sample reaches. γ is also same with $\sigma^{(-2)}$.

In Zhou et al. [21], this algorithm is performed and a good result can be gotten. From their work, the step of K-PLSR can be seen as followed. Supposed the input is a $(n \times N)$ matrix $\{x_i\}_{i=1}^{n}$ where n is the number of row and N is the number of column. Then our input will be mapped to the feature space by transforming it to a $(n \times M)$ matrix $\{\phi(x_i)\}_{i=1}^{n}$ where M is the number of transformed matrix column. After that, the steps of K-PLSR is shown as follows:

(a) Randomly initialize u variable which is a latent variable of the dependent variable.

(b) Calculate t variable which is a latent variable of the independent variable and then normalize it by using equation (3.9).

$$t = \phi\phi^T, t \leftarrow t / \|t\| \tag{3.9}$$

(c) Calculate c variable which is the weight of dependent variable by using equation (3.10).

$$c = Y^T t \tag{3.10}$$

(d) Calculate u variable and also normalize it by using equation (3.11).

$$u = Yc, u \leftarrow u / \|u\| \tag{3.11}$$

Where Y is the actual value.

(e) Repeat step 2 - 5 until convergence is reached.

(f) Deflate matrix $\phi\phi^T$ and Y by using equation (3.12).

$$\phi\phi^T \leftarrow (\phi - tt^T\phi)(\phi - tt^T\phi)^T, Y \leftarrow Y - tt^T Y \tag{3.12}$$

In this steps, we can apply "kernel-trick" for calculating $(n \times n)$ kernel Gram matrix which is

$K(x,y) = \phi\phi^T$. So that, the equation (3.12) can be equation (3.13).

$$K \leftarrow (I - tt^T)K(I - tt^T)^T \tag{3.13}$$

Where I is a $(n \times n)$ matrix identity of K.

(g) Repeat step 6 for each component.

(h) Calculate regression coefficient matrix B by using equation (3.14).

$$B = \phi^T U(T^T KU)^{-1} T^T Y \tag{3.14}$$

Where T and U is the matrix form of the variable t and u.

(i) The predicted result $\hat{Y}$ can be calculated by using equation (3.15).

$$\hat{Y} = \phi B = KU(T^T KU)^{-1} T^T Y \tag{3.15}$$

(j) After matrix B have been found, predicted result of testing data $\hat{Y}_t$ can be calculated by using equation (3.16).

$$\hat{Y}_t = \phi_t B = K_t U(T^T KU)^{-1} T^T Y \tag{3.16}$$

Where ϕ_t is transformed matrix of testing data, so K_t is the testing data $(n_t \times n)$ kernel matrix which represents by $K_t = K(x,y)$ where x is the testing data, and y is the training data.

However, parameters and kernel in K-PLSR play a role in deciding the accuracy of the result. Whereas every parameter and kernel will fit depends on the data. If the parameters and kernel are initialized constantly, it will give the same result. So, choosing which parameter and kernel of K-PLSR will be used is needed to optimize the K-PLSR result.

3.1.4 K-PLSR Optimized(K-PLSRO)

K-PLSRO is the combination of K-PLSR and iterative processes. The iterative process is only the simple iteration to search for the best candidate from the initialize group of the possible solution. By using this K-PLSRO, it is possible to optimize the K-PLSR result by searching the appropriate kernel and parameter according to the input. In general, this iterative process is as followed.

1. Initialize N which is the number of candidates and M which is candidates of K-PLSR model kernel and parameter.

2. Training K-PLSR by using initialize kernel and parameter.

3. Evaluate training result by calculating the accuracy of training result. The equation that is used is as

shown on the equation (3.17).

$$val_{accuracy-n} = 1 - \frac{\left[(\triangle L_{corrected}*)^2 + (\triangle a_{corrected}*)^2 + (\triangle b_{corrected}*)^2\right]^{\frac{1}{2}}}{\left[(\triangle L_{uncorrected}*)^2 + (\triangle a_{uncorrected}*)^2 + (\triangle b_{uncorrected}*)^2\right]^{\frac{1}{2}}} \tag{3.17}$$

Where $val_{accuracy-n}$ represents the accuracy of n-th candidate, n represents the index of candidate, the $L_{corrected}*, a_{corrected}*, b_{corrected}*$ represent the difference of $(L*, a*, b*)$ of those two respective corrected colors, and the $L_{uncorrected}*, a_{uncorrected}*, b_{uncorrected}*$ represent the difference of $(L*, a*, b*)$ of those two respective uncorrected colors.

4. Repeat second step and third step until all candidates is used.

5. Evaluate the best result by using equation (3.18).

$$F = M[max(val_{accuracy})] \tag{3.18}$$

Where F represents the final candidate model and $val_{accuracy}$ represents the list of accuracy.

3.1.5 Objective Evaluation

Objective evaluation is needed to evaluate which algorithm can achieve the best result for this thesis case. The objective evaluation is like a standard to measure the performance of the algorithm. In this thesis, euclidean distance will be used to measure the distance between the reference value and the predicted value. Usually, the formula to calculate the distance is as shown in the equation (3.19).

$$\triangle E(L*, a*, b*) = \left[(\triangle L*)^2 + (\triangle a*)^2 + (\triangle b*)^2\right]^{\frac{1}{2}} \tag{3.19}$$

Where $L*, a*, b*$ represent the difference of $(L*, a*, b*)$ of those two respective colors. As has been explained, CIELAB is a device-independent color space that represents human visuals. Moreover, in Zhou et al. [21], it is said that this color space can maximally reduce the color difference.

3.1.6 Evaluation Result

In this evaluation step, the experiment images are captured by using Xiaomi Redmi 8 Pro selfie camera which has 20 MP specifications. X-Rite color checker is used as the reference for color correction purposes. The image that is used in this evaluation is shown in figure (3.3).

There are three algorithms that will be performed to eliminate the best algorithm, those are PCC, RPCC, K-PLSR (RBF Kernel and $\sigma = 10$), and K-PLSR optimization which has RBF kernel with $\sigma = [10, 20]$, chi-squared kernel with $\sigma = [2, 20]$ and sigmoid kernel with $\sigma = [4, 20]$.

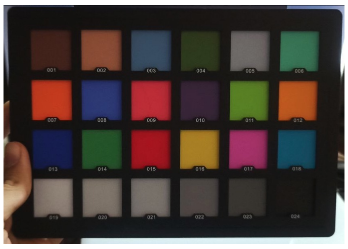


Figure 3.3: Uncorrected evaluation image.

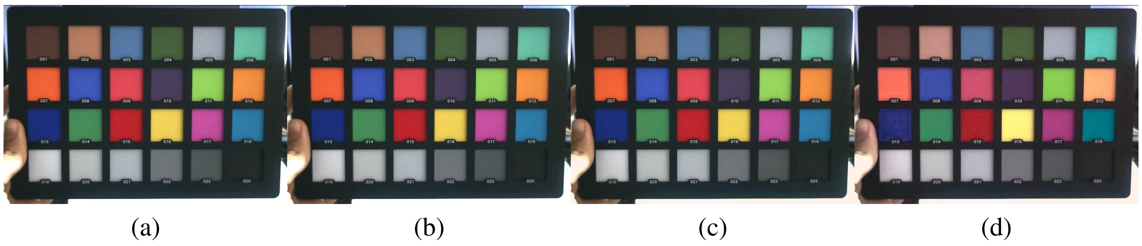


Figure 3.4: Corrected image. (a) K-PLSR optimization. (b) K-PLSR. (c) First-degree PCC. (d) Third-degree RPCC.

Table 3.2: Comparison result of different color correction algorithms.

Image	$\triangle E_{K-PLSR-optimized}$	$\triangle E_{K-PLSR}$	$\triangle E_{RPCC}$	$\triangle E_{PCC}$
Image 1	9.55	9.66	22.45	22.98
Image 2	12.61	12.90	23.06	24.48
Image 3	10.00	10.27	22.47	22.91

After the calculation of $\triangle E$ has been calculated, the result of those algorithms can be shown above. From table (3.2) and figure (3.4), it can be concluded that the result of RPCC can improve PCC which represents PCC, whereas K-PLSR optimization is the most superior one rather than the others.

Besides the $\triangle E$ result, we can show the time computation that is needed by each algorithms to perform. The time computation is performed in the computer with specification processor *Intel i3 generation 3* and RAM 10GB. The result can be seen on the table (3.3).

Table 3.3: Comparison average time computation of different color correction algorithms.

Algorithm	Average time computation (seconds)
K-PLSRO	0.862
K-PLSR	0.029
PCC	0.007
RPCC	0.014

Although the time difference between K-PLSRO and K-PLSR is around 28 times longer, but it can be implied that the time computation does not take long time which is still under 1 second. So, the time is considered acceptable for smartphone used.

3.2 Image Segmentation for TCCS System

3.2.1 Introduction

Image segmentation is a process to separate certain images by segmenting the intended image area. The intended image area can be gotten by using different ways which is the most current advanced one can be seen in figure (3.5).

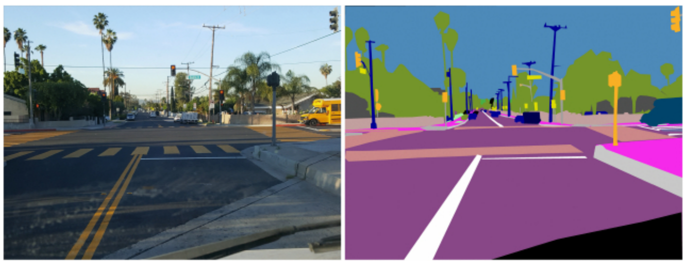


Figure 3.5: Image segmentation.
Source: https://dl.acm.org/

Image segmentation is useful to make an image easy to be analyzed. The application of this method is widely used in a different fields, such as machine vision, medical image, object detection, and image recognition. So, image segmentation also will be helpful for this thesis purpose to separate tongue image with the background of the image.

There are many algorithms that are very popular for doing image segmentation. Start from the conventional method such as threshold-based method and region-based method [29] until the newest method which is often used to handle image, that is classification method by using deep learning. Many improved deep learning architectures which often introduces in image segmentation, such as U-Net, Mask RCNN, SegNet, etc. [30]. However, image segmentation is not really affected the final result of this case because the consistent color result has nothing to do with image segmentation.

3.2.2 Existing Algorithm

In the introduction, image segmentation has been explained to have many different forms. So, in this section, there are three chosen algorithms to eliminate the best result, those are snake for representing the conventional method, ResSeg-Net, and U-Net for representing the deep learning method. Their explanation can be shown below.

1. Snake

Snake is also called active contour model. This is a framework that is used for many purposes such as object tracking, image segmentation, edge detection, etc. Basically, snake works by setting the initial snake position, then doing iteration to minimize the energy of snake until pulled toward the true contour. The position of snake can be set by put the parameter at $v(s) = (x(s), y(s))$ and the energy of it can be calculated by using equation (3.20).

$$E{*}_{snake} = \int_1^0 E_{internal}(v(s)) + E_{image}(v(s)) + E_{con}(v(s)))ds \tag{3.20}$$

Where $E_{internal}$ represents the internal energy of the spline due to bending, E_{image} represents image forces, and E_{con} represents external constraint forces.

In Saparudin et al. [31], snake is used to segment tongue image. There are 40 tongue images that are used in the experiment. As the result, there are 75% of tongue images that are successfully segmented by using snake.

2. Deep Learning

Deep learning in image segmentation is worked by classifying each pixel of the images into certain classes. Usually, deep learning architecture for image segmentation has two-part, which are the encoder part and the decoder part. The encoder part is worked by going to a deeper dimension to find the highlight feature, whereas the decoder part is worked by upsampling the highlight feature to the real size of the image for purpose of predicting the location of the feature in the image.

There are many kinds of architecture that are offered. However, this thesis will only choose some of them because image segmentation architectures quite vary. In order to compare the result for the image segmentation algorithm in this proposed framework, the latest architecture which is used in general and in the biomedical image will be compared. ResSeg-Net will represent architecture which is used not only for biomedical image, whereas U-Net will represent architecture which is used for biomedical image.

(a) ResSeg-Net

ResSeg-Net is the combination of ResNet at the encoder part and SegNet at the decoder part. ResNet is powerful rather than a traditional convolutional network(CN) because it can handle degradation problems well. In traditional CN, degradation problems or the disappearance of the gradient often happens. So, the improvement is by adding an identity layer to the network structure which makes it can follow the depth. As an advantage, the implementation of ResNet can handle the degradation problem which makes it not easy to get overfitting. Figure (3.6) shows architecture of ResSeg-Net in general.

In [32], their work proves how ResSeg-Net can give a high accuracy in segmenting food image which is over 90%. In general, 90% is very good accuracy for segmenting the image. Besides that, ResSeg-Net also can be used for segmenting biomedical image like for liver tumor segmentation [33].

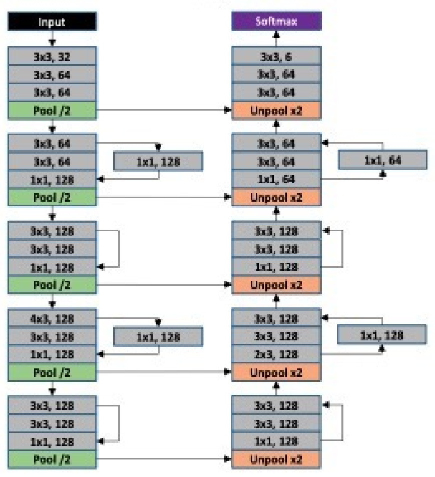


Figure 3.6: ResSeg-Net architecture.
Source: https://www.researchgate.net

(b) U-Net

U-Net is a very popular architecture for image segmentation of the biomedical image. Basically, it contains a fully convolutional network that a fully connected layer should fix the input image with any size and gives an output with the corresponding image. The idea of this architecture is by performing downsampling on the encoder part and upsampling on the decoder part. Besides that, there are shortcuts usually added between encoder and decoder in order to repair the details of the image. The structure of U-Net has a symmetrical right-left side and makes a form like an U shape. That is why it's called U-Net. Figure (4.10) shows architecture of U-Net in general.

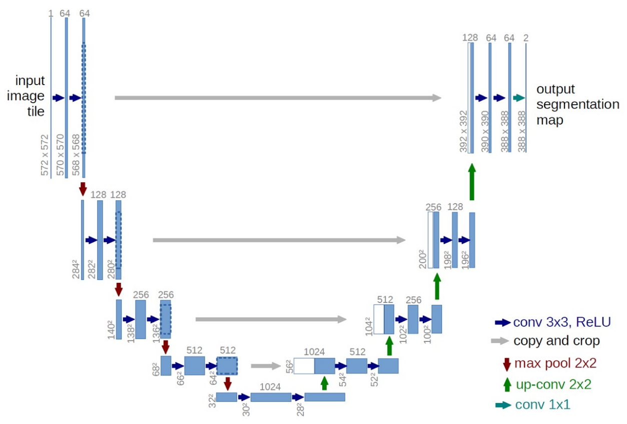


Figure 3.7: U-Net architecture.
Source: https://lmb.informatik.uni-freiburg.de

In [34], it shows that U-Net which is used for cardiac images can reach 93% accuracy. Not only for cardiac images, that review also shows many types of medical images that can be an input of this architecture. It shows how reliable U-Net in solving biomedical images segmentation.

3.2.3 Objective Evaluation

Objective evaluation of image segmentation is also needed. Without setting a certain standard, it will be only subjective evaluation by observing the result people by people. So, a certain standard is needed to eliminate which is the best algorithm that will be used in this thesis framework. There are four evaluation indicators, those are True Positive (TP), False Positive (FP), True Negative (TN), and False Negative (FN). The explanation of each indicator as followed.

1. True Positive

   True Positive is marked if the pixel in the result is correctly mapped to the tongue body image.

2. False Positive

   False Positive is marked if the pixel in the result is wrongly mapped to the tongue body image.

3. True Negative

   True Negative is marked if the pixel in the result is correctly mapped to the background of the image.

4. False Negative

   False Negative is marked if the pixel in the result is wrongly mapped to the background of the image.

After marking those four indicators, then accuracy, precision, recall, and f1-score can be calculated. Accuracy is the ratio of correctly predicted results over all total predictions. The percentage of accuracy can be calculated according to the equation (3.21).

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} * 100\% \tag{3.21}$$

Whereas precision is the ratio of correctly positive predicted result over total positive prediction. The percentage of precision can be calculated according to the equation (3.22).

$$Precision = \frac{TP}{TP+FP} * 100\% \tag{3.22}$$

Whereas recall is the ratio of correctly positive predicted result over total prediction in actual class. The percentage of recall can be calculated according to the equation (3.23).

$$Recall = \frac{TP}{TP+FN} * 100\% \tag{3.23}$$

Whereas f1-score is the average of precision and recall. The percentage of f1-score can be calculated according to the equation (3.24).

$$F1-score = \frac{2 * Precision * Recall}{Precision + Recall} * 100\% \tag{3.24}$$

3.2.4 Evaluation Result

The evaluation images of this step are also captured by using Xiaomi Redmi Note 8 pro selfie camera. The target image and the ground-truth image that is used can be shown on the figure (3.8).

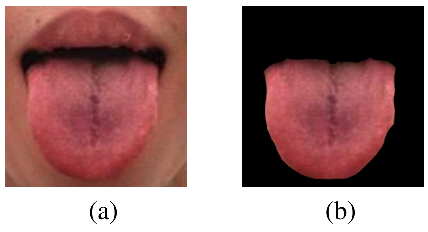


Figure 3.8: Evaluated image. (a) Target image. (b) Ground-truth image.

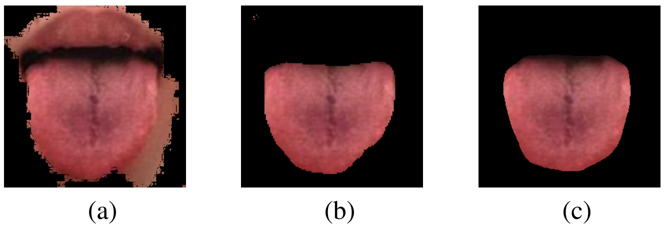


Figure 3.9: Segmented image. (a) Snake. (b) ResSeg-Net. (c) U-Net.

Table 3.4: Comparison result of different image segmentation algorithms.

Algorithm	Accuracy(%)	Precision(%)	Recall(%)	F1-score(%)
Snake	70.64	60.24	97.06	74.34
ResSeg-Net	91.86	99.81	81.6	89.79
U-Net	92.43	98.91	83.73	90.64

After the calculation of those indicators, the result of those three algorithms can be shown above. From figure (3.9) and table (3.4), it can be concluded that U-Net is the most superior algorithm rather than the other algorithm in almost of all aspects. Also besides the 4 indicators above, the computation speed of each algorithms will be compared too. The hardware specification that is used is the same as the color correction. The result of average computation speed can be seen on the table (3.5).

Table 3.5: Comparison average time computation of different image segmentation algorithms.

Algorithm	Average time computation (seconds)
Snake	96.09
ResSeg-Net	10.70
U-Net	9.45

From the computation speed, it can be seen that segmentation by using U-Net architecture is faster rather than the other algorithms although the difference between ResSeg-Net and U-Net are not really significance which is 0.1 times faster.

CHAPTER 4

DESIGN OF FRAMEWORK

4.1 Introduction

The TCCS system is a set of algorithms that work together to classify color of the tongue. The result can be varied. In this thesis, the result will focus on getting the consistent color of tongue classification without the dependency of smartphones and the environment. From the review in chapter 2, in general, the TCCS system consists of image acquisition, color correction, image segmentation, and color classification. In order to make improvement, the proposed framework offers different algorithms in color correction, image segmentation, and color classification. For understanding the flow, the proposed framework of the TCCS system in this thesis can be shown in the figure (4.1).

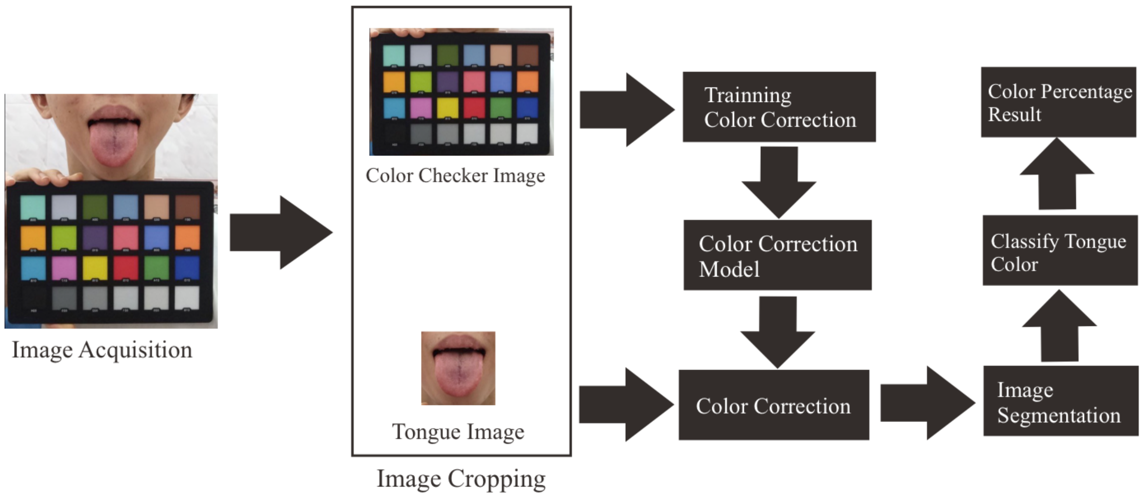


Figure 4.1: Proposed framework flow.

According to the previous chapter evaluation, The chosen algorithms for color correction and image segmentation can be decided. For more detailed explanation, it will be divided become four sections, which are image preprocessing, color correction, image segmentation, and tongue color classification.

4.2 Preprocessing

In this section, there are two subsections that will be discussed for showing what is the crucial steps before doing the color correction step. There are two steps that should be discussed here, those are image acquisition & image cropping, and the preprocessing itself.

4.2.1 Image Acquisition & Cropping

This thesis focus is processing an image. So, the step of collecting the data of the image is quite important. In a public online case, the device which will be used is a smartphone since this device is very popular nowadays. There are many considerations that should be thought about for getting a proper result and a successful experiment. So, the discussion of this step needs to be explained in detail too.

As the name, image acquisition is the step of taking required images from a smartphone. But in taking the images, there are rules that should be followed in this thesis. First, the color checker image and tongue image should be taken in one image directly rather than taking them separately. Then, the tongue image and the color checker image should not be too far, but the color checker should be just below the chin. For more specific, it can be shown in the figure (4.2).

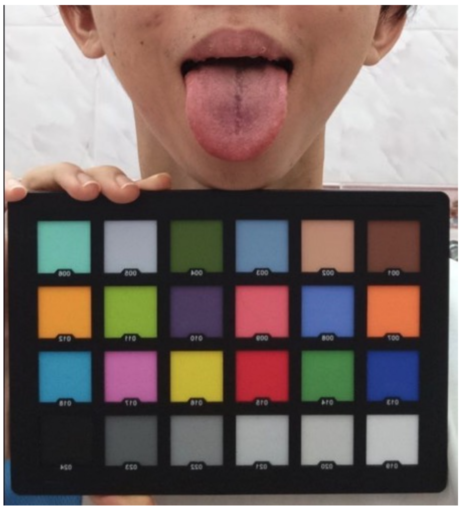


Figure 4.2: Image acquisition.

The size of the taken image will be various because every smartphone will have a different camera resolution. For the color correction and color extraction step, there is no requirement about image resolution, but there is for the image segmentation. So for solving it, the taken images need to be resized for the image segmentation step since it will need fixed resolution for the input of the algorithm that is $256x256$ pixels.

There is a specific reason why we have to take color checker image and tongue image in one image directly rather than separately. The reason is that in every smartphone, the software and setting of the smartphones are different from each other, there is some camera in smartphone especially selfie camera will change their aperture depends on the object that is detected by the camera. It will cause different illumination between tongue image and color checker image. If their illumination is different, it will cause a biased result.

After the image has been taken, the image should be cropped manually to separate the color checker and the tongue. The tongue image should be cropped with a square ratio with lips as the upper bound and above the chin bit as the lower bound. Whereas the color checker image can be cropped with less constraint, but as long as it is included in the color checker patches. For more detail, it can be shown in the figure (4.3).

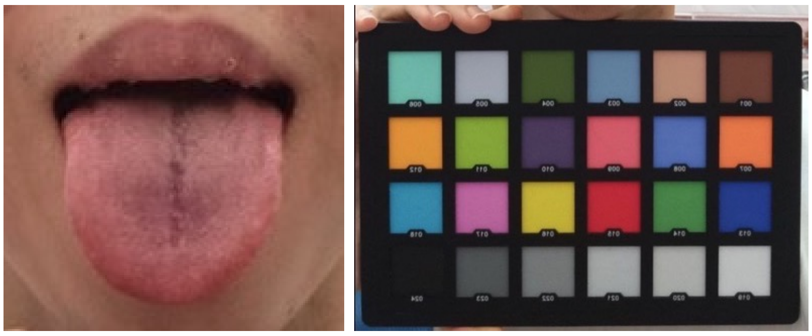

(a) (b)

Figure 4.3: Image cropping. (a) Tongue image. (b) Color checker image.

4.2.2 Preprocessing

After the images have been cropped, the preprocessing result can be performed to get a better result in the color correction. The preprocessing step is intended to bring the image to the normal distribution. The image should be in the normal distribution step for purpose of cheaper computational cost and also to get a better training result.

Normal distribution has two certain rules, that is the mean is 0 and the standard deviation is 1. For getting the normal distribution condition, the steps are as followed.

1. Calculate the mean of image RGB pixel.
2. Decrease each pixel of the image with the mean.
3. Calculate the standard deviation of the new image pixel.
4. Divided each new image pixel with the standard deviation.

4.3 Color Correction

Sensor independence will depend a lot on this step because this step will adjust the color to the correct one by shifting it to the appropriate value. For shifting the value, CCM has to be found by training it with color patches data from the X-Rite color checker. In fact, there are many algorithms to get the CCM as have been discussed. But this proposed framework focus is on the K-PLSRO algorithm because the evaluation step has proved K-PLSRO is the best algorithm for this case. Since this thesis is used for a public online purpose, then it will be impossible if we use DNN as our algorithm because there is no such a big dataset for every smartphone that is available now.

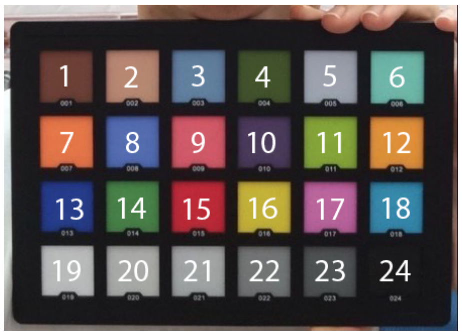


Figure 4.4: X-Rite's 24 color patches.

The first thing that needs to be done before doing color correction, the 24 color patches need to be gotten in the following order in figure (4.4). Then, the uncorrected color patches will be the independent variable in the training step. Whereas the actual value of 24 color patches which is shown in table (4.1) will be the dependent variable.

Table 4.1: X-Rite's 24 real color patches in RGB.

No	Color	R	G	B
1	Dark skin	115	82	69
2	Light skin	204	161	141
3	Blue sky	101	134	179
4	Foliage	89	109	61
5	Blue flower	141	137	194
6	Bluish green	132	228	208
7	Orange	249	118	35
8	Purplish blue	80	91	182
9	Moderate red	222	91	125
10	Purple	91	63	123
11	Yellow green	173	232	91
12	Orange yellow	255	164	26
13	Blue	44	56	142
14	Green	74	148	81
15	Red	179	42	50
16	Yellow	250	226	21
17	Magenta	191	81	160
18	Cyan	6	142	172
19	White	252	252	252
20	Neutral 8	230	230	230
21	Neutral 6.5	200	200	200
22	Neutral 5	143	143	142
23	Neutral 3.5	100	100	100
24	Black	50	50	50

The implementation of K-PLSR can be used for handling a small quantity of the training data and handling the linear problem in a color correction like in this case where the number of sample training data is only 24 color patches. If the quantity of training data is small, it will make the accuracy of the predicted result is reduced. But, K-PLSR which has the characteristics of PLSR can handle those problems well since

PLSR can solve multiple correlations among variables and a small amount of sample training data.

However, K-PLSR still can be optimized more because different sensor will fit with different kernel and parameter. The kernels which are available are many such as linear kernel, polynomial kernel, RBF kernel, sigmoid kernel, laplacian kernel, and chi-squared kernel. But after certain experiment, only several kernels that can be used for the improvisation. First is sigmoid kernel which can be seen on equation (4.1), second is RBF kernel which can be seen on equation (4.2), and third is chi-squared kernel which can be seen on equation (4.3).

$$k(x,y) = tanh(\gamma x^T y + c_0) \tag{4.1}$$

Where x is the reference image, y is the input image, and c_0 is the intercept.

$$k(x,y) = exp(-\gamma \|x-y\|^2) \tag{4.2}$$

Where x is the reference image, y is the input image, and γ is the parameter that defines how far the influence of a single training sample reaches.

$$k(x,y) = exp\left(-\gamma \sum_{i=0}^{n} \frac{(x[i]-y[i])^2}{x[i]+y[i]}\right) \tag{4.3}$$

Where x is the reference image, y is the input image, and γ is the parameter that defines how far the influence of a single training sample reaches, and n is the number of the pixels.

After deciding what kernels that can be used, the evaluation of which kernel that will be chosen can be performed automatically by the implementation of K-PLSRO below.

1. Define the possible kernels and parameters. The possible kernels and parameters will be the parameters of the function in this color correction. As have been said, the possible kernels and parameters are RBF kernel, sigmoid kernel, and chi-squared kernel. From the experiment, the range σ of each kernel is $[10,20]$ for RBF kernel, $[4,20]$ for sigmoid kernel, and $[2,20]$ for chi-squared kernel.

2. Pick each RGB color patch from the cropped image of the X-Rite color checker as shown in the figure (4.4). Then we will have 24 reference values and table (4.1) become actual value.

3. The actual value will be converted to the D65 illuminant to simulate the real condition of tongue observation. D65 illuminant has default white RGB $[255,249,253]$. So, the conversion of the D65 illuminant can done by using equation (4.4).

$$c(r,g,b) = \left[r\frac{255}{w_r}, g\frac{249}{w_g}, b\frac{253}{w_b}\right] \tag{4.4}$$

Where $c(r,g,b)$ is the function of conversion, r is R value, w_r is R value of white pixel from actual value, g is G value, w_g is G value of white pixel from actual value, and w_b is B value of white pixel from actual value.

4. Doing center scale to the dataset like step in the preprocessing steps. This step is a must because for making the image easy become convergence.

5. Perform transformation to the higher dimension by using kernel trick. But it should be noted that, if the kernel that is used is chi-squared, it means the center scale result of the dataset should be added by 10 because chi-squared cannot has negative values. The flowchart to perform it can be seen on the figure (4.5).

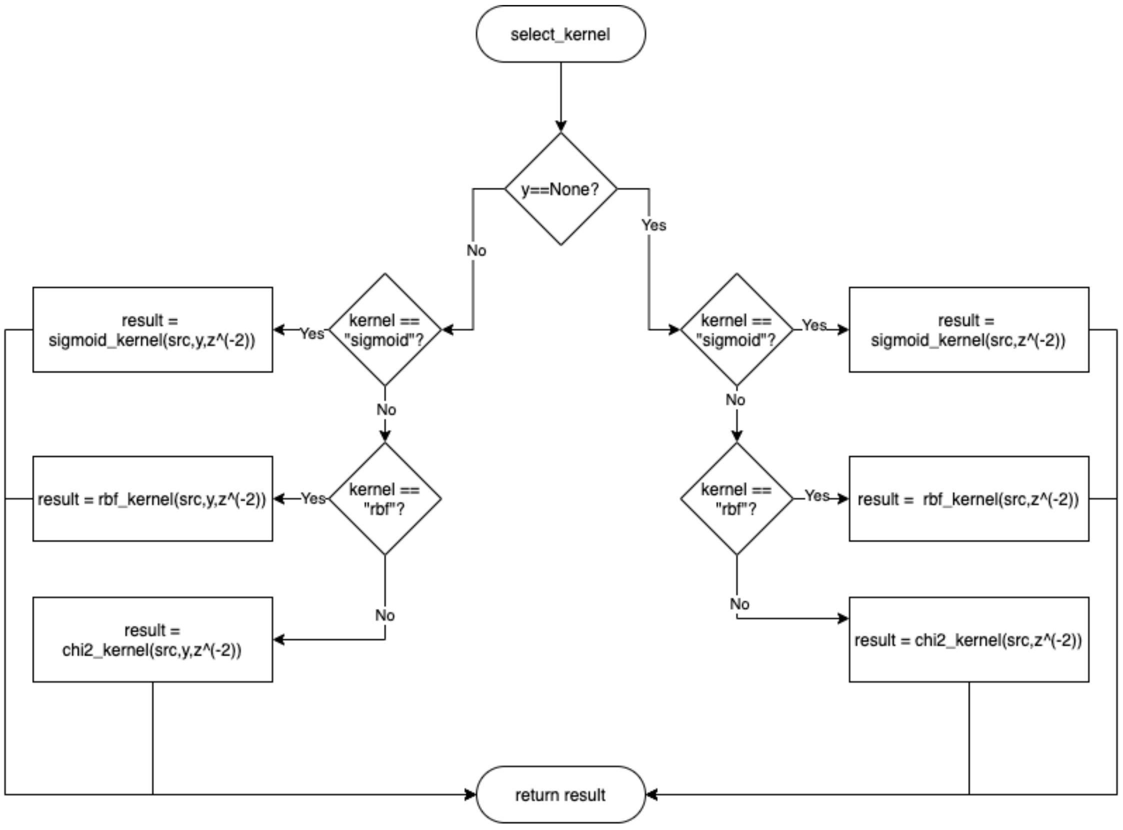

select_kernel
y==None?
No
Yes
kernel == "sigmoid"?
Yes
result = sigmoid_kernel(src,y,z^(-2))
No
kernel == "rbf"?
Yes
result = rbf_kernel(src,y,z^(-2))
No
result = chi2_kernel(src,y,z^(-2))
kernel == "sigmoid"?
Yes
result = sigmoid_kernel(src,z^(-2))
No
kernel == "rbf"?
Yes
result = rbf_kernel(src,z^(-2))
No
result = chi2_kernel(src,z^(-2))
return result

Figure 4.5: Transformation by kernel trick flowchart.

Where y variable is uncorrected input image, $kernel$ variable is selected kernel, src variable is 24 reference values, and z variable is σ.

6. After the transformation, K-PLSR algorithm will be performed. To perform K-PLSR algorithm, we need to define 4 variables, which are $x-weight$, $y-weight$, $x-score$, and $y-score$. $x-weight$ is $i \times n$ matrix, $y-weight$ is $k \times n$ matrix, $x-score$ is $j \times n$ matrix, and $y-score$ is $j \times n$ matrix.

   Where i is the size of reference value's second axis, j is the size of reference value's first axis, n is the number of component which is 4, and k is the size of the actual value's second axis.

7. Then we need to train by using the dataset n times. It means we will train by each components. The flowchart of it can be seen on the figure (4.6).

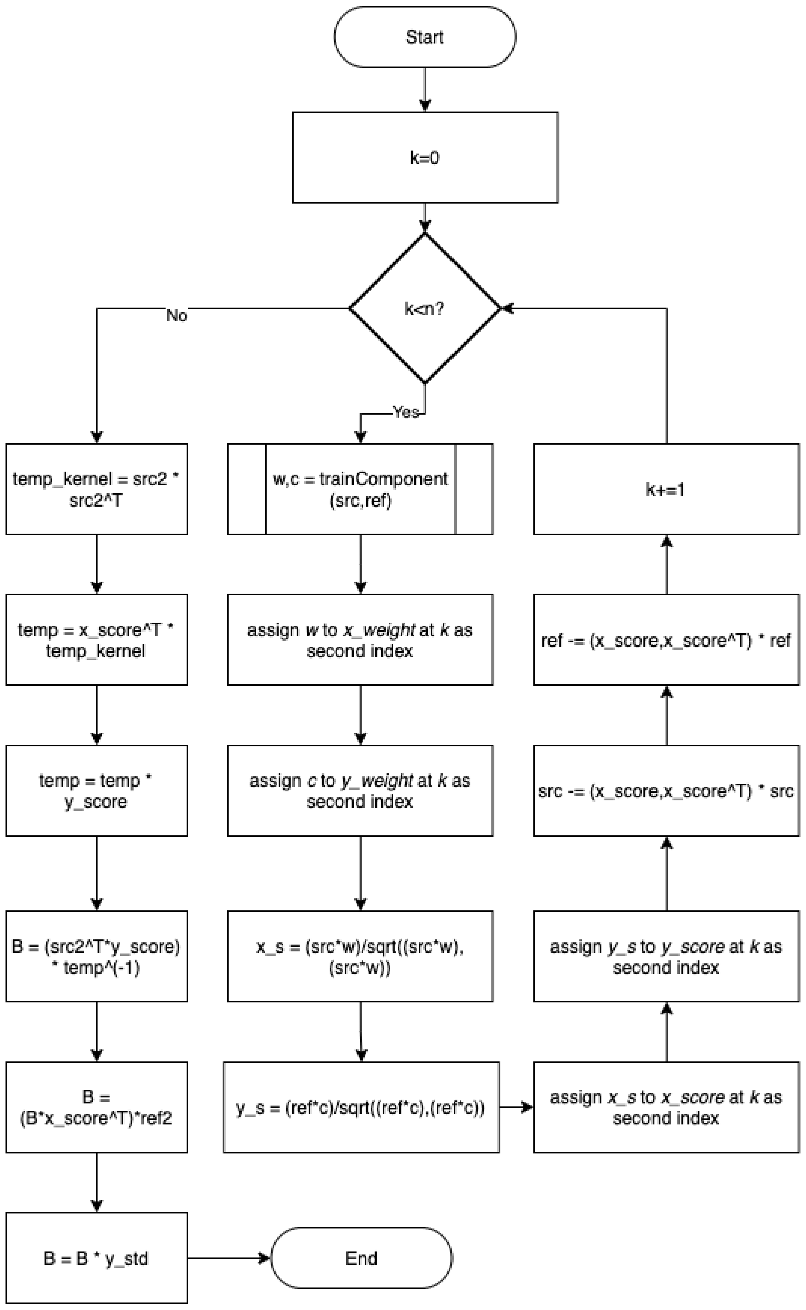

Start
k=0
k<n?
No
Yes
temp_kernel = src2 * src2^T
w,c = trainComponent (src,ref)
k+=1
temp = x_score^T * temp_kernel
assign *w* to *x_weight* at *k* as second index
ref -= (x_score,x_score^T) * ref
temp = temp * y_score
assign *c* to *y_weight* at *k* as second index
src -= (x_score,x_score^T) * src
B = (src2^T*y_score) * temp^(-1)
x_s = (src*w)/sqrt((src*w),(src*w))
assign *y_s* to *y_score* at *k* as second index
B = (B*x_score^T)*ref2
y_s = (ref*c)/sqrt((ref*c),(ref*c))
assign *x_s* to *x_score* at *k* as second index
B = B * y_std
End

Figure 4.6: Training part of K-PLSRO flowchart.

Where *src* variable is reference value and ref variable is the actual value.

Above flowchart is shown the iteration for each components according the K-PLSR algorithm in the chapter 3. As we can see, each components will get the convergence weight of reference image and actual image. Then calculate the score vector of reference image and actual image for getting the model. To make it become convergence, the *trainComponent* function should be used. In this function, it will be looped until 500 times until it convergence. For more detail, the flowchart of it can be shown on the figure (4.7).

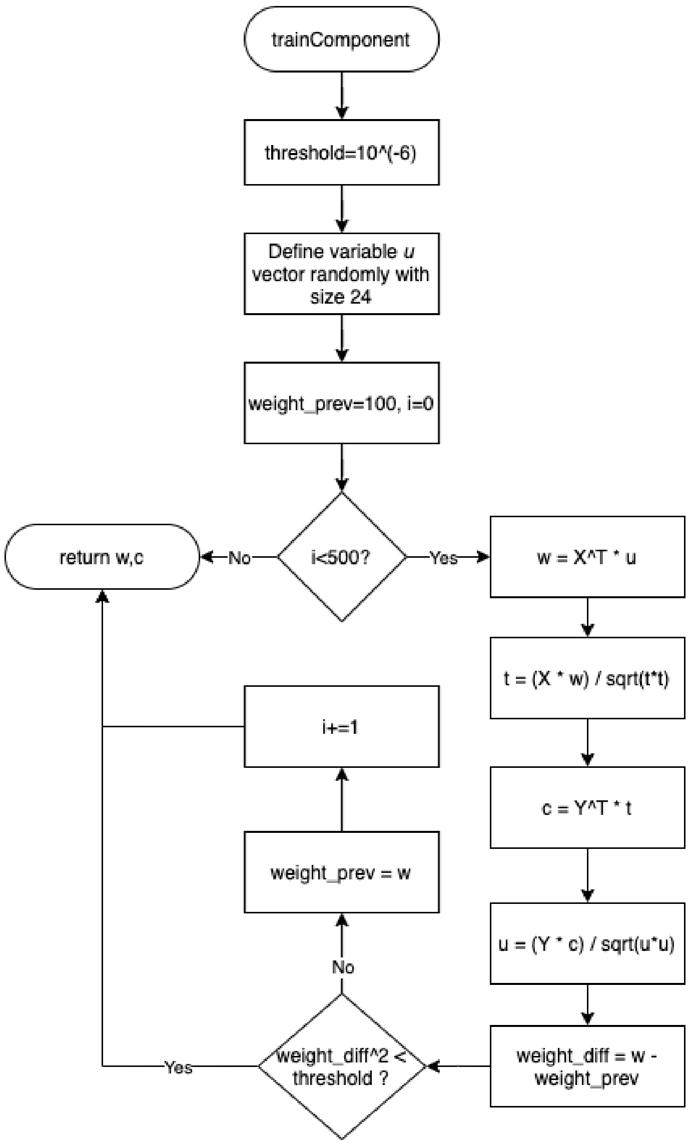
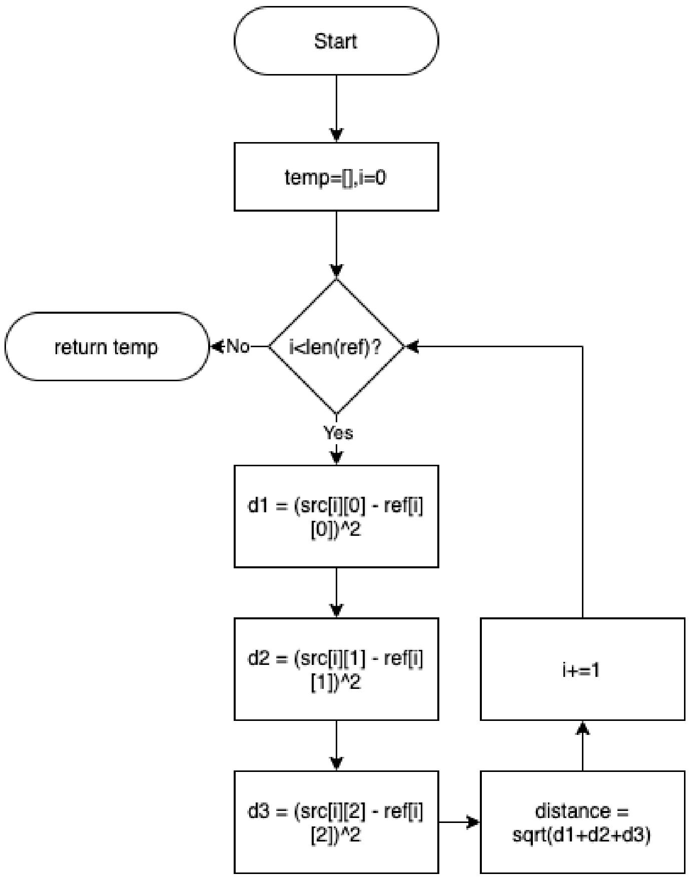


Figure 4.7: Training component of K-PLSRO flowchart.

Where *X* variable is reference value and *Y* variable is actual value.

The flowchart on the figure (4.6) will give an output as model which usually we called it mapping model. In the flowchart, mapping model is represented by variable *B*. Then the mapping model will be evaluated by using flowchart on the figure (4.8) and stored it in memory.


Start
temp=[],i=0
i<len(ref)?
No
return temp
Yes
d1 = (src[i][0] - ref[i][0])^2
d2 = (src[i][1] - ref[i][1])^2
d3 = (src[i][2] - ref[i][2])^2
distance = sqrt(d1+d2+d3)
i+=1
Figure 4.8: Evaluating mapping model flowchart.

Where *src* variable is corrected value and ref variable is actual value.

8. Iterate step 3-7. Because the defined parameters have 44 parameters, which are sigmoid has 16 parameters, RBF has 10 parameters, and chi-squared has 18 kernels, it means we will iterate the flowchart above 43 times.

9. After all mapping models is evaluated, we will search the minimum evaluation result which we got from figure (4.8).

10. As the result, we can get the best mapping model for the input. Then we will do center scale again to the input image and add the value by 10 if the best kernel is chi-squared.

11. Then, the final step is the prediction step. The prediction step can be performed by the flowchart which is shown on the figure (4.9). In the flowchart, The input image and reference value will be performed kernel trick to bring them to the higher dimension, then the result will be multiplied with the best mapping model. After that, we can see that if the value smaller than 0, we should bring it to the 0, and if the value bigger than 255, we should bring it to the 255. We must do it because RGB value range is only from 0 until 255. Finally, we will get the corrected result.

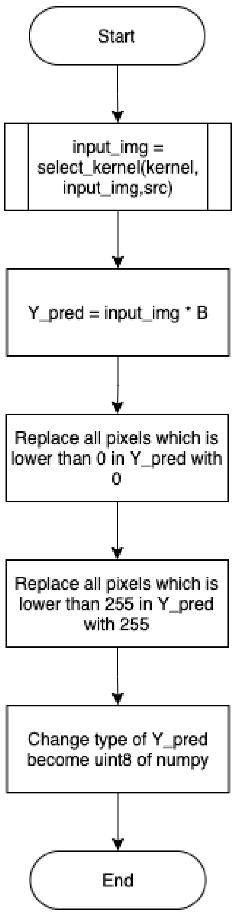

Start
input_img = select_kernel(kernel, input_img,src)
Y_pred = input_img * B
Replace all pixels which is lower than 0 in Y_pred with 0
Replace all pixels which is lower than 255 in Y_pred with 255
Change type of Y_pred become uint8 of numpy
End

Figure 4.9: Prediction flowchart.

Where *src* variable is reference value, *kernel* variable is selected kernel, and $input - img$ variable is uncorrected image in array.

4.4 Image Segmentation

After the corrected tongue image has been gotten, the tongue image should be separated from the background to do further process. This step is important because to calculate tongue body pixels, the background of images should be removed to make the tongue body distinguishable from the background. For doing so, an image segmentation process is needed to segment the image and the background by labeling them separately. If the segmented image was gotten, the background of the tongue image can be removed by doing masking.

In chapter 3, according to the evaluation result, the U-Net architecture will be used. The details of the architecture can be seen below.

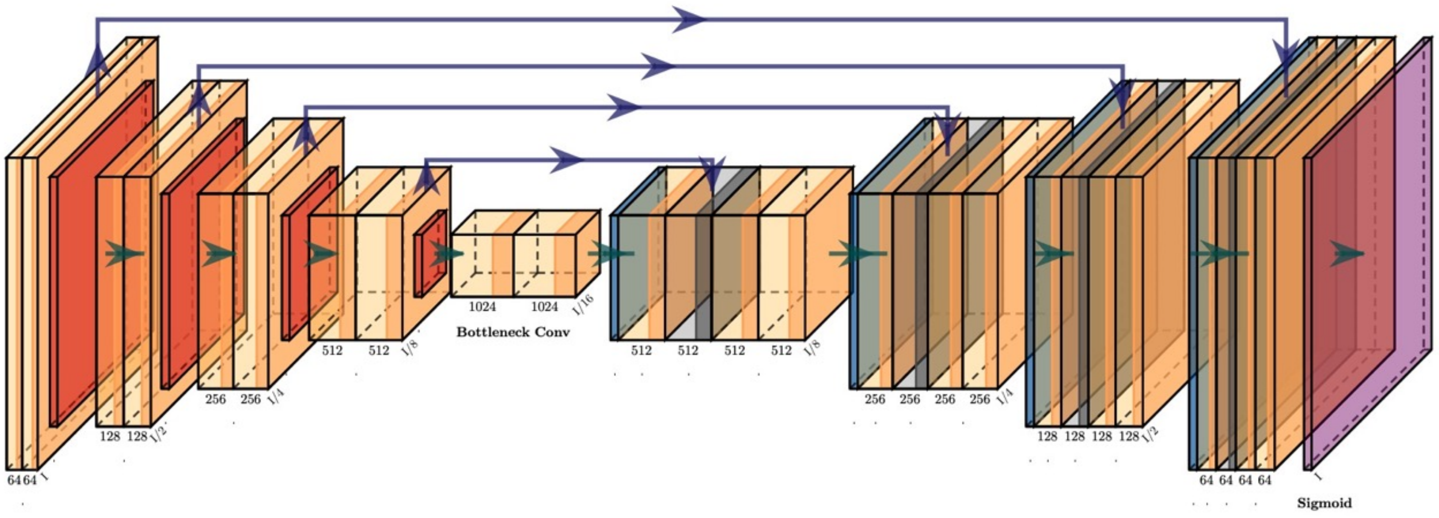


Figure 4.10: U-Net architecture.

As can be seen on the figure above. The architecture above is divided into two parts. Those are the encoder part and decoder parts. Each part will do a different task but meet each other's needs.

1. Encoder

Encoder is the first half part of the U-Net architecture. In U-Net, it is only simply a stack of convolutional layers and max-pooling layers. Here is where the downsampling is performed which means this part has a role to find "what" is important in the image by reducing the dimension of weight and height becomes a deeper dimension, but it will lose "where" information from the image. Here is the pattern of the layers.

$$convlayer_1 \rightarrow convlayer_2 \rightarrow maxpooling \rightarrow dropout(optional)$$

Each unity of the layer will be stacked and produce the deeper dimension from the wider dimension where each convolutional layer will use the ReLU activation function which is represented by equation (4.5).

$$f(x) = max(0,x) \tag{4.5}$$

Where x is the value of the output. The purpose of implementing the ReLU activation function is to activate the feature that will be used only which means only the value greater than 0 will be used and improve the speed of the computation. So, the training and prediction speed can be faster too.

2. Decoder

Decoder is the second half part of U-Net architecture. In U-Net, it is the transposed convolutional layers along with the regular convolutional layer. Here is where the upsampling is performed which means this part has a role to find "where" is the important part. To get the precise location, it needs to merge or concatenate the output of transposed convolutional layer with the feature from the encoder part from the same level. Here is the pattern of the merge.

(a) $conv_5 = conv_5 + conv_4$

(b) $up_6 = up_6 + conv_3$

(c) $up_7 = up_7 + conv_2$

(d) $up_8 = up_8 + conv_1$

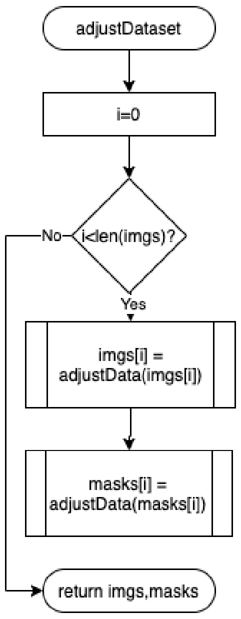
At the end of the decoder, the sigmoid function will be used since it will only need binary classification for the mask image of the tongue. The equation of sigmoid function is represented on equation (4.6).

$$f(x) = \frac{1}{1+e^{-x}} \tag{4.6}$$

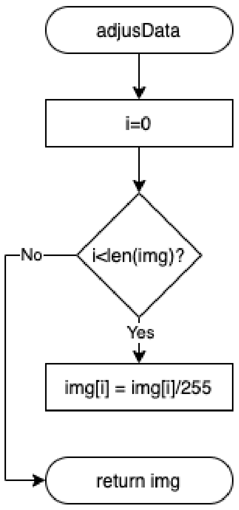
Where x is the final output of each pixel. The value will be classified between range [0,1] where 0 is black and 1 is white.

From the architecture, we can know that the predicted result will produce one channel image which means it will be a grayscale image. The predicted result of this step is the masking image that will be used to separate the tongue body and the background. The background in the image will be labeled as 0 and the tongue body will be labeled as 255. For more detail about image segmentation implementation, the steps will be as followed.

1. Load all dataset including the tongue image and the mask by using generator library.
2. After that, we should zip the image and mask become pairs.
3. Then, we should adjust the data become range $[0,1]$ to reduce the calculation cost and convergent can be reached faster. To adjust it, the flowchart which is shown on the figure (4.11) is used.


adjustDataset
i=0
i<len(imgs)?
No
Yes
imgs[i] = adjustData(imgs[i])
masks[i] = adjustData(masks[i])
return imgs,masks

(a)



(b)

Figure 4.11: Adjust dataset flowchart. (a) Adjust dataset. (b) Adjust each image.

Where *imgs* variable is dataset image of tongue and *masks* variable is ground-truth of dataset which is masks of *imgs* variable.

4. After the training generator is gotten, load the model architecture as shown on the figure (4.10).
5. Start the training phase with 10 epochs and 10 step per epoch and save the model.
6. After training the model, we can start the prediction. First, resize the image to the size of the architecture input which is 256×256 pixels. The example of before and after can be seen in the figure (4.12).

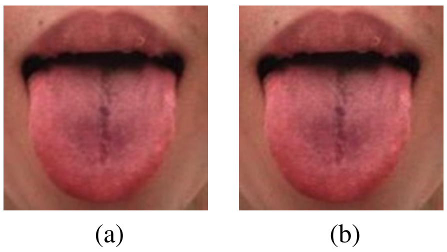


Figure 4.12: Resizing image. (a) Before. (b) After.

7. Because we already has the trained model, we can use it for the prediction step. Load the architecture of the model and also the parameter of the trained value. After that, predict the mask image according to the input. The example of input and output can be in the figure (4.13).

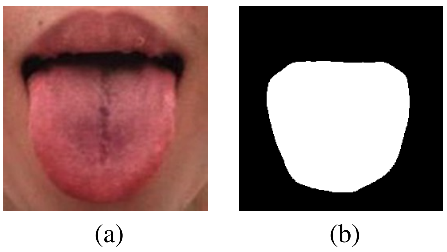


Figure 4.13: Predicting mask. (a) Input. (b) Output.

8. Resize the mask image to the actual size value of cropped tongue image.
9. After resizing done, masking the image by using the resized prediction result. The example of masking can be seen in the figure (4.14) which is the final result.

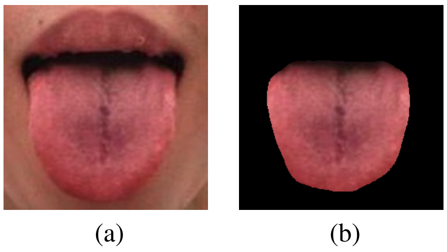


Figure 4.14: Masking. (a) Input. (b) Output.

4.5 Tongue Color Classification

Tongue color classification is the final step of this framework. This step is intended to classify the color of the segmented image according to the table (3.1). The table will be converted to the LAB color space since the evaluation will be held in the LAB color space. The LAB version is shown on the table below. The final result of this step is in the percentage of the classified color. For classifying it, euclidean distance will be used to calculate the similarity. Usually, there are many available algorithms that can be used for this step, such as K-Nearest Neighbor(K-NN), Neural Network(NN), SVM, etc. But, this proposed framework is considered to use euclidean distance.

Table 4.2: Tongue ten colors reference LAB version.

Color	L*	a*	b*
Cyan	76.07	-0.56	1.36
Red	52.25	34.84	21.30
Purple	69.47	42.47	-23.89
Deep Red	37.84	24.55	25.94
Light Red	69.46	28.49	13.39
Light Purple	76.06	24.32	-9.77
Black	37.84	3.96	20.58
Gray	61.65	5.71	3.73
White	70.97	10.98	8.29
Yellow	56.68	9.55	24.45

Euclidean is considered more appropriate with this case because this case only needs to classify each pixels to the closest color in the table (3.1) and it will be more expensive to only predict color pixels by using advance algorithms like DNN which needs big dataset. Getting a balance big dataset for twelve colors references is hard without image augmentation. However, doing image augmentation is not good for this case because the variation of color will be decreased. So, the classification is done by calculating the similarity is the best choice, not only cheaper, but it's more accurate without a big dataset.

For more further explanation about this subsection, following steps can be seen below.

1. Convert the image from RGB color space to the CIELAB color space.
2. Calculate the distance of each pixels with the color in the table (3.1) by using euclidean distance which is the formula is shown on the equation (4.7).

$$Distance = \sqrt{(L_r - L_t)^2 + (a_r - a_t)^2 + (b_r - b_t)^2} \tag{4.7}$$

Where L_r, a_r, b_r is the L*, a, b value of the reference color and L_t, a_t, b_t is the L*, a, b value of the pixel. To calculate and classify them, the flowchart that is shown on the figure (4.15) is used.

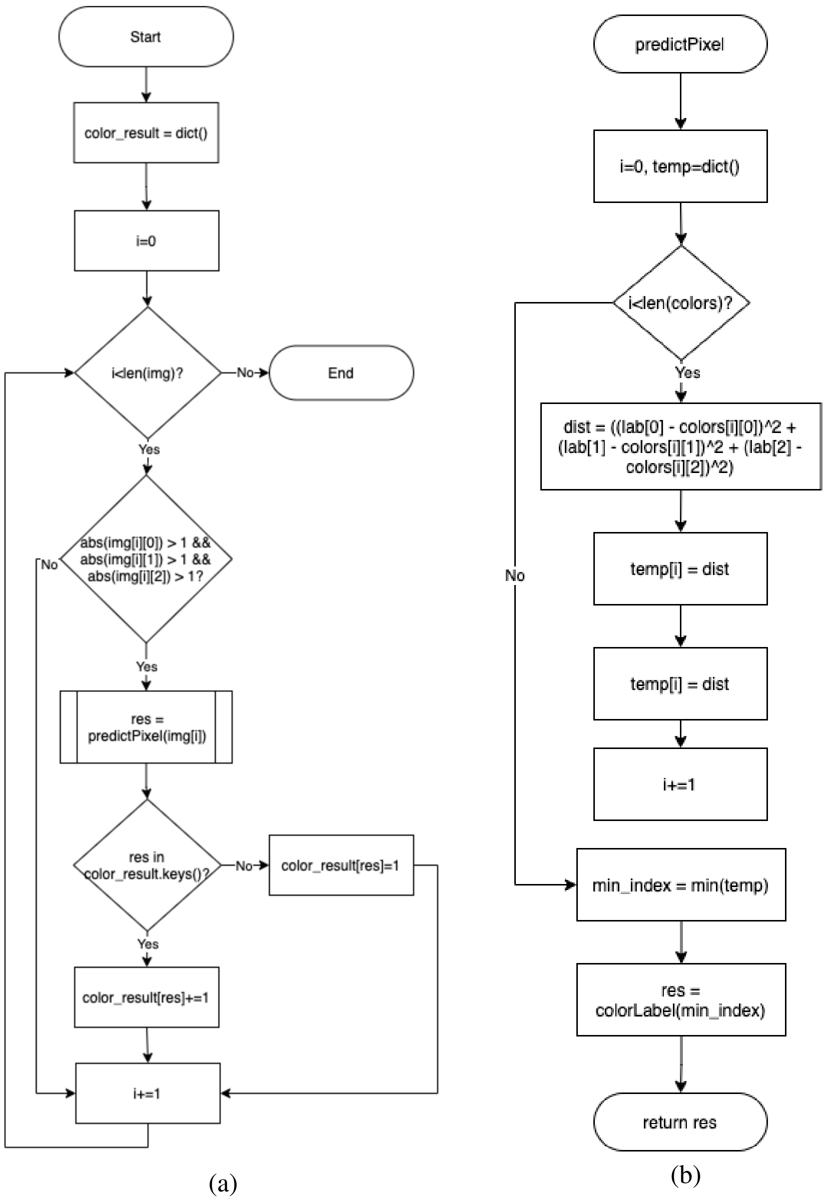

Start
color_result = dict()
i=0
i<len(img)?
No
End
Yes
abs(img[i][0]) > 1 && abs(img[i][1]) > 1 && abs(img[i][2]) > 1?
No
Yes
res = predictPixel(img[i])
res in color_result.keys()?
No
color_result[res]=1
Yes
color_result[res]+=1
i+=1
(a)
predictPixel
i=0, temp=dict()
i<len(colors)?
Yes
dist = ((lab[0] - colors[i][0])^2 + (lab[1] - colors[i][1])^2 + (lab[2] - colors[i][2])^2)
temp[i] = dist
temp[i] = dist
i+=1
No
min_index = min(temp)
res = colorLabel(min_index)
return res
(b)

Figure 4.15: Classify pixel flowchart. (a) Classifying each pixel. (b) Predict pixel function.

Where *img* variable is segmented image, *colors* variable is an array which contains LAB color space version value of table (3.1), and *colorLabel* variable is an array which contains label of table (4.2).

This flowchart will only use the pixel that is bigger than 1 and classify the pixel to the label which is gotten from the prediction result and place it according to the predicted label and position of the pixel in the actual value.

3. After all pixels has been classified, the total of each pixels is divided by the total of all pixels for getting the percentage and saved according to the color.

4. Besides the percentage, the median also will be calculated for the evaluation purpose only. The median result will be linear with the percentage result, but it can be used to see how the point is adjusted.

The median is calculated according to the each axis for the balance purpose. Because if it's not performed according to the each axis, if the case that one of the median is too high, the other will adjust to that value and the bias result will happen.

After the percentage of each colors have been gotten, the dominant color can be seen as the color of tongue. Regardless of that thing, the other classified color can be used too for separating the color of tongue body and coating. Then those data can be used to predict the disease. However, that purpose is outside of this scope since this thesis only intends to get a consistent color by using proposed framework.

CHAPTER 5

EXPERIMENTAL RESULT AND ANALYSIS

5.1 Experimental Setting

An experimental setting is required for setting the configuration of the experiment and make the same perspective when evaluating the result. There are two subsections that should be configured here, those are the experimental equipment and environment and also the parameters of the experimental algorithm.

5.1.1 Experimental Setting of Equipment and Environment

There are few things that should be configured and explained in this subsection. First, the devices that will be used in this experiment. There are three Xiaomi smartphones that will be used to capture the images, those are:

1. Redmi Note 8 Pro selfie camera
2. Mi Max Pro selfie camera
3. Mi 5 selfie camera

After that, the experiment will be held in two certain conditions. The first condition will be held in a constant lighting environment and using different smartphones. The second condition will be held in various lighting environments and using the same smartphone which is Redmi Note 8 Pro selfie camera.

The constant lighting environment is held in a certain room size with 100cm x 100cm x 300cm and with a certain configuration as shown in the figure (5.1). Whereas the various lighting condition is taken in the random place such as inside room, inside house, and outside the house.

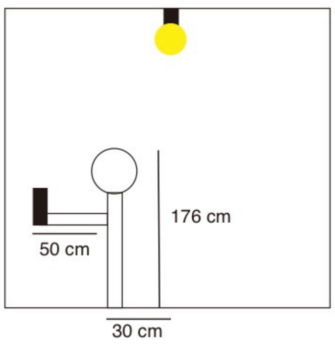


Figure 5.1: Condition 1 environment.

5.1.2 Experimental Setting of Algorithms Parameters

In this subsection, the parameters of the chosen algorithm will be figured out. The first algorithm that needs configuration is K-PLSR optimized. In the K-PLSR optimized, there are two candidates that should be set, those are σ value and kernel. After a certain experiment, the kernel candidates are RBF kernel, chi-squared kernel, and sigmoid kernel which is the best kernel for this case. The σ value candidates are set around range $[10,20]$ for RBF kernel, $[2,20]$ for the chi-squared kernel, and $[4,20]$ for the sigmoid kernel which is the safest range to avoid overfitting and underfitting.

Besides K-PLSR optimized, U-Net should be configured too especially about dataset and training phase. Dataset is one of the most important things in deep learning for making a good model. The dataset that will be used in this step is the tongue images dataset for making the model recognize which part is the tongue part and which is the background part. So, the ground-truth of the images will be the result of segmentation like shown in the figure (5.2) (b). Since it is quite hard for getting the tongue image and the amount of the dataset should be quite many to make a good model, so the dataset is gotten from the open-source which is from Github (https://github.com/BioHit/TongeImageDataset). There are 300 images total dataset that will be used.

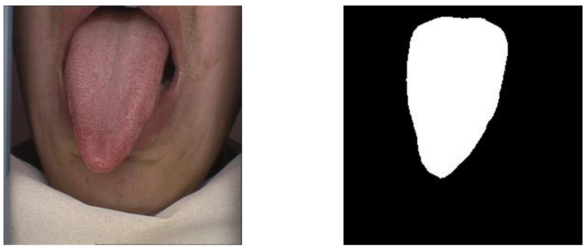

(a) (b)

Figure 5.2: Dataset of tongue image. (a) Training image. (b) Ground-truth image.

The size of the real dataset is 576 x 768 pixels. But since the model required 256 x 256 pixels for making the computation speed faster, then it should be resized. Then they will be training data and the model will be trained for 10 epochs and 10 steps per epoch. Finally, the training phase can reach 92.43% accuracy, 98.91% precision, and 83.73% recall.

5.2 Experimental Result

In this experiment, it has been said that there are two certain conditions that will be used. The evaluation itself is quite complex which is divided become two types. Those are subjective evaluation and objective evaluation. In subjective evaluation, it is performed by observing by people's perspective, whereas objective evaluation is performed by measuring values in a certain condition.

As has been configured in the previous section, the first condition of this experiment is held in the same environment and captured by using different smartphones. Figure (5.3) shows three captured images with condition 1. Without using certain measurements, it can be seen obviously the color difference between each captured image is a lot. Whereas figure (5.4) shows three predicted framework result. The improvisation of colors can be seen improved quite a lot. The color becomes quite similar rather than before.

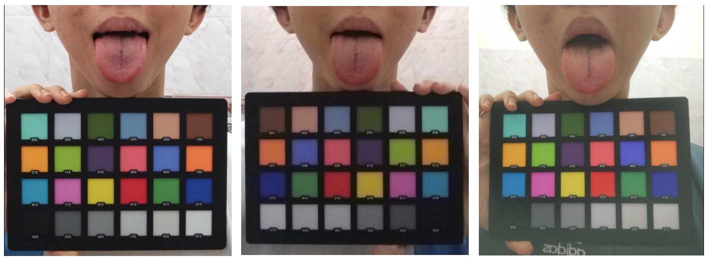


Figure 5.3: Input image. (a) Redmi Note 8 Pro. (b) Mi Max Pro. (c) Mi 5.

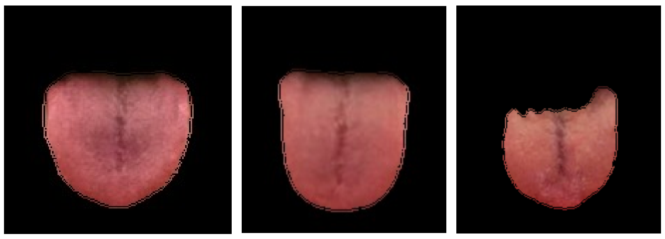
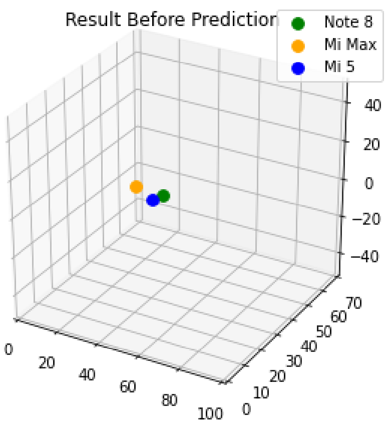


Figure 5.4: Result of framework. (a) Redmi Note 8 Pro. (b) Mi Max Pro. (c) Mi 5.

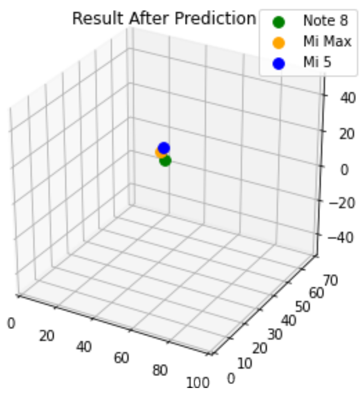
From the table (5.1) which R represents red, DR represents deep red, LR represents light red, BK represents black, W represents white, Y represents yellow, P represents purple, C represents cyan, LP represents light purple, and G represents gray, it can be seen that the results are going to classify the color percentage to the certain color which are red and dark red. It means the result is going to the consistent prediction result. For more clear, figure (5.5) below can represent how the framework shifts the median to one point closer and figure (5.6) below can see how each color pixel is classified to more consistent result.

Table 5.1: Percentage of color pixel classification before and after using framework in condition 1.

Smartphone	R(%)	DR(%)	LR(%)	BK(%)	W(%)	Y(%)	P(%)	C(%)	LP(%)	G(%)
Before										
Note 8 Pro	46.56	16.56	28.14	3.32	0.30	0.40	0.00	0.00	0.00	4.87
Mi Max Pro	6.72	1.10	0.00	12.19	1.29	63.10	0.00	0.00	0.00	0.00
Mi 5	7.16	11.05	0.00	7.52	1.37	66.96	0.00	0.00	0.00	5.87
After										
Note 8 Pro	79.76	14.76	1.69	3.72	0.00	0.00	0.00	0.00	0.00	0.00
Mi Max Pro	83.19	12.59	0.00	4.17	0.00	0.00	0.00	0.00	0.00	0.00
Mi 5	83.59	14.60	1.75	0.00	0.00	0.00	0.00	0.00	0.00	0.00

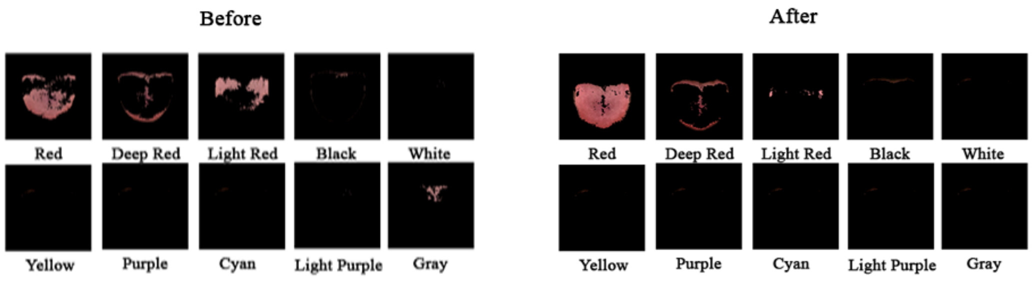


(a)

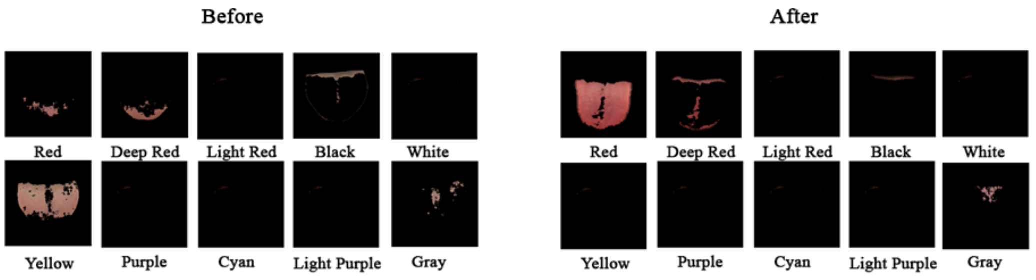


(b)

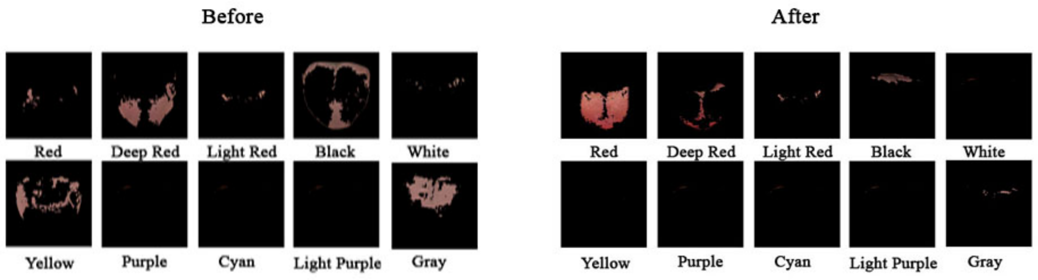
Figure 5.5: Diagram median color of condition 1. (a) Result before prediction. (b) Result after prediction.



(a)



(b)



(c)

Figure 5.6: Result pixel classification of condition 1. (a) Redmi Note 8 Pro. (b) Mi Max Pro. (c) Mi 5.

For making sure more whether this proposed framework can classify the tongue color to give a more consistent and objective result, the other tongue images of the other person which is shown on the figure (5.7) will be used to verify it.

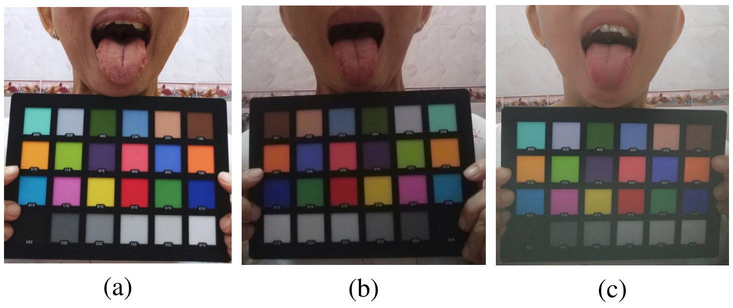


Figure 5.7: Additional input image. (a) Redmi Note 8 Pro. (b) Mi Max Pro. (c) Mi 5.

As a result, we can see from table (5.2) that the result is going to have more consistent result too which are the color red as a dominant and dark red get more percentage from the classification result.

Table 5.2: Additional percentage of pixel classification before and after using framework in condition 1.

Smartphone	R(%)	DR(%)	LR(%)	BK(%)	W(%)	Y(%)	P(%)	C(%)	LP(%)	G(%)
Before										
Note 8 Pro	16.02	38.48	5.52	3.56	2.96	33.22	0.00	0.00	0.00	0.13
Mi Max Pro	0.00	93.92	0.00	6.05	0.00	0.00	0.00	0.00	0.00	0.00
Mi 5	2.41	25.43	0.00	24.95	0.00	19.61	0.00	0.00	0.00	27.52
After										
Note 8 Pro	59.72	39.23	0.48	0.50	0.00	0.00	0.00	0.00	0.00	0.00
Mi Max Pro	60.18	37.38	0.00	2.40	0.00	0.00	0.00	0.00	0.00	0.00
Mi 5	70.84	28.49	0.00	0.62	0.00	0.00	0.00	0.00	0.00	0.00

After the experiment with condition 1, the experiment with condition two is held in various environments and with the same smartphone. Figure (5.8) shows three captured images with condition 2. It also can be shown obviously how different the color before using the proposed framework. Whereas figure (5.9) shows three predicted results after using the proposed framework. The improvisation of it also can be seen which the color is shift closer than before.

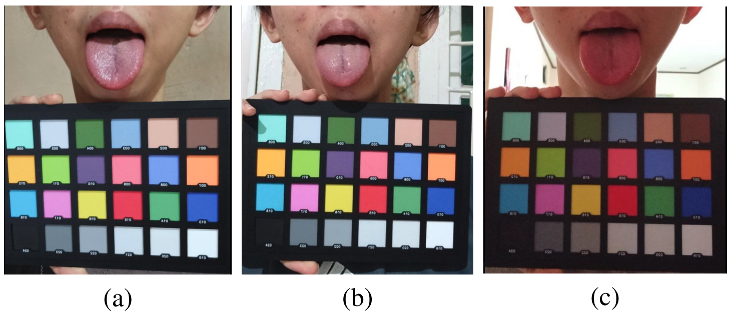


Figure 5.8: Input image. (a) Environment 1. (b) Environment 2. (c) Environment 3.

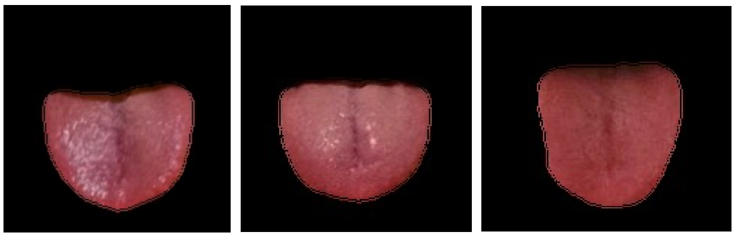
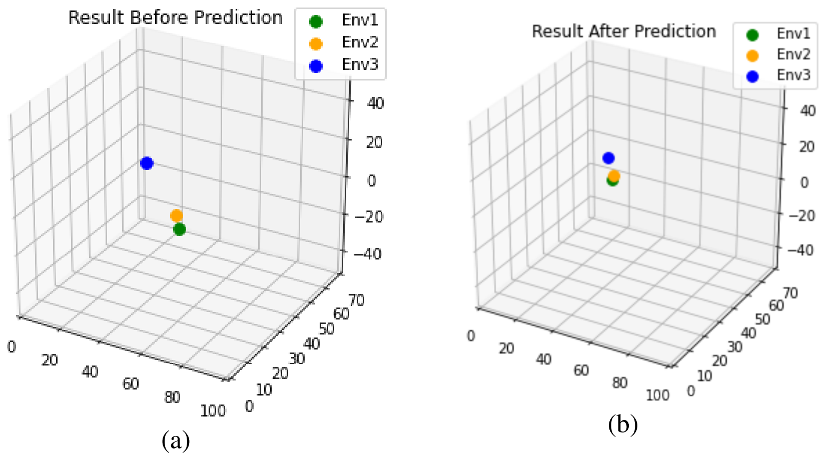

(a) (b) (c)

Figure 5.9: Result of framework. (a) Environment 1. (b) Environment 2. (c) Environment 3.

Also from the table (5.3), it can be seen that the results are going to classify the color percentage to the red and dark red. It means the result is going to the constant prediction result too. For more clarity, figure (5.10) below can represent how the framework shifts the median to one point closer rather than before prediction and figure (5.11) below can see how each color pixel is classified to more consistent result.

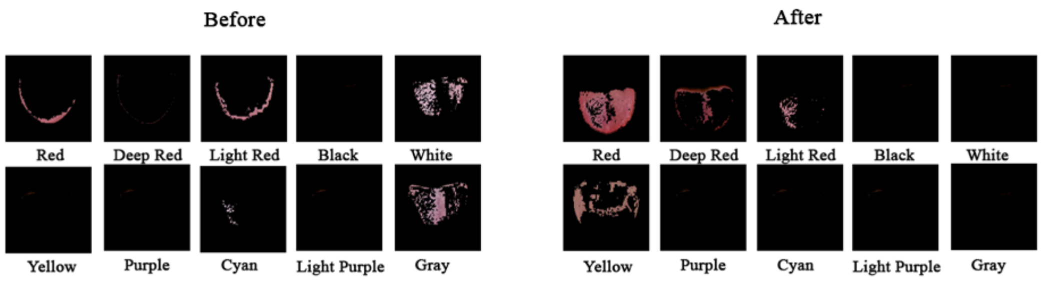
Table 5.3: Percentage of color pixel classification before and after using framework in condition 2.

Environment	R(%)	DR(%)	LR(%)	BK(%)	W(%)	Y(%)	P(%)	C(%)	LP(%)	G(%)
Before										
Environment 1	11.01	1.42	15.99	2.47	30.78	0.00	0.00	2.02	0.00	36.17
Environment 2	17.22	3.05	22.23	3.02	36.59	0.14	0.00	0.00	0.00	17.60
Environment 3	6.08	92.03	0.21	1.61	0.00	0.01	0.00	0.00	0.00	0.00
After										
Environment 1	74.35	17.87	7.47	0.25	0.00	0.00	0.00	0.00	0.00	0.00
Environment 2	76.25	19.83	1.91	1.93	0.00	0.00	0.00	0.00	0.00	0.00
Environment 3	42.54	53.05	0.00	4.35	0.00	0.00	0.00	0.00	0.00	0.00

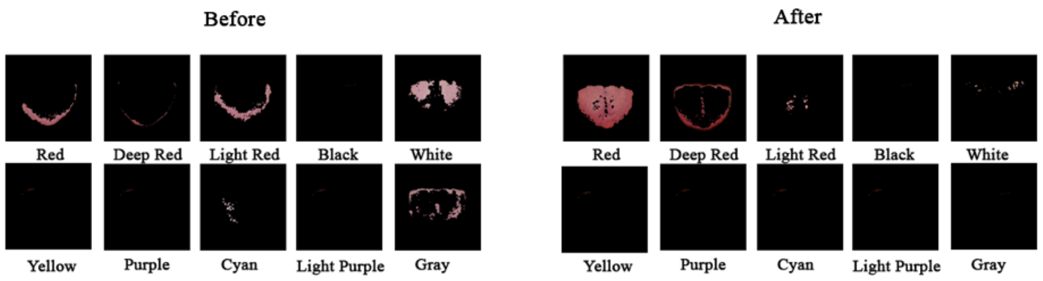


(a) (b)

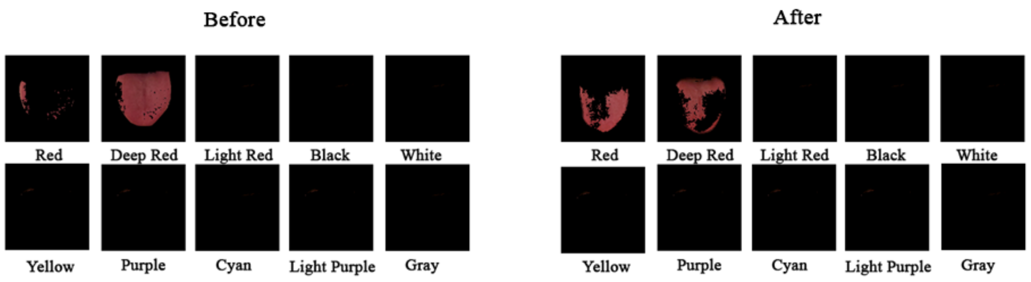
Figure 5.10: Diagram median color of condition 2. (a) Result before prediction. (b) Result after prediction.



(a)



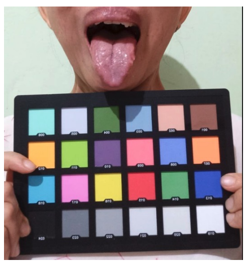
(b)

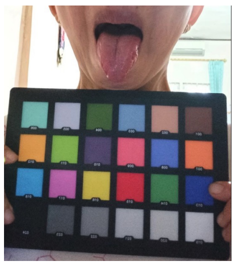


(c)

Figure 5.11: Result pixel classification of condition 2. (a) Environment 1. (b) Environment 2. (c) Environment 3.

In the first experiment, we add 3 more images of the other person's tongue to verify whether the proposed framework can successfully classify it to the more consistent result. So as in the second experiment, we will add more 3 images to verify it. The images are shown on the figure (5.12).


(a)


(b)

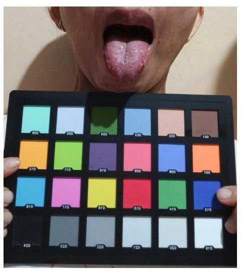

(c)

Figure 5.12: Additional input image. (a) Environment 1. (b) Environment 2. (c) Environment 3.

As a result, we can see from table (5.4) that the result is going to have more consistent result too which are the color red as a dominant and dark red get more percentage from the classification result like in the first experiment, although the color distribution is not really close.

Table 5.4: Additional percentage of pixel classification before and after using framework in condition 2.

Environment	R(%)	DR(%)	LR(%)	BK(%)	W(%)	Y(%)	P(%)	C(%)	LP(%)	G(%)
Before										
Environment 1	9.29	1.51	46.45	2.72	26.62	0.00	00.0	0.29	0.00	12.98
Environment 2	38.82	27.80	2.46	11.25	1.93	0.00	0.00	0.00	0.00	17.12
Environment 3	4.74	16.63	18.79	2.64	10.96	0.00	0.00	0.00	0.00	62.17
After										
Environment 1	74.66	21.36	3.73	0.00	0.15	0.00	0.00	0.00	0.00	0.01
Environment 2	51.07	45.07	0.97	2.82	0.00	0.00	0.00	0.00	0.00	0.00
Environment 3	50.91	40.98	6.69	1.07	0.03	0.00	0.00	0.00	0.00	0.21

5.3 Experimental Discussion

The proposed framework experiment is performed to show how this framework can improve captured image to get constant prediction results in a certain condition such as in different environments and with different smartphones. From the experiments that have been held, it can show that the experiments can bring the color to the more constant result which is measured in CIELAB color space. For further explanation in comparing the result, this section will discuss more about it.

The first experiment in condition 1 shows the different captured colors in different cameras can be adjusted to nearly the same color which is known from the dominant color, that is red. The improvement shows that the maximum distance of CIELAB median color distance changes from 15.32 to 3.95. From the CIELAB median also can be seen the adjusted color is going to shift the value to one same point in the diagram. It means the proposed framework can make a nearly constant result for the device-independent case from an objective evaluation perspective. Whereas from the subjective evaluation perspective, the color can be seen to have a slightly different color only.

Moreover, from the table (5.1), we can see that how the color classification distribution is changed. Before the proposed framework is used, smartphone Note 8 Pro is dominant at red with slightly light red and deep red. But smartphone Mi Max Pro is more dominant at deep red and yellow. Whereas Mi 5 is more dominant at yellow and slightly deep red and black. The color distribution of those smartphones are totally not the same with each other. But after the proposed framework is used. The color of smartphones are classified to the dominant red and slightly deep red which are red is around 79.76% - 83.59% and deep red is around 12.59% - 14.76%. So as the additional input, the result can be seen produce more consistent colors which are red is around 59.72% - 70.84% and deep red is around 28.49% - 39.23%. It can be implied that, the proposed framework can successfully classify the color of tongue.

The second experiment in condition 2 shows that from the figure (5.10) we can see how far the distance if the condition is covered by shadow can be improved. The proposed framework can shift the value of the maximum distance from 35.34 become 10.79. Although the value is not really close to zero, it shows the proposed framework can also support environment-independent quite well from the objective evaluation perspective. However, one of them which is environment 3 is kept dominant in the deep red but in the improvement state with the CIELAB color goes to the dominant red. Whereas from the subjective evaluation the color is not as good as the first experiment, but the color difference shows an improvement.

Like in the first experiment, table (5.3) shows the classification result can distribute percentage of color becomes in the same dominant color. Environment 1 and 2 show the red dominant with value 74.35% - 76.25% and deep red dominant with value 17.87% - 19.83%. Whereas the environment 3 is more dominant at deep red color with value 53.05% and red color with value 42.54%. So as the additional input, the result is distributed to the red dominant with value 50.44% - 74.66% and deep red dominant with value 21.36% - 45.69%. It can be said that the proposed framework can improved the result of environment-independent.

Although the result can be considered success, the computation speed of each steps also needs to be observed to know how effective the time consumption of the proposed framework. The measurement also performed with the same hardware specification like the evaluation algorithms in the chapter 3. The average time computation of each steps can be seen on the table (5.5).

Table 5.5: Comparison average time computation of the steps in the proposed framework.

Steps	Average time computation (seconds)
Color correction	0.88
Segmentation	9.45
Pixel Classification	16.45
Total	26.78

The result shows that the average time needed to get the final result of one tongue is 26.78 seconds. The taken time is considered quite longer for predicting only one tongue. Color correction and segmentation don't take too much longer time, but the pixel classification needs more time rather than both steps before since the classification needs to take the pixel one by one entirely. But since the used of the proposed framework not for real-time purpose, the average time is still can be performed for public online used. So that the improvisation to make it can be performed real-time is needed in the future.

However in fact the color of healthy tongue is light red, but in this experiment most of them are shown as red color dominant. However, for more further improvement for predicting disease, the related dataset with this proposed framework which is has the equal color parameter should be used to get an accurate result in the future. Regardless of that problem, this proposed framework is success to get more consistent result in the color classification of the tongue.

For more further, the classified colors in each tongue is not only represent the dominant color. But also they can be used to separate tongue body and coating which are important for ATD system in the future. It means the result of this proposed framework will become important data for input of future ATD system.

CHAPTER 6

CLOSING

6.1 Conclusion

According to the previous chapter's discussion, the conclusion is as follows:

1. This thesis can make an alternative framework for TCCS as a part of ATD system to get more consistent color classification result.

2. The proposed framework is intended to get a consistent result of tongue image for public online used case in case of sensor-independent.

3. The proposed framework experiment has successfully improved the result for the sensor-independent used case.

4. The proposed framework can help the improvement of the environment-independent used case.

6.2 Limitation & Future Works

The proposed framework's result in the environment-independent is still not as good as sensor-independent which is caused by different light spread in a different area. In order to do improvement in environment-independent case, it needs to add more layer in the framework to handle the light spread which cannot be done in this chance because of the time limitation and knowledge limitation.

In the future, this framework is expected to have a better result by adding a layer to improve the quality of the environment-independent case. Moreover, this framework still depends on the X-Rite color checker as a reference which will reduce the simplicity to do diagnose in public online used cases. For the improvement, it's expected to do estimation without X-Rite color checker in the future. Besides that, the improvisation for making it real-time also can be considered for the future research.

REFERENCES

[1] C. quan Ling, L. na Wang, Y. Wang, Y. hui Zhang, Z. fei Yin, M. Wang, and C. Ling, "The roles of traditional Chinese medicine in gene therapy," 2014.

[2] C. Q. Ling, X. Q. Yue, and C. Ling, "Three advantages of using traditional Chinese medicine to prevent and treat tumor," 2014.

[3] J. K. Anastasi, L. M. Currie, and G. H. Kim, "Understanding diagnostic reasoning in TCM practice: tongue diagnosis.," *Alternative therapies in health and medicine*, 2009.

[4] T. Wu, K. Wu, W. Hu, J. Sheen, C. Lu, J. Chiang, and Y. Hung, "Tongue diagnosis indices for upper gastrointestinal disorders: Protocol for a cross-sectional, case-controlled observational study," *Medicine (United States)*, 2018.

[5] M.-C. Hu, K.-C. Lan, W.-C. Fang, Y.-C. Huang, T.-J. Ho, C.-P. Lin, M.-H. Yeh, P. Raknim, Y.-H. Lin, M.-H. Cheng, Y.-T. He, and K.-C. Tseng, "Automated tongue diagnosis on the smartphone and its applications," *Computer Methods and Programs in Biomedicine*, vol. 174, pp. 51–64, jun 2019.

[6] L. Lo, Y. Chen, J. Chiang, T. Cheng, and N. Damdinsuren, "Education of Chinese medicine tongue diagnosis by automatic tongue diagnosis system," *Chinese Journal of Integrative Medicine*, 2015.

[7] D. Zhang, H. Zhang, and B. Zhang, *Tongue Color Analysis for Medical Application*, pp. 207–223. Singapore: Springer Singapore, 2017.

[8] A. Bosio, P. Bernardi, A. Ruospo, and E. Sanchez, "A Reliability Analysis of a Deep Neural Network," in *2019 IEEE Latin American Test Symposium (LATS)*, pp. 1–6, mar 2019.

[9] Y. Lu, X. Li, L. Zhuo, J. Zhang, and H. Zhang, "Dccn: A Deep-Color Correction Network For Traditional Chinese Medicine Tongue Images," in *2018 IEEE International Conference on Multimedia Expo Workshops (ICMEW)*, pp. 1–6, jul 2018.

[10] J. Hou, H. Su, B. Yan, H. Zheng, Z. Sun, and X. Cai, "Classification of tongue color based on CNN," in *2017 IEEE 2nd International Conference on Big Data Analysis (ICBDA)*, pp. 725–729, IEEE, mar 2017.

[11] H. Zhang, K. Wang, X. Jin, and D. Zhang, "SVR based color calibration for tongue image," in *2005 International Conference on Machine Learning and Cybernetics*, vol. 8, pp. 5065–5070 Vol. 8, aug 2005.

[12] A. S. R. Sinaga, "Color-based Segmentation of Batik Using the L*a*b Color Space," *SinkrOn*, vol. 3, no. 2, p. 175, 2019.

[13] H. Niu, Q. Lu, and C. Wang, "Color Correction Based on Histogram Matching and Polynomial Regression for Image Stitching," in *2018 IEEE 3rd International Conference on Image, Vision and Computing (ICIVC)*, pp. 257–261, jun 2018.

[14] G. D. Finlayson, M. Mohammadzadeh Darrodi, and M. Mackiewicz, "The alternating least squares technique for nonuniform intensity color correction," *Color Research & Application*, vol. 40, no. 3, pp. 232–242, 2015.

[15] X. Wang and D. Zhang, "An Optimized Tongue Image Color Correction Scheme," *IEEE Transactions on Information Technology in Biomedicine*, vol. 14, pp. 1355–1364, nov 2010.

[16] J. Sui, C. Xia, K. Yang, Y. Zhang, Y. Wang, H. Yan, and P. Qian, "Tongue image color correction method based on root polynomial regression," in *2019 IEEE 8th Joint International Information Technology and Artificial Intelligence Conference (ITAIC)*, pp. 1337–1342, may 2019.

[17] G. D. Finlayson, M. Mackiewicz, and A. Hurlbert, “Color Correction Using Root-Polynomial Regression,” *IEEE Transactions on Image Processing*, vol. 24, pp. 1460–1470, may 2015.

[18] G. D. Finlayson, M. Mackiewicz, and A. Hurlbert, “Root-polynomial colour correction,” in *Final Program and Proceedings - IS and T/SID Color Imaging Conference*, 2011.

[19] B. Wei, “The research on color reproduction and texture morphological analysis of tcm tongue analysis,” *Beijing University of Technology, Beijing*, 2004.

[20] M. H. Tania, K. T. Lwin, and M. A. Hossain, “Computational complexity of image processing algorithms for an intelligent mobile enabled tongue diagnosis scheme,” in *2016 10th International Conference on Software, Knowledge, Information Management Applications (SKIMA)*, pp. 29–36, Dec 2016.

[21] L. Zhuo, P. Zhang, P. Qu, Y. Peng, J. Zhang, and X. Li, “A K-PLSR-based color correction method for TCM tongue images under different illumination conditions,” *Neurocomputing*, 2016.

[22] Y. Cai, “A novel imaging system for tongue inspection,” in *Conference Record - IEEE Instrumentation and Measurement Technology Conference*, 2002.

[23] B. Zhang, X. Wang, J. You, and D. Zhang, “Tongue color analysis for medical application,” *Evidence-based Complementary and Alternative Medicine*, 2013.

[24] S. Han, X. Yang, Q. Qi, Y. Pan, Y. Chen, J. Shen, H. Liao, and Z. Ji, “Potential screening and early diagnosis method for cancer: Tongue diagnosis,” *International Journal of Oncology*, 2016.

[25] M.-C. Hu, G.-Y. Zheng, Y.-T. Chen, and K.-C. Lan, “Automatic Tongue Diagnosis Using a Smart Phone,” *2014 IEEE International Conference on Systems, Man, and Cybernetics*, 2014.

[26] Y. Zhou, K. Gao, Y. Guo, Z. Dou, H. Cheng, and Z. Chen, “Color correction method for digital camera based on variable-exponent polynomial regression,” in *Communications, Signal Processing, and Systems* (Q. Liang, X. Liu, Z. Na, W. Wang, J. Mu, and B. Zhang, eds.), (Singapore), pp. 111–118, Springer Singapore, 2020.

[27] X. Ding, Y. Wang, J. Zhang, and X. Fu, “Underwater image dehaze using scene depth estimation with adaptive color correction,” in *OCEANS 2017 - Aberdeen*, pp. 1–5, 2017.

[28] X. Cheng, B. Khomtchouk, N. Matloff, and P. Mohanty, “Polynomial Regression As an Alternative to Neural Nets,” *arXiv*, jun 2018.

[29] S. Yuheng and Y. Hao, “Image segmentation algorithms overview,” 2017.

[30] S. P. Mary, Ankayarkanni, U. Nandini, Sathyabama, and S. Aravindhan, “A Survey on Image Segmentation Using Deep Learning,” *Journal of Physics: Conference Series*, vol. 1712, no. 1, pp. 1–22, 2020.

[31] Saparudin, Erwin, and M. Fachrurrozi, “Tongue Segmentation Using Active Contour Model,” *{IOP} Conference Series: Materials Science and Engineering*, vol. 190, p. 12041, apr 2017.

[32] J. O. Pinzón-arenas, R. Jiménez-moreno, and C. G. Pachón-suescún, “ResSeg : Residual encoder-decoder convolutional neural network for food segmentation,” vol. 10, no. 1, pp. 1017–1026, 2020.

[33] Y. Xu, M. Cai, L. Lin, Y. Zhang, H. Hu, Z. Peng, and Q. Zhang, “PA-ResSeg : A Phase Attention Residual Network for Liver Tumor Segmentation from Multi-phase CT Images,” vol. di, pp. 1–23.

[34] G. Du, X. Cao, J. Liang, X. Chen, and Y. Zhan, “Medical image segmentation based on U-Net: A review,” *Journal of Imaging Science and Technology*, vol. 64, no. 2, pp. 1–12, 2020.