School of Computing
FACULTY OF ENGINEERING
UNIVERSITY OF LEEDS

Final Report

Investigation of Image Processing Techniques for Metal Detection in Water

Kang Liu

Submitted in accordance with the requirements for the degree of BSc Computer Science

2020/21

40 credits

The candidate confirms that the following have been submitted.

Items	Format	Recipient(s) and Date
Report - Digital	PDF	Minerva (10/05/2021)
Project Repository	GitHub Repository URL (See Appendix A)	Supervisor, Assessor (10/05/2021)

Type of project: Exploratory Software

The candidate confirms that the work submitted is their own and the appropriate credit has been given where reference has been made to the work of others.

I understand that failure to attribute material which is obtained from another source may be considered as plagiarism.

Kang Liu

© 2020/21 The University of Leeds and Kang Liu

Summary

Due to environmental concerns, it has become significant to accurately detect and monitor heavy metals in the industrial processes and environment. However, most of industrial techniques require complicated conditions and cost a lot. Thurs, there is a high demand for simple, efficient and portable detection methods.

As a result, this project has designed and implemented two image processing algorithms to detect metal in water. And it evaluates these two algorithms through comparison. What's more, this project has implemented an Android application and related web service to demonstrate the feasibility of the detection methods.

Acknowledgements

I would like to thank my project supervisor Muhammad Azhar lqbal, for his continuous support and guidance in both meetings and external communications throughout the whole project.

Contents

Chapter 1

Introduction

1.1 Project Background

Due to environmental concerns, it has become significant to accurately detect and monitor heavy metals in the industrial processes and environment. However, most of industrial techniques require complicated conditions and cost a lot. Thurs, there is a high demand for simple, efficient and portable detection methods.

1.2 Project Aims

This project focus on metal detection in water through image processing techniques. The main aim is to investigate, design and implement an image processing algorithm to achieve metal detection. This algorithm will use an image token by the user as input, it will then analyse and extract the features in the image to get the concentration of the metal. In addition, the evaluation of the algorithms is also important.

Alongside the main aim of the project, the secondary aim of the project is to develop and provide a generic mobile application platform. And a web service for the mobile application to access the image processing algorithm is also needed. This is to demonstrate the feasibility of the detection methods.

1.3 Objectives

1. Develop a base mobile application that allows users to upload data and get results.
2. Develop a web service that receives users' requests and sends the results back.
3. Design and evaluate image processing algorithms that detect the concentration of metals in water.
4. Reflect on limitations and possible future improvements to the mobile application, web service and image processing algorithms.

1.4 Deliverables

1. An android application developed by Android Studio.

2. A web service developed by Python Flask and deployed in Pythonanywhere.

3. Two image processing algorithms have been implemented and compared for evaluation.

4. Future improvements discussed in this report.

1.5 Initial Project Plan (Gantt Chart)

As is shown below in Figure 1.1, it is a Gantt Chart showing the initial project plan. The time line is arranged that there is one meeting with the supervisor every two weeks before the winter vacation and every one week after the vacation. The task: design and implementation of algorithms cost most of the time in the project, because it is the core aim of the project.

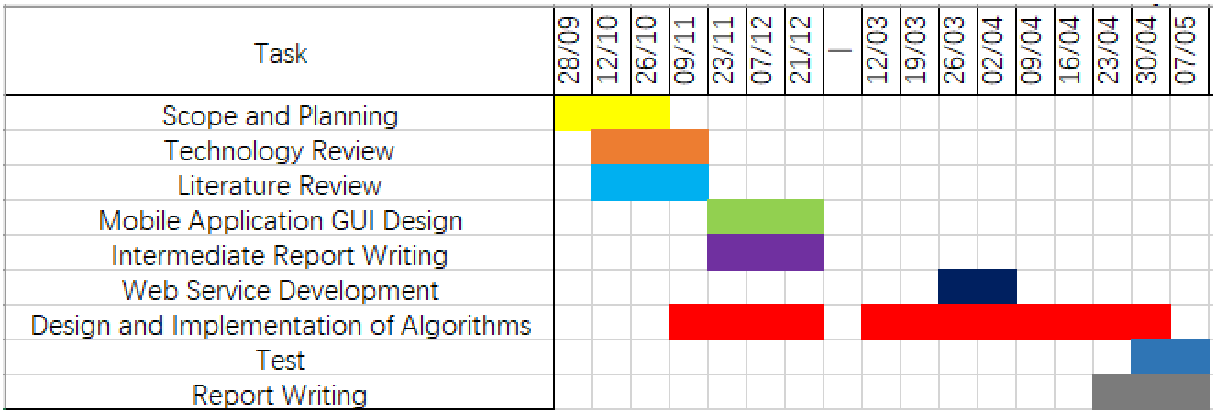


Figure 1.1: Gantt Chart for the project planning.

1.6 Version Control

This project use Git to control the version and manege the development of project. Git is also used for storing the code and important files in the project. The git repository url is shown in the Appendix A.

Chapter 2

Background Research

2.1 Introduction

Rapid development of industrial factories and progress in related technologies are causing increasing environmental problems [3, 7]. Most of heavy metals are non-biodegradable and can easily accumulate in ecological systems which makes them one of the most problematic pollutants. Heavy metals can also be harmful to public health. For example, cadmium (Cd) can accumulate in kidney and liver for over 10 years and affects physiological functions of a human's body eventually [8]. As a result, it has become significant to accurately detect and monitor heavy metals in the industrial processes and environment.

2.2 Current Detection Methods of Heavy Metals

Many techniques have been developed for the detection of heavy metals, including inductively coupled plasma mass spectrometry (ICP-MS), inductively coupled plasma-atomic/optical emission spectrometry (ICP-AES/OES), electrothermal atomic adsorption spectrometry (ETAAS), and so on. Many of these techniques have high sensitivity, specificity and accuracy. But these methods all require complicated facilities, professionals, and complex operation. Therefore, in developing countries and regions with a lack of sufficient infrastructure, professional experts, and appropriate environmental treatment, there is a high demand for simple, efficient and portable detection methods [6].

Considering above concerns, paper-based analytical devices (PADs) is one of the most promising solutions, which is cheap, simple, sensitive, specific, accurate, user-friendly, and environmental-friendly [6]. Thurs, we choose to use the results of PADs in this project.

2.3 Paper-based Analytical Devices and Sample Images

There are three main steps in the metal detection through PADs. First, the water sample is dipped on filter paper. Second, a specific reagent is applied to know the existence of particular metal. Third, the application of specific reagent to water on filter paper will produce only a single color showing the existence of particular metal and its concentration. There exist special types of chromogenic chemicals known as reagents which are reactant to a particular metal. For instance for cobalt(Co) and zinc(Zn), chromogenic reagents are 4-(5-bromo-2-pyridylazo)-N,N-diethyl-3-hydroxy-aniline (5-Br-PADAP) and dithizone (H2DZ), respectively.

Due to the limitation of the experimental conditions and available data, we focus on three metals' detection: cobalt(Co), hydrargyrum(Hg), and nickel(Ni). Figure 2.1 below shows examples of three metals' detection results, ppm (parts per million) is the unit of concentration which is the mass fraction of solute per million.

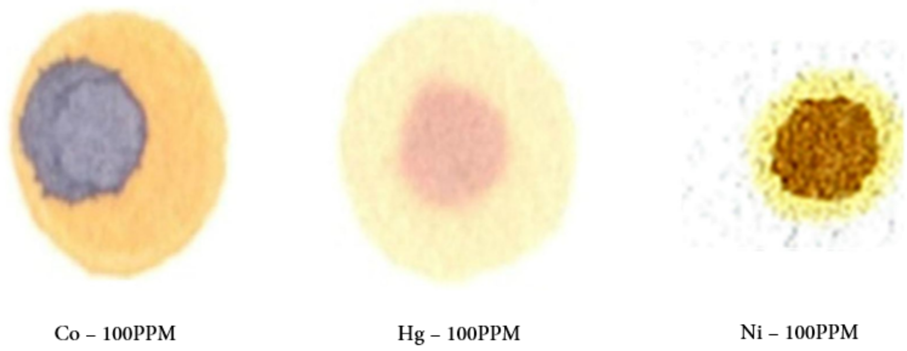


Figure 2.1: Examples of three metals' appearances on PADs with concentration 100PPM.

Figure 2.2 below shows a group of examples of cobalt(Co). It can be indicated that the concentration of a metal has relationship with the color in the PADs test result.

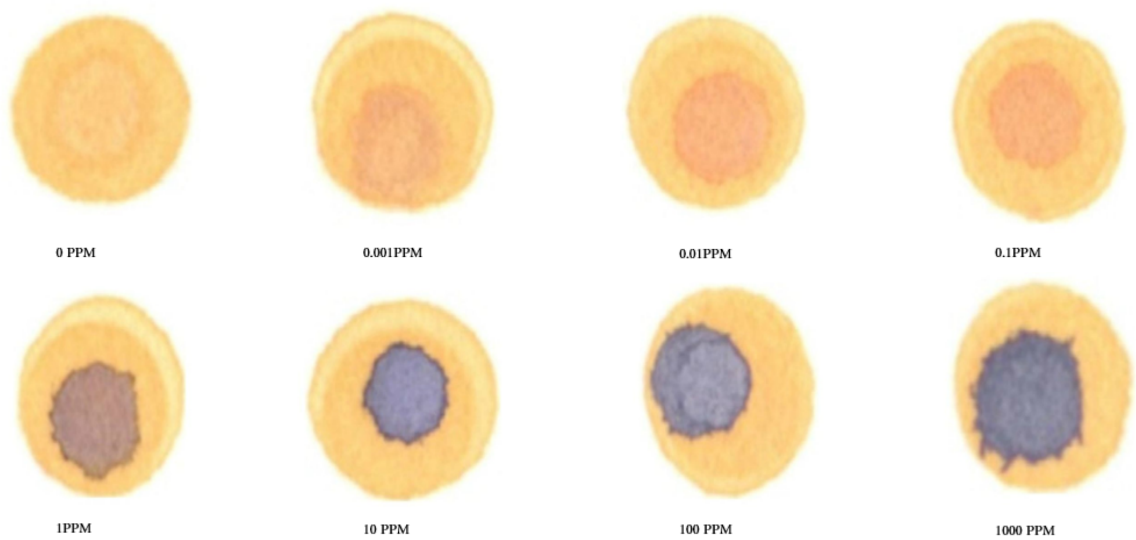


Figure 2.2: A group of examples showing cobalt(Co)'s appearances on PADs with different concentration ranging from 0 to 1000PPM.

2.4 Color Detection

Color detection is an important step in image processing. It is to classify pixels into different color regions from a given image [13]. Specifically, the aim of color detection in this project is to find the area showing the existence of particular metal and its intensity which is also called region of interest. As the results of this project are based on chemical reaction, environmental factors can easily cause decline in image quality like background noise, motion blur, or shadowing effects. So, image pre-processing needs to be carried out to partially reduce the impact of environmental factors on the images.

2.5 Color Space

There are many kinds of color spaces (also called color models) used for image representation, each with their own strengths and weaknesses [13].

First, RGB color space is the most frequently-used color model in computer storage. It has three color channel red, green and blue corresponding to the intensity of each primary color. The value of each color channel usually ranges from 0 to 255 in computer. The main advantage to use RGB color space is its simplicity which is easy to store, configure and display the image in all kinds of devices. However, some important color properties, such as brightness and purity, are embedded within the RGB color channels. So, it can be difficult to determine the specific colors and their reliable working ranges [13].

The other commonly used color spaces for image processing are the hue-based color models. They usually include: Hue-Chroma-Value (HCV), Hue-Chroma-Lightness (HCL), Hue-Saturation-Value (HSV) and Hue-Saturation-Lightness (HSL). In these models, hue represents the color angle, chroma or saturation represents the color purity, value or lightness represents the color brightness. Hue-based color models tend to be a favourable choice for colour detection because they separate important properties such as brightness and purity from the image. This allows better separation between different colours and thus they are the preferred colour models for various colour detection applications [11].

2.6 Color Distance in RGB Space

Color distance is a value to measure the difference between colors. It has different forms in different color space. According to the Euclidean distance between points in space, the fundamental form in RGB color space can be represented as:

$$\Delta E_{RGB} = \sqrt{\Delta R^2 + \Delta G^2 + \Delta B^2} \tag{2.1}$$

However, the human eye has a different sensitivity to the changes in the light intensity of the primary colors. These changes in the intensity of the colors cause different sensations [9]. This is taken into account by the following formula with the sensitivity coefficients to compensate for different color components of the eye. It has the form:

$$\Delta E_{RGB}' = \sqrt{3\Delta R^2 + 4\Delta G^2 + 2\Delta B^2} \tag{2.2}$$

2.7 Image Histogram Matching

Image histogram matching is a simple and basic method for image processing. It is usually used to compare, modify or select an image with a similar one [4]. This method changes the image into histogram by calculating the frequency of the pixels in grey scale or color distance in different color space. The histogram can show some features through the distribution of colors. As the histogram is easy to handle and analyse, the method is common in the pre-processing for data preparation in the bigger algorithms.

2.8 K-Means Algorithm

The K-Means algorithm is a clustering algorithm usually used in unsupervised learning. The algorithm is to partition a data set into k clusters, which groups all the data into K sets or classes. The method is to find k centers in the data set and assign other items to the nearest cluster which is represented by its center. After several iterations, when the centers have no changes larger than the error required, the algorithm finishes and the clusters determined by the last k centers are the objective [2].

In image processing, the K-Means algorithm can be used to classify the pixels into several groups with required features.

2.9 Linear Regression

Linear regression models are widely used to estimate the importance of parameters and predict the response variables at any point in the parameter space. [5]. There are many forms of linear regression model, one of the simpler forms of such models is:

$$y = wx_i + b \tag{2.3}$$

To solve the model, least square method is used to minimise the square loss while m is the size of the training set:

$$E_{(w,b)} = \sum_{i=1}^{m}(y_i - wx_i - b)^2 \tag{2.4}$$

And the solution is:

$$w = \frac{\sum_{i=1}^{m} y_i(x_i - \bar{x})}{\sum_{i=1}^{m} x_i^2 - (\sum_{i=1}^{m} x_i)^2/m} \tag{2.5}$$

$$b = \frac{1}{m}\sum_{i=1}^{m}(y_i - wx_i) \tag{2.6}$$

2.10 Existing Applications in Past Papers

There are two existing applications similar to this project in past papers. The first one is an on-site detection with a smartphone to detect free chlorine, hydrogen sulfide and formaldehyde in wastewater [1]. The other one is an on-site detection of heavy metals in wastewater which is the same as this project [10].

These two applications both use mobile application to take photo of paper test strip and get the result through on-site detection service. Their detection methods are also the same. They both use linear regression to get the linear model on a certain range of concentration and achieve a high precision and accuracy. The main advantage of this method is low cost and requirement to obtain a high accuracy.

These two applications have inspired me of the design in image processing algorithms and the demonstration application.

Chapter 3

Design and Implementation

3.1 Mobile Application

3.1.1 Technologies and Requirements Specification

First, the mobile application is developed in android platform. Android application is one of the most widely-used application in smart phone. So, it can easily demonstrate the feasibility of the algorithms which is one of the aims of the project. And the experience in development of android application in the past software project is the other significant reason. Then, we choose to use Android Studio to implement. Android Studio is a powerful development tool with its GUI designer, mobile simulator and project management tools.

It should be mentioned that this is a exploratory software project and the mobile application is designed to demonstrate the feasibility of algorithms, so the functions implemented in the android application are kept to a minimum. This application is aimed to use in the industrial processing or environmental monitoring. The user group of the application would be workers or inspector with limited background knowledge. So, the application doesn't focus on handling unavailable input data which could only be the wrong operations made by workers. As a result, the application mainly has two functions to be implemented:

1. Allow users to upload data.
2. Display the results.

3.1.2 User Interface Design and Implementation

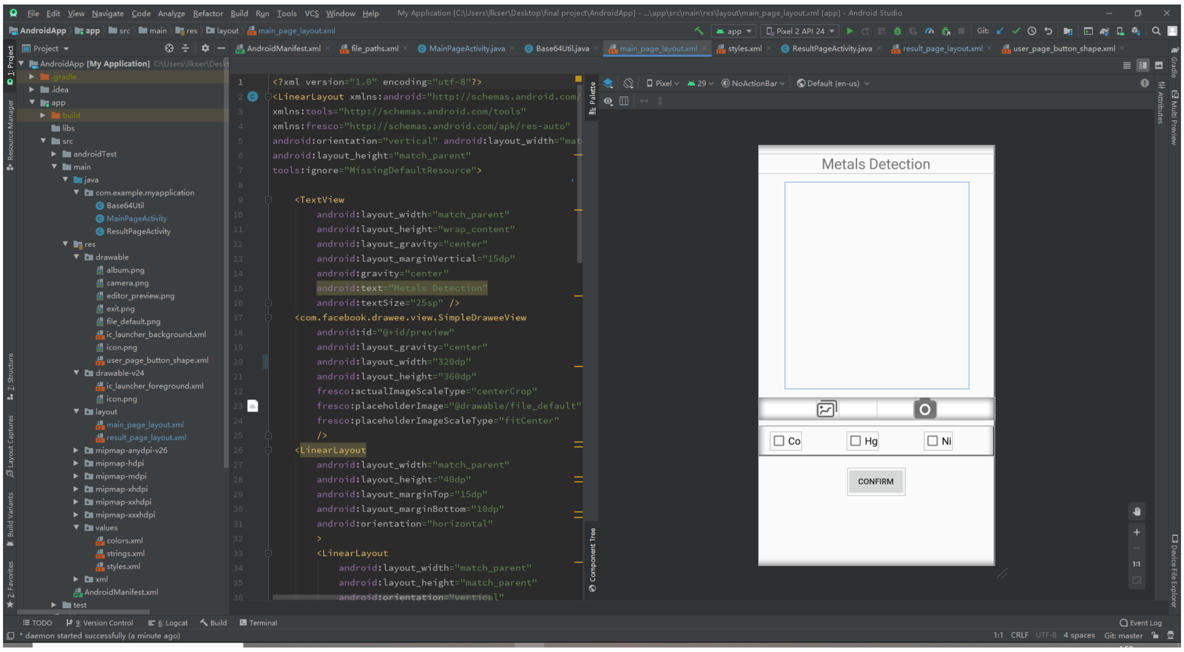
The mobile application mainly have two pages to be designed and implemented.

First is the home page. This page should allow users to upload data to the server. The data should include an image to be tested and a choice representing which metal would be detected. So, as is shown in Figure 4.1(a), the home page has the following elements implemented (from top to bottom):

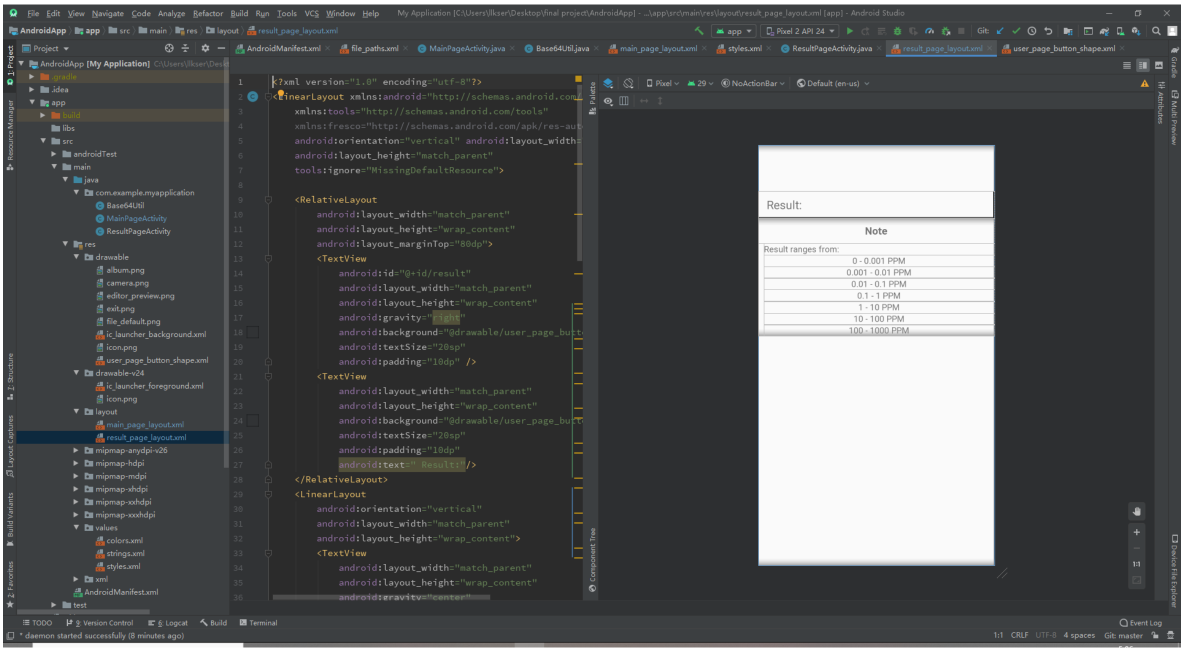
1. The title in a textbox.
2. The image window for preview.
3. Two icons. The left one is to access system album to choose photo, and the right one is to open camera.

4. Three checkboxes for user to choose which metal to be detected.

5. The confirm button.



(a) Main page



(b) Result page

Figure 3.1: Screenshots of GUI design and implementation in Android Studio.

Second is the result page. This page displays the results of user's test image. As this page is a activity called by the main page, this page has implemented an actionbar using theme in android: *android:theme="@style/AppTheme.ActionBar.Userpage.Transparent"*. Adding what is shown in Figure 4.1(b), the result page has the following elements implemented (from top to bottom):

1. The android actionbar which show the title and return button.
2. The result textbox to show the feedback from the server.
3. The note list show the ranges of detection.

3.1.3 Camera and Album Functionality

One of the core function for this application is the camera support and allowing users to access the system album to choose photo to be uploaded. The camera and system album can be accessed by pressing the icons in the home page shown in Figure 4.1(a). The camera support is provided using file provider in android. When the user press the icon, the application call the camera activity provided by the system and get the imageUri as the result to access the address of the image in the system storage. Similarly, the application allows users to access the system album by calling system activity and get the imageUri.

```
private void openAlbum(){
    Intent intent=new Intent( action: "android.intent.action.GET_CONTENT");
    intent.setType("image/*");
    startActivityForResult(intent,CHOOSE_PHOTO);
}

private void openCamera(){
    File outputImage = new File(getExternalCacheDir(),  child: "output_image.jpg");
    try {
        if (outputImage.exists()) {
            outputImage.delete();
        }
        outputImage.createNewFile();
    } catch (IOException e) {
        e.printStackTrace();
    }
    if(Build.VERSION.SDK_INT>=24){
        imageUri= FileProvider.getUriForFile( context: MainPageActivity.this,
                 authority: "com.example.cameraalbumtest.fileprovider",outputImage);
    }else{
        imageUri=Uri.fromFile(outputImage);
    }
    Intent intent=new Intent( action: "android.media.action.IMAGE_CAPTURE");
    intent.putExtra(MediaStore.EXTRA_OUTPUT,imageUri);
    startActivityForResult(intent,TAKE_PHOTO);
}
```

Figure 3.2: A screenshot of the code implementations for the camera and system album.

3.1.4 Uploading data

When the user clicks on the confirm button, the application will check for data validation. And after getting the imageUri as described above and user's choice for a certain metal detection, the application starts to form the http request to the server. As is shown in Figure 4.3, the application first encodes the image to string with Base64Util class implemented in the other source file. And then, it forms a POST http request using OkHttpClient class in android with those input data. Finally, the application sends the request to the server url and waits for data response.

```java
    Bitmap bitmap = MediaStore.Images.Media.getBitmap(getContentResolver(), imageUri);
    String picture=new Base64Util().bitmapToBase64(bitmap);

    OkHttpClient client = new OkHttpClient();
    FormBody.Builder formBuilder = new FormBody.Builder();
    formBuilder.add( name: "image", picture);
    if(metal_Co.isChecked())
        formBuilder.add( name: "metalCode", value: "1");
    else if(metal_Hg.isChecked())
        formBuilder.add( name: "metalCode", value: "2");
    else
        formBuilder.add( name: "metalCode", value: "3");
    Request request = new Request.Builder().url("http://llkserfinal.pythonanywhere.com/test").post(formBuilder.build()).build();
    final Call call = client.newCall(request);
    call.enqueue(new Callback()
```

Figure 3.3: A screenshot of the code implementation for the method to upload data in mobile application.

3.2 Web Service

3.2.1 Technology Choice

To host the mobile application and provide a platform to implement the image processing algorithms, a web framework is needed. As the image processing algorithms are implemented in Python, the web framework should also base on Python. So, we choose Python Flask to implement the web service due to its simplicity and several experiences in the past web applications. The other advantage for using Python Flask is that it is easy to deploy in the free server like Pythonanywhere. As a result, the web service is implemented and deployed in http://llkserfinal.pythonanywhere.com/.

3.2.2 Communication between Mobile Application and Web Service

The only web service is provided for mobile application to access. The web service is bound to the url '/metalDetection'. It receives Post HTTP request from mobile application and gets data from request form. The encoded image is then decoded by base64 module in Python and stored in a static image file in the server. After image processing algorithm, the result is packaged into Json format and sent to the mobile application finally.

```python
@app.route('/metalDetection', methods=['POST'])
def test():
    img = base64.b64decode(str(request.form['image']))
    metalCode = request.form['metalCode']
    image_data = np.fromstring(img, np.uint8)
    image_data = cv2.imdecode(image_data, cv2.IMREAD_COLOR)
    cv2.imwrite('sample.jpg', image_data)
    result = imageProcessing(metalCode)
    resp={
        "result":result
    }
    return jsonify(resp)
```

Figure 3.4: A screenshot of the code implementation for the web service.

3.3 Image Processing Algorithms

3.3.1 Technologies and Environment Require

The image processing algorithms in this project are implemented in Python. There are several reasons for choosing Python to use. First, Python is one of the most useful and powerful computer scientific tools nowadays [12]. It is easy to use and efficient to design algorithms. Second, Python has several effective module like OpenCV for image processing. And we have also learnt to use Python in the past projects.

The Python version in this project is 3.7.5 which is more stable than the latest version. The algorithms mainly use OpenCV module (v4.2.0) for image I/O and processing, numpy module (v1.17.4) for numerical calculation, and matplotlib module (v3.1.1) for plotting.

3.3.2 Data Analysis

As is shown in Figure 4.5, the image data usually have four features or regions.

1. The first feature is the most outside region. It usually appears in a single color and is same in all the data of a certain metal. This outside region shows the color of the specific reagent. Thurs, this feature doesn't have any information of the metal concentration which should be ignored in image processing.

2. The second feature is the inner region. It usually appears in a circular shape in the center of the image. The color in this region is not pure and the intensity deepens with the growth of metal's concentration. This inner region shows the color after the chemical reactions between the specific reagent and the metal. Thurs, this feature is the core of the image processing algorithms also called region of interest(ROI) in this project.

3. The third feature is some different color appearing in the inner region. This feature usually has similar color with the inner region. It may be caused by the experimental error or uneven distribution of metal. As the feature only appears in some data, the effect to the result can be ignored.

4. The fourth feature is some noise in the image. It can be some lighter color around the outside region or some dark spots in the background. These image noises should be removed to reduce its effect to the result.

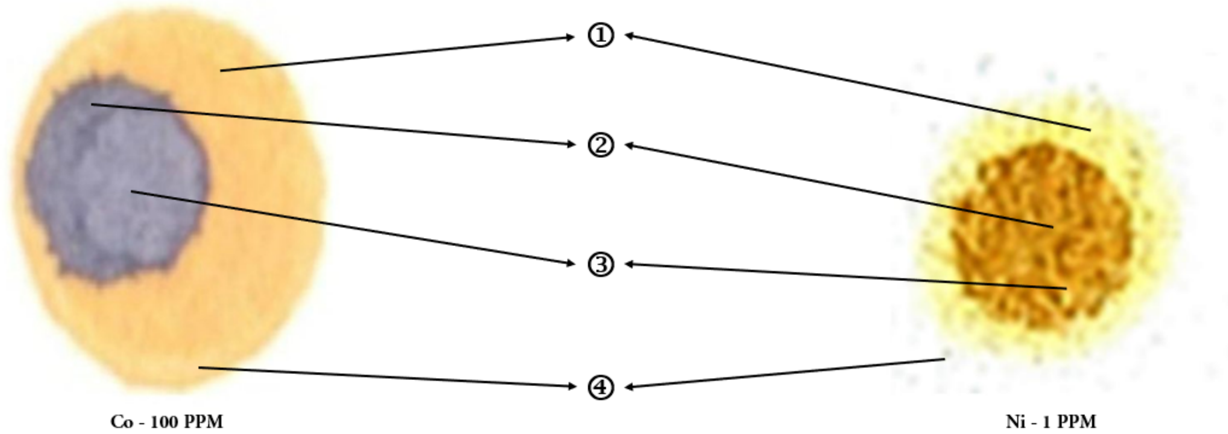


Figure 3.5: An example image of cobalt(Co) in 100PPM and nickel(Ni) in 1PPM showing four features in the data.

3.3.3 Image Histogram Matching

This algorithm is based on image histogram. An image histogram is usually represented by a vector to calculate the frequency of pixels with a same color value. Thurs, the image histogram shows the distribution of the color in the image. As the concentration of metal is related to the color of the image, we can compare two histograms and obtain their similarity to get the result.

Steps of Algorithm

The algorithm takes an image as the input, and outputs a predicted range of concentration of chosen metal. It follows below steps:

1. Input image I;
2. Change I to gray scale image G(I);
3. Calculate the histogram in gray scale: H(I) stored in a 256-dimensional vector;
4. Calculate the similarity between input vector and each vector in data set;
5. Choose the highest similarity and get the corresponding concentration range;
6. Return the result.

Building Image Histogram

As is shown in Figure 3.6, the original images first be changed to grey scale image by function in OpenCV module. The grey scale value ranges from 0 to 255. So, the application calculates the frequency of pixels with same scale value. The output is a 256-dimension vector which can be represented by a histogram.

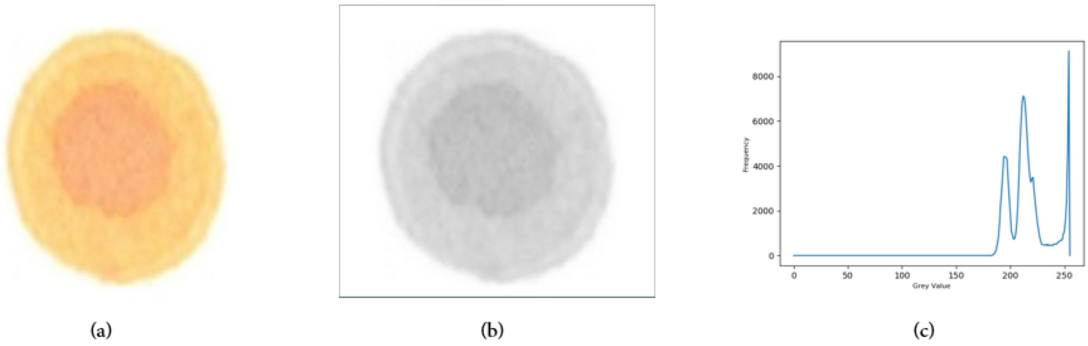


Figure 3.6: An example histogram building process from (a) the original image of cobalt(Co) in 0.1PPM to (c) the histogram of the image, while (b) is the Gray Scale Image.

Calculating Similarity using Cosine Similarity

After getting the histogram of the image, the next step is to find the most similar image in the data set. Cosine similarity is a method to measure the similarity between two vectors [14]. If we treat the histogram as a 256-dimension vector, the similarity between two histogram can be calculated by formula (3.1). And among all the images in the data set, the corresponding concentration of the most similar image in the data set can shows the range of objective concentration which is the result of this algorithm.

$$Similarity = \cos(\theta) = \frac{A \cdot B}{\|A\| \, \|B\|} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \times \sqrt{\sum_{i=1}^{n} B_i^2}} \tag{3.1}$$

3.3.4 K-Means & Linear Regression

This algorithm is inspired by the past projects discussed in Chapter 2.10. It has been mentioned that the intensity of color in the inner region of the image has linear correlation with the concentration of metal. Thurs, we use K-means clustering algorithm to classify the color in the image and find the region of interest. Then, we use linear regression to find the linear correlation equations between the intensity and concentration. Finally, we can get the intensity of ROI in the test image through same process and calculate the concentration of metal using the linear model obtained.

Steps of Algorithm

The algorithm takes an image as the input, and outputs a predicted value of concentration of chosen metal. It follows below steps:

1. Input image I;
2. Change I to a point set with its color distance: P(I);
3. K-Means cluster to find the region of interest(ROI);
4. Calculate the average RGB value in ROI and get its color intensity;

(0). (Pre-computing) Linear regression to get the linear model between color intensity and metals' concentration in training data set;

5. Calculate the metals' concentration by applying linear model to the color intensity in ROI;
6. Return the result.

Image Pre-processing

The image pre-processing refers to the second step described above. A pixel in the computer image has two kinds of information: 1. x, y position; 2. three values ranges from 0-255 representing the intensity of three primary color red, green and blue. In these information, we focus on the color information in the image. To perform K-Means algorithm, we need to reduce the dimension of RGB color space from three values to one value representing color difference. So, according to (2.2), we change the input image to the point set in the form (3.2), the color distance is the square to avoid the error of floating numbers.

$$[xPosition, yPosition, 3\Delta R^2 + 4\Delta G^2 + 2\Delta B^2] \tag{3.2}$$

K-Means Cluster to Find ROI

The K-Means Cluster algorithm is implemented by Python and an example result is shown in Figure 3.7. The choices of K for each metal have been tested and the best results are: K=4 for Cobalt(Co), and K=3 for hydrargyrum(Hg) and nickel(Ni). The region of interest need to be selected from K sets divided by K-means. As the ROI is usually in the inner region of the image, the algorithm choose the most central set to be the ROI. The results for ROI in the image data set are shown in the Appendix C.

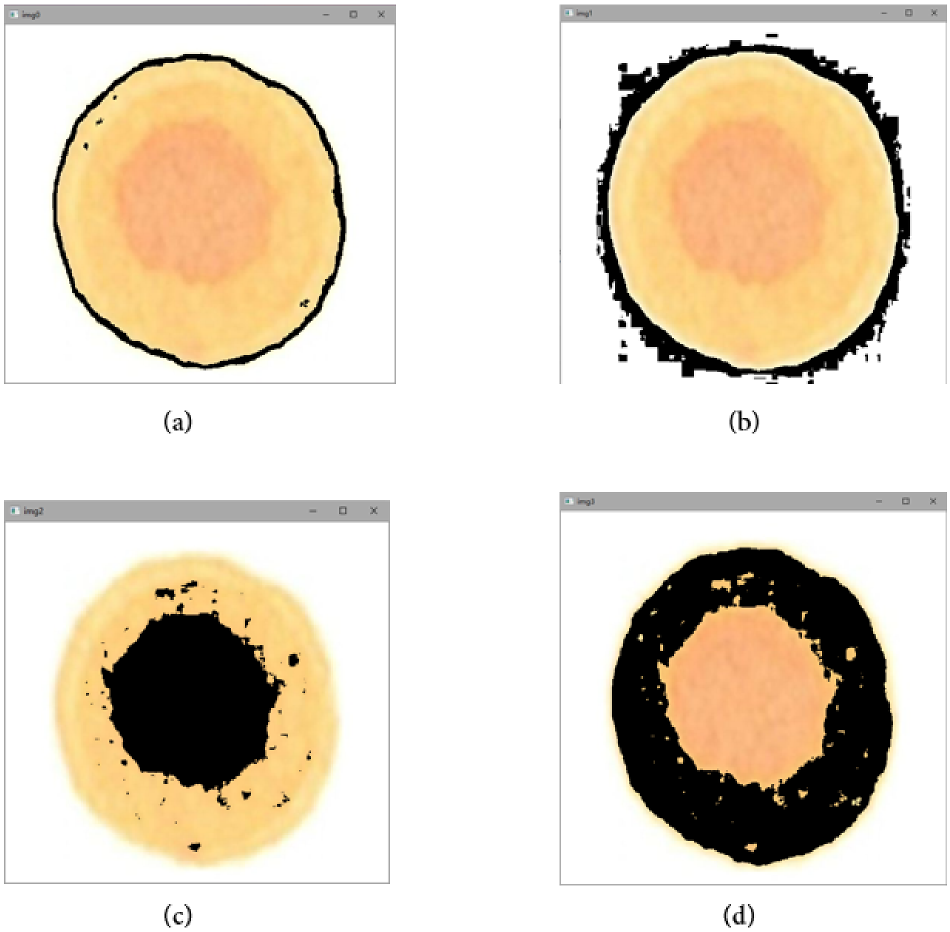


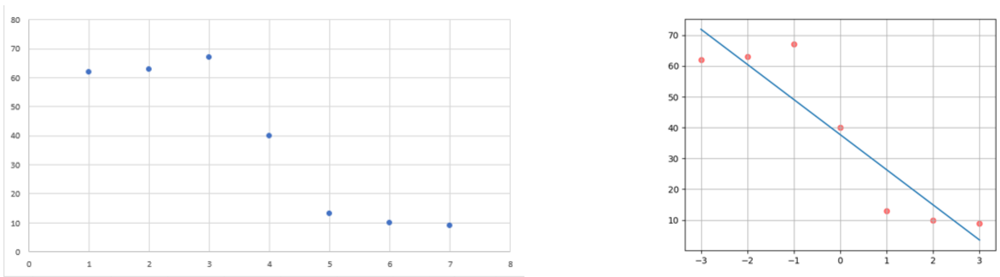
Figure 3.7: An example result of K-Means Cluster applied on the image of cobalt(Co) in 0.1PPM. The black parts in (a), (b), (c), (d) show four point sets divided by the algorithm. (c) is the region of interest in this image.

Linear Regression

After obtaining the results of all the images in the data set, the algorithm calculates the linear model using formulas (2.5) (2.6). As the concentrations of metal are the powers of ten, we use $y' = \log_{10} y$ as the objective parameter. So, the final linear model is:

$$\log_{10} y = wx_i + b \tag{3.3}$$

where y is the objective concentration of metal and x is the intensity of color in ROI. Figure 3.8 shows an example result of linear regression applied to the cobalt(Co) data set.



(a) (b)

Figure 3.8: An example result by applying linear regression in the cobalt(Co) data set. (a) shows the intensity of each image, and (b) gives the result of linear model.

Chapter 4

Software Testing and Evaluation

4.1 Introduction

The software testing is one of the necessary steps in software development as it is an important material to judge whether the implementation of the software meets its requirements. Due to difficulties in obtaining available experimental results, the project suffers a lack of data and insufficient test. To solve the problem, we choose to build test data from original data, which will be introduced in the followings. However, as the data for this project is generated in chemical experiment, those artificial building data couldn't satisfy the requirement for algorithms' tests because these data would generate wrong model and couldn't predict true results in real condition. As a result, the building test data is only used for the test for mobile application and web service.

The second part in this chapter is the comparison and evaluation for two algorithms. The two algorithms are compared and evaluated for their strengths and weaknesses.

4.2 Building Test Data

Due to a lack of data, we decide to build test data from original data. As is shown in Figure 4.1, a test data can be built by rotating and adding spots in an existing color to the original image. It is mentioned in the Chapter 3.3.2 that the features in the image are around the center of image and usually in circular. So, rotating the image for any angular would not affect the availability of data. Similarly, adding a little existing color or noise to the image would not change the features in the image greatly.

Figure 4.1: An example test data building from original image. (a) is the original data while (b) is the test data.

4.3 Test Results for Mobile Application and Web Service

As the web service is only provided for mobile application, the tests for the mobile application and web service are carried out together. As is shown in Figure 4.2, we have applied several operation tests to check the response of the application to different user inputs.

Test Operations	Expected Results	Actual Results
Choose image from album	The system album was opened for user to choose image. And after user's choice, the application displayed a preview in main page.	As expected
Take photo by camera	The camera was called by the application. And after user toke a photo, the photo was displayed in the preview.	As expected
Choose the metal to detect	Three checkboxes can be checked correctly, and two checkboxes can't be checked in a same time.	As expected
Try to upload data without any choice made	The application refused user's request and displayed an error information.	As expected
Try to upload data without image	The application refused user's request and displayed an error information.	As expected
Try to upload data without metal choice	The application refused user's request and displayed an error information.	As expected
Send test data with correct metal choice	The application displayed correct result and the test image was stored in a static file in server.	As expected
Send test data with wrong metal choice	The application displayed a result and show a warning for wrong choice.	Wrong result showed in the result page and no warnings.
Send a random image with random metal choice	The application refused user's request for inavilable input image..	A random result showed in the result page and no warnings.

Figure 4.2: A checklist to show the test results.

As a result, the application responses correctly to available user inputs or empty inputs and shows some information. But the application fails to judge the validity of the input data, because it treats any input from user as available data.

4.4 Comparison and Evaluation for Two Algorithms

4.4.1 Comparison

The comparison for two algorithms is mainly in five aspects as is shown in Figure 4.3.

Criterion	Image Histogram Matching	K-Means & Linear Regression
Input/Output	Input: an image Output: a predicted **range** of metal concentration	Input: an image Output: a predicted **value** of metal concentration
Time complexity	The time costs mainly in two processes: 1. O(n) for calculating the histogram of image, where n is the number of pixels; 2. O(m) for calculating the similarity, where m is the number of images in data set; So, total time complexity is O(n+m).	As the linear model is obtained previously, the time costs mainly in the K-Means to find ROI; So, total time complexity is O(nm), where m is the number of iterations; and n is the number of pixels.
Space complexity	External spaces required to store the histograms of images.	There are no external spaces required because the linear model is an equation with two parameters.
Main fator to affect the accuracy of the result	The **scale** of the data set.	The **quality** of the data set.
Tolerance to the noise data	High	Low

Figure 4.3: A table to show the comparison results.

4.4.2 Evaluation for Image Histogram Matching

Pros.

1. The algorithm is easy to understand and implement.
2. The accuracy of the algorithm can be easily improved by expanding the data set.
3. The tolerance to the noise is high because the algorithm only focus on similar image in the data set.

Cons.

1. The algorithm only provides a range or similar values of predicted concentration, which means that the algorithm can't give a certain value for prediction.
2. When there was no similar image in the data set, the algorithm would fail.
3. With the growth of the data set, the time cost would increase because the test image should be compared to each image in the data set.

Conclusion

This algorithm is preferred when the data set is large with some noises and the required precision is low.

4.4.3 Evaluation for K-Means & Linear Regression

Pros.

1. The algorithm outputs a certain concentration result with high precision.
2. The time complexity and resources cost are very low in the processing due to pre-computing of the linear model.

Cons.

1. The quality of the data have a great effect on the accuracy of the algorithm. Thurs, the algorithm can't be applied to a data set with a lot of noises.
2. If some data doesn't follow the linear correlation, the algorithm would fail in some ranges of parameters.

Conclusion

This algorithm is preferred when the data set is limited but the quality of the data is very high and the precision requirement is high.

Chapter 5

Conclusion

5.1 Project Objectives and Outcomes

There are four objectives described in Chapter 1.3. The outcomes of each objective will be discussed in the followings.

1. **Develop a base mobile application that allows users to upload data and get results.**

   As a result, the mobile application has been designed and implemented in Android platform using Android Studio. Although the functions in the android application are kept to a minimum due to the nature of this project, the application still demonstrates the feasibility of the algorithms and project. The test for the mobile application is carried out successfully and the demonstration video can be accessed in Appendix B.

2. **Develop a web service that receives users' requests and sends the results back.**

   This objective has been achieved by implementing a web service using Python Flask framework and deployed in Pythonanywhere. As the web service is only used for mobile application to access the image processing algorithms, there is no requirement for further development in the administration or database. And the test for web API is carried out with the test in mobile application.

3. **Design and evaluate image processing algorithms that detect the concentration of metals in water.**

   This objective describes the core outcomes in this project. Overall, the project has investigated, designed and implemented two image processing algorithms to detect metal in water. One is image histogram matching and the other is K-means & linear regression. The two algorithms have been fully analysed and compared for evaluation. The only problem is that lack of data causes insufficient tests and difficulties in improving precision.

4. **Reflect on limitations and possible future improvements to the mobile application, web service and image processing algorithms.**

   The reflections and future improvements will be discussed in the following sections.

5.2 Personal Reflection

This project is an impressive experience for me to learn to design and implement image processing algorithms. As the data in the project is simple to handle and features in images are obvious, I had miscalculated the scale of the problem and the difficulties to find a solution. Although I have implemented two algorithms in the project, the solutions could be better and more convinced.

This project is also a good chance to learn to develop mobile application and web server (CS model). As I have developed several client-server model in the past modules or projects, it is easy for me to implement such a demonstration application. However, due to the time limitation and the nature of the project, I failed to implement more functions in the mobile application.

The most serious problems in this project occur in two aspects. One is bad time and project management, the other is lack of data. The first one is caused by miscalculation and no software methodology applied. This teaches me a lesson to start planing earlier and the project methodology is vital in the software development. The second problem is caused by lack of experimental conditions and unfamiliarity with the chemical approaches. The problem is quite hard to solve and it could be only expected to have more communication and cooperation with professionals in the future projects.

The last meaningful experience in this project is to use LaTeX to write the report. LaTeX is really a powerful tool for typesetting, formula expression and charting. I'm going to use it more in the future paper workings.

5.3 Future Improvements to System Functionality and Architecture

It is mentioned in the previous chapters that the mobile application and server are only developed to a minimum level. If the system are going to be developed to a production, there are many functionalities to be added and improvements should be made in the architecture.

First, the mobile application can implement the login / register system. The login system allows the application to identify a user. It helps to refuse the malicious access to the application service and customized user interface can be implemented next.

Second, the web service can be improved by adding administrator's entry. In the previous design, when the manager of the website wants to change the algorithm, he need to modify the source code in the web server. And an administrator's entry can allow the administrator of the website to change the service easily.

At last, the architecture can be improved by adding database to the server. A database in the server can store the data securely, and allows the website to collect users' data to provide better service.

5.4 Future Improvements to Image Processing Algorithms

For the image histogram matching algorithm, the best method to improve its accuracy and precision is to expand the data set. This algorithm allows a little noise in the data set, while the scale of the data set is vital.

For the K-means & linear regression algorithm, it can be improved by changing the linear model higher level one like multivariate linear model. A more complicated model can usually provide a better accuracy.

5.5 Legal, Ethical, Social and Professional Issues

There are actually no particular legal, ethical or social issues to be addressed in the project. The data involved in the project are experimental results and no any privacy or secure data exists. And the mobile application actually doesn't obtain or store any users' data or operations, while the server is the same. As the mobile application in the project is provided for workers or simulators in the factory, the only professional issue is that an expert is needed to maintain and update the algorithms in the server.

References

[1] S. Arsawiset and S. Teepoo. Ready-to-use, functionalized paper test strip used with a smartphone for the simultaneous on-site detection of free chlorine, hydrogen sulfide and formaldehyde in wastewater. *Analytica Chimica Acta*, 1118, 2020.

[2] A. Bansal, M. Sharma, and S. Goel. Improved k-mean clustering algorithm for prediction analysis using classification technique in data mining. *International Journal of Computer Applications*, 157(6):35–40, 2017.

[3] M. Bhuiyan, S. B. Dampare, M. A. Islam, and S. Suzuki. Source apportionment and pollution evaluation of heavy metals in water and sediments of buriganga river, bangladesh, using multivariate analysis and pollution evaluation indices. *Environmental monitoring and assessment*, 187(1):4075, 2015.

[4] T. M. Chan and J. Zhang. *An Improved Super-Resolution with Manifold Learning and Histogram Matching.* Advances in Biometrics, 2005.

[5] P. J. Joseph, K. Vaswani, and M. J. Thazhuthaveetil. Construction and use of linear regression models for processor performance analysis. In *Twelfth International Symposium on High-performance Computer Architecture*, 2006.

[6] Y. Lin, D. Gritsenko, S. Feng, Y. C. Teh, X. Lu, and J. Xu. Detection of heavy metal by paper-based microfluidics. *Biosensors and Bioelectronics*, 83:256–266, 2016.

[7] Y. Lu, S. Song, R. Wang, Z. Liu, J. Meng, A. J. Sweetman, A. Jenkins, R. C. Ferrier, H. Li, W. Luo, and T. Wang. Impacts of soil and water pollution on food safety and health risks in china. *Environment International*, 77:5–15, 2015.

[8] A. L. Marzo, J. Pons, D. A. Blake, and A. Merko?I. All-integrated and highly sensitive paper based device with sample treatment platform for cd2+ immunodetection in drinking/tap waters. *Analytical Chemistry*, 85(7):3532–3538, 2013.

[9] W. S. Mokrzycki and M. Tatol. Color difference delta e - a survey. *Machine Graphics and Vision*, 20(4):383–411, 2011.

[10] S. Muhammad-Aree and S. Teepoo. On-site detection of heavy metals in wastewater using a single paper strip integrated with a smartphone. *Analytical and Bioanalytical Chemistry*, 412(6):1395–1405, 2020.

[11] P. Sebastian, V. V. Yap, and R. Comley. The effect of colour space on tracking robustness. In *IEEE Conference on Industrial Electronics and Applications*, 2008.

[12] A. G. Silva, R. Lotufo, R. C. Machado, and A. Saúde. Toolbox of image processing using the python language. In *International Conference on Image Processing*, 2003.

[13] F. Su, G. Fang, and J. Zou. A novel colour model for colour detection. *Journal of Modern Optics*, pages 1–11, 2016.

[14] J. Ye. Cosine similarity measures for intuitionistic fuzzy sets and their applications. *Mathematical & Computer Modelling*, 53(1-2):91–97, 2011.

Appendices

Appendix A

Code Repository

The code for this project can be accessed from: https://github.com/llkser/FinalProject.

Appendix B

Video Demonstration

The video demonstration for mobile application can be accessed from:
https://github.com/llkser/FinalProject/blob/master/Report/videoDemonstration.mp4

Appendix C

Results of ROI of the Images in Data Set

K=4	Total Pixels (Not white)	Pixels in ROI	R	G	B	Intensity
Co1						
Co2	179737	43183	246	184	123	62
Co3	164334	45270	250	187	124	63
Co4	162487	44359	250	183	122	67
Co5	157525	41107	186	146	124	40
Co6	182375	34365	153	140	155	13
Co7	169032	45536	157	147	158	10
Co8	182297	61057	141	132	144	9
K=3	Total Pixels (Not white)	Pixels in ROI	R	G	B	Intensity
Hg1						
Hg2	174740	43484	251	227	186	24
Hg3	164721	35720	251	222	181	29
Hg4	166465	43875	251	219	180	32
Hg5	167098	39529	251	220	184	31
Hg6	159855	28971	250	208	175	42
Hg7	187535	34348	243	194	171	49
Hg8	177961	65223	234	184	162	50
K=3	Total Pixels (Not white)	Pixels in ROI	R	G	B	Intensity
Nickle1						
Nickle2	95321	31056	245	242	95	3
Nickle3	93511	30988	243	233	81	10
Nickle4	104644	31669	240	210	57	30
Nickle5	118363	39034	209	147	27	62
Nickle6	108033	30127	193	120	16	73
Nickle7	108027	27403	170	107	11	63