**Found & Unlocked: Key Matching & Identification Using an Android Mobile App**

*Senior Project*

*In Partial Fulfillment of Bachelor of Science in Computer Science*

*Yessenia Rodriguez*

*Supervisor: Dr. Mahmoud K. Quweider*

*December 10, 2018*

**Table of Contents**

1. Deliverables, Timeline and Status for the Project
2. Abstract
3. Android Mobile App
4. Capture an Image from Camera
5. Extract Features
6. Store Data of User’s Key
7. Compare Keys
8. Appendix
9. References

**I.** **Deliverables, Timeline and Status for the Project**

|  |  |  |
| --- | --- | --- |
| **#** | **Deliverable** | **Expected Date of Completion** |
| 1 | Learn how to use Android Studio | Sep 16 |
| 2 | Design an Android Mobile app | Sep 23 |
| 3 | Design an Android Mobile app (…continued) | Sep 30 |
| 4 | Build app | Oct 7 |
| 5 | Be able to get image from Camera | Oct 14 |
| 6 | Extract features from key image | Oct 21 |
| 7 | Store data of users’ keys | Oct 28 |
| 8 | Compare key image to stored keys | Nov 4 |
| 9 | Display results and options after comparison | Nov 11 |
| 10 | Display results and options (…continued) | Nov 18 |
| 11 | Testing and bug | Nov 25 |
| 12 | Testing and bug (… continued) | Dec 2 |
| 13 | Demo & presentation | Dec 10 |

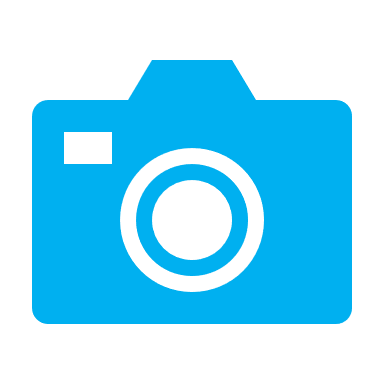
**II. Abstract**

According to an esure insurance poll, the average person has nine keys in his or her keyring [10]. However, from those nine keys, the individual does not know what at least three keys are for [10]. Based on this poll, many could benefit from having additional help keeping track of one’s keys. Thus, this project aimed to provide another method of storing key information using images. This project is an Android mobile application that is able to identify and match keys. The user uses the camera of the Android phone or device to capture an image of a key, obtaining the key’s features. These features are compared to the features of keys previously stored in a database by the user. If the key matches another, the application displays the matching key’s image and name. If not, the option of adding the key to the database of keys is provided. Otherwise, the user can try again to match the key. Moreover, the keys in the database can be viewed – displaying the keys’ names and images. Regarding the storage of keys, the new key’s name, image and features are stored in the database to either be viewed by the user or used to compare to other keys.

Deliverables were in the form of a mobile application. The application was developed using Android Studio with Java as the primary programming language. Additionally, SQLite was used to create a local database.



The figure below shows different components involved in the project.



Information of Image Stored:

1. Key ID
2. Key Name
3. Key Description
4. Key Image
5. Key Features

Store Information

Retrieve/Display Information

**III. Android Mobile App**

In this project, an Android Mobile App was designed and built. To accomplish this, the Android Studio was used. This Integrated Development Environment (IDE) provides many tools and features for building an Android app, including emulators and environment that allows the development of an Android app for all Android devices. XML files were used for designing the app, and Java was used to build the app and its operations. [9]

**A. Designing Mobile App**

Within Android Studio, a folder titled res can be found. This folder includes all the non-code resources. Among them are XML files used to define the layout, color, and string values. Additionally, images used in the app are also found in this folder. Using these files and adding XML layouts, the design of the app was created. To help support these layouts, additional XML files like color.xml and string.xml were used. The XML layouts are listed below, and screenshots of some of these layouts can be found in Appendix A.

* activity\_main.xml
* activity\_camera\_ocv.xml
* activity\_extract.xml
* activity\_search.xml
* activity\_keys.xml
* activity\_keyinfo.xml
* edit\_key.xml
* activity\_nomatch.xml
* activity\_match.xml
* activity\_match2.xml
* activity\_add.xml

**B. Building Mobile App**

Each XML layouts listed above has a Java file. These files organize the XML layouts by determining when and how the layouts are displayed. For instance, activity\_main.xml has two buttons. If one button is selected, a different xml file would be displayed. The change in xml files are done by beginning a new activity with a different ContentView, different XML file. Thus, there is a Java file for every activity in the application. Each Java file has the operation onCreate that create the activity’s view in the app. Additionally, additional classes were needed to build the mobile app. A class for the keys was created called KeyInformation, and a class to adapt the key viewer, the list of the keys’ stored. These two classes were used to organize and update the key’s information being displayed or inserted by the user. [13]

**IV. Capture an Image from Camera**

“OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library” [1]. This means a “live stream” of the camera can be offered by OpenCV, obtaining images an application can work with. First, an XML file was used to design the layout in which the open camera could be displayed. Then, a Java file that corresponded with this XML file was created to control the camera and handle the operations within this “activity.” When it comes to the camera, the camera on the Android mobile device is used – in essence, the Camera application. To connect to the camera services, special permission was granted to the application by using the file Manifest.xml.

**A. Designing Open Camera**

Within Android Studio, the folder res contains the XML file activity\_camera\_ocv.xml. This file was used to design the *camera* that the user uses to capture the image of their key. It is in this file that the open camera, a JavaCameraView, and a button to capture images are displayed. The JavaCameraView is a placeholder, accessed by the OpenCV library to provide the “live stream” open camera.

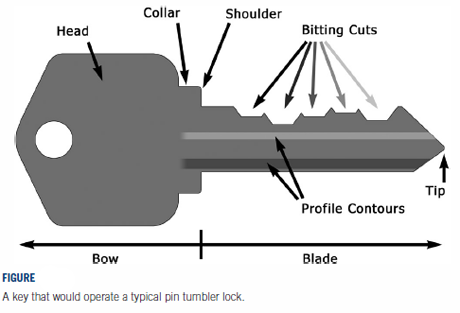
**B. Building Open Camera**

The java file used to control the operations and display the XML layout file discussed above is the java file called CameraPreview.java. It was placed in a different package to differentiate from the rest of the activities. Within this file, the OpenCV libraries are used to display the open camera that is “live streaming.” This open camera will return a mat to the java file – allowing it to obtain the image which will be used to compare the key. In order to obtain this image, the button in the XML layout is used. The button’s actions are controlled in this java file. This button captures an image from the “live stream” camera by storing the current mat. This mat is then pre-processed. These pre-processing steps are done to obtain the contour, or original shape, of the key, isolating the key from the rest of the image. This contour is used to obtain the features. These steps are outlined and visually represented in Appendix B. To facilitate these steps and get the most accurate contour of the key, a slider is utilized in the binarization step. This is shown in the Appendix A, figure 5. Furthermore, the operations to perform these steps are in the java file ImageProcessor.java, also found in the package for the camera. This java file utilized operations from the OpenCV libraries to complete the steps. The code for CameraPreview.java was obtained from the video *OpenCV in Android Studio Part 3 (Camera Preview with Button),* modified to capture an image.[3]

**V. Extract Features**

Every image has certain features, or properties, that can identify the image and distinguish it from the other images. This idea is utilized to compare keys, accomplishing this by isolating the key from the rest of the image. This isolation is accomplished by having the image undergo the pre-processing steps, which were previously mentioned, obtaining the key’s contour. This contour is the basic shape of the key. In this project, shape is very important because within the shape of a key the true *key* is found. The main part of any key is the blade. This will go into the lock, and if the bitting cuts matches with the lock mechanism, the key will work and unlock the lock. Thus, considering the importance of a key’s shape, a shape-oriented feature detection was utilized.

Provided below are a few images that label the parts of a key to help visualize what was discussed above.



**A. Fourier Descriptors**

The boundary, our contour, of our two-dimensional image is obtained using OpenCV. The points that make up the boundary undergo the Fourier Transform to obtain complex number, (x – iy). Before the transformation, the boundary is sampled to a fixed number of points. To make the process faster and more efficient, the Fast Fourier Transform was used, which is a derivation of the Discrete Fourier Transform. This transformation will be easier if the sampled points are a power-of-two integer. This was kept in mind when considering the number of features that are obtained. [19]

The formula for Discrete Fourier Transform is provided below for reference. [19]

A centroid of the image is also found and transformed into a complex number. This centroid was added to eliminate any effects a transformation or placement change of the key can have on the features. When obtaining these features, a specified number of sample points will determine the accuracy of the matching. The greater the sample points the more accurate the matching will be, for it will have more features. Granted, there is a downside to using to many descriptors – the run time taken to obtain and compare to other feature sets. Furthermore, these complex coordinates were normalized. According to Dr. Dengsheng Zhang and Professor Guojun Lu, “the normalization successfully eliminates the noise and small details of the shape…while successfully [extracting] the outline features from the shape and also [keeping] the key salient points which is important to shape representation” [19], giving us the best results and features to work with. Thus, their modified formula was used, assuming the normalization is to N. Their formula is provided below for reference. [19]

**Finishing with the following features set:**

The FD0 is used for index and normalization. In addition, the FD is used to represent the distances between the complex coordinate number and the centroid. In addition, only have of the of the number of samples are different frequencies in the Fourier Transform. Thus, only have of the samples are needed. This is denoted in the formula by N/2. [19]

**B. Obtaining Features**

A class called Complex, written by Robert Sedgewick and Kevin Wayne, was used to obtain and convert to complex coordinates [12]. For the Fast Fourier Transformation (FFT), the class FFT was derived based on the algorithm in the video *The FFT Algorithm – Simple Step by Step* [18]. Both java files were placed inside the package Descriptors. Additionally, operations to extract the features were added to the class ImageProcessor.java, which was previously mentioned when discussing the OpenCV camera. These operations use the two classes in the package Descriptors to obtain normalized complex features.

**VI. Store Data of User’s Key**

In order to identify or match a key, there needs to be a group of keys that will be used for comparison. When a user takes a picture of a key to be identified, this key is compared to all the keys stored in the database. Thus, the application required a database of stored keys.

**A. Designing Database**

The database was designed based on the needs of the application. First, the data being stored was analyzed. Only data of the keys are stored – thus, the table “Keys” was created with the unique identifier, KeyID, as the primary key. Additionally, the user would have a name and description for the key. Thus, the columns KeyName and KeyDesc were added to the table being created. Finally, the image and features of a key are the main component used to compare keys. Thus, these two items were also stored. This was done by adding the columns KeyImage and KeyFeat. Using this database, the name, description, and image of the keys stored can be viewed. This layout was created in the file activity\_keys.xml, previously mentioned. A screenshot of this layout can be found in Appendix A, figure 3. Furthermore, the overall design of the database table “keys” is visually represented in Appendix C.

**B. Building Database**

The java file MyDBHandler.java was created to handle the database, using a code posted in the DZone website by Ngoc Minh Tran as a basis [14]. It utilizes SQLite to create the database and contains the operations used to modify and access the database. These operations include loading the complete table into an ArrayList. This ArrayList is used to display the list of keys stored by the user in the view activity\_keys.xml, which is handled by the file MyKeysActivity.java. Within this view, the operations to edit a keys information and delete a key were added. Additionally, an operation to add to the database was defined – given as an option if the key is not found. This operation is used by the view activity\_add.xml and its corresponding java file AddActivity.java. A screenshot of this view can be found in Appendix A, figure 9. Thus, the java files MyKeysActivity.java and AddActivity.java both work with the MyDBHandler class to add, edit, or delete a key, modifying the database. Moreover, because the same database needs to be accessed by several activities, the MyDBHandler class follows a singleton design pattern. This allows the same database to be accessed by any activity or operation in the application, ensuring users access the same updated database every time. [14]

**VII. Compare Keys**

Once there are keys to compare the “unknown key,” and the features of this “unknown key” are obtained, we can begin to compare the keys and get possible matches.

**A. Comparison Process**

The comparison, or search, for possible matches, is done in the activity SearchActivity.java. This activity handles the comparison and has a view in case the search takes a long time, which is activity\_search.xml. Within this view, a message is displayed letting the user know the application is searching through the database. For the comparison, the first step is to access the database – getting an ArrayList of KeyInformation objects, the keys stored in the database. The features of all the keys in this ArrayList are compared to the features obtained from the image captured. This comparison is done by using the distance formula, obtaining the distance between the feature sets. The one with the smallest distance is the closest match. Because of the margin of error caused by the variability of the environment captured by the image, the three keys with the three smallest distance are obtained, being stored in an ArrayList in order from smallest to largest distance. This ArrayList is passed to the activity that handles displaying the results.

**B. Displaying Results**

After possible matches were found or the database was found empty, the app displays the results of the comparison to the users – identifying the key for the user if the key is in the database.

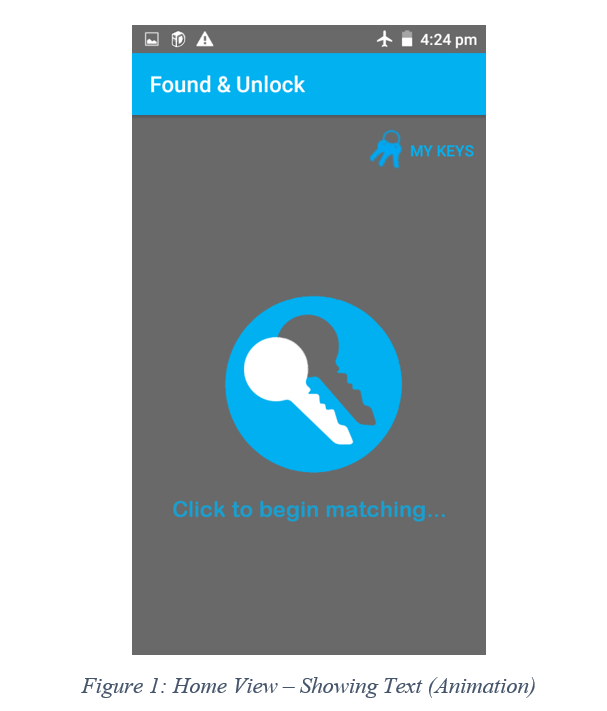
**B. 1. Designing Display of Results**

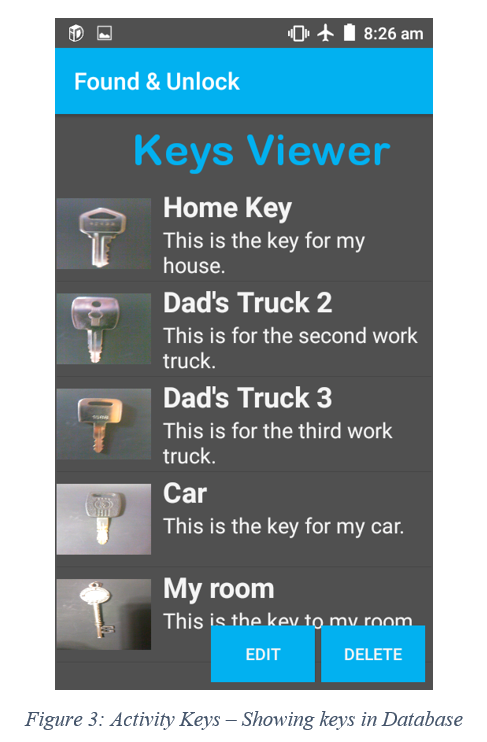
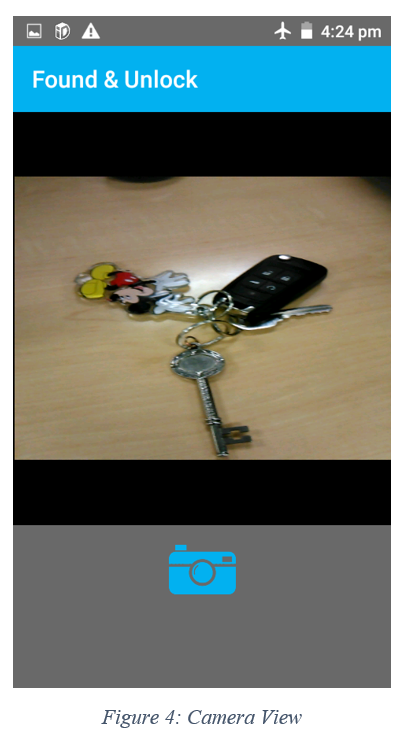
There are two possible results that can occur – possible matches were found, or the database was found empty, in which case no comparisons could be made. Thus, two XML layouts were created to handle the two possible outcomes – activity\_match2.xml and activity\_nomatch.xml. When the database is empty, a message is displayed that informs the user that no possible matches could be found due to no keys being stored. When there are keys stored, the possible matches found are displayed. This is done by accessing the ArrayList of the top three closest matches to the key being identified. If there are three or less keys than all the keys will be displayed as the application displays the top *three* matching keys. Granted, it could be that the key is not in the database. If this is the case, the user can select to add the key to the database – an option available to the user that was previously mentioned. Screenshots displaying the views defined in the two XML files are found in Appendix A, figures 7 and 8.

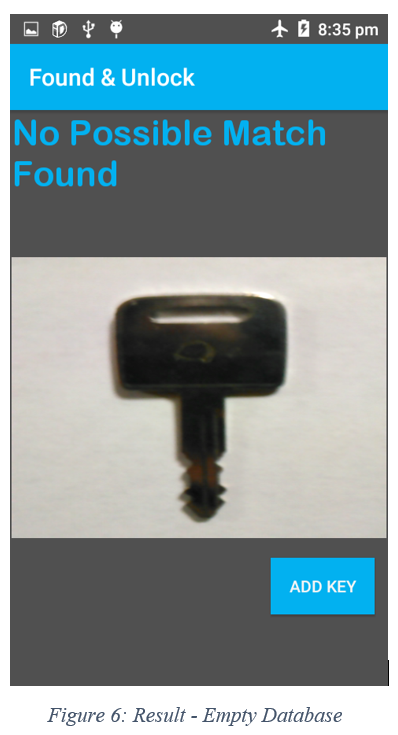
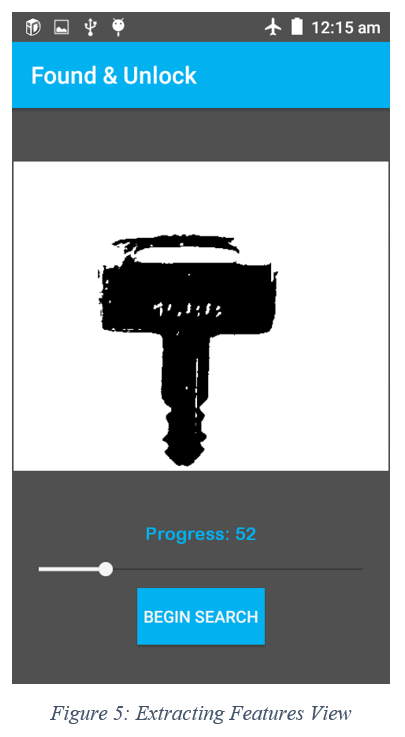
**B. 2. Building Display of Results**

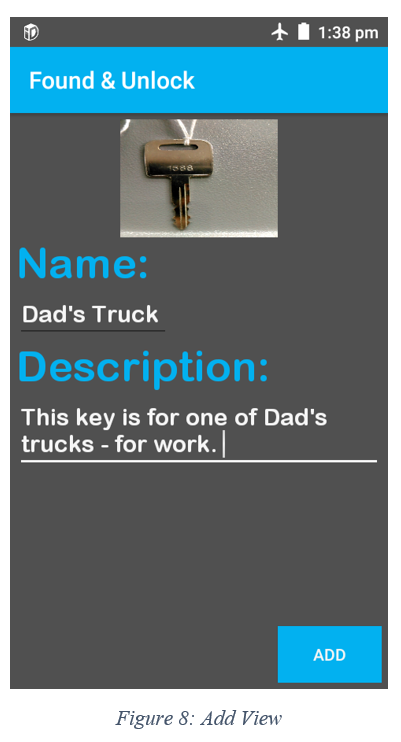
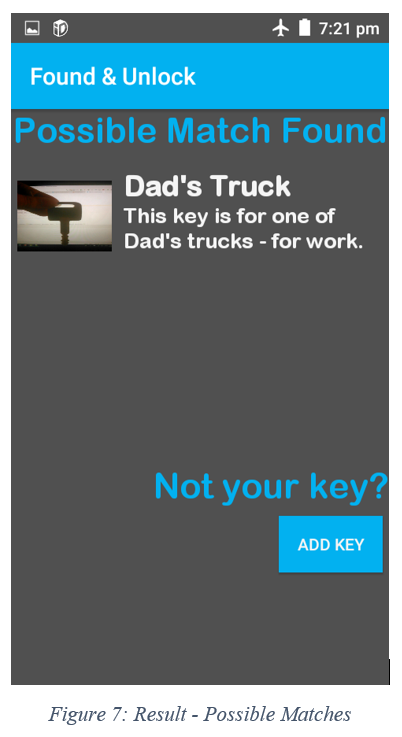
The java files corresponding to the two XML files used to display the results are NoMatchActivity.java and MatchActivity2.java. These two files display the XML files and obtain the necessary information to display. In the case that possible matches were found, the file MatchActivity2.java gets access to the ArrayList with the matching keys. These keys’ name, description, and image are displayed to using the XML file activity\_match2.xml. In the case that the database is empty, the java file sets the image captured to be displayed again when informing the user that no possible matches could be found. In addition, both java files have a button that allows the user to add the key the user wished to identify to the database. This button is handled by opening a new activity, AddActivity.java, that allows users to input a name, and description for the key.

**Appendix A**

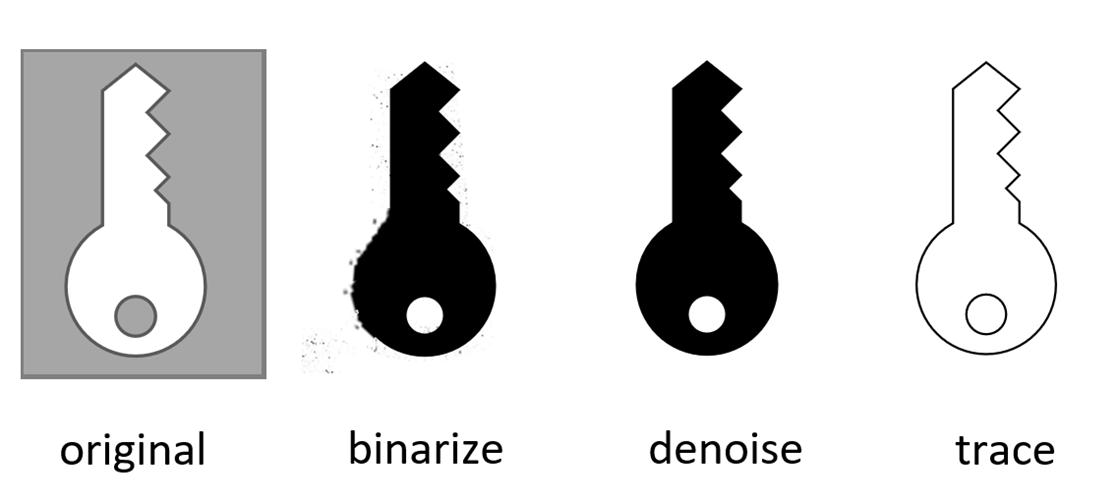






**Appendix B**



**Appendix C**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Key ID** | **Key Name** | **Key Description** | **Key Image** | **Key Features** |
| **1** | **Dad’s Truck** | **This key is for one of the three work trucks.** | **BLOB** | **BLOB** |
| **2** | **…** | **…** | **…** | **…** |
| **…** | **…** | **…** | **…** | **…** |

**References**

1. About. (n.d.). Retrieved from <https://opencv.org/about.html>
2. ArrayAdapter. (n.d.). Retrieved from <https://developer.android.com/reference/android/widget/ArrayAdapter#summary>
3. [Easy Learning]. (2018, April 26) *OpenCV in Android Studio Part 3 (Camera Preview with Button)* [Video File]. Retrieved from <https://www.youtube.com/watch?v=PQeMGRZZRzI>
4. Error Attempt to invoke virtual method 'android.text.Editable android.widget.EditText.getText()' on a null object reference. (n.d.). Retrieved from <https://stackoverflow.com/questions/36701022/error-attempt-to-invoke-virtual-method-android-text-editable-android-widget-edi>
5. Fonts in XML. (n.d.). Retrieved from <https://developer.android.com/guide/topics/ui/look-and-feel/fonts-in-xml>
6. ImageView. (n.d.). Retrieved from <https://developer.android.com/reference/android/widget/ImageView#setImageDrawable(android.graphics.drawable.Drawable)>
7. Implementing a slider (SeekBar) in Android. (n.d.). Retrieved from <https://stackoverflow.com/questions/8629535/implementing-a-slider-seekbar-in-android>
8. Layouts. (n.d.). Retrieved from <https://developer.android.com/guide/topics/ui/declaring-layout#CommonLayouts>
9. Meet Android Studio. (n.d.). Retrieved from <https://developer.android.com/studio/intro/>
10. Mirror.co.uk. (2012, February 03). People have nine keys on their keyring on average, says survey. Retrieved from <https://www.mirror.co.uk/news/uk-news/people-have-nine-keys-on-their-keyring-on-average-115264>
11. Ray-rodriguez. (2018, May 09). Ray-rodriguez/DIP\_Final\_KeyDetection. Retrieved from <https://github.com/ray-rodriguez/DIP_Final_KeyDetection/tree/master/ImageCapture> [previous group project - used some of the code]
12. Sedgewick, R., & Wayne, K. (2017, October 20). Complex.java. Retrieved from <https://introcs.cs.princeton.edu/java/32class/Complex.java.html>
13. Start Another Activity. (n.d.). Retrieved from <https://developer.android.com/training/basics/firstapp/starting-activity#java>
14. Tran, N. M. (2017, December 05). Create a Database Android Application in Android Studio 3.0 - DZone Mobile. Retrieved from <https://dzone.com/articles/create-a-database-android-application-in-android-s>
15. Tutorialspoint.com. (n.d.). Android Animations. Retrieved from <https://www.tutorialspoint.com/android/android_animations.htm>
16. Tutorialspoint.com. (n.d.). Android list Fragment. Retrieved from <https://www.tutorialspoint.com/android/android_list_fragment.htm>
17. Xu, S. (2015, August 3). *Discrete Fourier Transform* [Video File]. Retrieved from <https://www.youtube.com/watch?v=mkGsMWi_j4Q&t>
18. Xu, S. (2015, August 10). *The FFT Algorithm – Simple Step by Step* [Video File]. Retrieved from <https://www.youtube.com/watch?v=htCj9exbGo0>
19. Zhang, D., & Lu, G. (2001). A Comparative Study on Shape Retrieval Using Fourier Descriptors with Different Shape Signatures.

Note: Some of these references were not referred in this report. However, they were included because they were used when writing the code.