Using Smart Glasses for Facial Recognition

Gabriella A. Mayorga, Xuan Do, & Vahid Heydari

Computer Science Department, Rowan University, Glassboro, NJ

https://doi.org/10.33697/ajur.2019.003

Students: mayorgag4@students.rowan.edu, dox0@students.rowan.edu
*Mentor: heydari@rowan.edu**

ABSTRACT
Facial recognition is one of the most promising applications of smart glasses and can help many organizations become more efficient. For example, police traditionally identify criminals by manually going through pictures in a database which makes face matching a slow process. However, with the combination of facial recognition software, smart glasses, and databases, the police can quickly scan through multiple databases of faces to find a match. The police would also be able to spot criminals in crowds, identify unknown victims at crime scenes, retrieve background information on individuals, and verify if someone is a missing person. The Transportation Security Administration (TSA) can also use this combination to identify potential terror suspects or verify the identity of travelers. Lastly, academia can benefit from these tools by being able to identify individuals at events (*e.g.* conferences) and display relevant information about them. The goal of this project is to write an Android program that takes a photo via Google Glass, compares it with a predefined sample database held within the smartphone, and outputs information based on its analysis. The results are displayed with an accuracy acceptance level to the user both on their Android smartphone and on their Google Glass.

KEYWORDS
Face Detection; Facial Recognition; Smart Glasses; Android Smartphone; Mobile Application; Google Glass; Java; SQLite

INTRODUCTION

Facial recognition is the process of capturing an image, analyzing it, and identifying someone based on that still, video, or live image of their face. It is not new to the world of cybersecurity and law enforcement. Facial recognition is becoming one of the most popular surveillance technologies and it is being increasingly used by law enforcement. It has evolved to the point where it can be used in both photos and videos to find a match in a database of faces. A step beyond simple facial recognition would be facial recognition combined with smart glasses technology. Law enforcement can benefit from this combination because it can help them determine if someone is a person of interest even if they are hidden in a crowd. However, there is still a debate about privacy when using this technology. The authors understand the importance of privacy so this research focuses on recognizing only the student authors' images and the purpose of this research is for educational use only.

Real World Examples

Facial Recognition software on its own is available through multiple companies. Amazon developed their own facial recognition Application Programming Interface (API) known as Rekognition that can recognize faces and objects, however, it comes at a cost.[1] Microsoft also offers their own facial recognition API called Face, however, depending on the number of facial recognition transactions you want to perform, you also have to pay a fee.[2] A quick Google search will show more companies marketing their own facial recognition software, but the cost factor associated with using their software may deter people from exploring the world of facial recognition. One algorithm that performs facial recognition and does not come at a cost was created by OpenCV.[3] Their facial recognition code is free for academia.[4] OpenCV is a library that is focused on real-time computer vision programming functions and is mainly written in C++.[5] A counterpart to OpenCV is JavaCV.[6] JavaCV uses JavaCPP Presets which acts as a bridge between OpenCV's C++ code and Java code.[7]

Facial recognition is already being used by law enforcement. According to The New York Times, when a suspect for the 2018 Capital Gazette newsroom shooting refused to identify himself, the police used facial recognition to identify him.[8] The New York Times also noted that facial recognition tools are becoming standard for law enforcement in the United States.[8]

While facial recognition alone is a great tool, being able to use it in combination with smart glasses makes it even more useful to law enforcement. There is already a real-world example of combining facial recognition and smart glasses. LLVision is a Chinese company that has created such a combination[9] and has allowed Chinese law enforcement to identify car license plates and faces based on a database while the police are in the field.[10] Being able to use smart glasses allows the Chinese law enforcement to have a hands-free tool that can quickly and accurately identify someone. As long as the technology is used to protect citizens and not abused, facial recognition can be extremely beneficial.

Related Works

Researchers at University of Oulu, located in Finland, created a program that can perform real-time face detection and recognition while running independently on smart glasses.[11] However, while their smart glasses can perform all the face detection and recognition, their program can only display the face position and identifier and did not include additional information about the person.[11]

The Technical University of Munich, located in Germany, created a cognitive assistant application that also uses real-time facial recognition and runs on smart glasses.[12] Their application uses fog computing which, they stated, is the process of inserting a fog layer between the client and the cloud server in order to use less power and run faster than other real-time applications.[12]

This Study

The Android application created for this project is designed to receive a photo via Google Glass, compare it with a sample database held within the smartphone, and display the results to the user on their Google Glass and Android smartphone. The application also accounts for partial matches by giving the user an acceptance level. Depending on the acceptance level, the user will be informed if there is a match, potential match, no match, or no face detected. The results will be sent to the user's Android smartphone and smart glasses via a notification. **Figure 1** is a simplified visualization of the interaction between the Google Glass and the Android smartphone. The application is called Face2Rec because it stands for Face to Recognition and this paper will give an overview of how it was created and provide insight into its code. This study will also look at the reliability of our chosen facial recognition algorithm.

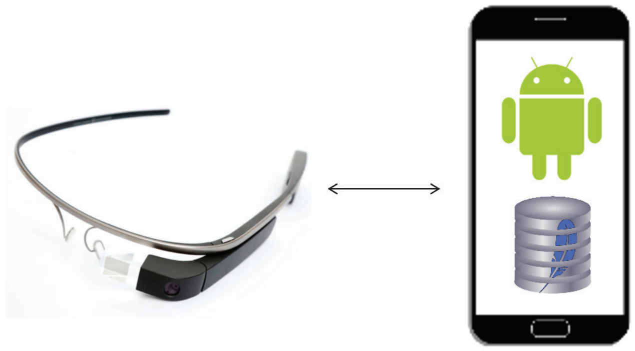


Figure 1. Interaction between smartphone and Google Glass.

Based on our research of all the facial recognition algorithms, APIs, and tools available, JavaCV was selected due to cost factors and its compatibility with Android Studio. The application was written using Java code, so using OpenCV indirectly through JavaCV was advantageous. This is due to OpenCV being written in C++ which means using it in Android Studio, which supports Java programming, adds extra complexities. Also, as undergraduate researchers delving into facial recognition for the first time, OpenCV and specifically JavaCV libraries are very beneficial because they allowed the authors to develop and learn basic concepts of face detection and recognition without worrying too much about creating a face detection and recognition algorithm from scratch. **Figure 2** shows an overview of how the application works.

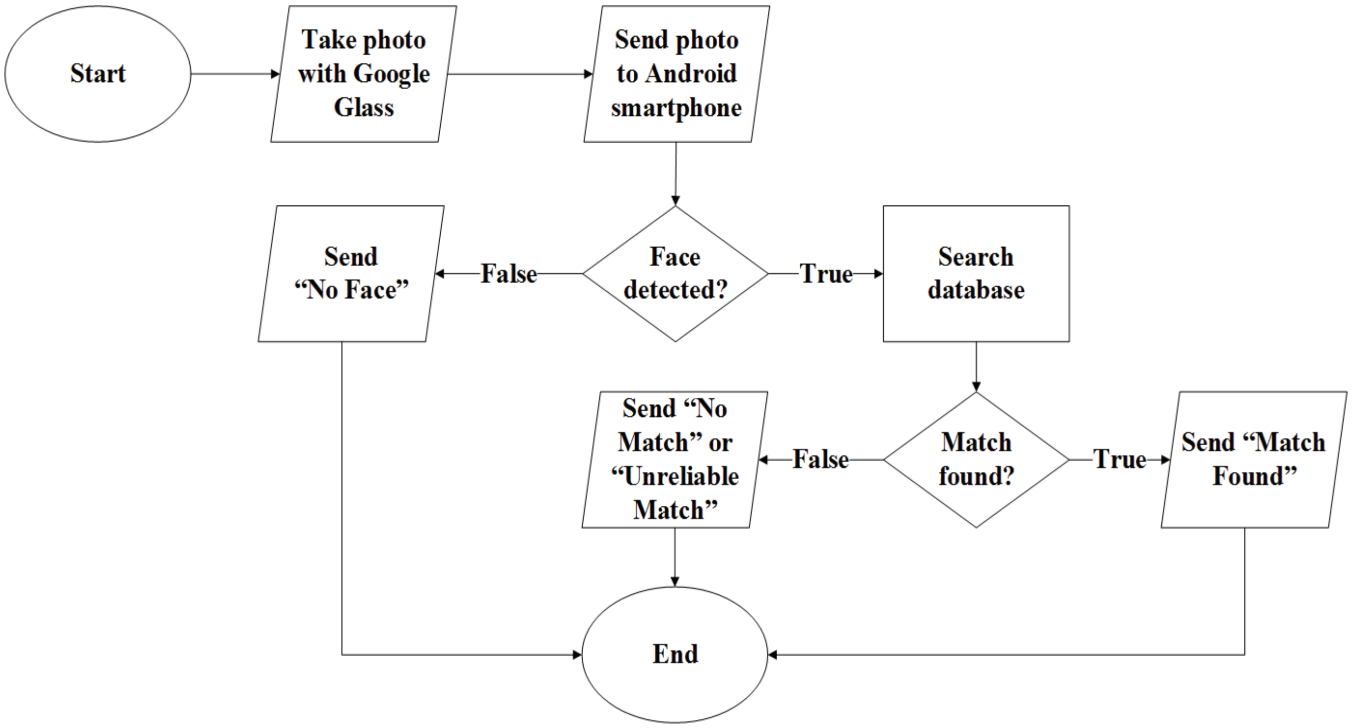

Start
Take photo with Google Glass
Send photo to Android smartphone
Face detected?
False
True
Send "No Face"
Search database
Match found?
Send "No Match" or "Unreliable Match"
Send "Match Found"
End

Figure 2. Overview of how Face2Rec works.

TOOLKIT

Our toolkit comprised of five different components as shown in **Figure 3**. The authors relied on OpenCV and JavaCV for the face detection and facial recognition algorithms. Android Studio is used as the integrated development environment (IDE) to do all the programming in Java. SQLite, the database management system, is used to hold all the match information. With regards to hardware, a Samsung Galaxy S9 is used as the smartphone and Google Glass is selected as the smart glass.

Figure 3. The tools that were used.

FACE DETECTION AND RECOGNITION IMPLEMENTATION

Face Detection

The image file type used for this project is called a bitmap. However, the raw image that is saved in the smartphone may not be in a bitmap format so it must be converted. Once a bitmap image is available on the smartphone, it must be converted to Mat. A Mat is a class that can contain a color or grayscale image that is broken down into an array of numbers.[13] OpenCV must first take the color image and turn it into a color Mat. After OpenCV has the color Mat, it can take it and finally convert it to the grayscale Mat.[14] A grayscale Mat is needed for the face detection to properly work. Now that Face2Rec has a grayscale Mat of an image, it can use OpenCV's CascadeClassifier class to detect objects within the selected data.[15] The function loadClassifierCascade loads the frontalface.xml classifier file. In order for Face2Rec to detect faces in an image, the Haar Feature-based Cascade Classifiers is used.[16] Once Face2Rec finds all the faces in an image, a green rectangle is placed around the detected faces. **Figure 4** shows a flowchart of the face detection algorithm implementation and how the results display on the screen.

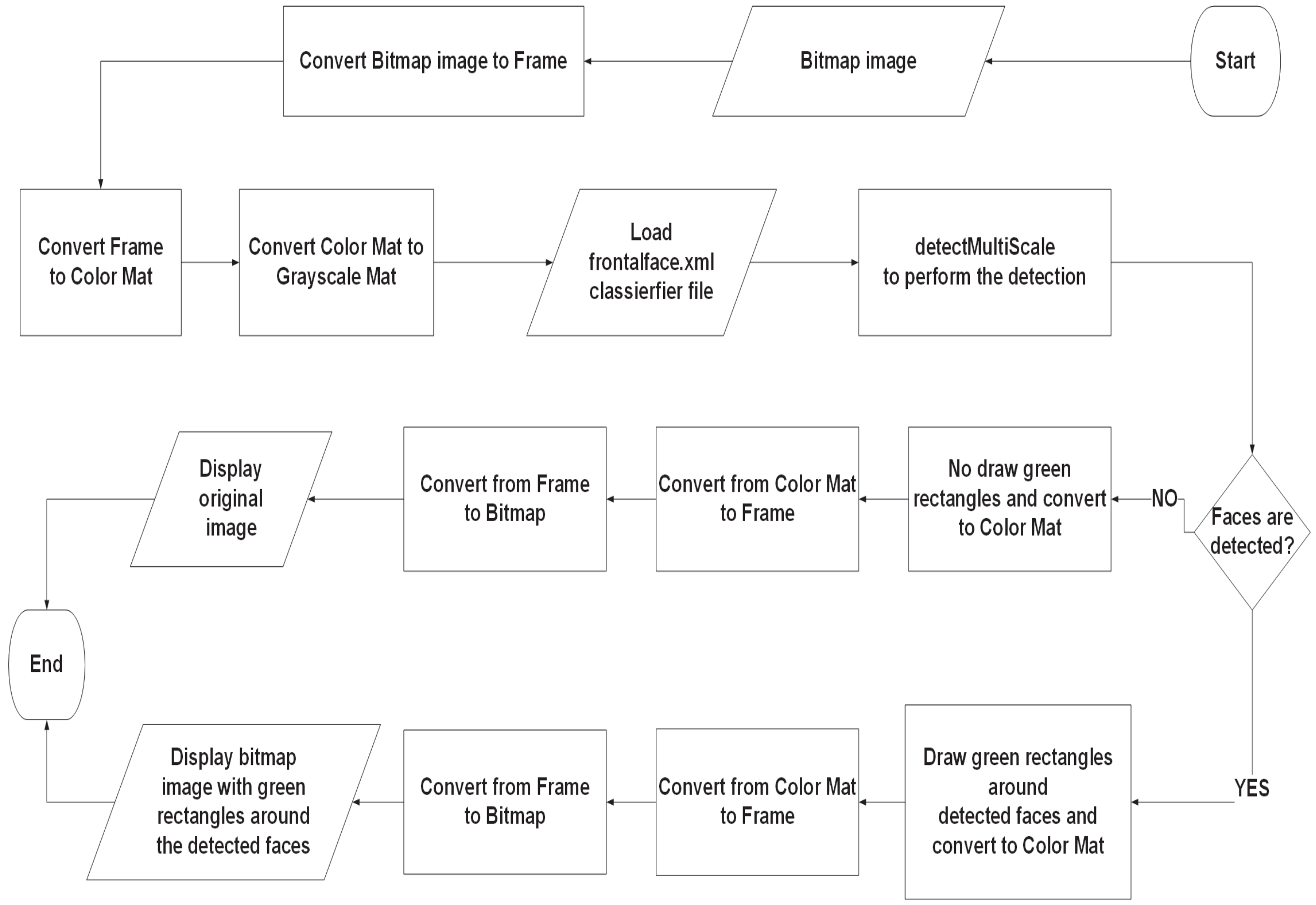

Start
Bitmap image
Convert Bitmap image to Frame
Convert Frame to Color Mat
Convert Color Mat to Grayscale Mat
Load frontalface.xml classierfier file
detectMultiScale to perform the detection
Faces are detected?
NO
No draw green rectangles and convert to Color Mat
Convert from Color Mat to Frame
Convert from Frame to Bitmap
Display original image
End
YES
Draw green rectangles around detected faces and convert to Color Mat
Convert from Color Mat to Frame
Convert from Frame to Bitmap
Display bitmap image with green rectangles around the detected faces

Figure 4. The face detection and display flowchart.

Model Creation

A functionality that is not released in the final version of Face2Rec is model creation. Model creation is the process of saving photos in a specific folder on a Samsung smartphone that adheres to OpenCV/JavaCV's model rules. An example of a model rule is that the photo must be in grayscale. Model creation is a feature of OpenCV/JavaCV that users who are not equipped with the background in OpenCV/JavaCV may mishandle. This is due to the need to use Android Studio to ensure the Face2Rec code can link to the model and ensure the database matches with the correct model. The process of model creation will still be explained here for future reference. After implementing face detection and capturing all the detected faces in a photo, a model will be created of each individual. For this study, the images used to create models belonged to the student authors. Through our testing, it was found that using more faces for training made the facial recognition more accurate. As a result, the model has about forty images for each student author. Initially, our images were captured using the Google Glass and then the images were automatically sent to the gallery. After that, our selected images were detected and stored in a training folder located in the external storage of the smartphone with the correct grayscale format and custom unique names. Finally, by using the create and train methods of EigenFaceRecognizer class,[17] a model of our faces is created using all the images in the training folder. The model is saved in the same folder as the training images. The EigenFaceRecognizer class has methods available in both OpenCV libraries and in Bytedeco's JavaCV.[18] To elaborate further on EigenFaceRecognizer, it has a create method that creates an object of the FaceRecognizer which contains the facial recognition algorithms.[19] The EigenFaceRecognizer's train method takes in two parameters, the trained images and the labels that correspond to the images. The labels are the keys for the facial recognition implementation. In other words, each person has a label attached to them and people are recognized by their different labels. **Figure 5** shows a flowchart of our model creation.

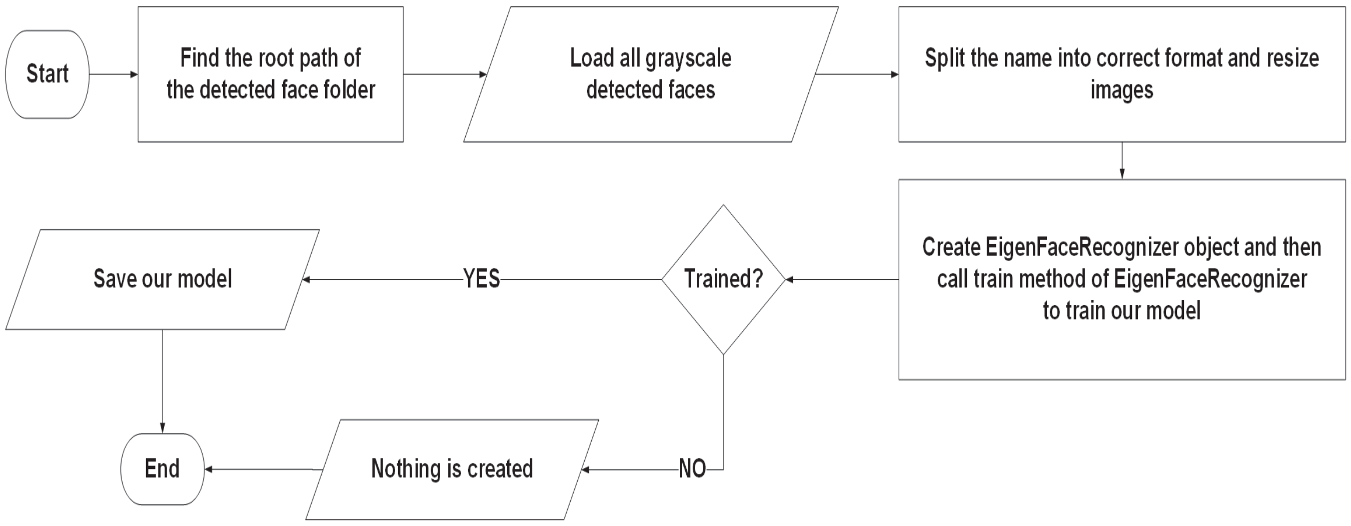

Start
Find the root path of the detected face folder
Load all grayscale detected faces
Split the name into correct format and resize images
Create EigenFaceRecognizer object and then call train method of EigenFaceRecognizer to train our model
Trained?
YES
Save our model
NO
Nothing is created
End

Figure 5. The model creation flowchart.

Database Setup

After creating a model, the SQLite external database is created using Android SQLiteAssetHelper.[20] Each person has an ID number as their primary key, a second field which contains their name, and a third field which contains their general background information. Creating and modifying the database was not hard because of the simplicity of SQLiteAssetHelper. As a result, the authors could concentrate on the more important parts of the application which was the facial recognition implementation and the communication between the Google Glass and smartphone.

Outcome

After creating the model and database, the EigenFaceRecognizer class was used to implement facial recognition. It utilized a prediction number and acceptance level number to match faces in a photo to faces in the models. Every image is assigned a specific label before the training, so when JavaCV's predict function is called, the label from a matching face was received from the model. By getting the label of the matching face, the prediction number was available and the correct matching face was known.

The acceptance level is a number that tells the user how reliable the results are. The closer the number is to zero, the more reliable the results. For best results, the acceptance level is set to the lowest number that would perform the best accuracy. The acceptance level was determined by trial and error. Face2Rec was tested multiple times to decide what would be the lowest possible acceptance level that is still accurate. If an acceptance level is higher than 4000, then the subject will be listed as unknown. If the acceptance level is greater than or equal to 3500 and less than or equal to 4000, then that is a middle acceptance level and the subject is a potential match. Lastly, if the acceptance level is less than 3500, then that is a low acceptance level and the match is reliable. If Face2Rec indicates a positive match, then the closer the acceptance level is to 3500 the more the user must scrutinize the results to determine its accuracy. **Figure 6** shows the facial recognition and display flowchart.

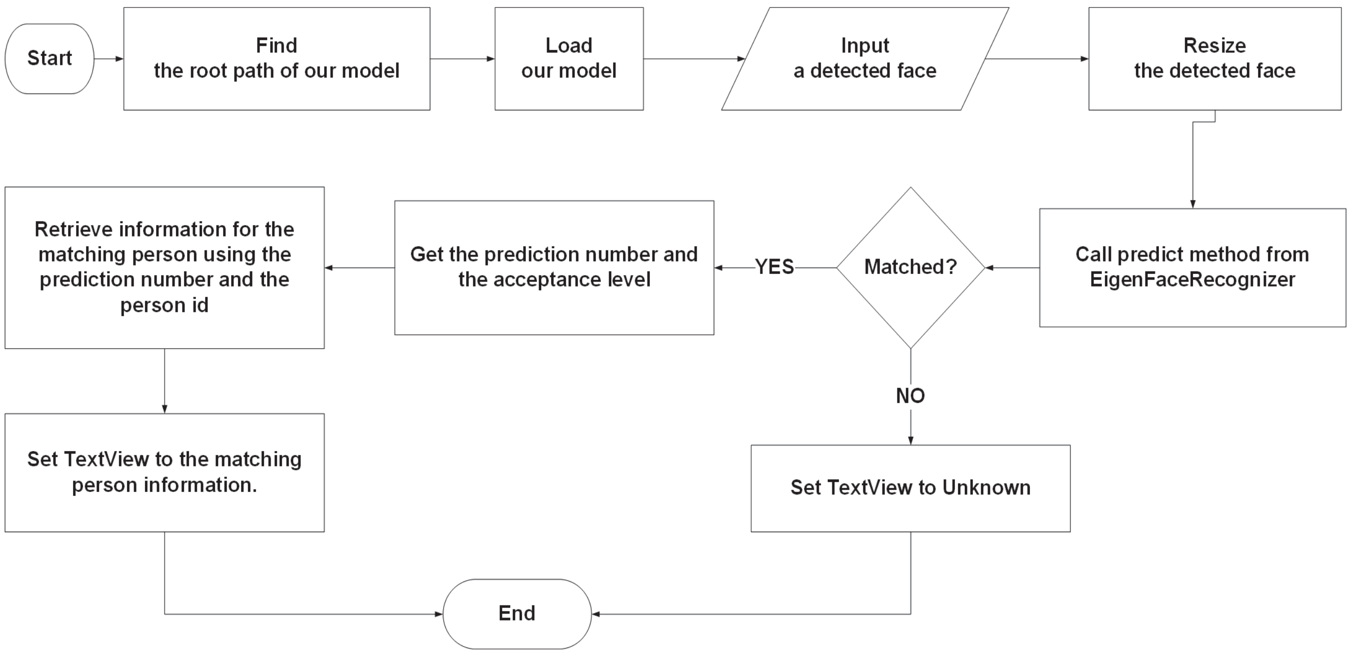

Start
Find the root path of our model
Load our model
Input a detected face
Resize the detected face
Call predict method from EigenFaceRecognizer
Matched?
YES
NO
Get the prediction number and the acceptance level
Retrieve information for the matching person using the prediction number and the person id
Set TextView to the matching person information.
Set TextView to Unknown
End

Figure 6. The facial recognition and display flowchart.

COMMUNICATION BETWEEN SMART GLASSES AND SMARTPHONE

Notifications Overview

Android provides online developer guides in order to educate users on how to program using Android Studio.[21] Within Face2Rec's code, more than thirty classes, packages, *etc.* are imported such as NotificationCompat,[22] to send notifications, and bitmap,[23] an image file type, in order to create this project. The NotificationCompat can only process bitmaps, so in order to send notifications, all imported images from the smartphone's gallery will be converted to bitmaps. The function notifications()[24] is programmed to use NotificationCompat so it was possible to tap into Google Glass' built-in feature of displaying notifications from a connected smartphone.[25]

Detecting Latest Image in Gallery

When Face2Rec first opens, it will check for images in the gallery by running the function lastPhotoInGallery() and display the latest photo in the gallery[26] or a blank main menu if there is no photo. The function moves a cursor to first position using moveToFirst() in order to detect for images in the gallery. Face2Rec uses the class GalleryObserver based on The Engineer's Cafe's DirectoryFileObserver[27] which extends the Android class FileObserver[28] in order to detect if there are changes in the smartphone's gallery. However, our GalleryObserver checks for modifications instead of if something was created. The Google Glass will automatically send any photos it takes to the smartphone's gallery and the GalleryObserver is constantly checking for those photos. A Handler was required to create a delay[29] to check for images because, based on code testing, the new images would not be detected by the GalleryObserver without a slight delay of two seconds. Once an image is detected in the gallery, BitmapFactory's decodefile[30] function converts the image into a bitmap for processing. That bitmap photo would be saved into a variable called bitmapAutoGallery.

Sending Latest Image as a Notification

In the function lastPhotoInGallery()[26], if there is an image and bitmapAutoGallery is updated, then that image will be sent to the function detectDisplayAndRecognize() to go through face detection and recognition. There, if bitmapAutoGallery has no faces, then it will go to the function notifications(). If there are faces detected, then the image will go to the function recognizeMultiple(), then displayMatchInfo(), and finally the image would be sent to notifications().

Opening the Gallery

Another option that is available for the user is to use a gallery viewer which displays all photos in gallery mode using the openGallery() function.[31] The user is shown a button on the main screen called "Select A New Face To Recognize". The images displayed are taken from the Google Glass since the Glass photos are delivered there directly. The MyGlass application provides this photo sync option and is allowing direct communication from the Google Glass to the Android smartphone.[32] In the function onActivityResult(), selected photos are converted to bitmaps[33] and saved under a variable called bitmapSelectGallery. If the user selects to open the gallery, but does not choose a photo, then an alert will appear stating no gallery image selected. If an image is selected from the gallery and bitmapSelectGallery is not null or empty, then the image will go to the function detectDisplayAndRecognize(). There, if no faces are detected, the image will go to notifications(). If there are faces detected, then the image will go to recognizeMultiple() and then displayMatchInfo(). Lastly, the image would be sent to notifications(). The two options users have to send their photos through Face2Rec's code is shown in **Figure 7**.

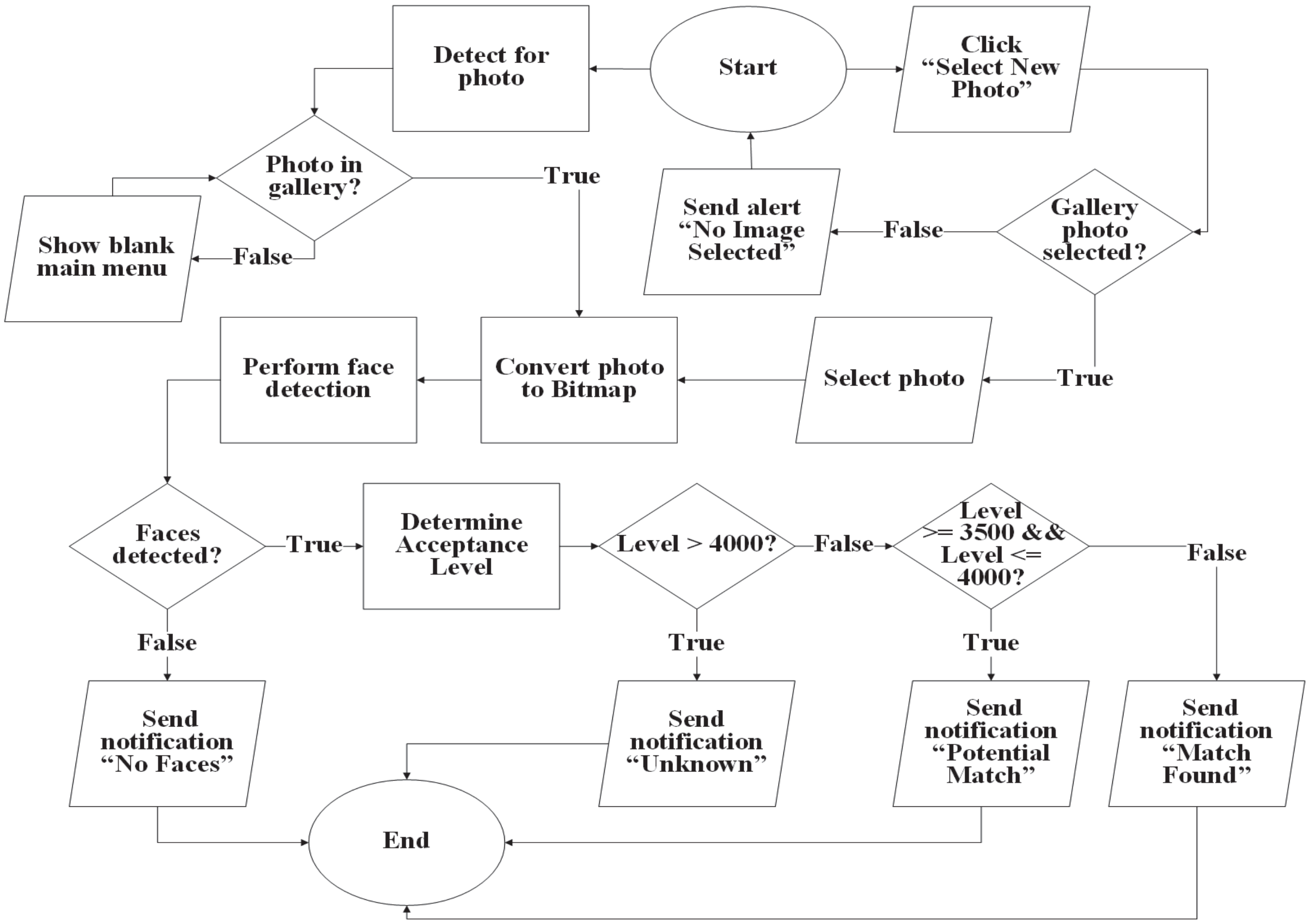

Start
Detect for photo
Click "Select New Photo"
Photo in gallery?
True
False
Show blank main menu
Send alert "No Image Selected"
Gallery photo selected?
Perform face detection
Convert photo to Bitmap
Select photo
Faces detected?
Determine Acceptance Level
Level > 4000?
Level >= 3500 && Level <= 4000?
Send notification "No Faces"
Send notification "Unknown"
Send notification "Potential Match"
Send notification "Match Found"
End

Figure 7. The notifications flowchart.

Sending Notification Images

Now that it is established how images are sent to notifications(), Face2Rec must decide which image to display when sending notifications. This is done by a simple if/else statement that checks to see if bitmapSelectGallery and bitmapAutoGallery are not null. At the end of running notifications(), bitmapSelectGallery is set to null in order to free the variable for a new selected image. So if bitmapAutoGallery is not null, then the image gets saved into a variable called finalBitmapPic, but if bitmapSelectGallery is not null, then that image will be saved in finalBitmapPic. The bitmap variable finalBitmapPic is required because NotificationCompat can only accept one variable in its method setLargeIcon. This method displays images to the smartphone and Google Glass once a notification is sent. The application knows what text to display based on what is saved in the variables matchText and moreInfo that was defined in the function detectDisplayAndRecognize() if there are no faces detected or displayMatchInfo() if there are faces detected. The text that is displayed is pulled from a previously created database. If the text that is saved in the database is too long for the smartphone screen, then the text will display with the option to scroll through the text on the main menu. This was done by setting scrollbars to equal vertical in the TextView under layout. The notification that appears on the smartphone can also be tapped in order to expand the notification. The Google Glass automatically enables a scrolling feature to view more information of any notification that is sent to it. To enable the scrolling feature, the user only needs to tap the Glass when the notification appears.

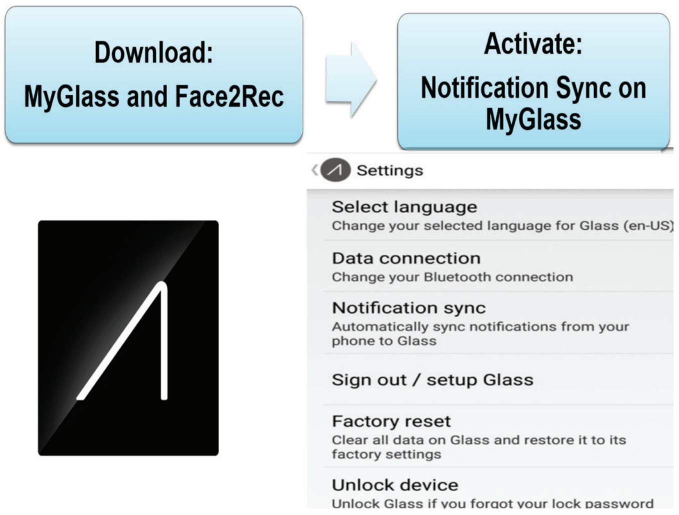
SETTING UP THE DEVICES

Downloads

To begin using Face2Rec, the user must download the Google Glass MyGlass[34] application from the Google Play store and download Face2Rec onto their smartphone. Our Face2Rec application is not available on the Google Play store, so the user must add it via Android Studio. They can do this by cloning Face2Rec from Github[35] into Android Studio and then running it on their Android smartphone. Preferably, their smartphone will be similar to our Samsung Galaxy S9 due to the processing power the two applications will require. The MyGlass application enables the user's smartphone to connect with their Google Glass and the Face2Rec application will be where the face detection and recognition will occur. After downloading both applications, the user must set them up.

Linking Devices

On the MyGlass application, the user must link the Google Glass to their phone by following the Google Glass instructions that are displayed on the Google Glass and MyGlass application. In order for Face2Rec to work, the user must enable Notification Sync and Photo Sync in MyGlass. **Figure 8** shows the icon for MyGlass and the settings option[34] on MyGlass that enables Notification Sync. The Photo Sync option may display on the main page instead of in settings. Notification Sync is required so Face2Rec's results can be displayed onto the Google Glass. Photo Sync is required so the Google Glass can save images to the user's smartphone gallery. The user must also allow Face2Rec to have read/write privileges to their device which can be done once the application first launches on their smartphone or in their phone's settings when they download the application.



Settings

Select language
Change your selected language for Glass (en-US)

Data connection
Change your Bluetooth connection

Notification sync
Automatically sync notifications from your phone to Glass

Sign out / setup Glass

Factory reset
Clear all data on Glass and restore it to its factory settings

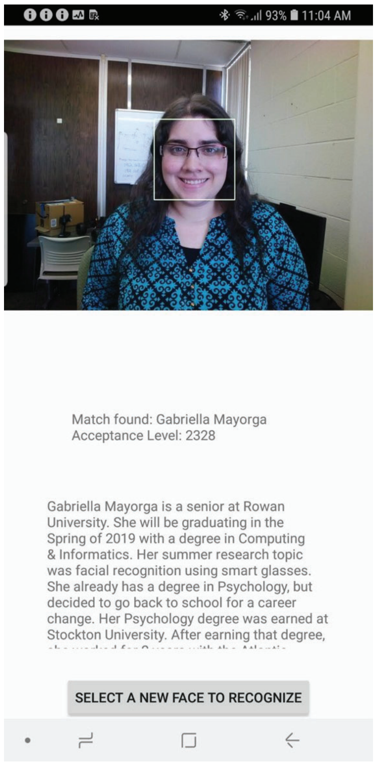
Unlock device
Unlock Glass if you forgot your lock password

Figure 8. Setting up Notification Sync.[34]

USER EXPERIENCE

Main Menu

Once the user has set up both applications, their smartphone, and the Google Glass, they can then begin to use Face2Rec to perform facial recognition. Upon opening Face2Rec, the user will see the most recent photo on their device displayed on the main menu page or a blank menu if there are no photos. **Figure 9** shows what the user may see. Face2Rec will automatically search for a match based on what is on the main menu and display one of four outcomes to the user: no match, potential match, match, or no face detected. When the user takes a new photo with the Google Glass, Face2Rec will automatically detect a new photo has been added to the smartphone via the Google Glass MyGlass application and update the Face2Rec application to begin its facial recognition process. The user also has the option to manually open their phone's gallery via Face2Rec to select a photo for facial recognition processing.



Match found: Gabriella Mayorga
Acceptance Level: 2328

Gabriella Mayorga is a senior at Rowan University. She will be graduating in the Spring of 2019 with a degree in Computing & Informatics. Her summer research topic was facial recognition using smart glasses. She already has a degree in Psychology, but decided to go back to school for a career change. Her Psychology degree was earned at Stockton University. After earning that degree,

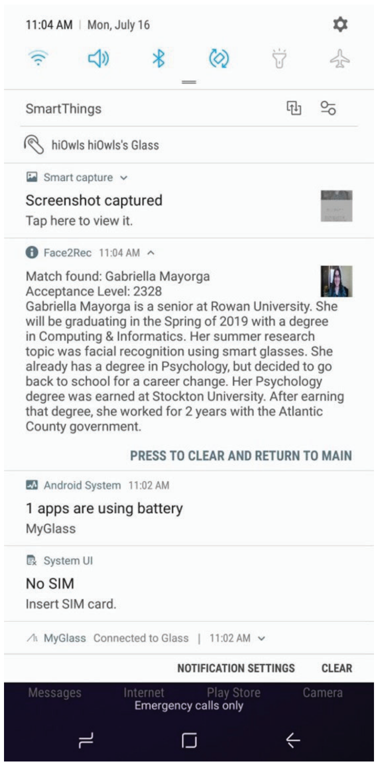
Figure 9. Face2Rec main menu.

Outcome

If Face2Rec performs its facial recognition processing and finds a match, information about the match that is stored in the database will be displayed on both the user's smartphone and Google Glass via a notification and on the main menu. If there is no match, partial match, or no face detected then the user would be sent a notification with that information. The user will also be given an acceptance level in order to determine the reliability of the information that is being displayed. The acceptance level is set in the recognizeMultiple() function and the displayMatchInfo() function determines how to categorize the level. The higher the acceptance level, the less reliable the match. More information about acceptance level can be found in the *Face Detection and Recognition Implementation* section. While an acceptance level may be worryingly close to 3500, the user must determine if glasses, hair being in the subject's face, or anything else affected the level before deciding the accuracy of the results.

Scrolling and Dismissing Notifications

Google Glass and Face2Rec are equipped to handle long text once a photo is done being processed for a match. If the text sent to the user about the processed photo is too long to fully display on either the smartphone or Google Glass, then the option to scroll through the text is available. Face2Rec allows the user to scroll directly on the application using touch, notifications on the smartphone can be opened to accommodate more text, and the Google Glass also gives the user the option to open the notification as scrollable text right on the Google Glass display. Once the user is done viewing the notification, they can dismiss it from their screen. If they wish to not dismiss the notification, they can simply ignore it and it will remain as a notification until it is dismissed. New notifications from Face2Rec will not overwrite old ones due to each notification having its own notification ID. This allows the user to see their recent activity. **Figure 10** shows how a notification may appear on an Android smartphone and **Figure 11** shows the same notification on Google Glass.


11:04 AM | Mon, July 16

SmartThings

hiOwls hiOwls's Glass

Smart capture

Screenshot captured
Tap here to view it.

Face2Rec 11:04 AM

Match found: Gabriella Mayorga
Acceptance Level: 2328
Gabriella Mayorga is a senior at Rowan University. She will be graduating in the Spring of 2019 with a degree in Computing & Informatics. Her summer research topic was facial recognition using smart glasses. She already has a degree in Psychology, but decided to go back to school for a career change. Her Psychology degree was earned at Stockton University. After earning that degree, she worked for 2 years with the Atlantic County government.

PRESS TO CLEAR AND RETURN TO MAIN

Android System 11:02 AM
1 apps are using battery
MyGlass

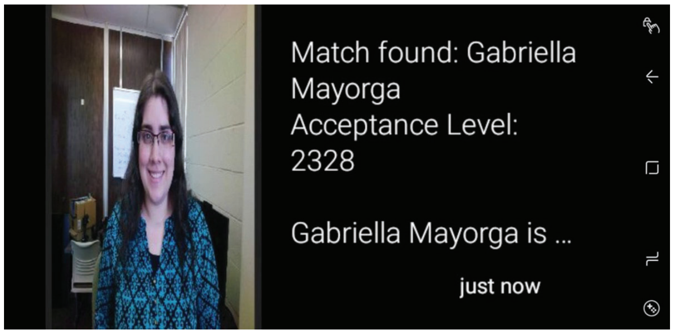
System UI
No SIM
Insert SIM card.

MyGlass Connected to Glass | 11:02 AM

NOTIFICATION SETTINGS CLEAR

Messages Internet Play Store Camera
Emergency calls only

Figure 10. Android device notification.


Match found: Gabriella Mayorga
Acceptance Level: 2328

Gabriella Mayorga is ...

just now

Figure 11. Google Glass notification. If Glass tapped, the option to scroll through the text is provided.

RESULTS

To test OpenCV/JavaCV's algorithm accuracy, a group photo and a solo photo was used. In a group photo, Face2Rec scanned all the faces to find one who is in the database. If no one in the photo was in the database, then Face2Rec would display an "unknown" notification. If it found someone who is in the database, then it will calculate and display an acceptance level. During our group photo testing, the algorithm was able to pick out the individual who was in the database and provide an acceptance level. However, if the individual was far from the camera, then the individual's acceptance level may qualify the match as only potential. This could be because of the image quality being too low due to the distance. If the individual in the database uses glasses, then that can also confuse the algorithm. In one instance, the algorithm recognized one student author as the other student author potentially because they both wear glasses and the image was not a close-up. A solo photo had the same level of accuracy as a group photo. Another OpenCV and JavaCV issue was that it was unable to process large file sizes. Face2Rec can potentially crash due to an image being greater than a couple hundred Kilobytes. Fortunately, the Google Glass photos are small so Face2Rec does not crash. However, photos taken with the Android smartphone does crash the application due to their high quality. OpenCV and JavaCV also may not always perform accurate facial recognition when a head is tilted in a photo. Based on this information, OpenCV/JavaCV's facial recognition algorithm was not very accurate. It is a good introduction to the potentials of facial recognition, but due to the noted cases, it should not be relied upon. The application, Face2Rec, however, did perform as expected. Face2Rec was able to take in input from Google Glass and the smartphone's gallery and output results based off OpenCV/JavaCV's algorithm calculations. The Google Glass took a few seconds, as great as ten seconds during some test runs, to send photos to the Android smartphone. This means Face2Rec did not work as quickly as it should. However, since Google Glass Photo Sync was used, the speed to send photos is out of our control.

DISCUSSION

The overall purpose of this project was to create an Android application that communicated with Google Glass and performed facial recognition on images. Face2Rec successfully executed all our objectives for this project. Face2Rec is able to identify the two student authors and can send notifications to all devices. However, while the application itself is able to perform all our goals, it is hindered by OpenCV/JavaCV's unreliable algorithm and by the Google Glass slowness.

Future Updates

A future version of Face2Rec may involve a custom function to send photos to the Android phone faster. Other algorithms besides OpenCV/JavaCV need to be used that can handle all the variables as noted in our *Results* section. Face2Rec also requires an update in order for non-programmers to easily update the database with their persons of interest. At this time, the user needs to use Android Studio to manually input more people into the Face2Rec database. Lastly, Face2Rec can be improved by incorporating a web crawler that can search online photos and pull information about individuals once it finds a match.

CONCLUSIONS

Face2Rec is a good starting point into what could be a highly useful application. After working on it for only ten weeks, it progressed quickly and has a lot of potential. If Face2Rec is improved upon as noted in our *Discussion* section, then it can make the world a safer place. At this time, the facial recognition algorithm used in Face2Rec is not 100% accurate. Therefore, it should not be officially used by law enforcement, the government, or any other institution. However, facial recognition is being continuously worked on by multiple companies so an accurate algorithm is expected one day. The code of this project is open source and can be used by anyone in order to improve it. The project was saved in multiple parts, but the combined final version of the code can be found on GitHub.[35]

REFERENCES

1. Amazon, Amazon Rekognition – Video and Image, *https://aws.amazon.com/rekognition/* (accessed Nov 2018)
2. Microsoft, Face API - Facial Recognition Software, *https://azure.microsoft.com/en-us/services/cognitive-services/face/* (accessed Nov 2018)
3. OpenCV, Face Recognition with OpenCV – OpenCV 2.4.13.7 documentation, *https://docs.opencv.org/2.4/modules/contrib/doc/facerec/facerec_tutorial.html* (accessed Nov 2018)
4. OpenCV, OpenCV, *https://opencv.org/* (accessed Nov 2018)
5. OpenCV, About, *https://opencv.org/about.html* (accessed Nov 2018)
6. Bytedeco, Java interface to OpenCV, FFmpeg, and more, *https://github.com/bytedeco/javacv* (accessed Nov 2018)
7. Bytedeco, javacpp, *https://github.com/bytedeco/javacpp* (accessed Nov 2018)
8. Metz, C. and Singer, N., Newspaper Shooting Shows Widening Use of Facial Recognition by Authorities, *https://www.nytimes.com/2018/06/29/business/newspaper-shooting-facial-recognition.html* (accessed Nov 2018)
9. LLVision, About Us, *https://www.llvision.com/#/about* (accessed Nov 2018)
10. Li, P. and Cadell, C., China eyes 'black tech' to boost security as parliament meets, *https://www.reuters.com/article/us-china-parliament-surveillance-idUSKBN1GM06M* (accessed Nov 2018)
11. Alvarez Casado, C., Bordallo Lopez, M., Holappa, J., and Pietikainen, M. (2015) Face detection and recognition for smart glasses, *ISCE. https://www.doi.org/10.1109/ISCE.2015.7177783*
12. Hellmund, T., Seitz, A., Haladjian, J., and Bruegge, B. (2018) IPRA: real-time face recognition on smart glasses with fog computing, *ACM. http://doi.acm.org/10.1145/3267305.3274122*
13. OpenCV, cv::Mat Class Reference, *https://docs.opencv.org/3.2.0/d3/d63/classcv_1_1Mat.html* (accessed Nov 2018)
14. Bytedeco, Introducing JavaCV frame converters, *http://bytedeco.org/news/2015/04/04/javacv-frame-converters/* (accessed Nov 2018)

15. OpenCV, OpenCV: Cascade Classifier, *https://docs.opencv.org/3.4.1/db/d28/tutorial_cascade_classifier.html* (accessed Nov 2018)
16. OpenCV, OpenCV: Face Detection using Haar Cascades, *https://docs.opencv.org/3.4.1/d7/d8b/tutorial_py_face_detection.html* (accessed Nov 2018)
17. Bytedeco, Class opencv_face.EigenFaceRecognizer, *http://bytedeco.org/javacpp-presets/opencv/apidocs/org/bytedeco/javacpp/opencv_face.EigenFaceRecognizer.html* (accessed Nov 2018)
18. Bytedeco, Home, *http://bytedeco.org/* (accessed Nov 2018)
19. Bytedeco, Class opencv_face.FaceRecognizer, *http://bytedeco.org/javacpp-presets/opencv/apidocs/org/bytedeco/javacpp/opencv_face.FaceRecognizer.html* (accessed Nov 2018)
20. Gilfelt, J., android-sqlite-asset-helper, *https://github.com/jgilfelt/android-sqlite-asset-helper* (accessed Nov 2018)
21. Android, Developer Guides, *https://developer.android.com/guide/* (accessed Nov 2018)
22. Android, NotificationCompat.Builder, *https://developer.android.com/reference/android/support/v4/app/NotificationCompat.Builder* (accessed Nov 2018)
23. Android, Bitmap, *https://developer.android.com/reference/android/graphics/Bitmap* (accessed Nov 2018)
24. Android, Create a Notification, *https://developer.android.com/training/notify-user/build-notification* (accessed Nov 2018)
25. Google, Notifications, *https://support.google.com/glass/answer/3086048?hl=en* (accessed Nov 2018)
26. Wei, T. , Get the latest image from external storage in android, *https://stackoverflow.com/a/41442632* (accessed Nov 2018)
27. Kanzariya, M., How to create FileObserver in Android, *https://theengineerscafe.com/create-fileobserver-android/* (accessed Nov 2018)
28. Android, FileObserver, *https://developer.android.com/reference/android/os/FileObserver* (accessed Nov 2018)
29. Pomber, java - How to call a method after a delay in Android, *https://stackoverflow.com/a/28173911* (accessed Nov 2018)
30. Android, BitmapFactory, *https://developer.android.com/reference/android/graphics/BitmapFactory* (accessed Nov 2018)
31. Androidbitmaps, Android Bitmaps Tutorials: Loading Images in Android Part III: Pick images from Gallery, *http://androidbitmaps.blogspot.com/2015/04/loading-images-in-android-part-iii-pick.html* (accessed Nov 2018)
32. Google, Sharing your pictures and videos - Google Glass Help, *https://support.google.com/glass/answer/3079691?hl=en* (accessed Nov 2018)
33. Yupi, Issue converting Image Uri from gallery to bitmap Android, *https://stackoverflow.com/a/47214198* (accessed Nov 2018)
34. Google, MyGlass, *https://play.google.com/store/apps/details?id=com.google.glass.companion&hl=en_US* (accessed Nov 2018)
35. Mayorga, G. and Do, X., Face2Rec, *https://github.com/GMayorga/Face2Rec* (accessed Nov 2018)

ABOUT THE STUDENT AUTHORS

Gabriella A. Mayorga and Xuan Do completed this project while earning a Bachelor of Arts in Computing and Informatics from Rowan University. They both plan to graduate in the Spring of 2019. They were selected by Dr. Heydari to participate in Rowan University's Summer Undergraduate Research Program (SURP) in 2018 which provided their funding for 10 weeks to develop this application. Gabriella A. Mayorga plans to focus on cybersecurity and is interested in software development and mobile applications. Xuan Do plans to pursue her career in web development.

PRESS SUMMARY

Using facial recognition and detection to identify people is a useful tool for law enforcement, the government, and universities. Combining that technology with smart glasses gives those agencies the ability to identify people in a hands-free manner. Our project, Face2Rec, is designed to do that by using an Android smartphone that communicates with Google Glass. Face2Rec searches for faces in a Google Glass photo and if a face is found, it compares those faces against a user-created database of people. The user is informed if there is no face detected, no match found, partial match found, or match found. If there is a match, the user is given additional information. All results are displayed on their Android smartphone and smart glass. The user is also given an acceptance level for each result to independently determine the reliability of the information.