Face Recognition using Microsoft Cognitive Service Face API

Martino Christanto Khuangga - 13514084
Informatics/Computer Science Program
School of Electrical Engineering and Informatics
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13514084@std.stei.itb.ac.id

***Abstract*—nowadays humans use technology to boost their work performance. Technologies, which is usually a machine, have a better speed than human at some work. But, some of technologies lack of security aspect. Because of that, biological features of human used as input to improve securities. One of the biological features that improve security is face. Therefore people start to develop face recognition algorithm, and many developers start to use the algorithm. To help the developers build their application with face recognition easily, many corporation start build API, so developers don't need to create the face recognition algorithm from the beginning again. One of the API is Microsoft Cognitive Service Face API. This paper provides how to use Microsoft Face API, and what the function does, so people still understand the system, even they don't create the face recognition algorithm.**

Keywords—security, technologies, face recognition, Microsoft Cognitive Service, Face API

I. Introduction

These days, many softwares are developed to make people for more efficient. However, there are many challenges to develop a good software that users really need. Some of the parameters are security and good design of the software.

Many softwares use authentication to ensure the safety. But, security issues grow as technology does too. Security right now can't be guaranteed just by password, which is a string (text). As we know that these days many people complain about security in ATM, website, online shop, etc. Therefore, people try to develop something that very secure and yet easy enough for people to use. And one of the method is using face recognition.

Face recognition is a method for machine to learn an image to find human face on the image. After the face(s) are being detected, machine start to get some information from it, for example gender, age, pupil pattern, lip pattern, etc. The process for a machine to learn face pattern from many images is called training [1]. Training is a useful method when we detect a face of an unknown person, so machine can recognize the face easily in the future.

From Microsoft face API documentation, we know that face recognition is used in scenario like security, natural user interface, and image content anaylisis and management. Microsoft Face API contains four function: face verification, finding similar faces, face grouping, and person identification [3].

Face recognition is widely used in many platforms and scenarios. Many developers use this face recognition algorithm to enhance their application. There are face recognition library developed by many corporation, and one of them is Microsoft, which provide ease of use in many platform to develop. But this ease of use sometimes make developers don't understand what happens behind the code. So this paper will also explain what a function does.

This paper provide organized into five section. The first section is introduction. The second section is related works about Microsoft Face API. The third section contain how face API used for training process. The fourth section is experiments of the Face API and the implementation of the algorithm to get the experiments result. And the last section we give conclusion of the experiments and suggestion for future works.

II. Related Works

Microsoft Face API, as described in the previous section, contain four main function, there are face verification, finding similar faces, face grouping, and person identification [3].

A. *Face Verification*

Face verification is a function to verify whether two faces is similar or a face is belong to a person [3]. As we know that this method is similar to simple validation in basic programing. The differences is, this function need to train face image first. After the training set produced, we can verify (test) another face, whether it's belong to someone in our training set or still unknown. This function also can be used for validating two faces whether it's similar or not directly.

B. *Finding Similar Faces*

This function is like searching in basic programming. Basically, this method has a target face and candidate faces. This function contain two modes, the first one is called matchPerson and the second one is called matchFaces [3]. The matchPerson mode implementing same person threshold, which is derived from face verification function [1]. This mode have more precision in actual life, because this mode also calculate another factors beside face.

In opposite, matchFace verify face based on face landmarks and expression only. That's why this mode usually return a more general result, which we can say have bias.

C. *Face Grouping*

This function will group several unknown faces based on similarities [3]. This function is a development from another function above. This function have a weakness, where faces from a same person might be grouped to several group. Face grouping need at minimal 2 faces, and 1000 at most [4]. There is also a special group called messyGroup in this function. Messy group is a group of faces that can not find any similar counterpart face from original faces.

D. *Person Identification*

Face identification used to identify a person by face. For example, this function can determine the name of a person from face we input to the API [3]. The identification can be done after a group of faces made. A group of faces is trained so the machine find the face characters from a person. After a group of face trained, an identification can be done. This function return a person object to user [3].

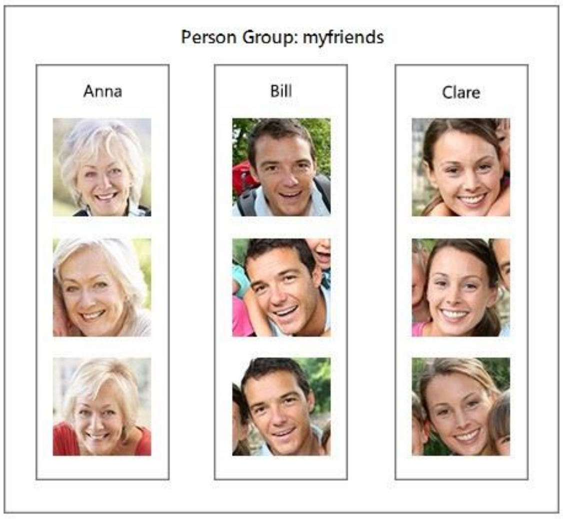
III. Face Training

Fortunately, Microsoft has a really helpful library, so we can implement the Face API code easily. This library is easily used in many platform. In this paper, we will use the library in C# and Android platforms. We use C# to create the train data, and use the Android to identify (test) image input. In C# we can browse this library via NuGet, the library called "Microsoft.ProjectOxford.Face" [2].

First, we include the Microsoft Project Oxford Face Library in our projects. Before implement the code, it is recommended that we have some faces of a same person first in a folder, in order to make data train process efficient. Beside that, we need to declare our client object first in our code, in order to request to Microsoft Face API. We can get the subscription key via Microsoft Cognitive Service Official Website [3].

A. *Create A Face Group*

After we have faces of a same person, we must create a group first. So in this paper, we do our face grouping manually, not using Face Grouping function from the API [2]. See picture 1 for an example of face grouping. Faces in the same rectangle (e.g labeled "Anna"), are a group of faces we recognize (manually) as Anna. And all faces, are grouped again into one big group labeled "myfriends".



Picture 1 Example of face grouping
Source : https://docs.microsoft.com/en-us/azure/cognitive-services/Face/images/person.group.clare.jpg

To start create group, we simply call CreatePersonGroupAsync method from the client we declared before. The method require person group id and person group name. Both are string. A success response will create the group on Microsoft Server [8].

B. *Add Face to Person Object*

When a group has been created, we can freely add person object to the group. To create a person object, first we must declare a CreatePersonResult object, which we initialized with return value of CreatePersonAsync method from the client object [5]. The CreatePersonAsync method need two input parameters, person group id and person name. The person name is just like the person group id before. We can give a name to a person object freely [2]. Becaus the person group just being initialized, it's lack of person information. So every faces we input might be unidentified by the API at first.

The add person process also need our image input. To add the face image to person object we create before, we simply can call AddPersonFaceAsync method from the client object [2]. The method need three parameters, person group id, person (object) id, and the stream of the face image file. By this method we are not training the data yet. This method just simply add multiple images which we (manually) identified as the person faces.

C. *Train Faces of A Person*

After we add the faces to a person, we must get the pattern of the faces to make the API learn a person face pattern. To get the face pattern, we use training method [6] just like in artificial intelligence algorithm. To train a person we call a method TrainPersonGroupAsync from the client object. This method will train the all image data from a person group. Therefore, this method needs a person group id as its input parameter. Before we start face identification, we should confirm that the training method is finished.

Microsoft provide a method to check whether the training has been finished yet or not. The method called GetPersonGroupTrainingStatusAsync from the client object. Same as the TrainPersonGroupAsync method, this method require person group id. This method return a string that represents the status of the training [7]. There are four possible return value: “notstarted”, “running”, “succeeded”, “failed”. In this part, we only want to check if the training has been done or not. To do that, we can add a infinite loop, which will break when the the status of the training is not “running” [2]. Because we will not do the face identification in this part, so we don’t need to check if the training success or not.

IV. Face Identification Implementation And Result

In this paper, we identify the face in different platforms (Android) to simplified the process of data training and data test. Before start to identify faces, we need to setup the library first. In Android Studio, we should open the project gradle, and add “mavenCentral()” in allprojects repositories part [2]. After that, we add dependencies in app gradle. To do that, we simply add “compile ‘com.microsoft.projectoxford:face:1.0.0’” in the dependencies part [2]. And also, we need to add permission to use internet in our manifest file. To begin the code section, we need to initialize client object first just like in the face training section.

A. *Import Image File to Bitmap*

First thing to do, after we set up the environment, we need to get an image file bitmap first, because Android can not process the raw image file. There are two ways to load the image, in this paper, we will use user input. This method means user can browse their own image and analyze it. Assume that the image is valid (contain at least one face, and the face belong to a person, whose face has been trained before), user can analyze every image as long as the image is on user’s phone.

To convert raw image into bitmap, we must create an InputStream object and initialize with the return value of the openInputStream method in ContentResolver class [2]. After that, we can easily get the image bitmap using decodeStream method in BitmapFactory class. This method require an inputStream in its contructor, and this stream is the one we create before.

B. *Detect Face(s)*

After we get image bitmap, we must detect the face in the picture first. To do that, we need to convert the bitmap into ByteArrayInputStream object. Using this ByteArrayInputStream, we can detect face(s) easily. To do face detection, we call detect function in client object. This function need the ByteArrayInputStream object we declared before [2]. Face(s) detected from the image will be stored in array of Face object.

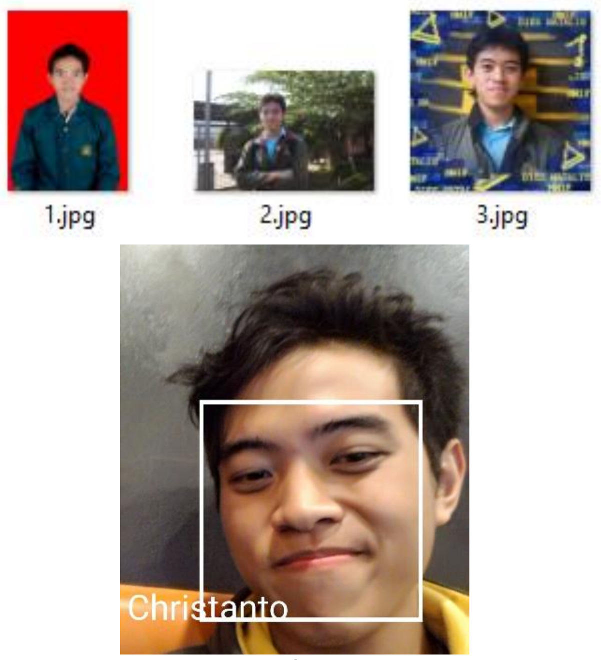
C. *Identify The Face(s)*

This section is only check whether the training data in the saved in the server is available or not. The method to detect the status is quite similar with the face train section before. To detect the status, we can use getPersonGroupTrainingStatus method in client object [2]. If the status is not “succeeded” then the program will be stoped. If the status is “succeeded” then the program will continue to detect who are the persons in the image.

D. *Person Detection*

Person detection simply use the method getPerson from the client object. The method require person group id and UUID. The UUID (person ID) can be gotten from the identify result from previous section. Continuously, we can call these function on identify result: candidates, get, and get the person ID [2]. Assume that the image is valid, these method won’t return a null, but if the image contain face not belonging to anyone in training data, the application will be crashed.

After we detect a person, we can draw rectangle on face in the image and write their name on the image. To draw a rectangle and person’s name, we need to modify the bitmap we save earlier. We can use Canvas and Paint class to draw the rectangle and the person name. To draw the rectangle exactly at the face, we can use the coordinates from the face objects. The example of the experiment result can be seen in picture 2.



Picture 2 Experiments result of a person named “Christanto”. 1.jpg, 2.jpg, and 3.jpg are the trained data.

V. Conclusion

In this paper we have presented how to use Microsoft Face API. As described in second section, Microsoft Face Api is really easy to use and to understand. However, the implementation of the API need to be more careful because some of the "fail" condition return null value.

Developers who wants to use this API needs to handle all the error. So the user will not face a crash in the software. The suggestion also goes to Microsoft, to minimize the null return value, so it will become easier to use by many developer.

REFERENCES

[1] Emami, Shervin. 2012. "Introduction to Face Detection and Face Recognition". Retrieved May 03, 2017, from http://www.shervinemami.info/faceRecognition.html

[2] EMDTDev team, "Android Studio Tutorial – Face Identification". Retrieved May 03, 2017, from http://www.edmtdev.com/

[3] Microsoft Cognitive Service Team, "Face API," Washington: Microsoft, 2017. Retrieved May 03, 2017, from https://docs.microsoft.com/en-us/azure/cognitive-services/Face/Overview

[4] Microsoft Cognitive Service Team, "Face API," Washington: Microsoft, 2017. Retrieved May 03, 2017, from https://westus.dev.cognitive.microsoft.com/docs/services/563879b61984550e40cbbe8d/operations/563879b61984550f3039523a

[5] Microsoft Cognitive Service Team, "Face API," Washington: Microsoft, 2017. Retrieved May 03, 2017, from https://westus.dev.cognitive.microsoft.com/docs/services/563879b61984550e40cbbe8d/operations/563879b61984550f30395238

[6] Microsoft Cognitive Service Team, "Face API," Washington: Microsoft, 2017. Retrieved May 03, 2017, from https://southeastasia.dev.cognitive.microsoft.com/docs/services/563879b61984550e40cbbe8d/operations/563879b61984550f30395244

[7] Microsoft Cognitive Service Team, "Face API," Washington: Microsoft, 2017. Retrieved May 03, 2017, from https://docs.microsoft.com/en-us/azure/cognitive-services/face/face-api-how-to-topics/howtoidentifyfacesinimage#a-namestep3a-step-3-train-the-person-group

[8] Microsoft Cognitive Service Team, "Face API," Washington: Microsoft, 2017. Retrieved May 03, 2017, from https://westus.dev.cognitive.microsoft.com/docs/services/563879b61984550e40cbbe8d/operations/563879b61984550f30395247

STATEMENTS

This paper is originally create by the author, not a copy or tranlation from another paper, and not an act of plagiarism.

Bandung, May 4th 2017

[illegible]

Martino Christanto Khuangga - 13514084