Implementing Face Recognition With OpenCV Using Python Programming Language

Alson Cahyadi - 13514035
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13514035@std.stei.itb.ac.id

***Abstract*—Face Recognition is one of the implementations of AI (Artificial Intelligence), which focuses on detecting a face given a frame, and identifying whose face(s) is/are given. The focus of this paper is to implement face recognition with OpenCV using Python as the main language, and Haar cascade as the classifier. The result of this implementation includes the datasets, model, and source code. From the model created in this implementation, the model is able to recognize faces more or less 70% of the frames accurately given a good dataset, which is enough but still needs improving.**

Keywords—artificial intelligence; face recognition; OpenCV; Python; Haar cascade

I. INTRODUCTION

The background behind this paper is our urge to learn how *Artificial Intelligence* works with conjunction to image processing. We learn the workflow of supervised *Artificial Intelligence* to have a hands-on experience of making datasets, train a model, and classifying data tests, in the form of video file, or input from a webcam.

Even though face recognition can be done easily by humans, computers have a hard time doing it. This is unfortunate, since the automation of face recognition can open the door to eye opening innovations.

Face detection is the capability to detect faces within a given frame, and pinpointing the face position. In this particular case, the ability to pinpoint the coordinate of the face and

On the other hand, face recognition is the capability of detecting faces within a frame, and label the faces given a model which was trained beforehand using available datasets. This capability can be achieved through a series of action as follows: generating datasets, training models, and classifying inputs.

II. LITERATURE STUDY

A. Supervised Learning

Supervised learning is the task of inferring a function from labeled training data, which is one of the most common training methods of machine learning. This learning method produces a model which can be used to label test data. Each training data is a pair that consists of the object, and the label associated with it.

To be able to label data test accurately, this learning algorithm is required to find patterns within the training data, and make a generalization in the form of a function that is able to classify data test to a specific label. The function's performance is measured by the percentage of accuracy of classifying data, that should be a group of separate data from the training data. There is a wide range of learning algorithms, each with specific tasks, with their own strengths and weaknessses.

B. Haar Cascade Face Detection

Haar feature-based cascade classifiers is an effective object detection method which is approached from the machine learning perspective where a cascade function (or model) is trained from a large amount of positive and negative images, which can be used to classify other objects in other images.

The implementation of these cascade classifiers are often stored in an xml file. Each xml file have a specific task of identifying a specific object, including but not limited to frontal face, eye, mouth and nose.

C. LBPH Face Recognition

There are three face recognition classifiers available in OpenCV 2: Eigenface, Fisherface, and LBPH (Local Binary Patterns Histograms). While eigenface and fisherface extract features of the training data as a whole, LBPH analyze each image individually, which results in a simpler implementation. Although simple, it is particularly good to create a model of a small amount of training data. This encourages us to use LBPH classifier as our main classifier.

To create an LBPH classifier, we need to summarize the local structure in an image by comparing pixel by pixel, each pixel with its neighbours. LBPH takes a center pixel, then compare its intensity with its neighbour, and extract features

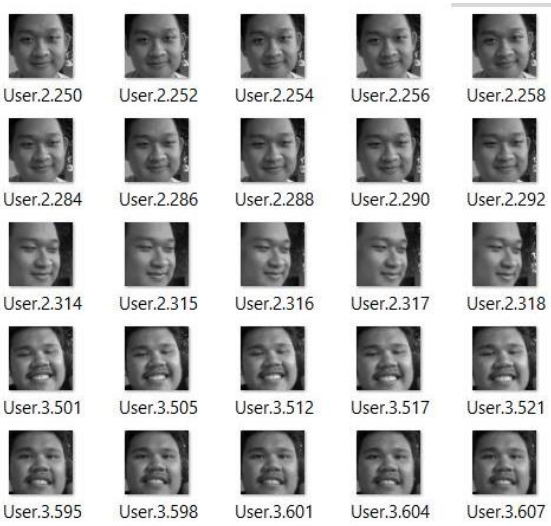
III. IMPLEMENTATION

This section explains thoroughly about the series of actions required to achieve face recognition ability, which has been stated in the introduction.

The full implementation source code can be seen in https://github.com/alsoncahyadi/FaceRecognitionPythonOpenCV.

A. *Generating Datasets*

The datasets are in the form of pairs consisting of an image and an id associated to the image. In order to generate a large amount of this dataset, detected faces on the current frame of webcam will be saved to the dataSets folder, with the id written on the file name each 100 milliseconds continuously, until the user prompt the program to stop. The datasets should consist of one particular person per id, with varying face angles. Below is the datasets we use in this implementation.



The name convention is as such:

User.<user id>.<train datum id>.jpg

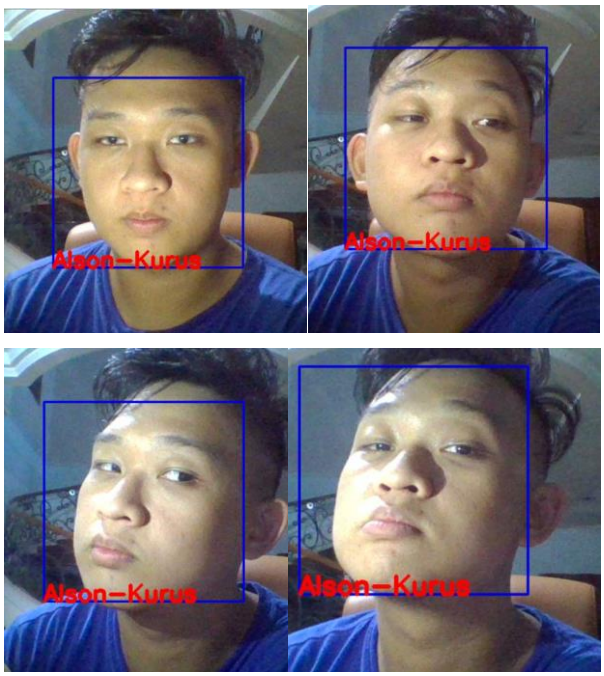
So that each file name would contain the required user id, while having unique names because of the datum id. In this dataset, we use four different user ids, indicating different faces.

B. *Training a Model*

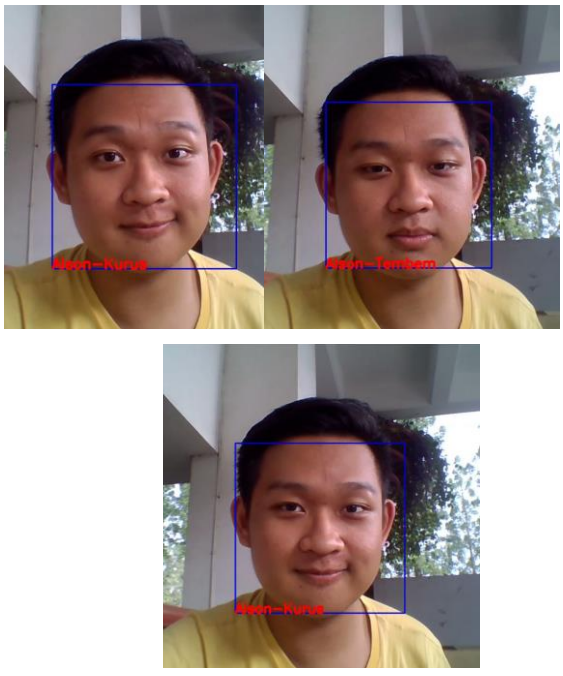
Training a model essentially produces a model that would later be used to classify the images, by using the dataset generated before as inputs, and OpenCV will do the training for us after calling `recognizer.train()` . An LBHP model will be saved in the trainer folder, named "trainer.yml". This file is the model that will later be used to classify the images.

C. *Classifying inputs*

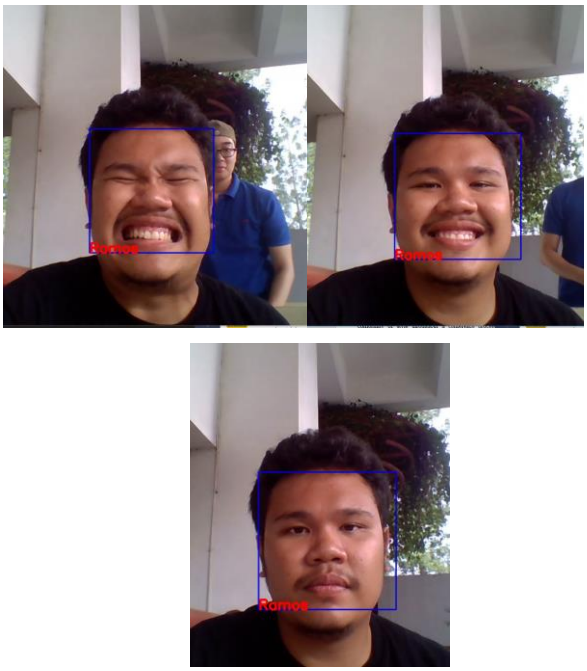
The inputs that need classifying come from the live webcam that is present in our laptop, or using a prerecorded video file. The previously made model is loaded and used to predict the label of the input's label. Predictions with confidence value of less than 80 are considered valid (lower confidence value means higher probability of being right), while confidence value of 80 or more will be labeled "unknown".



The figure above shows how the classifier can still be able to predict the label that corresponds to the given image regardless of different angle and lightning, given a large amount of training data varying in lightning exposure and angle.



The figure above shows a wrongly predicted label. Despite of the correct label being Alson-Tembem, the classifier sometimes still classify the images as Alson-Kurus. This may be the result of the same person being trained as different ids, whilst only differing a little in cheek size (Alson-Tembem being the thicker one, Alson-Kurus being the thinner one).



The figure above shows how the classifier can correctly label faces with different facial expressions, given the large amount of training dataset varying in facial expressions.

IV. Conclusion

Face recognition can be done with three steps: generating training datasets, training a model, and classifying inputs. Implementing face recognition with OpenCV using Python can be done with a relatively high accuracy, although it can use some adjustments and optimizing parameters to ramp the accuracy up.

Acknowledgment

First and foremost, I would like to thank God for His blessings so that I can finish this paper. I also would like to thank my parents for without their support I would never be where I'm at now. Last but not least, I want to express my sincere gratitude to my Socio Informatics lecturers: Dr. Ir. Rinaldi Munir, MT, Dr. Eng. Ayu Purwarianti, ST.,MT, and Dessi Puji Lestari ST,M.Eng.,Ph.D. for without their guidance I would not be motivated to write this paper.

References

[1] https://en.wikipedia.org/wiki/Supervised_learning accessed on May 5th 2017, 10:31 WIB

[2] http://docs.opencv.org/ accessed on May 5th 2017, 10:42 WIB

[3] http://thecodacus.com/ accessed on May 5th 2017, 10:45 WIB

Statement

I hereby declare that this paper is my own work and not a copy, translation, nor plagiarism of somebody else's work.

Bandung, May 5th 2017

Alson Cahyadi