COMP7116 – COMPUTER VISION

REPORT

FACE DETECTION AND RECOGNIZER

Kevin Kurniawan 2101658685

LA08

Master Track of Information Technology

Computer Science

BINUS UNIVERSITY

Daftar Isi

Bab I: Pendahuluan

Penggunaan komputer saat ini merupakan salah satu hal yang paling banyak digunakan dan merupakan salah satu hal yang terpenting untuk dikuasai baik untuk kepentingan sehari-hari seperti hiburan sampai kebutuhan kerja untuk membantu pengguna dalam meringankan pekerjaannya. Perkembangan komputer bertumbuh dengan sangat cepat seiring dengan adanya perlombaan antara perusahaan-perusahaan besar yang terus menerus berlomba untuk meningkatkan kualitas perangkat lunak maupun perangkat keras mereka untuk sebaik mungkin daripada lawan perusahaan mereka. Salah satu bidang yang merupakan bidang yang terus menerus dikembangkan dan diminati oleh banyak orang yaitu bidang *Computer Vision.*

Computer Vision merupakan bidang *Aritificial Intelligence* atau yang biasa dikenal dengan AI yang mencoba untuk meniru cara kerja visual manusia seperti mengenali benda atau objek, wajah, pemandangan, dan lain-lain. Cara kerja visual manusia merupakan hal yang sangat kompleks untuk dipelajar dikarenakan 1/3 bagian dari otak manusia didedikasikan hanyak untuk pengelihatan manusia. *Computer Vision* merupakan teknik-teknik untuk mengestimasi ciri-ciri objek di dalam citra, pengukuran ciri yang berkaitan dengan geometri objek dan menginterpretasi informasi geometri tersebut seperti menentukan posisi objek, dimana posisi horizontal diwakili oleh sumbu X, posisi vertikal diwakili oleh sumbu Y dan jarak dari kamera ke suatu titik objek diwakili oleh sumbu Z yang berada dalam ruang tiga dimensi. Salah satu hasil dari pengembangan bidang *Computer Vision* yaitu *Face Detection and Recognition*, yang merupakan kemampuan komputer untuk mengenali wajah dan dapat menentukan wajah siapa yang dideteksi tersebut. Pada laporan kali ini, akan dijelaskan cara implementasi *Face Detection and Recognizer* pada bahasa pemrograman python dengan bantuan library opencv, os, dan numpy.

Bab II: Metode Pengerjaan

Library yang akan di-import ke dalam python yaitu library opencv, os, dan numpy.

```
import cv2
import os
import numpy as np
```

Berikut adalah alur pengerjaannya, train root path merupakan tempat folder test data dan train data:

```
if __name__ == "__main__":
    '''
        Please modify train_root_path value according to the location of
        your data train root directory

        -------------------
        Modifiable
        -------------------
    '''
    train_root_path = "dataset/train"
    '''
        -------------------
        End of modifiable
        -------------------
    '''

    train_names = get_path_list(train_root_path)
    image_path_list, image_classes_list = get_class_names(train_root_path, train_names)
    train_image_list = get_train_images_data(image_path_list)
    train_face_grays, _, filtered_classes_list = detect_faces_and_filter(train_image_list, image_classes_list)
    classifier = train(train_face_grays, filtered_classes_list)

    '''
        Please modify test_image_path value according to the location of
        your data test root directory

        -------------------
        Modifiable
        -------------------
    '''
    test_root_path = "dataset/test"
    '''
        -------------------
        End of modifiable
        -------------------
    '''

    test_names = get_path_list(test_root_path)
    test_image_list = get_test_images_data(test_root_path, test_names)
    test_faces_gray, test_faces_rects, _ = detect_faces_and_filter(test_image_list)
    predict_results = predict(classifier, test_faces_gray)
    predicted_test_image_list = draw_prediction_results(predict_results, test_image_list, test_faces_rects, train_names)
    final_image_result = combine_results(predicted_test_image_list)
    show_result(final_image_result)
```

Cara mengambil nama-nama folder dari tempat root:

```
def get_path_list(root_path):
    '''
        To get a list of path directories from root path

        Parameters
        ----------
        root_path : str
            Location of root directory

        Returns
        -------
        list
            List containing the names of the sub-directories in the
            root directory
    '''
    path_list = []
    for paths in os.listdir(root_path):
        path_list.append(paths)
    return path_list
```

Cara mengambil nama kelas/kategori foto yang akan di train/latih:

```
def get_class_names(root_path, train_names):
    '''
        To get a list of train images path and a list of image classes id

        Parameters
        ----------
        root_path : str
            Location of images root directory
        train_names : list
            List containing the names of the train sub-directories

        Returns
        -------
        list
            List containing all image paths in the train directories
        list
            List containing all image classes id
    '''
    image_path = []
    class_list = []

    for index, class_name in enumerate(train_names):
        full_path = root_path + '/' + class_name
        for images in os.listdir(full_path):
            full_image_path = full_path + '/' + images
            image_path.append(full_image_path)
            class_list.append(index)

    return image_path, class_list
```

Cara mengambil data untuk di train/latih:

```
def get_train_images_data(image_path_list):
    '''
        To load a list of train images from given path list

        Parameters
        ----------
        image_path_list : list
            List containing all image paths in the train directories

        Returns
        -------
        list
            List containing all loaded train images
    '''
    image_list = []

    for images in image_path_list:
        img = cv2.imread(images)
        image_list.append(img)

    return image_list
```

Mendeteksi wajah dengan bantuan ekstensi xml "haarcascade_frontalface_default" yang berisikan training fitur-fitur wajah, jika dalam foto tersebut terdapat lebih atau kurang dari 1 wajah, maka akan di-skip / dilewati:

```
def detect_faces_and_filter(image_list, image_classes_list=None):
    '''
        To detect a face from given image list and filter it if the face on
        the given image is more or less than one

        Parameters
        ----------
        image_list : list
            List containing all loaded images
        image_classes_list : list, optional
            List containing all image classes id

        Returns
        -------
        list
            List containing all filtered and cropped face images in grayscale
        list
            List containing all filtered faces location saved in rectangle
        list
            List containing all filtered image classes id
    '''
    image_grayscale = []
    image_rectangle = []
    image_classes = []

    face_cascade = cv2.CascadeClassifier("haarcascade_frontalface_default.xml")

    for index, images in enumerate(image_list):
        img_gray = cv2.cvtColor(images, cv2.COLOR_BGR2GRAY)
        detected_faces = face_cascade.detectMultiScale(img_gray, 1.2, 5)
        if (len(detected_faces) > 1 or len(detected_faces) < 1):
            continue

        for x,y,w,h in detected_faces:
            img_face = img_gray[y : y + h, x : x + w]
            image_grayscale.append(img_face)
            image_rectangle.append([x,y,w,h])

            if image_classes_list != None:
                image_classes.append(image_classes_list[index])

    return image_grayscale, image_rectangle, image_classes
```

Melatih data-data yang telah ditampung:

```
def train(train_face_grays, image_classes_list):
    '''
        To create and train classifier object

        Parameters
        ----------
        train_face_grays : list
            List containing all filtered and cropped face images in grayscale
        image_classes_list : list
            List containing all filtered image classes id

        Returns
        -------
        object
            Classifier object after being trained with cropped face images
    '''
    training = cv2.face.LBPHFaceRecognizer_create()
    training.train(train_face_grays, np.array(image_classes_list))
    return training
```

Cara mengambil test data:

```
def get_test_images_data(test_root_path, image_path_list):
    '''
        To load a list of test images from given path list

        Parameters
        ----------
        test_root_path : str
            Location of images root directory
        image_path_list : list
            List containing all image paths in the test directories

        Returns
        -------
        list
            List containing all loaded test images
    '''
    image_list = []

    for images in image_path_list:
        image = cv2.imread(test_root_path + '/' + images)
        image_list.append(image)
    return image_list
```

Memprediksi test data tersebut merupakan class/kategori yang mana dari data yang telah dilatih:

```
def predict(classifier, test_faces_gray):
    '''
        To predict the test image with classifier

        Parameters
        ----------
        classifier : object
            Classifier object after being trained with cropped face images
        train_face_grays : list
            List containing all filtered and cropped face images in grayscale

        Returns
        -------
        list
            List containing all prediction results from given test faces
    '''
    prediction = []
    for images in test_faces_gray:
        img_class, _ = classifier.predict(images)
        prediction.append(img_class)

    return prediction
```

Menggambar kotak hijau disekitar wajah yang telah dideteksi dan juga menambahkan teks tentang class/kategori paling cocok untuk wajah yang terdeteksi tersebut:

```
def draw_prediction_results(predict_results, test_image_list, test_faces_rects, train_names):
    '''
        To draw prediction results on the given test images

        Parameters
        ----------
        predict_results : list
            List containing all prediction results from given test faces
        test_image_list : list
            List containing all loaded test images
        test_faces_rects : list
            List containing all filtered faces location saved in rectangle
        train_names : list
            List containing the names of the train sub-directories

        Returns
        -------
        list
            List containing all test images after being drawn with
            prediction result
    '''
    result = []
    for index, images in enumerate(test_image_list):
        prediction = predict_results[index]
        count = -1
        for x,y,w,h in test_faces_rects:
            count += 1
            if count != index:
                continue
            cv2.rectangle(images, (x,y), (x+w, y+h), (0,255,0), 2)
            text = train_names[prediction]
            cv2.putText(images, text, (x, y), cv2.FONT_HERSHEY_PLAIN, 1, (0,255,0, 2))

        result.append(images)
    return result
```

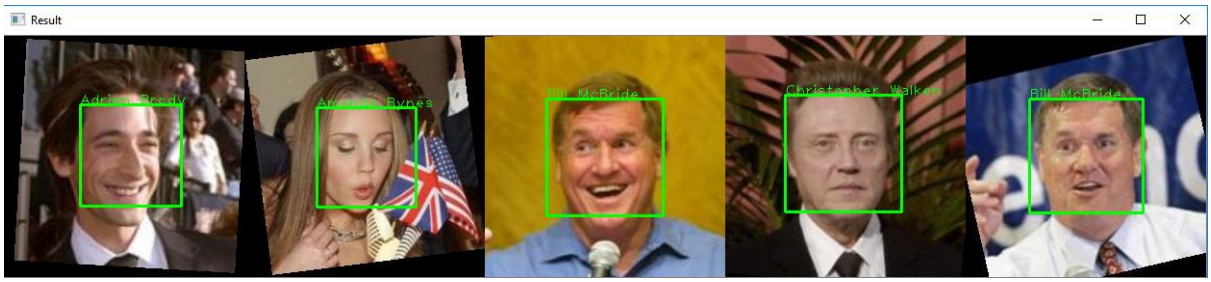
Mengkombinasikan gambar-gambar yang telah ditambahkan kotak tersebut agar dapat ditampilkan dalam 1 window:

```
def combine_results(predicted_test_image_list):
    '''
        To combine all predicted test image result into one image

        Parameters
        ----------
        predicted_test_image_list : list
            List containing all test images after being drawn with
            prediction result

        Returns
        -------
        ndarray
            Array containing image data after being combined
    '''
    combined = np.hstack(tuple(predicted_test_image_list))
    return combined
```

Menampilkan gambar:

```
def show_result(image):
    '''
        To show the given image

        Parameters
        ----------
        image : ndarray
            Array containing image data
    '''
    cv2.imshow("Result", image)
    cv2.waitKey(0)
```

Hasil akhir:



Untuk training data, lebih banyak data yang masuk untuk di train, maka akan lebih akurat *face recognizer* untuk test data.

Daftar Pustaka

http://maulanagilbert.blogspot.com/2013/11/penerapan-computer-vision-untuk.html

https://github.com/opencv/opencv/blob/master/data/haarcascades/haarcascade_frontalface_default.xml