Getting Face recognition to work

1. Introduction

For face recognition using Turtlebot's Orbbec Astra pro camera, we found **face_recogniton** package (https://github.com/procrob/face_recognition) to be the best among the pool of readily available face recognition packages. Before cloning the package, make sure to have Catkin workspace in your home directory. If you haven't created a Catkin workspace yet, please follow the instructions from the link below:

Creating a Catkin WS - http://wiki.ros.org/catkin/Tutorials/create_a_workspace

Once you have your Catkin workspace ready, you can proceed on to cloning the face_recognition package as explained in the following section.

2. Installing the face_recognition package

Type the following commands in a new terminal.

```
$ cd ~/catkin_ws/src
$ git clone https://github.com/procrob/procrob_functional.git --branch catkin
$ cd ~/catkin_ws
$ catkin_make
$ source ~/catkin_ws/devel/setup.bash
```

After catkin_make, you should have the executables generated without any errors. You might end up with a couple of warnings though, which, isn't a problem.

3. Making the necessary changes in the package

- Go to **catkin_ws → src → procob_functional → src → face_recognition.cpp**
- In **line 84**, change the first argument in it_.subscribe() from "/camera/image_raw" to "/astra_usb_cam/image_raw". (This takes RGB input from AstraPro Camera of turtlebot).
- Change the confidence score in **line 57** of the code from 0.88 to some value in the range of 0.72 to 0.75. This will account for the varying lighting conditions.

4. Tutorial for running the face_recognition package

4.1. Adding a new face into database

Let's start with adding a new person's images into database. Follow the following steps in sequence and you should be able to add face images of a new person without any problem.

Source the **setup.bash** file.

```
$ source ~/catkin_ws/devel/setup.bash
```

Run the Fserver executable file in the same terminal where you sourced the setup.bash file.

```
$ rosrun face_recognition Fserver
```

Open a new terminal tab and again source the setup.bash file.

```
$ source ~/catkin_ws/devel/setup.bash
```

Now, run the Fclient executable in this terminal.

```
$ rosrun face_recognition Fclient
```

Open a new terminal tab and again source the setup.bash file.

```
$ source ~/catkin_ws/devel/setup.bash
```

In this tab, type in the following command

```
$ rostopic pub -1 /fr_order face_recognition/FRClientGoal -- 2 "your_name"
```

Running the above command, an image window would pop up and the code starts looking for a face. Make sure that you're the only person in front of the camera and are facing the camera. Try to change expressions for different frames.

Once the process stops, type in the following command to end the process of acquiring new images for database. Also perform a **Ctrl + C** operation in Fclient terminal.

```
$ rostopic pub -1 /fr_order face_recognition/FRClientGoal -- 4 "none"
```

To add another person into database, repeat the whole process. But you don't have to source the **setup.bash** file from now. Also, you can take a look into data folder (catkin_ws → src → procob_functional→ data) to see if your face images are added.

4.2. Recognizing continuously

Note: If you just added the images of a new face into data folder, make sure you delete the **facedata.xml** file in your procob_functional folder before proceeding with the following process.

Source the **setup.bash** file.

```
$ source ~/catkin_ws/devel/setup.bash
```

Run the Fserver executable file in the same terminal where you sourced the setup.bash file.

```
$ rosrun face_recognition Fserver
```

Open a new terminal tab and again source the setup.bash file.

```
$ source ~/catkin_ws/devel/setup.bash
```

Now, run the Fclient executable in this terminal.

```
$ rosrun face_recognition Fclient
```

Open a new terminal tab and again source the setup.bash file.

```
$ source ~/catkin_ws/devel/setup.bash
```

In this tab, type in the following command

```
$ rostopic pub -1 /fr_order face_recognition/FRClientGoal -- 1 "none"
```

Running the above command, an image window would pop up and the code starts looking for a face. If the face detected matches with the database images, it would provide the name of the person on the frame.

This will keep running until you do a **Ctrl+ C** or you pass the following command.

```
$ rostopic pub -1 /fr_order face_recognition/FRClientGoal -- 4 "none"
```

5. Fserver and Fclient

Fserver is a ROS node that provides a simple actionlib server interface for performing different face recognition functionalities in video stream.

Fclient is a ROS node that implements an actionlib client example for the face_recognition simple actionlib server (i.e. 'Fserver'). 'Fclient' is provided for demonstration and testing purposes.

SAI PRAJWAL KOTAMRAJU .

DEPT. OF ELECTRICAL ENGINEERING

ARIZONA STATE UNIVERSITY, TEMPE