NAO Face Recognition - Documentation

Project documentation for the course
Programming Intelligent Applications
at:
Stuttgart Media University,
08.02.2017

by:
Ina De Marco
Lisa Möcking
Yann Philippczyk

1 Introduction

This document serves as documentation and manual for the “Nao face recognition” package (from here on abbreviated as NFR), implemented as a project for the course “Programming Intelligent Applications”. It describes an implementation of the OpenFace framework (https://cmusatyalab.github.io/openface/ ; by Brandon Amos et al.) for usage on Nao robots from Aldebaran Robotics. The source code for NFR can be found at https://github.com/naofaces/nao_face_recognition .

Part 2 of this document describes prerequisites and installation of the software, part 3 covers the usage for face recognition with the robot, and part 4 describes details of the implemented code base.
The content of this document refers to the version of the Nao face recognition implementation and the state of requirements and packages at the time of writing.

Usage of the entirety of the software and the documentation is at your own risk. There is no guarantee for correctness, safety, or any other particular functionality.

2 Installation

2.1 Prerequisites

The entire code for NFR is written in Python 2.7 and was run and tested on Ubuntu 14.04. Both OpenFace and the NAOqi Framework are required for running NFR. For details, see the following sections. Installing OpenFace will require some experience in building and installing applications on Linux, as stated on the OpenFace website.

2.2 Python

A base Python 2.7 installation, as well as the Scipy stack, PyQt and certain additional modules are needed. Refer to the source code for the exact modules required, especially *face_recognition_gui.py* and *face_recognition_methods.py*.
The required modules can be installed by using apt-get, pip, or any other suitable packaging tool.

2.3 OpenFace

For installing the OpenFace framework, follow the instructions at: https://cmusatyalab.github.io/openface/setup/
A local installation is required, i.e. follow the instructions described in the “By Hand” section of the webpage linked above. OpenFace will further require the installation of OpenCV, dlib and Torch as detailed on the page. Since the installation process is dependent on the exact system you run, refer to the webpages of the respective frameworks for further instructions.

In order to complete this step and test if everything was installed correctly, run the Real-time Web Demo at https://cmusatyalab.github.io/openface/demo-1-web/
Take note of the additional “Manual Setup” steps described on the bottom of the page linked above.

2.4 NAOqi

The NAOqi framework must be downloaded from the Aldebaran website at https://community.aldebaran.com/en/resources/software (requires a log-in). For NFR, only the pynaoqi-package (for Linux 64 bit) itself is required. You may extract the package to any location. The directory path to the package will be required in the configuration file *face_recognition_config.py*, for details see the respective section. No further setup steps are required for NAOqi.

2.5 NFR

The NFR source code can be found at https://github.com/naofaces/nao_face_recognition within the *facerecog_scripts* package (*testscripts* is not required). Download/clone the package to any location of your system, from which you are able to execute scripts from. The final step for setting up NFR is to edit the file *face_recognition_config.py* . Open it with any suitable text editor and replace the values for the variables *ip*, *naoqi_root* and *openface_root* with the IP-address of your Nao robot, the path to your local pynaoqi-package as well as to your OpenFace installation directory, respectively.

3 Usage

3.1 General Usage and Training

After booting up the NAO robot and connecting to it via Ethernet or Wifi, make sure that the correct IP-address to the robot is set in *face_recognition_config.py*, as described in section 2.5.
The implementation can then be started by launching the script *face_recognition_gui.py*, which will automatically connect to the robot and initialize the user interface. In order to do so, open a terminal, navigate to the directory containing the script, and execute the following command:

python face_recognition_gui.py

You might first have to allow the execution of the script with this command:

sudo chmod +x face_recognition_gui.py

After launching the script, a live stream from Nao's camera will be displayed, with a bounding box around any detected face, probably labelled with "unknown".
In order to train a person, first enter a name in the text field next to the "Add person" button, then press the button. Now, the name you entered will be added to the list of known names. To start the actual training process, press the button "Turn training on". Nao will now take pictures (no bounding boxes will be displayed in the meantime), which are used to train the neural network and the support vector machine to recognize the person whose name was last added. Only one person can be trained at a time. For best results, only the person to be trained should be within the frame. It is recommended to let the script take pictures of the face from varying angles, distances and facial expressions. The number of pictures taken will be shown in the lower text field (which is only used for output) About 20-30 pictures are enough, the recognition may work on less as well and is robust against lighting conditions. However no exact tests for NFR's performance have been conducted at the time of writing.

To complete the training, press "Turn training off". The person trained can now be recognized and will be marked with a bounding box and the name. In order to train additional persons, repeat the process exactly as described.

Take note that only one single face can be recognized in a given frame, currently. Furthermore, the recognition algorithm will try to match the closest trained identity to any detected face, i.e. a person, which was not trained yet, will not be marked as "unknown", but with the name of any of the trained identities.

3.2 Greeting mode

For demonstration purposes, a greeting mode has been implemented, which can be activated by pressing “Turn greetings on”. When in greeting mode, the Nao robot will initially greet any recognized person with a short spoken sentence at the moment of first recognition. The exact greeting is randomly chosen from a list (see the respective subsection in section 4 of this documentation for details). If the same person is recognized repeatedly after a certain amount of time, Nao will again say a sentence each time, this time chosen randomly from another list for this purpose. Pressing “Turn greetings off” will deactivate the greeting mode, without completely resetting it, i.e. persons, which have already been greeted initially, will be greeted with repeated sentences when greeting mode is reactivated.

3.3 Tips

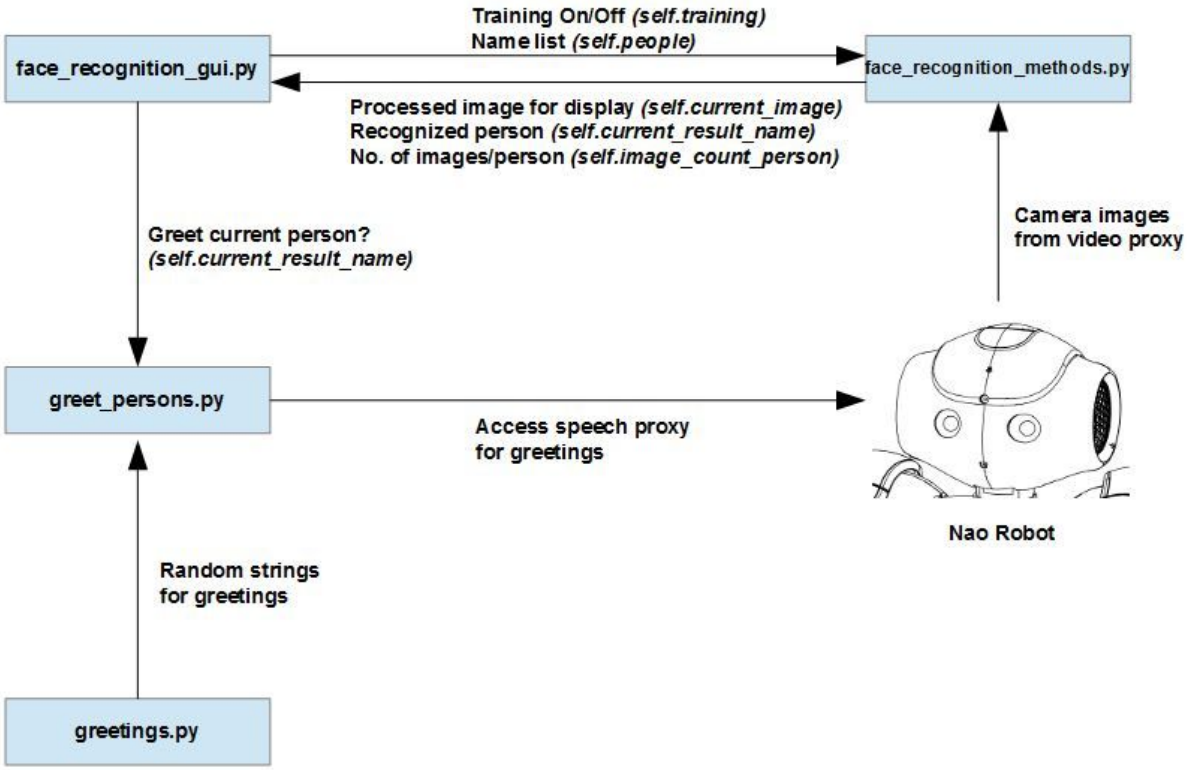
- While the training is robust against lighting conditions, background clutter and somewhat against distance, tilting the head to roughly 45° degrees or more sideways or turning it too much (esp. if one eye is out of view) will prevent face detection
- Similarly, the eyes, nose area and lower parts of the face should not be obstructed for training/good recognition performance
- It is possible to take additional training pictures for the **last** person added to the name list, however not for previous ones. Contrary to the OpenFace Real-time Web demo, pictures taken can not be reassigned to other trained identities at the moment.
- Nao’s video proxy is prone to crashing. If this happens, no pictures can be retrieved from the cameras. In this case, a failure image will be displayed in the user interface and no training/recognition is possible. The fastest way to restore proper function is to close the user interface, and reboot the robot from its local webpage (can be accessed via *nao.local* in the browser, login information is required). After rebooting, NFR should be able to start normally.
- The robot’s voice volume can be set on the robot webpage as well.
- While greeting a person, the user interface will be blocked.

4 Implementation Details

4.1 Overview

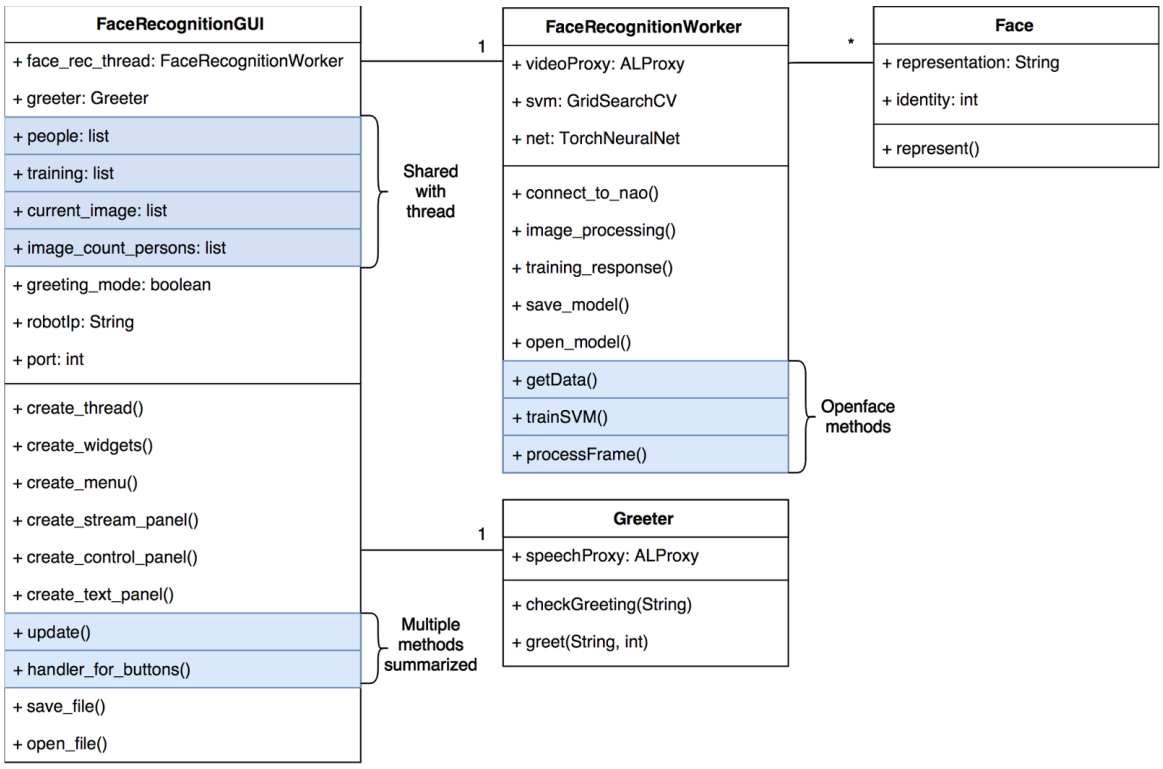
This section will provide a general description of the architecture and dataflows of NFR, as well as some implementation details and design choices, which might be not immediately clear. For additional details, refer directly to the source code and the respective comments.

The following diagram shows the dataflows between the NFR Python modules (the variable names holding the respective data is shown in italics):



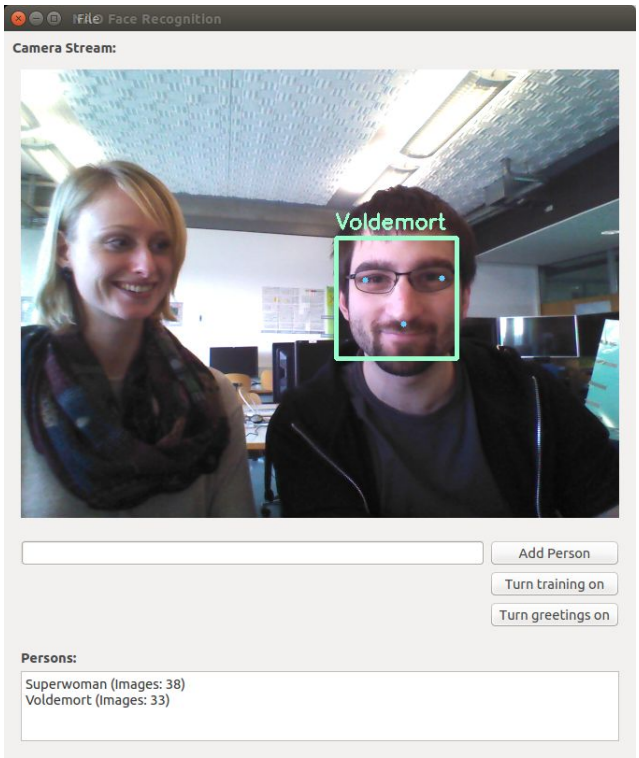
The further descriptions of the modules can be found in the subsections below. The *face_recognition_config.py* is accessed by several modules, but is not shown in the diagram to increase readability, as it only provides static configuration parameters during start-up.

The following UML class diagram shows an overview of all methods and variables of each class:



4.2 face_recognition_gui.py

Contains the code for creating the user interface and initializing all components of the application, including the image processing thread and the greeting module, as well as the code for file saving and opening dialogues.
The following image shows the user interface, including information about two trained persons in the lower text field:



This module holds certain variables which are shared with the image processing thread (for example the list with trained persons). These variables are commented in the code as such. Lists had to be used for holding the shared variables, because in Python, lists are a mutable data type, while e.g. strings are not. As such, the lists work as references to the respective variables.

4.3 face_recognition_methods.py

Contains methods to train the recognition and forward images to the artificial neural network of OpenFace. The module is started as a worker thread of the GUI. However, due to the Global Interpreter Lock, the thread still blocks the user interface temporarily.
The module establishes the connection to the Nao video proxy to retrieve the images. Names of persons are kept in a list shared with the python GUI.

In training mode, the last person in this list is the one for which training is performed. In recognition mode, images are forwarded to OpenFace to recognize persons on the current

image retrieved from NAO. After detection and recognition of faces, green rectangles marking the bounding boxes around the faces, as well as the respective names are drawn.

The actual saving and loading of trained models, including needed information, e.g. names and number of images, is done in the thread as well. Serialization of the SVM object and the name list is performed with the Pickle module. The saving process creates a pickle file, which should use the extension **.p*.

4.4 face_recognition_config.py

Contains required paths and addresses for the application, as described in subsection 2.5. This script is supposed to be modified during the installation by the user.

4.5 greet_persons.py

Contains the code for the greeting mode. Time delays for initial and between repeated greetings can be adjusted with the variables *initial_time_gap, repeated_time_gap* and *global_time_gap* in seconds. The class is initialized during startup and will open a connection to the robot's text to speech proxy for greetings.
check_greeting is supposed to be called repeatedly and will check, whether the person recognized at the given time was already greeted and when.
greet is called when a person is supposed to be greeted and will randomly generate a greeting, which is then spoken by the robot.

4.6 greetings.py

Contains the two lists of string from which greetings, which Nao should speak, are randomly chosen. The lists can be arbitrarily extended with additional strings to be used for greetings, *%s* will hold the name of the person to be greeted at the given moment.