

Implementing The Face Detection Authentication in the Nao Robot

Documentation

Kristi Prifti

Krutarth G. Patel

Hue Do

Hao Loi (Faculty Sponsor)

Department of Computer Science, Quinsigamond Community College

This project consists of **two different methods** of face recognition using nao's built-in cameras.

1. Method A: Using face recognition python library
2. Method B: Using build-in face recognition

Both methods are efficient. Method A is more accurate but slower. Method B is faster but less accurate.

Table of Contents

Implementing The Face Detection Authentication in the Nao Robot	1
Documentation	1
Method A	3
How to install the face_recognition library	3
Features.....	3
How does face_recognition library works.....	5
Finding all the faces.....	5
Posing and Projecting Faces	6
Encoding Faces	7
Needed libraries	8
Get_encoded_faces():	8
Unknow_image_encoded(img):	9
Classify_face(im)	9
Posting data using request library	12
Mapping.....	12
Modifys_the_start_attedance_time.py	13
This program takes a picture using naos cameras	13
How to run face recognition module.....	16
Common issues.....	17
Example of face recognition module	18
Work Cited :	19
INDEX	21
Implementing the Face Detection Authentication in the NAO Robot	22
Introduction to Choregraphe	23
What is Choregraphe?	23
Connecting the NAO robot to Choregraphe	23
Main Panels of Choregraphe:.....	24
Make the NAO learn face of any person	24
Face recognition Using Choregraphe	26
Work Cited	28
Research paper (part of documentation)	29
Attendance Website	29
Table shows database attendance record (present mark as 1, absent masked as 0 in attendance column; late masked as 1, on time masked as 0 in the late column).....	30

Method A

The face recognition library was written by [Adam Geitgey](https://github.com/ageitgey/face_recognition)
https://github.com/ageitgey/face_recognition

Built using [dlib](#)'s state-of-the-art face recognition built with deep learning. The model has an accuracy of 99.38% on the [Labeled Faces in the Wild](#) benchmark. (Ageitgey)

Face_recognition is very powerful when it comes to learning and matching faces.

How to install the face_recognition library

```
pip3 install face_recognition
```

```
pip3 install cmake
```

```
pip3 install dlib
```

```
pip3 install numpy
```

```
pip3 install opencv-python
```

Features

Find all the faces that appear in a picture:

```
import face_recognition
image = face_recognition.load_image_file("your_file.jpg")
face_locations = face_recognition.face_locations(image)
```



Input



Output (Ageitgey)

Identify faces in pictures

Recognize who appears in each photo.

```
import face_recognition
known_image = face_recognition.load_image_file("biden.jpg")
unknown_image = face_recognition.load_image_file("unknown.jpg")

biden_encoding = face_recognition.face_encodings(known_image)[0]
unknown_encoding = face_recognition.face_encodings(unknown_image)[0]

results = face_recognition.compare_faces([biden_encoding], unknown_encoding)
```



Input



**Picture contains
"Joe Biden"**

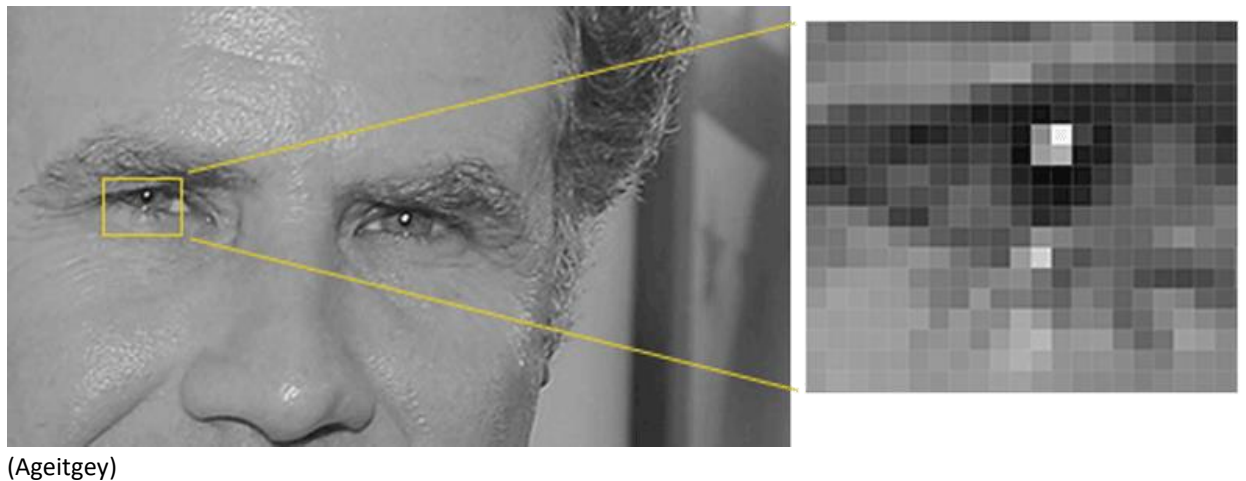
Output

(Ageitgey)

How does face_recognition library works

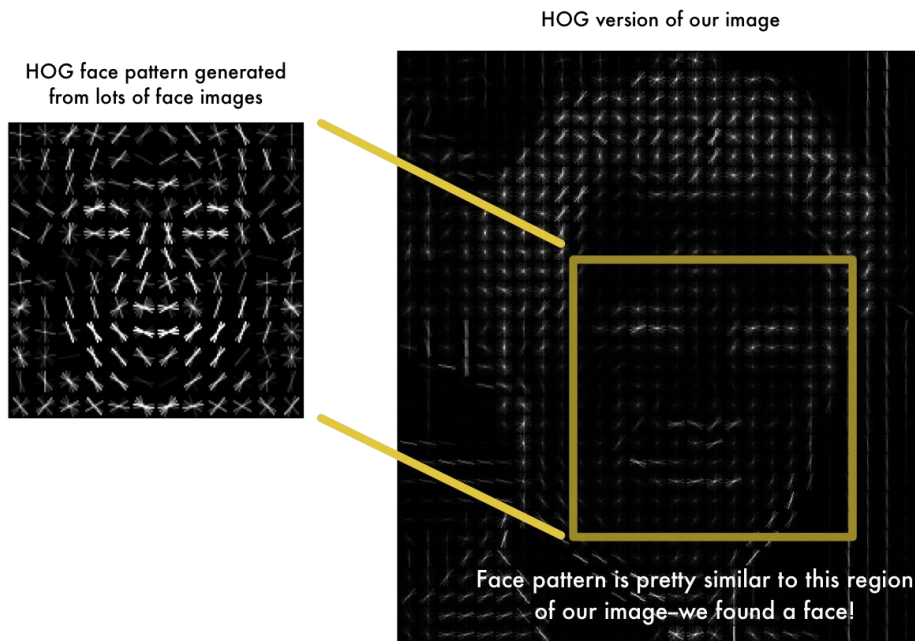
Finding all the faces

- a. To find faces in an image, face_recognition library starts by making our image black and white because we don't need color data to find faces:
- b. Then we'll look at every single pixel in our image one at a time. For every single pixel, we want to look at the pixels that directly surrounding it:
- c. Our goal is to figure out how dark the current pixel is compared to the pixels directly surrounding it. Then we want to draw an arrow showing in which direction the image is getting darker:



(Ageitgey)

- d. If you repeat that process for every single pixel in the image, you end up with every pixel being replaced by an arrow. These arrows are called gradients and they show the flow from light to dark across the entire image:
- e. The end result is we turn the original image into a very simple representation that captures the basic structure of a face in a simple way:
- f. To find faces in this HOG image, all we have to do is find the part of our image that looks the most similar to a known HOG pattern that was extracted from a bunch of other training faces: Using this technique, we can now easily find faces in any image:

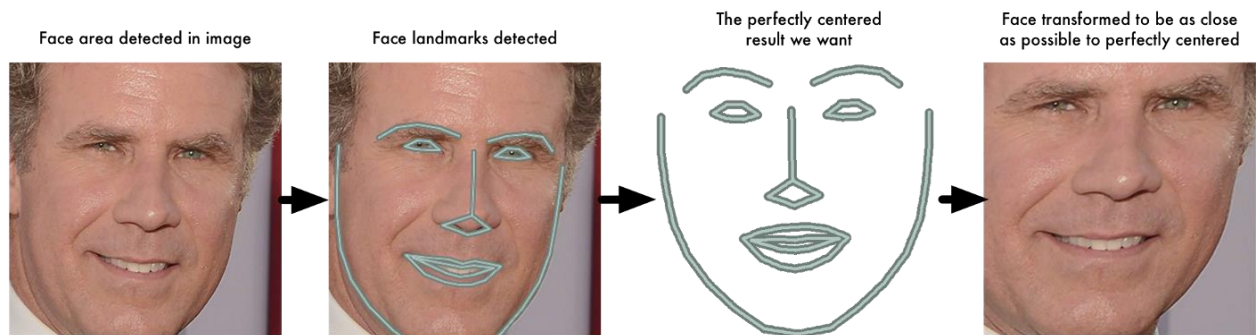


g.

(Ageitgey)

Posing and Projecting Faces

- To do this, the face_ recognition library is going to use an algorithm called **face landmark estimation**. There are lots of ways to do this, but we are going to use the approach [invented in 2014 by Vahid Kazemi and Josephine Sullivan](#). (Ageitgey)
- The basic idea is we will come up with 68 specific points (called *landmarks*) that exist on every face — the top of the chin, the outside edge of each eye, the inner edge of each eyebrow, etc. Then we will train a machine-learning algorithm to be able to find these 68 specific points on any face(Ageitgey)

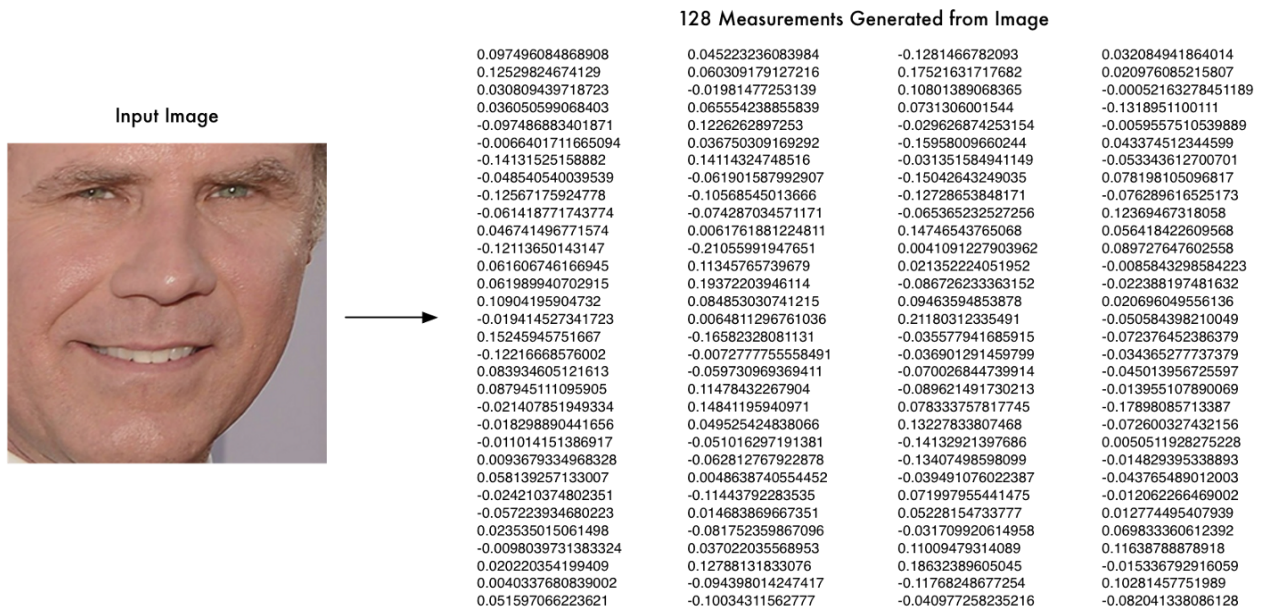


c.

(Ageitgey)

Encoding Faces

- a. So all we need to do ourselves is run our face images through their pre-trained network to get the 128 measurements for each face. Here's an example measurements:



(Ageitgey)

Integrating face recognition in [Face Recognition Module.py](#)

Needed libraries

```
import face_recognition as fr
import os
import cv2
import face_recognition
import numpy as np
from time import sleep
import datetime
import time
import sys
import subprocess
import requests
from bs4 import BeautifulSoup
```

Get_encoded_faces():

```
def get_encoded_faces():
    """
    looks through the faces folder and encodes all
    the faces

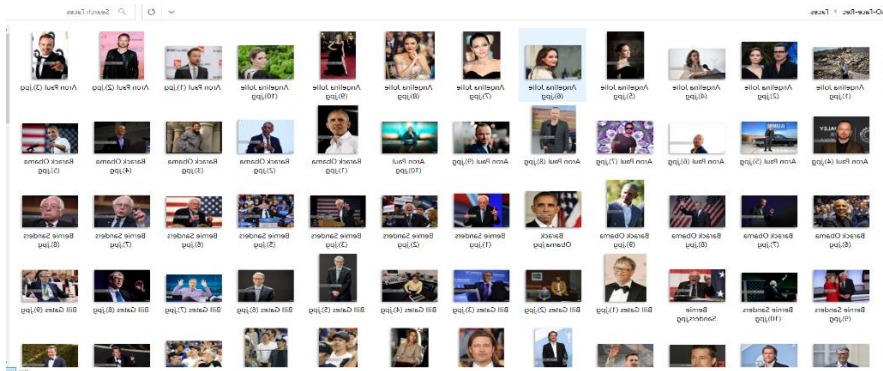
    :return: dict of (name, image encoded)
    """
    encoded = {}

    for dirpath, dnames, fnames in os.walk("./faces"):
        for f in fnames:
            if f.endswith(".jpg") or f.endswith(".png"):
                face = fr.load_image_file("faces/" + f)
                encoding = fr.face_encodings(face)
                encoded[f.split(".")[0]] = encoding

    return encoded
```


The function `get_encoded_faces()` reads through all the faces in the faces folder and it encodes all faces into a format that the machine learning model inside the `face_recognition` module can use to detect the faces.

It's essential to name the faces in the faces folder what they are. It will utilize those labels to name your face if your face matches one of the faces in the faces folder



`Unknown_image_encoded(img):`

```
def unknown_image_encoded(img):  
    """  
    encode a face given the file name  
    """  
    face = fr.load_image_file("faces/" + img)  
    encoding = fr.face_encodings(face)[0]  
  
    return encoding
```

The function `unknown_image_encoded(img)` gets the image given and encodes that image into a format that machine learning module `face_recognition` can read and returns that encoding

`Classify_face(im)`

`classify_face(im)` takes the name of an image and draws a box around the face and perform the face detection and show the face and also is going to return a list of faces.

`faces = get_encoded_faces()` - Start by getting all the encoded faces

`faces_encoded = list(faces.values())` - Turning these face values into a list

`known_face_names = list(faces.keys())` - All the names this is returned as a dictionary

the face is encoded is all the values and the names of all those faces are all of the keys

`img = cv2.imread(im, 1)` - Read the given image

`face_locations = face_recognition.face_locations(img)` - Find all the locations of the

face in the image by passing IMG which is read in by OpenCV

It finds all the locations for our faces.

`unknown_face_encodings = face_recognition.face_encodings(img, face_locations)`

- Encode unknown face (test.jpg)

`matches = face_recognition.compare_faces(faces_encoded, face_encoding)` - Compare

all the faces that we know against the faces in the images and see if any of them are

the same. If they are a box will be drawn and the name of the face. If they are not a box with "Unknown " will be written in the test.png image

```

def classify_face(im)
    """
    will find all of the faces in a given image and label
    them if it knows what they are
    :param im: str of file path
    :return: list of face names
    """

    faces = get_encoded_faces()
    faces_encoded = list(faces.values())
    known_face_names = list(faces.keys())

    img = cv2.imread(im, 1)
    """
    Resize optinal
    """

    #img = cv2.resize(img, (0, 0), fx=0.5, fy=0.5)
    #img = img[:, ::-1]
    face_locations = face_recognition.face_locations(img)
    unknown_face_encodings = face_recognition.face_encodings(img, face_locations)

    face_names = []
    for face_encoding in unknown_face_encodings:
        # See if the face is a match for the known face(s)
        matches = face_recognition.compare_faces(faces_encoded, face_encoding)
        name = "Unknown"

        # use the known face with the smallest distance to the new face
        face_distances = face_recognition.face_distance(faces_encoded, face_encoding)
        best_match_index = np.argmin(face_distances)

```

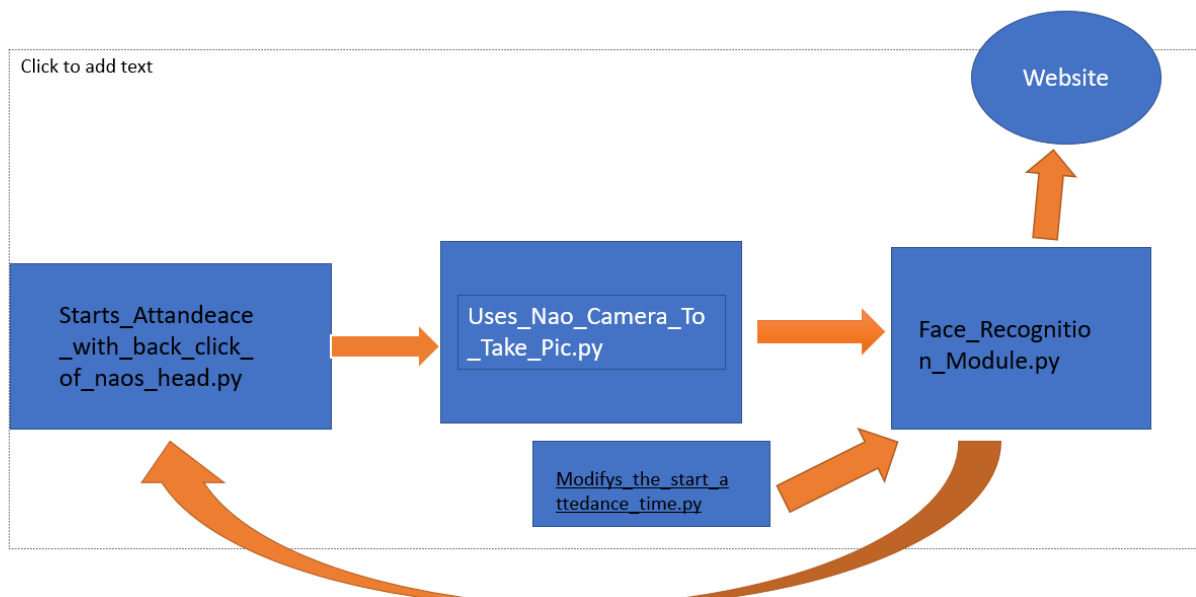
Posting data using request library

```
login_data = {
    'Course': courseid,
    'FirstName': firstname,
    'LastName': lastname,
    'Date': datetime2,
    'Attendance': 'on',
    'Late': latev,
    'submitButton': 'Submit'
}
if(fullname == "Unknow"):
    print("I-dont-know-you")
else:

    with requests.Session() as s:
        url = "https://rbattendance.000webhostapp.com/update.php"
        r = s.get(url)
        soup = BeautifulSoup(r.content, 'html5lib')
        r = s.post(url, data = login_data)
        #print(r.content)
```

Mapping

Files map



The first program that runs is start_attendance. If naos head is touched take_pic.py will be called after take_pic.py Face_recognition_module.py is called. The face_recognition_module.py uses a .txt file to tell when the class starts. The Face_recognition module tries to recognize the face in the photo taken by nao. If it finds a known person it will submit the Name, date, course id, and if the student is late. The name would go back to start_attendance for Nao to say the name and Time.

Modifys_the_start_attendance_time.py

```
dateandtime = input("Please enter date and time in this format ex:(04/11/2020 02:06AM) : \n")
```

```
courseID = input("Please enter courseID ex:(csc212): \n")
```

```
with open('class_starts_time.txt','w+') as f:
```

```
    f.write(dateandtime )
```

```
    f.write(courseID )
```

This simple python script creates a text file that is later used to by face recognition module to get the time class starts and course id.

[Uses Nao Camera To Take Pic.py](#)

This program takes a picture using naos cameras

```
import os
```

```
import qi
```

```
import argparse
```

```
import sys
```

```
import time
```

```
from PIL import Image
```

```

def main(session):
    """
    First get an image, then show it on the screen with PIL.
    """

    # Get the service ALVideoDevice.

    video_service = session.service("ALVideoDevice")
    resolution = 2    # VGA
    colorSpace = 11   # RGB

    videoClient = video_service.subscribe("python_client", resolution, colorSpace,
5)

    t0 = time.time()

    # Get a camera image.
    # image[6] contains the image data passed as an array of ASCII chars.
    naoImage = video_service.getImageRemote(videoClient)

    t1 = time.time()

    # Time the image transfer.
    print("acquisition delay ", t1 - t0)

    video_service.unsubscribe(videoClient)

```

```

# Now we work with the image returned and save it as a PNG using ImageDraw
# package.

# Get the image size and pixel array.
imageWidth = naoImage[0]
imageHeight = naoImage[1]
array = naoImage[6]
image_string = str(bytearray(array))

# Create a PIL Image from our pixel array.

#def imageNo2d(self):
    im = Image.frombytes("RGB", (imageWidth, imageHeight), image_string)#
Save the image.
# image_name_2d = "images/img2d-" + str(self.imageNo2d) + ".png"
    im.save("test1.png", "png") # Stored in images folder in the pwd, if not present
then create one
# self.imageNo2d += 1
    #im.show()
if __name__ == "__main__":
    parser = argparse.ArgumentParser()
    parser.add_argument("--ip", type=str, default="192.168.1.141",
                        help="Robot IP address. On robot or Local Naoqi: use
'192.168.1.141'.")
    parser.add_argument("--port", type=int, default=9559,
                        help="Naoqi port number")

```

```

args = parser.parse_args()
session = qi.Session()
try:
    session.connect("tcp://" + args.ip + ":" + str(args.port))
except RuntimeError:
    print(("Can't connect to Naoqi at ip \"" + args.ip + "\" on port " + str(args.port)
+ ".\n"
        "Please check your script arguments. Run with -h option for help.))
    sys.exit(1)
main(session)

```

How to run face recognition module

1. First install all the libraries in the requirements text file.
2. You can change the date and time the class starts also add the course id
running `Modifys_the_start_attendance_time.py` which will create `class_starts_time.txt`
3. Change Nao's IP in (`Starts_Attandeece_with_back_click_of_naos_head.py`,
`Uses_Nao_Camera_To_Take_Pic.py`)
4. Make sure your face is in the faces folder and is labeled accordingly
5. Run `Starts_Attandeece_with_back_click_of_naos_head.py`

6. Touch Nao's front/middle/ for information. Touch the back of his head to start attendance
7. Depending on how many files are in the faces folder it will take around (1 - 1:30 seconds)
8. Everything should run normally. Nao should say the name and if the student is late.
9. Check the website <https://rbattendance.000webhostapp.com/report.php> if your name was posted.
Your name will only be posted once.

Common issues

Make sure your photo test file is in .png format.

Touching Nao's Bumpers or hands will cause an error

Make sure you are running the right version of python

Face_Recognition_Module.py --- python 3.6

Uses_Nao_Camera_To_Take_Pic.py --- python 2.7

Starts_Attendance_with_back_click_of_nao_head.py --- python 2.7

Example of face recognition module



Work Cited :

Ageitgey, Adam. "Ageitgey/face_recognition." *GitHub*, 20 Feb. 2020, github.com/ageitgey/face_recognition.

Geitgey, Adam. "Machine Learning Is Fun! Part 4: Modern Face Recognition with Deep Learning." *Medium*, Medium, 7 Nov. 2018, medium.com/@ageitgey/machine-learning-is-fun-part-4-modern-face-recognition-with-deep-learning-c3cffc121d78.

Tim, Tech with. *YouTube*, June 2019, youtu.be/D5xqGk6LEc.

A dark blue vertical bar is on the left side of the page. A blue arrow points to the right, starting from the vertical bar and pointing towards the title box.

Developer Handbook for Facial Recognition Using the NAO Robot

Krutarth Patel | Hue Do

Faculty Sponsor: Hao Loi

Department of Computer Science

Quinsigamond Community College

Several thin, curved lines in shades of blue and grey originate from the bottom left corner and curve upwards and to the right, creating a dynamic, abstract design element.

INDEX

Abstract of Our Project	1
Introduction to Choregraphe	3
Make the NAO learn the face of any person	5
Face recognition Using Choregraphe	7

Implementing the Face Detection Authentication in the NAO Robot

The NAO robot has a wide range of applications to automate many of our daily tasks. These robots are used for many different purposes. Shortly, they will become an essential education element. The NAO robot has built-in cameras that could be used to recognize faces. Our project is to use the NAO robot to take the attendance of students of any class using the NAO robot. This project will have two different methods including several steps. First, the NAO robot will learn different faces. These faces will be saved in its memory. Programs based on python language will be created to compare the faces from the memory of the NAO with the faces of the students present in the class. This can be done also by using Choregraphe software. The second method is to compare pictures to recognize faces. The NAO robot will capture a picture of a person and compare it with the pictures existing in our machine. After training NAO, a webpage will be build based on the Blackboard interface. The main purpose of this webpage is to keep the attendance report of all students. These two different methods will take attendance and will store the data on one common webpage. This project will be based on the Python programming language and Choregraphe software. The application of this project will have potential beyond the classroom. Eventually, it could be programmed to use in hotels, hospitals, airports, at work, etc.

Introduction to Choregraphe

What is Choregraphe?

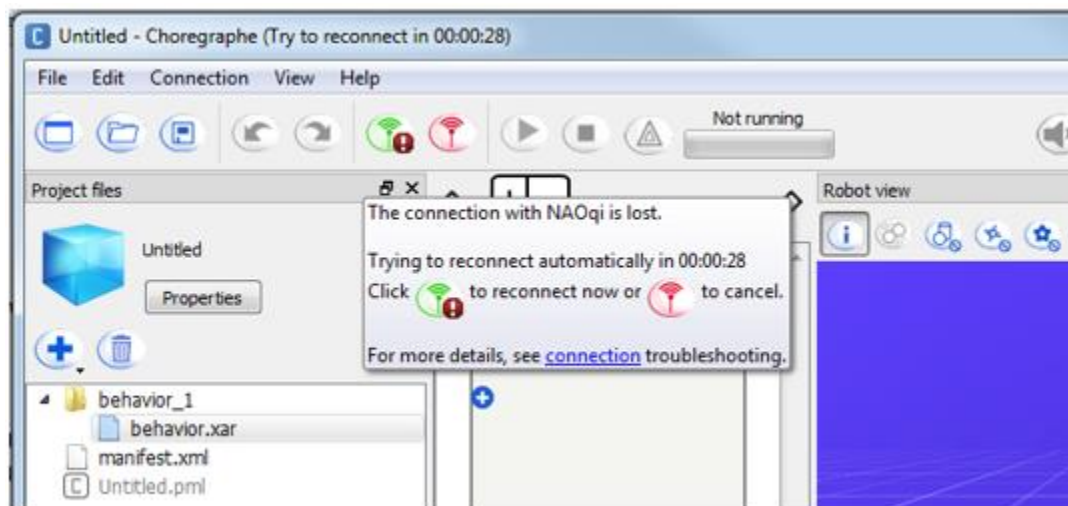
Choregraphe is a multi-purpose desktop software, enabling you to:

- Build animations, operations, and dialogs,
- Examine them on a simulated robot, or a real one,
- Observe and command your robot,
- Decorate Choregraphe behaviors with your Python code.

Choregraphe lets you build applications including Dialogs, assistance, and persuasive behaviors, such as interaction with people, dance, e-mails sending, without writing a single line of code. ¹

Connecting the NAO robot to Choregraphe

To make a connection between your machine with NAO robot, click on the green icon to begin. It will ask user to select the robot if there are multiple. The Connect To panel allows us to create a connection.



If another machine's choregraphe has already connected to the robot, it will not let you connect to it. When the connection is established, the title bar will display the name of your robot.

Main Panels of Choregraphe:

Project files Panel: The Project files panel displays the Project Properties and all the files connected to the current project.

Project objects panel: The Project objects panel showcases the Objects of the current Behaviors.

Box Panel libraries: A box library includes a list of Boxes. A default box library consists of actions and behaviors like speech, movement, animation, sensing, and many more.

Flow Diagram Panel: The Flow diagram panel is the space where you can create NAO's Behaviors.

Pose Library Panel: The Pose library panel presents distinct Timeline boxes including NAO preset positions. It allows you to simply define and locate standard key positions for NAO when Behavior is created.

Robot View: The Robot View presents in a 3-Dimensional view the robot connected. It lets you monitor and change the joint values.

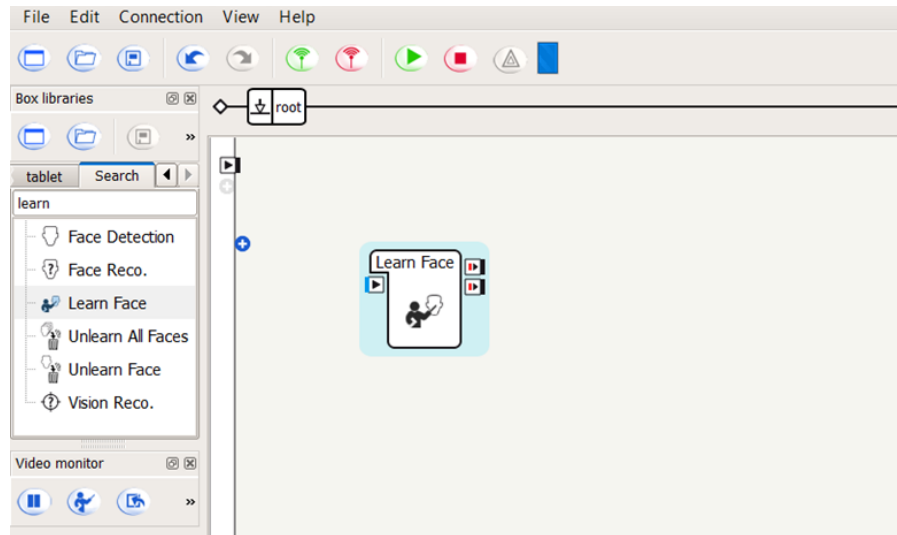
Inspector Panel: The Inspector panel is contextual presenting features of the chosen Object.

For More information you can refer to the [Softbank Robotics Documentation](#).

Make the NAO learn face of any person

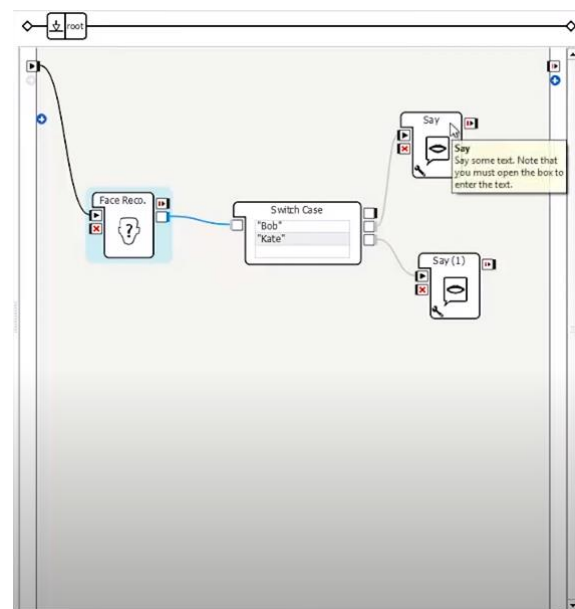
In here, we will provide you a detail explanation on how to program your robot to recognize faces. We will do this by exploring the learn face behavior. Obviously, it is a great tool to be used at places like school, hospitals, etc.

Search for 'Learn Face' box in Box Libraries and drag it to Flow Diagram Panel ².



As we can see, the learn face box have input of blue color. This indicates the here we will have to input a string.

After dragging the box into the flow diagram panel, we simply must run it. After running, double click on the blue color of input button. The function will ask you to input a string. Here, you will have to input the name of the person you are making the face learn. Place the face in such a way that it could be seen in your video monitor. When the light color of the NAO's changes to green, that indicates that the face is saved in its memory and the compilation is passed. If you want to test that the face is saved or not, then you can test it by using face recognition box.



Face recognition Using Choregraphe

After making the NAO robot learn faces, we are moving ahead to perform the face recognition using those faces. We want the NAO robot to say the name of the person every time it recognizes the face. To start with this task, we created a python language script that imports the NAOqi libraries. We also must subscribe to the HumanTracked event raised by ALFacedetection module ² and link it to a callback. This callback function will be called every time when the event will be raised. For example, run the script, bring the face of the person in front of the NAO robot. You should hear the name of this person and the information will be printed in consol.

To begin with the algorithm a class is created to get and rea the FaceDetected events. The class is named as attendancetaker(object). It includes various functions as shown below:

Functions of class attendancetaker(object):

➤ onTouched(strVarName, value):

This function aims to determine whether the robot is touched on its head. It contains Boolean behavior. So, whenever the robot is touched on head, the function will call sss(bodies) function.

➤ sss(bodies):

This function is called within onTouched function. It determines on what part is the head touched. Front, middle, or rear. We can add different speech output to those three buttons. For example, in our code, this buttons are used to greet students.

➤ `__init__(app)`

In this function., we are performing initialization of qi framework and event detection. It gets access to the NAO robot memory. It gets the service `ALMemory`, `ALTextToSpeech`, and `ALFaceDetection`. It connects the data from NAO memory to our program. It calls `OnHumanTracked` function.

➤ `On_Human_Tracked(value):`

When any face is detected, the Boolean variable in this function will change to true With the help of variable passed by `__init__` function, it will match that face array with person's face array. If the match is detected, it will say, print or upload the face information as required. This function also includes the algorithm to upload the face data to Attendance Report Webpage.

➤ `run():`

This function is called whenever user runs the program. It will look for an error. This error could be the user interruption. If it finds any, it will terminate the running program.

➤ Before running the python script, user will have to provide the Internet address of the connected NAO robot in `parser.add_argument()` function call, as a parameter. User can get this address by pressing the round button on the chest of the NAO robot. An object of `attendancetaker` class is introduced with name of `human_greeter`. This object calls `run(0)` function of the class.

Work Cited

1. To get familiar with **Choregraphe** interface, see:

http://doc.aldebaran.com/2-4/software/choregraphe/choregraphe_overview.html

2. ALFacedetection Module Documentation

<http://doc.aldebaran.com/2-1/naoqi/peopleperception/alfacedetection.html>

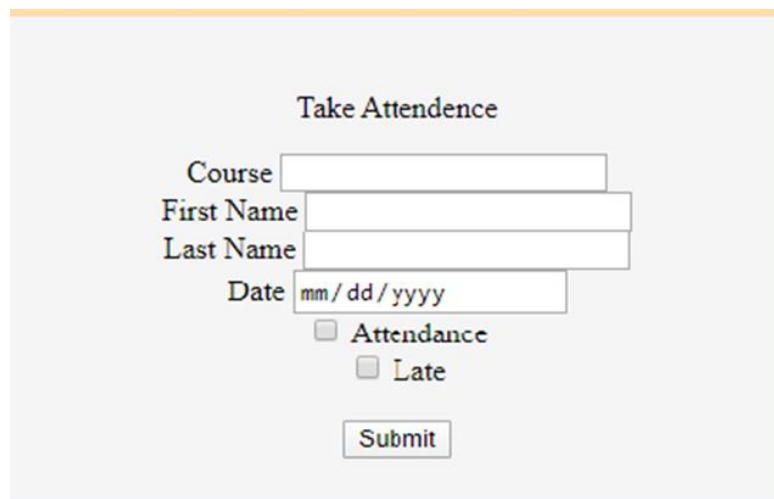
- 3.

Research paper (part of documentation)

In the traditional way to take attendance, the teachers must take attendance by hand, sometimes they wait for students coming for 15 minutes. By using the robot taking pictures of students and take attendance automatically that can save time for the professor, and the class won't be interrupted when students just coming.

Attendance Website

In the website, we simply create a form interface to read and insert data (includes Course, First Name, Last Name, Date and Attendance, and Late checkbox) into the database. After the NAO robot can recognize human faces successfully and give out all the information for taking attendance, we connect to the webpage that is already created using Python and working on library Beautiful Soup to automatically insert the information into the database. In the case, that robot does not work very well and miss some students, or it is out of battery so we just can take attendance by hand with this form below.



The image shows a web form titled "Take Attendance". It contains the following elements:

- A text input field for "Course".
- A text input field for "First Name".
- A text input field for "Last Name".
- A text input field for "Date" with a placeholder "mm/dd/yyyy".
- A checkbox labeled "Attendance".
- A checkbox labeled "Late".
- A "Submit" button.

Before the information can insert into the database, we need to create a database table in the phpMyAdmin. In the database table, we set the type of value for each variable such as course, the name of the student is varchar, and attendance's type is Boolean. By using PHP, we can connect the website with the database table in phpMyAdmin.

We also create a link connected to another page that can take the data from the database and display them on the table.

Course	First Name	Last Name	Date	Attendance	Late
CSC108	Krutarth	Patel	2020-05-02	1	0
CSC211	Kristi	Prifti	2020-05-02	1	1
CSC212	Hue	Do	2020-05-01	1	0

Table shows database attendance record (present mark as 1, absent marked as 0 in attendance column; late marked as 1, on time marked as 0 in the late column).

In the report page, we can search for the value we are looking for; for example, want to check that how many students present on that date, so just input the date in the filter form that will show all the results.

Course	First Name	Last Name	Date	Attendance	Late
CSC108	Krutarth	Patel	2020-05-02	1	0
CSC211	Kristi	Prifti	2020-05-02	1	1

Sort for the date.

Besides that the robot recognizes one student many times and gives out the same information, the same information won't be inserted into the table, or a student that is marked as absent but show up later, it will change that student from absent to present.

(Not sure if it's correct)

