OpenFace Documentation

Release 0.1.1

Carnegie Mellon University

February 28, 2017

Contents

- The code is available on GitHub at cmusatyalab/openface
- The main website is available at http://cmusatyalab.github.io/openface.

Contents:

CHAPTER 1

openface package

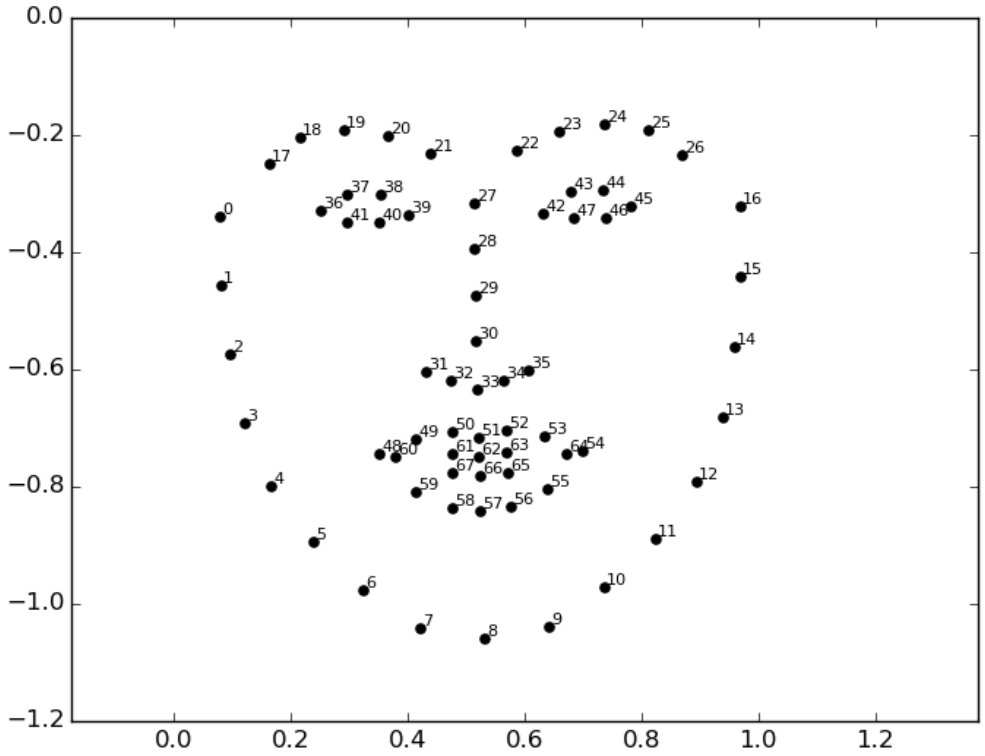
openface.AlignDlib class

class openface.**AlignDlib**(*facePredictor*)

Use dlib's landmark estimation to align faces.

The alignment preprocess faces for input into a neural network. Faces are resized to the same size (such as 96x96) and transformed to make landmarks (such as the eyes and nose) appear at the same location on every image.

Normalized landmarks:



Instantiate an 'AlignDlib' object.

Parameters **facePredictor** (*str*) – The path to dlib's

INNER_EYES_AND_BOTTOM_LIP = [39, 42, 57]
Landmark indices corresponding to the inner eyes and bottom lip.

OUTER_EYES_AND_NOSE = [36, 45, 33]
Landmark indices corresponding to the outer eyes and nose.

align (*imgDim*, *rgbImg*, *bb=None*, *landmarks=None*, *landmarkIndices=INNER_EYES_AND_BOTTOM_LIP*)
Transform and align a face in an image.

Parameters

- **imgDim** (*int*) – The edge length in pixels of the square the image is resized to.
- **rgbImg** (*numpy.ndarray*) – RGB image to process. Shape: (height, width, 3)
- **bb** (*dlib.rectangle*) – Bounding box around the face to align. Defaults to the largest face.

- **landmarks** (*list of (x,y) tuples*) – Detected landmark locations. Landmarks found on *bb* if not provided.
- **landmarkIndices** (*list of ints*) – The indices to transform to.

Returns The aligned RGB image. Shape: (imgDim, imgDim, 3)

Return type numpy.ndarray

findLandmarks(*rgbImg*, *bb*)
Find the landmarks of a face.

Parameters

- **rgbImg** (*numpy.ndarray*) – RGB image to process. Shape: (height, width, 3)
- **bb** (*dlib.rectangle*) – Bounding box around the face to find landmarks for.

Returns Detected landmark locations.

Return type list of (x,y) tuples

getAllFaceBoundingBoxes(*rgbImg*)
Find all face bounding boxes in an image.

Parameters **rgbImg** (*numpy.ndarray*) – RGB image to process. Shape: (height, width, 3)

Returns All face bounding boxes in an image.

Return type dlib.rectangles

getLargestFaceBoundingBox(*rgbImg*)
Find the largest face bounding box in an image.

Parameters **rgbImg** (*numpy.ndarray*) – RGB image to process. Shape: (height, width, 3)

Returns The largest face bounding box in an image, or None.

Return type dlib.rectangle

openface.TorchNeuralNet class

class openface.**TorchNeuralNet**(*self*, *model=defaultModel*, *imgDim=96*, *cuda=False*)
Use a Torch subprocess for feature extraction.

Instantiate a 'TorchNeuralNet' object.

Starts openface_server.lua as a subprocess.

Parameters

- **model** (*str*) – The path to the Torch model to use.
- **imgDim** (*int*) – The edge length of the square input image.
- **cuda** (*bool*) – Flag to use CUDA in the subprocess.

defaultModel = '/home/docs/checkouts/readthedocs.org/user_builds/openface-api/checkouts/stabl
The default Torch model to use.

forward(*rgbImg*)
Perform a forward network pass of an RGB image.

Parameters **rgbImg** (*numpy.ndarray*) – RGB image to process. Shape: (imgDim, imgDim, 3)

Returns Vector of features extracted from the neural network.

Return type numpy.ndarray

forwardPath(*imgPath*)
Perform a forward network pass of an image on disk.

Parameters **imgPath** (*str*) – The path to the image.

Returns Vector of features extracted with the neural network.

Return type numpy.ndarray

openface.data module

Module for image data.

class openface.data.**Image**(*cls*, *name*, *path*)
Object containing image metadata.

Instantiate an 'Image' object.

Parameters

- **cls** (*str*) – The image's class; the name of the person.
- **name** (*str*) – The image's name.
- **path** (*str*) – Path to the image on disk.

getBGR()
Load the image from disk in BGR format.

Returns BGR image. Shape: (height, width, 3)

Return type numpy.ndarray

getRGB()

Load the image from disk in RGB format.

Returns RGB image. Shape: (height, width, 3)

Return type numpy.ndarray

openface.data.**iterImgs**(*directory*)

Iterate through the images in a directory.

The images should be organized in subdirectories named by the image's class (who the person is):

```
$ tree directory
person-1
-- image-1.jpg
-- image-2.png
...
-- image-p.png

...

person-m
-- image-1.png
-- image-2.jpg
...
-- image-q.png
```

Parameters **directory** (*str*) – The directory to iterate through.

Returns An iterator over Image objects.

openface.helper module

OpenFace helper functions.

openface.helper.**mkdirP**(*path*)

Create a directory and don't error if the path already exists.

If the directory already exists, don't do anything.

Parameters **path** (*str*) – The directory to create.

CHAPTER 2

Indices and tables

- genindex
- modindex
- search

O

A

D

F

G

I

M

O

T