pfla Documentation

Release 1.0.0

Maxime Rousseau

May 15, 2020

Contents

CHAPTER 1

Introduction

A simple command line interface to automate facial analysis. `pfla` uses a pre-trained neural networks to detect faces and annotate them with 68 landmarks. The program also compyte four commonly used facial metrics. The output is saved to a file to allow for easy statistical analysis by the user.

CHAPTER 2

Contents

2.1 Installation and Usage

2.1.1 Requirements and Dependencies

- Python 3.5 (or higher)
- Python packages: numpy, pandas, pillow,

2.1.2 Installation

To install with pip:

```
pip install -r requirements-pytorch.txt\ # pytorch for CPU
     -f https://download.pytorch.org/whl/torch_stable.html
pip install -r requirements.txt # other dependencies
pip install pfla
```

2.1.3 Usage

```
usage: pfla [-h] [-d] [-l] [-m] [-o OUTPUT] [-v] path

PFLA: python facial landmark analysis. This program will read the image(s)
given as input and can apply a face detection algorithm, landmark placement
and computation of metrics. The results are returned as a text stream.

positional arguments:
  path                  path to the image or directory of images

optional arguments:
  -h, --help            show this help message and exit
```

(continues on next page)

(continued from previous page)

```
  -d, --detect          detect faces and output bounding box
  -l, --landmark        annotate detected faces and output coordinates
  -m, --metrics         compute metrics and output results
  -o OUTPUT, --output OUTPUT
                        specify output filename and format/filetype of the
                        data
  -v, --verbose         increase output verbosity

AUTHOR: Maxime Rousseau LICENSE: MIT
```

2.2 Overview

2.2.1 Image Processing

This program takes as inputs facial image(s) (supported formats: jpg, png tiff, bmp) for initial processing and prepare for landmarking and analyis. The image(s) are then processed as follows: facial detection with MTCNN, 68 landmark face annotation, computation of metrics.

2.2.2 Output

By default full image processing is done and outputed into to file (default: `out.csv`). The user may specify what output is desired as well as the desired output file (supported formats: csv, pkl, h5, xlsx). See usage for details.

2.3 Structure

The __init__.py file comprises of the main method calls while the different classes are stored in the fcn/ directory. Under this directory, we find: img_prep.py which will prepare the image by scaling and transforming it to grayscale, face_detect.py which runs the haar cascade detecting the face on the prepared image, annotate.py which places the landmarks on the detected faces of the image, analyze.py calls the stats.R script which runs the statistical analyses for the study.

The output images are stored as they are processed in their respective directories: img_raw/ for the raw inputed images, img_prep/ for the prepared images, img_proc/ for the processed images (faces detected and landmarks placed).

The data/ directory contains the cascade classifier and shape predictor. Under faces/ are stored the coordinates of the rectangles from the detected faces in each image. The ldmks/ directory contains the matrices of the landmarks for each groups to be analyzed using the R script.

The gross structure of the package is outlined below:

```
pfla
├── contributing.md
├── docs
│   ├── build
│   ├── make.bat
│   ├── Makefile
│   └── source
│       ├── analyze.rst
│       ├── annotate.rst
│       ├── conf.py
```

(continues on next page)

(continued from previous page)

```
        ├── face_detect.rst
        ├── img_prep.rst
        ├── index.rst
        ├── install.rst
        ├── modules.rst
        ├── overview.rst
        └── structure.rst
├── LICENSE.txt
├── MANIFEST.in
├── paper
    ├── histo_02.png
    ├── paper.bib
    ├── paper.md
    └── pfla.png
├── pfla
    ├── annotate.py
    ├── cli.py
    ├── face_detect.py
    ├── img_prep.py
    ├── __init__.py
    ├── linear.py
    ├── logger.py
    ├── metrics.py
    └── tests
        ├── data
            ├── __init__.py
            ├── m01.jpg
            ├── m02.jpg
            ├── m03.jpg
            ├── m04.jpg
            └── m05.jpg
        ├── __init__.py
├── PROGRESS.md
├── README.md
├── requirements-pytorch.txt
├── requirements.txt
└── setup.py
```

2.4 API Documentation

2.4.1 img_prep module

class img_prep.**ImgPrep**(*PATH*, *GRAY=False*)

Bases: object

Raw images to be prepared for processing.

Will read the raw image from the folder, scale it, turn it to grayscale, and save it to /img/img_prep/ under its identification number.

Parameters

- **PATH** (*string*) – Path to the image or image directory.
- **GRAY** (*boolean*) – Convert image to grayscale (default: False)

Returns **np_im** – Numpy array of image(s)

Return type numpy array

grayscale(*image*)

prepare_dir()
Load images and return as numpy array

prepare_file()
Load image and return as numpy array

2.4.2 face_detect module

class face_detect.**FaceDetect**(*IMG*, *IS_FILE*)
Bases: object

Detect faces on images

Parameters **IMG** (*numpy array*) – Numpy array of prepared image(s).

mtcnn_box()
Bounding boxes from the MTCNN face detector

2.4.3 annotate module

class annotate.**FaceAnnotate**(*IMG*, *BOX*, *IS_FILE*)
Bases: object

Face annotation with 68 landmarks

Parameters

- **IMG** (*numpy array*) – Numpy array of the image to be processed for landmarks
- **BOX** (*typle*) – Bounding box of the detected face

get_ldmk()
Get landmark coordinates for detected face

Returns **ldmk** – Numpy array containing the coordinates of the landmarks

Return type numpy array

2.4.4 metrics module

class metrics.**Metrics**(*LDMK*, *IS_FILE*)
Bases: object

Compute various metrics based on landmarks

Parameters

- **LDMK** (*numpy array*) – Array containing the landmarks for the detected face
- **IS_FILE** (*boolean*) – Is the input a file

Returns **metrics** – Numpy array of the metrics computed

Return type numpy array

compute_metrics()

compute_ratio(*coord*)

2.4.5 logger module

class logger.**Logger**(*VERBOSE*)
Bases: object

Logging functionality

Parameters **VERBOSE** (*boolean*) – Value from the verbose argument

info(*MSG*, *LEVEL*)
Log message based on level

Parameters

- **MSG** (*string*) – Message to be logged
- **LEVEL** (*int*) – Level of the message: (0 = info, 1 = error)

2.4.6 linear module

class linear.**Linear**(*coords_ax*, *coords_ay*, *coords_bx*, *coords_by*)
Bases: object

This class is a linear mathematical function

euc_dist()
Compute the Euclidean distance between 2 landmarks

Calculates the Euclidean distance between 2 landmarks and returns it as output.

Parameters **NONE**

Returns **distance** – Euclidean between the two landmarks

Return type numpy array

CHAPTER 3

Modules

- genindex
- modindex
- search

Python Module Index

Index