Menpo Documentation

Release 0.4.0+0.gb4bdac6.dirty

Joan Alabort-i-Medina, Epameinondas Antonakos, James Booth,

May 21, 2016

Contents

Welcome to the MenpoDetect documentation!

MenpoDetect is a Python package designed to make object detection, in particular face detection, simple. MenpoDetect relies on the core package of Menpo, and thus the output of MenpoDetect is always assumed to be Menpo core types. If you aren't sure what Menpo is, please take a look over at Menpo.org.

A short example is often more illustrative than a verbose explanation. Let's assume that you want to load a set of images and that we want to detect all the faces in the images. We could do this using the Viola-Jones detector provided by OpenCV as follows:

```
import menpo.io as mio
from menpodetect import load_opencv_frontal_face_detector

opencv_detector = load_opencv_frontal_face_detector()

images = []
for image in mio.import_images('./images_folder'):
    opencv_detector(image)
    images.append(image)
```

Where we use Menpo to load the images from disk and then detect as many faces as possible using OpenCV. The detections are automatically attached to each image in the form of a set of landmarks. These are then easily viewed within a Jupyter notebook using the MenpoWidgets package:

```
%matplotlib inline
from menpowidgets import visualize_images

visualize_images(images)
```

CHAPTER 1

Supported Detectors

MenpoDetect was not designed for performing novel object detection research. Therefore, it relies on a number of existing packages and merely normalizes the inputs and outputs so that they are consistent with core Menpo types. These projects are as follows:

- cypico - Provides the detection capabilities of the Pico detector. This has similar performance to a Viola-Jones detector but allows for in-plane rotation detection (detecting faces that are rotated in the Roll angle).
- cyffld2 - Provides the detection capabilities of the FFLD2 project. This is an FFT based DPM detection package. It also ships with the highly performant detector provided by Mathias et. al.
- dlib - Provides the detection capabilities of the Dlib project. This is a HOG-SVM based detector that will return a very low number of false positives.
- OpenCV - Provides the detection capabilities of the OpenCV project. This is only available for Python 2.x due to limitations of the OpenCV project. OpenCV implements a Viola-Jones detector and provides models for both frontal and profile faces as well as eyes.

We would be very happy to see this collection expand, so pull requests are very welcome!

1.1 The MenpoDetect API

This section attempts to provide a simple browsing experience for the MenpoDetect documentation. In MenpoDetect, we use legible docstrings, and therefore, all documentation should be easily accessible in any sensible IDE (or IPython) via tab completion. However, this section should make most of the core classes available for viewing online.

1.1.1 `menpodetect.detect`

This module contains a base implementation of the generic detection method. It also provides other helper methods that are useful for all detectors. In general you will never instantiate one of these directly.

Core

detect

menpodetect.detect.**detect**(*detector_callable*, *image*, *greyscale=True*, *image_diagonal=None*, *group_prefix='object'*)

Apply the general detection framework.

This involves converting the image to greyscale if necessary, rescaling the image to a given diagonal, performing the detection, and attaching the scaled landmarks back onto the original image.

uint8 images cannot be converted to greyscale by this framework, so must already be greyscale or `greyscale=False`.

Parameters

- **`detector_callable`** (*callable* or *function*) – A callable object that will perform detection given a single parameter, a *uint8* numpy array with either no channels, or channels as the *last* axis.
- **`image`** (*menpo.image.Image*) – A Menpo image to detect. The bounding boxes of the detected objects will be attached to this image.
- **`greyscale`** (*bool*, optional) – Convert the image to greyscale or not.
- **`image_diagonal`** (*int*, optional) – The total size of the diagonal of the image that should be used for detection. This is useful for scaling images up and down for detection.
- **`group_prefix`** (*str*, optional) – The prefix string to be appended to each each landmark group that is stored on the image. Each detection will be stored as group_prefix_# where # is a count starting from 0.

Returns **bounding_boxes** (*menpo.shape.PointDirectedGraph*) – A list of bounding boxes representing the detections found.

Convenience

menpo_image_to_uint8

`menpodetect.detect.`**`menpo_image_to_uint8`**(*image*)

Return the given image as a uint8 array. This is a copy of the image.

Parameters **`image`** (*menpo.image.Image*) – The image to convert. If already uint8, only the channels will be rolled to the last axis.

Returns **uint8_image** (*ndarray*) – *uint8* Numpy array, channels as the back (last) axis.

1.1.2 `menpodetect.dlib`

This module contains a wrapper of the detector provided by the Dlib [1] [2] project. In particular, it provides access to a frontal face detector that implements the work from [3]. The Dlib detector is also trainable.

Detection

DlibDetector

class `menpodetect.dlib.`**`DlibDetector`**(*model*)

Bases: `object`

A generic dlib detector.

[1] http://dlib.net/

[2] King, Davis E. "Dlib-ml: A machine learning toolkit." The Journal of Machine Learning Research 10 (2009): 1755-1758.

[3] King, Davis E. "Max-Margin Object Detection." arXiv preprint arXiv:1502.00046 (2015).

Wraps a dlib object detector inside the menpodetect framework and provides a clean interface to expose the dlib arguments.

__call__(*image*, *greyscale=False*, *image_diagonal=None*, *group_prefix='dlib'*, *n_upscales=0*)
Perform a detection using the cached dlib detector.

The detections will also be attached to the image as landmarks.

Parameters

- **image** (*menpo.image.Image*) – A Menpo image to detect. The bounding boxes of the detected objects will be attached to this image.
- **greyscale** (*bool*, optional) – Convert the image to greyscale or not.
- **image_diagonal** (*int*, optional) – The total size of the diagonal of the image that should be used for detection. This is useful for scaling images up and down for detection.
- **group_prefix** (*str*, optional) – The prefix string to be appended to each each landmark group that is stored on the image. Each detection will be stored as group_prefix_# where # is a count starting from 0.
- **n_upscales** (*int*, optional) – Number of times to upscale the image when performing the detection, may increase the chances of detecting smaller objects.

Returnsbounding_boxes (*menpo.shape.PointDirectedGraph*) – The detected objects.

load_dlib_frontal_face_detector

menpodetect.dlib.**load_dlib_frontal_face_detector**()
Load the dlib frontal face detector.

Returnsdetector (*DlibDetector*) – The frontal face detector.

Training

train_dlib_detector

menpodetect.dlib.**train_dlib_detector**(*images*, *epsilon=0.01*, *add_left_right_image_flips=False*, *verbose_stdout=False*, *C=5*, *detection_window_size=6400*, *num_threads=None*)
Train a dlib detector with the given list of images.

This is intended to easily train a list of menpo images that have their bounding boxes attached as landmarks. Each landmark group on the image will have a tight bounding box extracted from it and then dlib will train given these images.

Parameters

- **images** (*list* of *menpo.image.Image*) – The set of images to learn the detector from. Must have landmarks attached to **every** image, a bounding box will be extracted for each landmark group.
- **epsilon** (*float*, optional) – The stopping epsilon. Smaller values make the trainer's solver more accurate but might take longer to train.

- **add_left_right_image_flips** (*bool*, optional) – If `True`, assume the objects are left/right symmetric and add in left right flips of the training images. This doubles the size of the training dataset.
- **verbose_stdout** (*bool*, optional) – If `True`, will allow dlib to output its verbose messages. These will only be printed to the stdout, so will **not** appear in an IPython notebook.
- **C** (*int*, optional) – C is the usual SVM C regularization parameter. Larger values of C will encourage the trainer to fit the data better but might lead to overfitting.
- **detection_window_size** (*int*, optional) – The number of pixels inside the sliding window used. The default parameter of `6400 = 80 * 80` window size.
- **num_threads** (*int* > 0 or `None`) – How many threads to use for training. If `None`, will query multiprocessing for the number of cores.

Returns **detector** (*dlib.simple_object_detector*) – The trained detector. To save this detector, call save on the returned object and pass a string path.

Examples

Training a simple object detector from a list of menpo images and save it for later use:

```
>>> images = list(mio.import_images('./images/path'))
>>> in_memory_detector = train_dlib_detector(images, verbose_stdout=True)
>>> in_memory_detector.save('in_memory_detector.svm')
```

References

1.1.3 menpodetect.ffld2

This module contains a wrapper of the detector provided by the FFLD2 [1] [2] project. This module also provides the very powerful DPM model provided by [3].

The FFLD2 detector is also trainable.

Detection

FFLD2Detector

class menpodetect.ffld2.**FFLD2Detector**(*model*)

Bases: `object`

A generic ffld2 detector.

Wraps an ffld2 object detector inside the menpodetect framework and provides a clean interface to expose the ffld2 arguments.

__call__(*image*, *greyscale=True*, *image_diagonal=None*, *group_prefix='ffld2'*, *padding=6*, *interval=5*, *threshold=0.5*, *overlap=0.3*)

Perform a detection using the cached ffdl2 detector.

[1] https://www.idiap.ch/scientific-research/resources/exact-acceleration-of-linear-object-detectors

[2] Dubout, Charles, and François Fleuret. "Exact acceleration of linear object detectors." Computer Vision–ECCV 2012. Springer Berlin Heidelberg, 2012. 301-311.

[3] Mathias, Markus, et al. "Face detection without bells and whistles." Computer Vision–ECCV 2014. Springer International Publishing, 2014. 720-735.

The detections will also be attached to the image as landmarks.

Parameters

- **image** (*menpo.image.Image*) – A Menpo image to detect. The bounding boxes of the detected objects will be attached to this image.
- **greyscale** (*bool*, optional) – Whether to convert the image to greyscale or not.
- **image_diagonal** (*int*, optional) – The total size of the diagonal of the image that should be used for detection. This is useful for scaling images up and down for detection.
- **group_prefix** (*str*, optional) – The prefix string to be appended to each each landmark group that is stored on the image. Each detection will be stored as group_prefix_# where # is a count starting from 0.
- **padding** (*int*, optional) – Amount of zero padding in HOG cells
- **interval** (*int*, optional) – Number of levels per octave in the HOG pyramid
- **threshold** (*double*, optional) – Minimum detection threshold. Detections with a score less than this value are not returned. Values can be negative.
- **overlap** (*double*, optional) – Minimum overlap in in latent positive search and non-maxima suppression. As discussed in the Face Detection Without Bells and Whistles paper, a sensible value for overlap is 0.3

Returnsbounding_boxes (*menpo.shape.PointDirectedGraph*) – The detected objects.

load_ffld2_frontal_face_detector

menpodetect.ffld2.**load_ffld2_frontal_face_detector**()

Load the ffld2 frontal face detector. This detector is the DPM baseline provided from [1].

Returnsdetector (*FFLD2Detector*) – The frontal face detector.

References

Training

train_ffld2_detector

menpodetect.ffld2.**train_ffld2_detector**(*positive_images*, *negative_images*, *n_components=3*, *pad_x=6*, *pad_y=6*, *interval=5*, *n_relabel=8*, *n_datamine=10*, *max_negatives=24000*, *C=0.002*, *J=2.0*, *overlap=0.5*)

Train a DPM using the FFLD2 framework. This is a fairly slow process to expect to wait for a while. FFLD2 prints out information at each iteration but this will not appear in an IPython notebook, so it is best to run this kind of training from the command line.

This method requires an explicit set of negative images to learn the classifier with. The non person images from Pascal VOC 2007 are a good example of negative images to train with.

Parameters

[1] M. Mathias and R. Benenson and M. Pedersoli and L. Van Gool Face detection without bells and whistles ECCV 2014

- **positive_images** (*list* of *menpo.image.Image*) – The set of images to learn the detector from. Must have landmarks attached to **every** image, a bounding box will be extracted for each landmark group.
- **negative_images** (*list* of *menpo.image.Image*) – The set of images to learn the negative samples of the detector with. **No** landmarks need to be attached.
- **n_components** (*int*) – Number of mixture components (without symmetry).
- **pad_x** (*int*) – Amount of zero padding in HOG cells (x-direction).
- **pad_y** (*int*) – Amount of zero padding in HOG cells (y-direction).
- **interval** (*int*) – Number of levels per octave in the HOG pyramid.
- **n_relabel** (*int*) – Maximum number of training iterations.
- **n_datamine** (*int*) – Maximum number of data-mining iterations within each training iteration.
- **max_negatives** (*int*) – Maximum number of negative images to consider, can be useful for reducing training time.
- **C** (*double*) – SVM regularization constant.
- **J** (*double*) – SVM positive regularization constant boost.
- **overlap** (*double*) – Minimum overlap in in latent positive search and non-maxima suppression.

Returnsmodel (*FFLDMixture*) – The newly trained model.

References

1.1.4 menpodetect.opencv

This module contains a wrapper of the detector provided by the OpenCV [1] project. At the moment, we assume the use of OpenCV v2.x and therefore this detector will not be available for Python 3.x. We provide a number of pre-trained models that have been provided by the OpenCV community, all of which are implementations of the Viola-Jones method [2].

Detection

OpenCVDetector

class menpodetect.opencv.**OpenCVDetector**(*model*)

Bases: object

A generic opencv detector.

Wraps an opencv object detector inside the menpodetect framework and provides a clean interface to expose the opencv arguments.

__call__(*image*, *image_diagonal=None*, *group_prefix='opencv'*, *scale_factor=1.1*, *min_neighbours=5*, *min_size=(30, 30)*, *flags=None*)

Perform a detection using the cached opencv detector.

[1] http://opencv.org/

[2] Viola, Paul, and Michael Jones. "Rapid object detection using a boosted cascade of simple features." Computer Vision and Pattern Recognition, 2001. CVPR 2001.

The detections will also be attached to the image as landmarks.

Parameters

- **image** (*menpo.image.Image*) – A Menpo image to detect. The bounding boxes of the detected objects will be attached to this image.
- **image_diagonal** (*int*, optional) – The total size of the diagonal of the image that should be used for detection. This is useful for scaling images up and down for detection.
- **group_prefix** (*str*, optional) – The prefix string to be appended to each each landmark group that is stored on the image. Each detection will be stored as group_prefix_# where # is a count starting from 0.
- **scale_factor** (*float*, optional) – The amount to increase the sliding windows by over the second pass.
- **min_neighbours** (*int*, optional) – The minimum number of neighbours (close detections) before Non-Maximum suppression to be considered a detection. Use 0 to return all detections.
- **min_size** (*tuple* of 2 ints) – The minimum object size in pixels that the detector will consider.
- **flags** (*int*, optional) – The flags to be passed through to the detector.

Returnsbounding_boxes (*menpo.shape.PointDirectedGraph*) – The detected objects.

load_opencv_frontal_face_detector

menpodetect.opencv.**load_opencv_frontal_face_detector**()
Load the opencv frontal face detector: haarcascade_frontalface_alt.xml

Returnsdetector (*OpenCVDetector*) – The frontal face detector.

load_opencv_profile_face_detector

menpodetect.opencv.**load_opencv_profile_face_detector**()
Load the opencv profile face detector: haarcascade_profileface.xml

Returnsdetector (*OpenCVDetector*) – The profile face detector.

load_opencv_eye_detector

menpodetect.opencv.**load_opencv_eye_detector**()
Load the opencv eye detector: haarcascade_eye.xml

Returnsdetector (*OpenCVDetector*) – The eye detector.

References

1.1.5 menpodetect.pico

This module contains a wrapper of the detector provided by the Pico [1] project. In particular, it provides access to a frontal face detector that implements the work from [2]. At the moment no other Pico models can be loaded due to a technical limitation in how the models are provided.

Pico is of particularly useful for images where the face has undergone in-plane rotation as Pico is capable of performing in-plane detections.

Detection

PicoDetector

class menpodetect.pico.**PicoDetector**(*model*, *detector=<class 'menpodetect.pico.detect._pico_detect'>*)

Bases: object

A generic pico detector.

Wraps a pico object detector inside the menpodetect framework and provides a clean interface to expose the pico arguments.

At the moment this isn't particularly useful as loading Pico models is complex.

__call__(*image*, *image_diagonal=None*, *group_prefix='pico'*, *max_detections=100*, *orientations=0.0*, *degrees=True*, *scale_factor=1.2*, *stride_factor=0.1*, *min_size=100*, *confidence_cutoff=3.0*, *axis_aligned_bb=True*)

Perform a detection using the cached pico detector.

The detections will also be attached to the image as landmarks.

Parameters

- **image** (*menpo.image.Image*) – A Menpo image to detect. The bounding boxes of the detected objects will be attached to this image.
- **image_diagonal** (*int*, optional) – The total size of the diagonal of the image that should be used for detection. This is useful for scaling images up and down for detection.
- **group_prefix** (*str*, optional) – The prefix string to be appended to each each landmark group that is stored on the image. Each detection will be stored as group_prefix_# where # is a count starting from 0.
- **max_detections** (*int*, optional) – The maximum number of detections to return.
- **orientations** (list of *float's or 'float*, optional) – The orientations of the cascades to use. `0.0` will perform an axis aligned detection. Values greater than `0.0` will perform detections of the cascade rotated counterclockwise around a unit circle. If a list is passed, each item should be an orientation in either radians or degrees around the unit circle, with `0.0` being axis aligned.
- **degrees** (*bool*, optional) – If `True`, the `orientations` parameter is treated as rotations counterclockwise in degrees rather than radians.

[1] https://github.com/nenadmarkus/pico

[2] N. Markus, M. Frljak, I. S. Pandzic, J. Ahlberg and R. Forchheimer, "Object Detection with Pixel Intensity Comparisons Organized in Decision Trees", http://arxiv.org/abs/1305.4537

- **scale_factor** (*float*, optional) – The ratio to increase the cascade window at every iteration. Must be greater than 1.0
- **stride_factor** (*float*, optional) – The ratio to decrease the window step by at every iteration. Must be less than 1.0, optional
- **min_size** (*float*, optional) – The minimum size in pixels (diameter of the detection circle) that a face can be. This is the starting cascade window size.
- **confidence_cutoff** (*float*, optional) – The confidence value to trim the detections with. Any detections with confidence less than the cutoff will be discarded.
- **axis_aligned_bb** (*bool*, optional) – If `True`, the returned detections will be axis aligned, regardless of which orientation they were detected at. If `False`, the returned bounding box will be rotated by the orientation detected.

Returnsbounding_boxes (*menpo.shape.PointDirectedGraph*) – The detected objects.

load_pico_frontal_face_detector

`menpodetect.pico.`**`load_pico_frontal_face_detector`**`()`

Load the pico frontal face detector.

Returnsdetector (*PicoDetector*) – The frontal face detector.

References

Symbols

D

F

L

M

O

P

T