UNIVERSITY OF CALGARY

Department of Electrical and Computer Engineering

ENEL 619.76

Biometric Technologies and Systems Design

CODE REPORT

Cristobal Martinez

December 15, 2016

Instructor: Dr. S. Yanushkevich

INSTRUCTIONS

The menpo project homepage have great instructions in how to start using their tools. For full instructions go here: http://www.menpo.org/installation/
I will produce a summarized instructions for someone that has never used python.
The easiest way to get started is to install Conda. Conda is a popular open-source framework for distributing Python applications.

1. Install miniconda from this page: http://conda.pydata.org/miniconda.html
2. From the command line create a fresh conda environment by using: `conda create -n menpo python`
3. Activate the environment by executing: `activate menpo`
4. Install the whole Menpo Project and all of its dependencies: `conda install -c menpo menpoproject`

Menpo was created with the idea to be used on Jupyter Notebooks. A tutorial on Jupyter Notebooks can be found here: https://www.youtube.com/watch?v=Rc4JQWowG5I
Also you may can download the training database from here: http://ibug.doc.ic.ac.uk/resources/facial-point-annotations/

CODE

This first part of the code loads all the training images to RAM

```
#import libraries needed from menpo project
%matplotlib inline
import menpo.io as mio
from menpo.visualize import print_progress
from menpo.landmark import labeller, face_ibug_68_to_face_ibug_68_trimesh
from menpowidgets import visualize_images
from pathlib import Path

#here we have stored all the images from training
path_to_images = 'C:\\Users\\ch9fod\\Documents\\lfpw\\trainset'
training_images = []
for img in print_progress(mio.import_images(path_to_images, verbose=True)):
    # convert to greyscale
    if img.n_channels == 3:
        img = img.as_greyscale()
    # crop to landmarks bounding box with an extra 20% padding
    img = img.crop_to_landmarks_proportion(0.2)
    # rescale image if its diagonal is bigger than 400 pixels
    d = img.diagonal()
    if d > 400:
        img = img.rescale(400.0 / d)
    # define a TriMesh which will be useful for Piecewise Affine Warp of HolisticAAM
    labeller(img, 'PTS', face_ibug_68_to_face_ibug_68_trimesh)
    # append to list

    training_images.append(img)
```

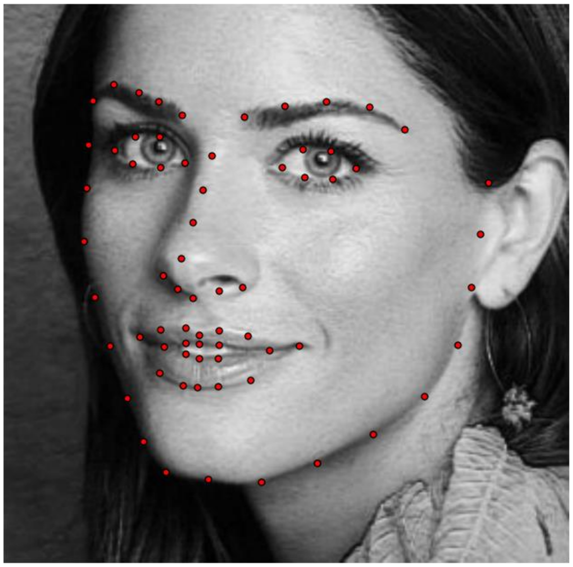
This code generates the following output:

```
Found 811 assets, index the returned LazyList to import.
[====================] 100% (811/811) - done.
```

With the code below you can visualize the training images:

```
from menpowidgets import visualize_images
visualize_images(training_images)
```

This code generates an output like this:



With this code you train the AAM:

```
from menpofit.aam import HolisticAAM
from menpo.feature import fast_dsift

# build Holistic AAM
aam = HolisticAAM(
    training_images,
    group='face_ibug_68_trimesh',
    verbose=True,
    holistic_features=fast_dsift,
    diagonal=120,
    scales=(0.5, 1.0)
)
```

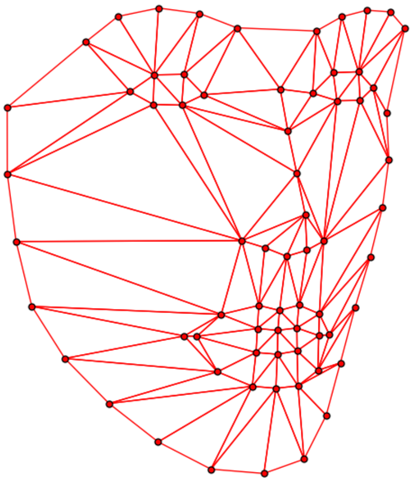
And gives the following output:

```
- Computing reference shape                         Computing batch 0
- Building models
  - Scale 0: Done
  - Scale 1: Done
```

To view the shapes model you use:

```
#with this widget we can view the shapes model
aam.view_shape_models_widget()
```

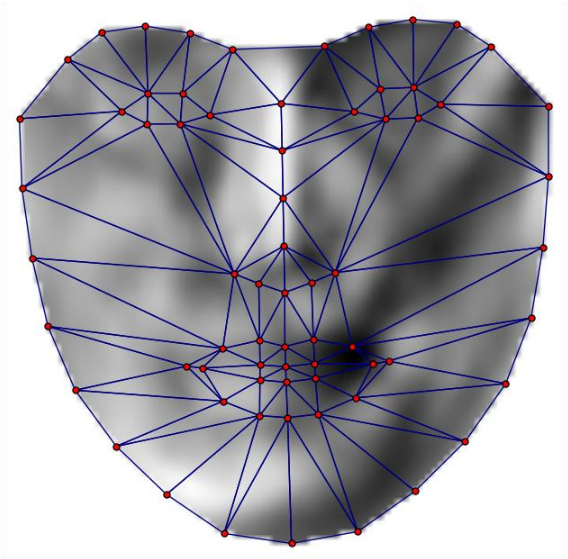
And gives an output like this:



To view the appearances model you use:

```
#with this widget we can view the appearance model
aam.view_appearance_models_widget()
```

And gives an output like this:



To view both at the same time you use:

```
#with this widget we can view both parameters
aam.view_aam_widget()
```

With the next code we prepare to do the fitting. We create the fitting object and load the new dataset to RAM:

```
#I use a LucasKanade fitter that uses the previous model created
from menpofit.aam import LucasKanadeAAMFitter
fitter = LucasKanadeAAMFitter(aam, n_shape=0.9, n_appearance=0.9)
path_to_lfpw = Path('C:\\Users\\ch9fod\\Documents\\lfpw\\')
# load test
test_images = []
bboxes = []
#for i in mio.import_images(path_to_lfpw / 'myset', max_images=12, verbose=True):
for i in mio.import_images(path_to_lfpw / 'myset', verbose=True):
    # crop image
    i = i.crop_to_landmarks_proportion(0.2)
    # convert it to grayscale if needed
    if i.n_channels == 3:
        i = i.as_greyscale(mode='luminosity')
    # append it to the list
    test_images.append(i)
```

And produces (depending on the dataset size):

```
Found 32 assets, index the returned LazyList to import.
```

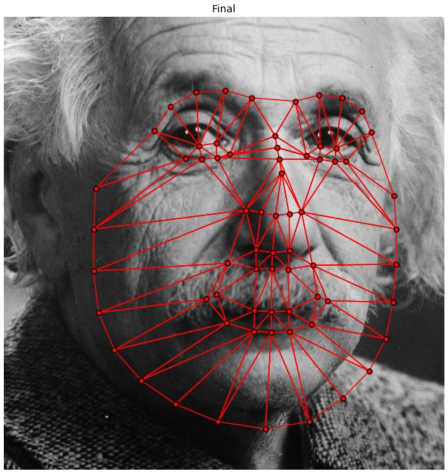
Finally, we fit the new dataset:

```
fitting_results = []
# fit images
for i in test_images:
    # obtain ground truth (original) landmarks
    gt_s = i.landmarks['LJSON'].lms

    # generate initialization shape
    initial_s = noisy_shape_from_bounding_box(gt_s, gt_s.bounding_box())

    # fit image
    fr = fitter.fit_from_shape(i, initial_s, gt_shape=gt_s)
    fitting_results.append(fr)
```

And we can visualize the results with the following widget:

```
from menpofit.fitter import noisy_shape_from_bounding_box
#here we view the fitting results
from menpowidgets import visualize_fitting_result
visualize_fitting_result(fitting_results)
```

That produces the results windows like this:



If a new user has any question you can always ask here: https://groups.google.com/forum/#!forum/menpo-users
Or contact the team members by email, which you can get from here: http://www.menpo.org/team.html
Everything I created for this project can also be viewed at my github page: https://github.com/ch9fod/619_Project