For usage of each file see inside it, its description. If we want to automatically classify images in a folder that contains images we need to run ‘classification.py’ only. Files ‘palsyfinal.py’ along with ‘patients\_landmarks.dat’, are also needed in order for the above to run. Instructions of how to create ‘patients\_landmarks.dat’ are provided at the end.

palsyfinal.py: Automatically detects the face and the landmarks in one image. Its arguments are the trained shape predictor (which detects the landmarks) and the image path

finpalrun.py: Runs palsyfinal.py many times, one for each image in a specified folder. Its arguments are the path which contains the shape predictor and the path of the folder with the images in which landmarks will be detected. The latter folder should only contain images with ‘png’ or ‘jpg’ format.

manual\_annot.py: If we want to manually annotate images we run this file. The only argument is the directory which contains the images. It is possible to also have subdirectories in that directory but these subdirectories must contain only images. When we run this file a window appears in which we can click 68 landmarks in each image. After that the next image will appear. We can delete a landmark we just clicked by pressing the key ‘a’. To exit from a specific image and move to the next one click ‘Esc’. To exit the program at any time click ‘Ctrl + \’.

pointcheck.py: This file checks if the previously manually annotated images have correctly located the landmarks. It is for verification purposes. Its argument is the same folder used in manual\_annot.py and it creates lines between landmarks that belong to the same areas of the face (eg. mouth, eyes, brows etc.). For examples, look at thesis/paper.

ibugformat.py: This file is used for retraining the landmark detector with the new manually annotated images of patients. It gets as the first argument the location of the folder that contains the new images to be inserted in the iBUG format (‘foretrain’ here) as well as the npy files of the landmarks of these images and the npy files of the face boxes of them. This folder should be a subfolder inside the iBUG folder that contains the initial images as well. The second argument is a txt file that contains the format of the iBUG dataset for just one image ('train\_format\_original.txt'). The last argument is the original file with the iBUG's dataset training files ('labels\_ibug\_300W\_train.xml'), which can be found in the original dataset. The output of this program is the file ‘traininglast.txt’ which can be found inside the folder of images specified in the first argument (‘foretrain’ here). This output file will be used below for retraining the landmark predictor.

Images of peripheral and central palsy are putted together in the same folder. Some of them are renamed since there might be conflicts with the names (from central folder). The initial images are in the folder “Central\_and\_Peripheral\_New” while the processed ones (renamed and run íbugformat.py’on them) can be found in the folder “foretrain”. Please note that the original i-BUG dataset needs to be downloaded from here: <https://ibug.doc.ic.ac.uk/resources/300-W/>

typeofpalsy.py: This file is used to decide which metrics and thresholds are the best for the cases of automatic and of manual detection of the landmarks. It is assumed here that we already have classified images in folders (healthy-central-peripheral) in order to establish a baseline for our model. This file takes as its first two arguments the image location folder with the peripheral images (should contain only the images with their corresponding npy files – one of the face box and one of the landmarks) and the image location folder with the central images (should contain the same files as peripheral folder). The third argument is the folder with the healthy images (same files in it as before). The last argument is a 1 or 0 depending on whether we have manual annotation or not.

classification.py: This file is used to automatically classify random images as containing a healthy individual, a patient with a central or a patient with a peripheral palsy. It can also be used if we have manually annotated landmarks. It gets as arguments the folder with the images that we want to classify, the location of the shape predictor folder, and a 1 or 0 as a third argument depending on whether we have manual annotation or not. There is also an optional argument that is used only when we have manual annotation. Its value should be set to 1 if we are in a folder which contains images of a central or peripheral palsy and 0 otherwise.

**Train Shape Predictor**

It is preferred to train a shape predictor using Google Colaboratory since it is significantly reduces the time required for training (we can also run it locally). To do that we have to upload in our google drive the folder ‘train\_shape\_predictor’. This folder should contain the original iBUG dataset with all the non-patient images as well as those of patients. The files of the original iBUG datasetcan be putted in a folder which was created and named ‘ibug’ (the ‘foretrain’ folder which contains the files created by running ‘ibugformat.py’ should be putted inside that folder as well). All the other files inside the ‘train\_shape\_predictor’ folder are used for evaluation (optional), and for setting parameters of our model. The file that we will use is the ‘predictor.ipynb’. It first gets access to our google drive and then it goes to the folder with the folder ‘train\_shape\_predictor’ (first two commands). Then we using the output file of ‘ibugformat.py’ (‘traininglast.txt’), which can be found inside the ‘foretrain’ folder, to create an .xml file that is used for retraining (‘labels\_ibug\_300W\_train\_patients.xml’) with the command that includes ‘parse\_xml.py’. It is suggested to output this file inside the ‘ibug’ folder. At last, retraining is performed by using the command with ‘train\_shape\_predictor.py’. It gets as input the .xml file outputted above and creates the model that we want. The file ‘patient\_landmarks’ is the file that emerges after the above procedure. If someone wants to create a predictor which was trained only on non-patient images he should replace the file ‘labels\_ibug\_300W\_train.xml’ instead.

The locations of the 68 landmarks are presented in the Figure below:

