Tutorial for 3D facial emotion recognition system

Daniel Barmaimon

July 20, 2015

Contents

1 Folder distribution

The folder distribution for the correct performance of the system should be set previously any trial. There is a folder that contains the 3D point cloud sequences and the 2D images that match with each one of the clouds. There are 41 subjects and each of these subjects has been recorded in 8 different situations that are going to be referred as tasks from now on. Each frame has three files with the 3D point cloud, 2D image and a metafile with the illumination parameters.

An image of the folder tree that allows the work with this tool is given in Figure 1.

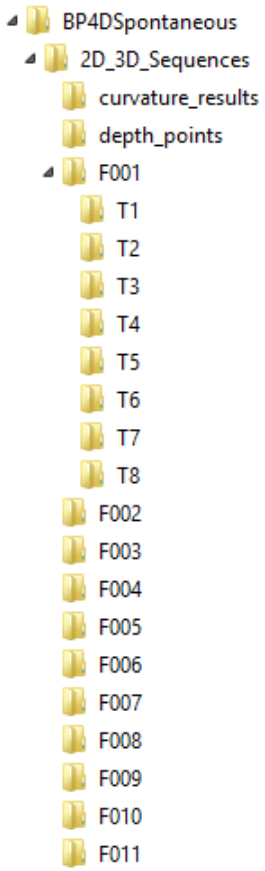

Figure 1: Folder tree for the use of the tool

As not all the images/clouds has been labeled with AU, the sequences should be modified for its use, removing those frames that are not needed. In order to know which of the frames are useful there is an excel file for each of the tasks and subjects. These files are collected in a folder called AU_OCC located inside

the *BP4DSpontaneous* folder. There is a delay of one frame in the names given in the excel files and the real name of each of the frames in the sequence (for more detail look at the document for using the database). The folder for each task has an approximate size of 4.5 Gb after removal of non-desired frames and before any processing. Be sure to have enough space to manage the amount that is desired to be analyzed before start.

The folders *curvature_results* or *depth_points* are created automatically after running some of the scripts as it will be explained later in this document.

2 Preprocessing

In the preprocessing step all the operations related with the alignment of the frames, resolution adjustment and noise addition are performed. The main task is the calculation of the curvature.
Before creating the automation for all the tasks and subjects' sequences, the analysis of a single point cloud could be performed using the function *main*. The folder structure to use should be as in Figure 2 As in the previous section,

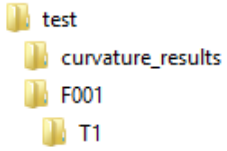

Figure 2: Folder tree for the use of the tool

the folder *curvature_results* will be created automatically in case it does not exist.

Two folders are needed to use this function: *toolbox_graph*, to read *.obj* files and create edges and faces if needed, and *icp*, to perform Iterative Closest Point algorithm for registration.

The parameters used for this function are listed below.

- *savingFlag*, allows to save and visualize the resulting curvature maps.
- *results_main_folder*, full path of the folder where the results will be stored.
- α, relationship between original number of points in the cloud and the points in the sorted and reduced point cloud (always ≥ 1).
- *reduction*, crops the face using the maximum and minimum values for coordinates x and y in a given point cloud ($0 < reduction < 1$).

- *rdist*, controls the area (normalized between -1 and 1) to compute the curvature of a chosen point. All the neighbors in this area will be considered.
- *noiseFlag*, boolean flag to add Gaussian noise to the point cloud
- *sigma*, in case that *noiseFlag* is true, value of the standard deviation (given in the same units of the point cloud, in our case mm.)
- *mapping2DFlag*, use of Viola-Jones method to find the nose area. It needs the point cloud and the 2D image as well.

After running the script for the first time, the files will be stored as a Matlab matrix with the name *data* in the same folder of the sequence. Next time the script runs, it will read only the Matlab file avoiding long times for reading the files.

If the new tests performed worked as they were expected, changes could be done over the function *processCurvature* that is the function called when *mainParallel* script runs.

The experiments that were performed related with the process of the curvature had the following flow:

1. Read the data from *.obj* files or from Matlab file.
2. Remove the back part of the head using the greatest value of y coordinate as reference.
3. Add noise if the flag is set to *true*.
4. Normalize the point cloud with values between -1 and 1 for each of the coordinate axis. (It should be changed in a future version to avoid aspect ratio changes).
5. Iterative Closest Point to reduce registration errors.
6. Calculation of the grid dimensions (it should be squared).
7. Finding the nose, looking between the minimum depth value between $0.4 * Y_{Min} < ROI < 0.4 * Y_{Max}$. In a future version use Viola-Jones.
8. Calculation of the principal directions for the point cloud.
9. Creation of a plane using the nose, the principal components, reduction parameter and number of points for the grid.
10. Grid interpolation over the point cloud.
11. Computation of the curvature for each interpolated 3D point give *rdist*.

2.1 Possible improvements to do

There are several changes to be done in order to have a more robust preprocessing pipeline. The function *mainNew* is a trial where some of the changes have been done.

First change is to base the pipeline in the nose-tip from the very beginning. Instead of cropping by the highest point (removing the back of the head using the greatest y coordinate value) a sphere with a certain value (90 mm. in our case) is centered at the nose-tip to get the cropped face.

Another change would be avoiding the normalization or create one normalization step that does not change the aspect ratio. The problem of this new step is that each frame will have different maximum and minimum values for the its axis coordinates. This means that all the frames should be analyzed before the normalization.

The most significant change should be done in the function *createPlaneGridReduced*. This function returns the grid values before interpolating over the point cloud. The old function was using standard x, y axis instead of taking advantage of the principal directions for a proper registration. The region of interest (ROI) is delimited by the maximum and minimum values of the projections of all the points onto the two principal directions that created the plane. As the grid should be done using this two directions, a homography is needed to reduce and grid the space. It is just an implementation problem due to the lack of a function that allow to grid the space in specific directions in Matlab. A initial version of this function is given as *createPlaneGridReduced*.

A last small change is to save the 2D images as a Matlab matrix in the same file *data* where *.obj* files were stored.

3 Feature extraction

The curvature maps could be considered 2D images that contains information of invariant points of the face. They are divided into squared patches after a cropping process that centers the patches at the center of the curvature map. The features chosen are the values for histogram's bin.
To get the features it is needed to run the script *getFeatures*. In the same script the only two conditions needed are to set the main folder, and have in the *curvature_results* folder the curvature maps from the preprocessing step (the parameters to configure the features extraction are commented below when the function *features* is described). The resulting features are stored in a folder located inside *curvature_results* folder as it is shown in Figure 3.

Each sequence will have a file with a notation as follows *F001_T1_a_1_00_feat*,

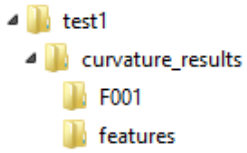

Figure 3: Folder tree for the features storage

where the four first digits define the subject (male or female and a representation number), then the values for alpha with two decimal numbers and finishing with the *feat* to denote that is a feature file. It should be careful when more than one level of noise is used because the features files could be overwritten.

The use of a sliding window for the emotion detection has been considered, and the extraction of the features has been developed as well. It is commented in the same script.

The features are extracted using the square curvatures maps. A function *features* is called given the location of the curvature maps, the number of bins for the histogram, the upper and lower limit for the bins of the histogram and N (where NxN will be total amount of patches that the curvature map is wanted to be divided into).

In this function a special crop for the curvature map is done, in such a way that the exact number of square patches divides the image with the most of the information possible. The function that performs this is called *divideIntoPatches*, and it takes care of square images, independently if the side size is even or odd, centering the central patch in the center of the curvature map (supposedly the nose-tip).

4 Classification

The classification could be performed after having the features. But for training and classifying emotional labels are needed. They are given in form of Action Units (AU) in excel files. To avoid the analysis and computation of useless data it is recommended to modify the original 3D video sequences and 2D images, removing those that have no information about AUs (as it was mentioned in Section 1).

To get this labels (using the same criteria the authors of the database used) the function *emotionArrays* should be used. It only needs the main folder as input. It will look *AU_OCC* folder and convert for each of the excel files with the action units given for each frame, into an array of numeric labels with the emotions.

Two different models are used for training and classification. First is support vector machines (SVM), that will measure how easy is to detect an emotion with respect to all the rest. This is same as saying how 'distinctive' or 'representative' is the emotion by itself. The second one is a binary tree, that classifies all the emotions together, giving a quantitative value of the performance of the whole system.

To get the classification the function *train* is used and it returns SVM models for each of the emotions, and prints on the screen the values for error for both binary trees and SVM models using a 10-cross fold validation of the model.

A manual structured cross validation has also been used and it is commented in the code. The reason of this last analysis is for those emotions with low appearance. With this division we train with half of the positive and negative samples in all of the classes. It is important in case very few information of an specific class is given.