Implementation of a cascaded regression model for facial landmark localisation

Arnaud Dapogny, Kevin Bailly

The objective of this practical session is to design a facial landmark alignment method inspired by [1]. At the end of the session, you should provide provide a commented Jupyter Notebook with the output of the cells.

This notebook (without the training and test data) should be submitted on the moodle plateform before the deadline.

1 Data preprocessing

First, you should prepare the date that will be used during the training and the evaluation of your method.

1.1 Data download

Data are available here: [this link] (1Go). While waiting for the download to complete, you can work on a sample of the database available via [this link] (the tree structure of this sample is the same as that of the main database)

This dataset gathers 4 data subsets (Helen, AFW, LFPW et IBug) containing directories with the same name as the subsets. Each directory contains both the images (in .jpg or .png) and the corresponding annotations (in .pts)

You will also find .txt files containing the list of data to be used for training – `300w_train_images.txt` and `300w_train_landmarks.txt` for the list of images and annotations respectively– as well as for the tests.

1.2 Data visualisation

To discover the data, the first step is to parse the `300w_train_images.txt` and to randomly display a dozen of images with the corresponding landmarks.

1.3 Data augmentation

For each image of the training set, you should:

1. Compute the parameters of the bounding box of the facial landmarks
2. Widen this bounding box by 30 %, crop the image with these new dimensions, and resize the image in $128 * 128$ and save it on the disk.
3. Compute the new landmark coordinates for this resized image. This new position is the ground truth (you should display some images with facial landmarks to check if this preprocess went well).

4. Compute the mean position of each landmark on the training set. It constitutes the mean shape of the face.

5. Initialize the mean shape in each image and generate 10 random perturbations (in translation and scaling) around this position. The range of these perturbations will be $\pm 20\%$ and $\pm 20px$ for scaling and translation respectively. Why do we generate these perturbations? How could they be estimated automatically?

At the end of this step, you have a clean dataset that you can use to train and evaluate your model. You should apply the same modifications to your test set (without the random perturbations) to be in the same condition for evaluation.

2 Training of a single linear regression model

In this part, we want to estimate a displacement of the landmarks, starting from a mean model (*i.e.* using computer vision and machine learning tools, to learn a function $f : I, s_0 \rightarrow \delta_s$ such that $s_0 + \delta_s \approx s^*$ in the sense of least squares).

2.1 Feature extraction

To do so, a first step is to extract a local representation around each landmark.

1. Why do we not directly use the raw value of the image pixels as a representation ?

2. Create, for each current landmark of the mean model $s^i_{0 i=0,\dots,68}$, a cv2.keyPoint object by specifying its coordinates, and setting the size of the surrounding patch to 20px.

3. Use the OpenCV *sift.compute* function to compute a SIFT feature around each landmark. What is the dimension of each feature?

4. For each image, concatenate all the computed features from each landmarks. What is the dimension of this feature vector?

2.2 Dimensionality reduction

At this stage we get a matrix $\mathbf{X}_0 \in \mathbb{R}^{M \times N}$ whose columns contain the different images from the database, and lines corresponds to the extracted features. A second step consists in reducing the dimension of the obtained descriptors.

Reminder: by computing the PCA (principal component analysis), we get a projection matrix $\mathbf{A}_0 \in \mathbb{R}^{M' \times M}$, with $M' < M$, and such that the total variance of the projected data $\tilde{\mathbf{X}}_0 = \mathbf{A}_0 \mathbf{X}_0$ is greater or equal to 98% of the total variance of the original data $\mathbf{X}_0$.

1. What is the main interest of this dimensionality reduction? Could you cite some other dimensionality reduction methods for machine learning?

2. Use the PCA (OpenCV or skimage) to keep 98% of the total variance of the features

3. What are the dimensions of the new resulting matrix $\tilde{\mathbf{X}}_0$?

2.3 Displacement estimation

Starting from this new representation $\tilde{\mathbf{X}} = \mathbf{A}_0\mathbf{X}_0$, we want to estimate the linear regression that predict δ_s, the displacement of the landmarks from the reduced extracted features. Formally:

$$\delta_s^0 = argmin_{\delta_s}||\delta_s^* - \delta_s||^2 \tag{1}$$

with $\delta_s^* \in \mathbb{R}^D$ ($D = 136$), the optimal displacement vector from the initial position s_0. By setting $\delta_s = R_0\tilde{\mathbf{X}}_0 + b_0$, it is equivalent to:

$$argmin_{R_0,b_0}||\delta_s^* - R_0\tilde{\mathbf{X}}_0 - b_0||^2 \tag{2}$$

1. Compute the least square estimation of R_0 and b_0 (b_0 can be removed by adding and extra row of ones to $\tilde{\mathbf{X}}_0$)

2. Compute the prediction error (mean absolute error) on the training set. Display s_0 the initial position of the model (in red) and $s_0 + \delta_s$ the displaced landmarks (in green) for the first image of the training set. What can we conclude ?

3. Why this prediction error is not relevant to evaluate our methods ?

4. Compute the prediction error on the test set and display s_0 the initial position of the model (in red) and $s_0 + \delta_s$ the displaced landmarks (in green) for the 5 first images of the test set. What can we conclude ?

References

[1] Xuehan Xiong and Fernando De la Torre. "Supervised Descent Method and Its Applications to Face Alignment". In: CVPR '13. 2013, pp. 532–539.