Blockwise Linear Regression for Face Alignment

Enrique Sánchez-Lozano[1]
esanchez@gts.uvigo.es
Enrique Argones-Rúa[2]
eargones@gradiant.org
José Luis Alba-Castro[1]
jalba@gts.uvigo.es

[1] Multimedia Technologies Group
AtlantTIC Research Center
University of Vigo
Vigo, Spain

[2] Multimodal Information Area
Gradiant
Vigo, Spain

Abstract

Parameterized Appearance Models, such as Active Appearance Models (AAM), Morphable Models, or Boosted Appearance Models, have been extensively used for face alignment. Discriminative methods learn a mapping function between appearance features and shape parameters. Different mapping functions have been studied in the literature, including linear regression, which has proved to perform well when close to the true solution. Despite its easiness, it still suffers from two major drawbacks: 1) It takes the whole data without highlighting relations among different regions of the face, and 2) it is computationally expensive both in time and memory. In this paper, we analyze the covariance of the training data, and propose a way to find related information. By clustering those patches that are related, we reach a noise-reduced regression matrix. Then, we construct a clean mapping matrix, with reduced dimensionality, taking only the relevant training information. Experiments show that this method outperforms linear regression for face alignment.

1 Introduction

Linear Regression for face alignment was early presented by Cootes and Taylor [6]. Once an AAM is built, alignment is formulated as the minimization of the Taylor expansion of the residual. This implies that it is necessary to recalculate the Jacobian at each iteration, which might be very expensive. To avoid this problem, only one pre-calculated mapping matrix is learned by using numerical differentiation, displacing each parameter from the known optimal value on typical images and computing an average over the training set. This method for obtaining the mapping matrix is proportional to a linear regression between a collection of perturbations and the residuals sampled on these perturbations. The behaviour of the relation between residuals and perturbations can be modelled linearly when close to the true solution [20]. In the same manner, it is no necessary to deal directly with the residuals [4]. Working with the texture, instead of with the residual, leads to improve the generalization. In such cases, an special issue of the appearance model is implicit in the regression matrix. This

easiness leads to fit images in challenging scenarios, such as in mobile phones [21]. However, despite its easiness and efectiveness when close to the true solution, linear regression suffers from two major drawbacks: 1) It is computationally expensive both in time and memory, and 2) it does not take into account what is relevant in the training data. In this paper, we present a novel way to analyze linear regression as a method for obtaining the mapping matrix. We propose finding which information is intrinsically related, removing those parts of it that might not be relevant. Instead of taking the whole data for regression, we take only this relevant data, based on a measure of the correlation between different structures (patches) of the face.

2 Related Work

Parameterized Appearance Models [7, 12, 14, 17] have proven to be a useful way to register faces, a crucial step in applications such as face recognition, tracking and synthesis. Since the work of Cootes and Taylor [6], many methods have been proposed to fit shape models to new instances. Basically, they can be classified into generative [7, 14, 17] and discriminative [13]. Recently, several successful approaches combined generative and discriminative methods [18]. There are two main approaches for discriminative fitting. The first set of methods learns a classifier to decide whether the alignment is correct or not. In this category, Liu *et al.* [10, 12], proposed several algorithms to perform gradient descent on the shape parameters to align w.r.t. the classifier score. A second set of methods learn a mapping function between image features and shape parameters [8, 17, 19, 20]. For learning this mapping, a variety of regressors have been proposed: Tresadern *et al.* [20], and Sauer *et al.* [19], used a pool of weak classifiers and random forests; Saragih and Göcke [17] proposed to learn a mapping function for each step in the fitting process. Donner *et al.* [8] proposed to use canonical correlation analysis to learn a fast regressor for AAM. Rivera and Martinez [15] explored a discriminative approach to learn a manifold from graylevel values directly to landmarks. Sánchez-Lozano *et al.* [16] proposed a continuous approach that does not need to sample the space. Cootes *et al.* [5] used Random Forest regressors to cast votes of each pixel for the optimal position. Recently, Cao *et al.* [3] proposed to learn an explicit regression matrix using boosting. Also, Tresadern *et al.* [21] used Haar-features to train the regressor. Among these methods, Tresadern *et al.* [21] found that linear regression is an useful way to fit images in challenging scenarios, such as in mobile phones. The easiness and speed when fitting a shape model with linear regression catch the attractiveness of having a real-time algorithm, for working in non fast devices. Considering that, in this paper we make a deeper analysis of linear regression for face alignment, towards making it more accurate.

3 Linear Regression

Let $\mathbf{d} \in \Re^{p\times 1}$ be the features corresponding to an specific image, where p represents the number of features. Typically, these features are taken within a set of t patches and rearranged into the column vector $\mathbf{d}$. These patches are located around a set of t points, which follow a Point Distribution Model, modelled by a shape model with s parameters. Let $\mathbf{p} \in \Re^{s\times 1}$ be the shape parameters. Linear regression methods for face alignment learn a mapping matrix $\mathbf{R} \in \Re^{s\times p}$ between perturbed shape parameters $\delta\mathbf{p} \in \Re^{s\times 1}$ and the image features $\mathbf{d}$ in the perturbed image space. $\mathbf{R}$ is desired to be able to predict the displacement of the shape

parameters towards the final and true positions. Once the matrix is learned, fitting is formulated as an iterative process, sampling the image and obtaining the shape parameters: First $\delta\mathbf{p} = \mathbf{R}\mathbf{d}$ is obtained, and then the parameters are updated $\mathbf{p} \leftarrow \mathbf{p} + \delta\mathbf{p}$. For a large set of training images (N), and a large set of perturbations (M), standard linear regression methods for training the matrix $\mathbf{R}$ minimize the following error:

$$\sum_{k=1}^{N}\sum_{l=1}^{M} \|\delta\mathbf{p}_k^l - \mathbf{R}\,\mathbf{d}_k(\mathbf{f}(\mathbf{x};\mathbf{p}_k^0 + \delta\mathbf{p}_k^l))\|_2^2, \tag{1}$$

w.r.t. $\mathbf{R} \in \Re^{s\times p}$, where $\mathbf{d}_k\big(\mathbf{f}(\mathbf{x};\mathbf{p}_k^0 + \delta\mathbf{p}_k^l)\big)$ are the image features in the image perturbed space, $\mathbf{f}(\mathbf{x};\mathbf{p})$ is a geometric transformation and $\mathbf{p}_k^0$ represent the ground-truth shape parameters of the k-th image. Here, k indexes images and l is the perturbation number. Onwards, for the sake of clarity, k will illustrate both images and perturbations. After re-arranging the features (i.e. $\mathbf{d}_k$) into the columns of $\mathbf{D} \in \Re^{p\times(NM)}$, and all the perturbations into the matrix $\mathbf{P} \in \Re^{s\times(NM)}$, the previous problem can be formulated as: $\min_{\mathbf{R}} \|\mathbf{P} - \mathbf{R}\mathbf{D}\|_F^2$, where $\|\cdot\|_F$ denotes the Frobenius norm. The optimal $\mathbf{R}$ is

$$\mathbf{R} = \mathbf{P}\mathbf{D}^T(\mathbf{D}\mathbf{D}^T)^{-1}, \tag{2}$$

(assuming that the inverse exists). Let $\mathbf{d}^i$ be the features belonging to the i-th patch, and $\mathbf{D}^i$ the rearranged data matrix for this patch. Let $\mathbf{M}^{ij} = (\mathbf{D}^i)(\mathbf{D}^j)^T = \sum_{k=1}^{NM}(\mathbf{d}_k^i)(\mathbf{d}_k^j)^T$ be the cross-products among patches; the structure of $\mathbf{D}\mathbf{D}^T$ can be seen as follows:

$$\mathbf{D}\mathbf{D}^T = \begin{pmatrix} \mathbf{M}^{11} & \cdots & \mathbf{M}^{1t} \\ \vdots & \ddots & \vdots \\ \mathbf{M}^{t1} & \cdots & \mathbf{M}^{tt} \end{pmatrix}. \tag{3}$$

Figure 1 (left) shows an example of $\mathbf{D}\mathbf{D}^T$. Let's consider that $\mathbf{d}_k^i = \widetilde{\mathbf{d}}_k^i + \mathbf{d}_0^i + \varepsilon_k^i$, where $\widetilde{\mathbf{d}}_k^i$ is the zero mean features from the i-th patch of the k-th training feature vector, $\mathbf{d}_0^i$ is a vector containing the mean features for the i-th patch, computed over $\mathbf{D}^i$, and ε_k^i is a sample noise vector, assumed to be zero mean gaussian, independent respect to the image features. We are assuming that each sample is a contribution of three terms: the zero-mean patch-wise features, the mean patch-wise vector and the noise. Expanding $\mathbf{M}^{ij}$, and removing those terms that are equal to zero, we obtain

$$\sum_{k=1}^{NM}(\mathbf{d}_k^i)(\mathbf{d}_k^j)^T = \sum_{k=1}^{NM}\big((\mathbf{d}_0^i)(\mathbf{d}_0^j)^T + (\widetilde{\mathbf{d}}_k^i)(\widetilde{\mathbf{d}}_k^j)^T + (\varepsilon_k^i)(\varepsilon_k^j)^T\big). \tag{4}$$

Considering that $\mathbf{R} \propto (\mathbf{D}\mathbf{D}^T)^{-1}$, it will include the noise, as well as the outer product of patches' means. So, it is clear that a regular regression will learn from the three terms, which yields lack of accuracy. Since fitting is formulated as $\delta\mathbf{p} = \mathbf{R}\mathbf{d}$, this noise will be mixed with the input features. Moreover, we should not expect, in general, that far apart patches jointly contribute to shape parameters estimates. So, it seems that a simplification of the regression matrix could lead to a more accurate model. Let's have a closer look to Eqn. (4). This equation is the sum of three terms: $\sum_{k=1}^{NM}(\mathbf{d}_0^i)(\mathbf{d}_0^j)^T = NM(\mathbf{d}_0^i)(\mathbf{d}_0^j)^T$ is the sum of the outer products between the means, $\sum_{k=1}^{NM}(\widetilde{\mathbf{d}}_k^i)(\widetilde{\mathbf{d}}_k^j)^T$ is the covariance matrix between the patch i and the patch j and $\sum_{k=1}^{NM}(\varepsilon_k^i)(\varepsilon_k^j)^T$ is the covariance of the noise. It can be shown that the covariances between patches are masked by the mean products. Figure 1

shows the matrix in Eqn. (3) (left) and the covariance matrix of the sampled data (second and third term on Eqn. (4)) (right). As can be seen, the outer products of patches' means mask the covariance terms. Filter them out it is highlighted that there are only few terms having a significative covariance. What we propose is to assume that those non-significative covariances are caused by the noise. Once the noise is removed, we can cluster the data to reduce the regression dimensionality, leading to a more informative matrix, which is free of most of the sample noise. We have to decide first whether a part of the matrix is only noise or not (Section 4.1). After that, clustering (Section 4.2) will remove all non relevant parts of the matrix. Then, the new regression method is presented (Section 4.3).

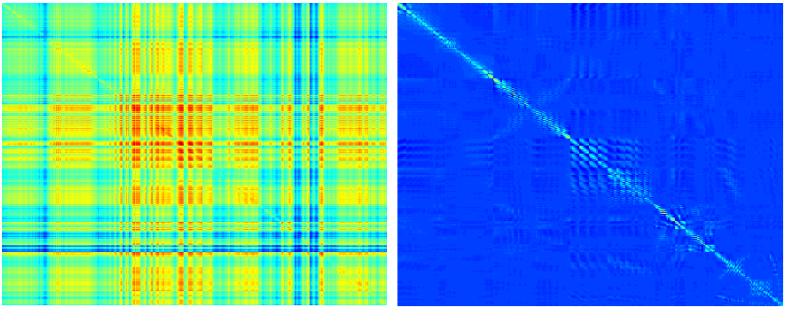

Figure 1: **Left:** Cross products between samples across the training set (Eqn. (3)) (best viewed in color). **Right:** Covariance matrix. Left image illustrates a noisy matrix where it is difficult to highlight patch-wise relationships. Right image illustrates how there are few terms that should actually be taken into account for training.

4 Regression Analysis

This section provides a deeper analysis of the linear regression method, highlighting how relationship between patches influences the final result.

4.1 Correlation

Let $\widetilde{\mathbf{D}}^i = \mathbf{D}^i - \mathbf{d}_0^i \mathbf{1}_{NM}^T$ be the zero-mean data belonging to the i-th patch[1]. Then, $\widetilde{\mathbf{D}} = \{\widetilde{\mathbf{D}}^i\}_{i=1...t}$, and $\widetilde{\mathbf{D}}\widetilde{\mathbf{D}}^T$ is the covariance of the training data. Our first aim is to discover whether two patches are related or not. Let $\sigma^{ij} = \|\left(\widetilde{\mathbf{D}}^i\right)\left(\widetilde{\mathbf{D}}^j\right)^T\|_F$ be the Frobenius norm of the covariance matrix between patch i and patch j. We define the *modified Pearson correlation coefficient* as:

$$\gamma^{ij} = \frac{(\sigma^{ij})^2}{\sigma^{ii}\sigma^{jj}} = \frac{\|\left(\widetilde{\mathbf{D}}^i\right)\left(\widetilde{\mathbf{D}}^j\right)^T\|_F^2}{\|\left(\widetilde{\mathbf{D}}^i\right)\left(\widetilde{\mathbf{D}}^i\right)^T\|_F \|\left(\widetilde{\mathbf{D}}^j\right)\left(\widetilde{\mathbf{D}}^j\right)^T\|_F}. \tag{5}$$

$\gamma^{ij} \in [0,1]$ is a measure of correlation between structures of independent variables. More clearly, a patch is a set of features which follow a distribution. The norm of the covariance

[1] $\mathbf{1}_{NM} \in \Re^{NM\times 1}$ is a NM-dimensional vector whose elements are all equal to 1

matrix is a measure of the global covariance between all the distributions inside each structure (patches in this case). If we reduce these structures to the isolated features, then σ^{ij} is directly the covariance (scalar) between feature i and feature j. Then γ^{ij} is reduced to the absolute value of the Pearson correlation coefficient. Considering that we do not care about the sign of the correlation, but only whether the structures (patches) are related or not, the *modified Pearson correlation coefficient* is enough for measuring whether patches are related among themselves. Let $\mathcal{P}$ be the set of patches: $\mathcal{P} = \{P^1, \ldots, P^t\}$. Two patches P^i and P^j are related if $\gamma^{ij} \geq \theta$, where θ is a threshold.

4.2 Clustering

As a first idea, we would consider to cluster the data based on whether two patches are related or not. However, two patches can be related with a third one, but not between each other. Then, we should consider clustering these three patches together. That is, if P^i is related to P^h, and P^h is related to P^j, P^i must be clustered together not only with P^h, but also with P^j. More specifically, P^i is clustered with P^j if, and only if, there exists some patch P^h (that can be P^i or P^j itself) that is related both to patches P^i and P^j:

$$P^i \sim P^j \iff \{\exists P^h \in \mathcal{P} \,|\, \gamma^{ih} \geq \theta \,,\, \gamma^{jh} \geq \theta\}, \tag{6}$$

where $\sim$ represents the "clustering" operation. Now, we will divide the set of patches $\mathcal{P}$ into c clusters of patches (represented as $\mathcal{C}$), so that

$$\begin{aligned} \mathcal{C}_1 \cup \cdots \cup \mathcal{C}_c &= \mathcal{P} \\ \mathcal{C}_l \cap \mathcal{C}_m = \emptyset &\iff l \neq m \\ P^i, P^j \in \mathcal{C}_l &\iff P^i \sim P^j. \end{aligned} \tag{7}$$

Eqn. (7) separates patches into disjoint clusters. Each patch must meet Eqn. (6) with the rest of the patches belonging to its cluster, and there will be no patches outside its cluster with which it meets the equation. Now, let consider the binary matrix

$$\mathbf{Z} = \{z^{ij} = 1_{(\gamma^{ij} \geq \theta)}\}_{i,j=1,\ldots,t}, \tag{8}$$

where 1_a is a binary element, which is equal to 1 if a is true, and is equal to 0 otherwise. $\mathbf{Z}$ shows which patches are related, and shows quickly which patches belong to the same cluster. Algorithm 1 summarizes how to cluster the data from the input matrix $\mathbf{Z}$.

Algorithm 1 Clustering

Require: $\mathbf{Z}$

Compute $\mathcal{C}_i = \{\, j \,|\, z^{ij} = 1\}_{i=1,\ldots,t}$
for i = 1...t **do**
  for j = i+1...t **do**
    if $\mathcal{C}_i \cap \mathcal{C}_j > \emptyset$ **then** $\mathcal{C}_i \leftarrow \mathcal{C}_i \cup \mathcal{C}_j$, $\mathcal{C}_j \leftarrow \emptyset$
    end if
  end for
end for
return $\{\mathcal{C}_i > \emptyset\}$

4.3 Regression

Looking back to Eqn. (4), we can see that it is not possible to divide the covariance of the sampled data (left side of the equality) into each different term (right side). However, we can assume that those covariances that are less than a relative threshold are due to the noise. Let $\widetilde{\mathbf{M}}^{ij} = \sum_{k=1}^{NM} \left((\widetilde{\mathbf{d}}_k^i)(\widetilde{\mathbf{d}}_k^j)^T + (\boldsymbol{\varepsilon}_k^i)(\boldsymbol{\varepsilon}_k^j)^T \right)$ be the covariance of the sampled data, between patch i and patch j. In order to mathematically reduce the effect of the noise, we define the modified covariance as:

$$\widetilde{\mathbf{M}}_\theta^{ij} = z^{ij}\widetilde{\mathbf{M}}^{ij}, \tag{9}$$

for a given threshold θ. Once clusters are obtained with Algorithm 1, we can sort the training data so that patches corresponding to an specific cluster stay together. This is equal to permute the rows of $\widetilde{\mathbf{D}}$ using a permutation matrix $\mathbf{L}$. If we use the permuted matrix $\widetilde{\mathcal{D}} = \mathbf{L}\widetilde{\mathbf{D}}$, we find that $\mathbf{R}$ is transformed into $\mathbf{R}\mathbf{L}^T$. That is, the final matrix will have the same permutation, but in columns. Notice that we also have to permute the input data when fitting, in order to maintain the solution:

$$\delta\mathbf{p} = (\mathbf{R}\mathbf{L}^T)(\mathbf{L}\mathbf{d}) = \mathbf{R}(\mathbf{L}^T\mathbf{L})\mathbf{d} = \mathbf{R}\mathbf{d}. \tag{10}$$

So, the way we permute the data is indifferent because the solution remains the same. If we use $\mathbf{L}$ to stick all the patches belonging to a corresponding cluster, we have:

$$\mathbf{L}\widetilde{\mathbf{D}}\widetilde{\mathbf{D}}^T\mathbf{L}^T = \widetilde{\mathcal{D}}\widetilde{\mathcal{D}}^T = \begin{pmatrix} \widetilde{\mathcal{M}}_{11} & \cdots & \widetilde{\mathcal{M}}_{1c} \\ \vdots & \ddots & \vdots \\ \widetilde{\mathcal{M}}_{c1} & \cdots & \widetilde{\mathcal{M}}_{cc} \end{pmatrix}. \tag{11}$$

Where

$$\widetilde{\mathcal{M}}_{lm} = \left(\widetilde{\mathcal{D}}_l\right)\left(\widetilde{\mathcal{D}}_m\right)^T = \{\widetilde{\mathbf{M}}^{ij} | P^i \in \mathcal{C}_l, P^j \in \mathcal{C}_m\}. \tag{12}$$

Now, $\widetilde{\mathcal{D}}_l$ corresponds to all the features from the matrix $\widetilde{\mathbf{D}}$ belonging to the l-th cluster (remind that $\widetilde{\mathbf{D}}^i$ were all the features belonging to the i-th patch). Now, we will replace each $\widetilde{\mathbf{M}}^{ij}$ by the modified covariance $\widetilde{\mathbf{M}}_\theta^{ij}$, as well as $\widetilde{\mathcal{M}}_{lm}$ by $\widetilde{\mathcal{M}}_{lm,\theta} = \{\widetilde{\mathbf{M}}_\theta^{ij} | P^i \in \mathcal{C}_l, P^j \in \mathcal{C}_m\}$. It is trivial to see that $\widetilde{\mathcal{M}}_{lm,\theta} = \mathbf{0} \iff l \neq m$. Then, we have to compute only $\widetilde{\mathcal{M}}_{ll,\theta}$, which implies computing only the products in the l-th cluster whose *modified Pearson coefficients* are greater than a threshold. The other products within the cluster will be set to zero. Following the structure defined in Eqn. (12), we assume that $\widetilde{\mathcal{M}}_{lm,\theta} = (\widetilde{\mathcal{D}}_{l,\theta})(\widetilde{\mathcal{D}}_{m,\theta})^T$. We have put the training data so that they are sorted by clusters instead of patches, and we have set to zero those products among patches that are not related. Working with this structure implies that Eqn. (11) is replaced by

$$\left(\widetilde{\mathcal{D}}_\theta\widetilde{\mathcal{D}}_\theta^T\right) = \begin{pmatrix} \widetilde{\mathcal{M}}_{11,\theta} & \cdots & \mathbf{0} \\ \vdots & \ddots & \vdots \\ \mathbf{0} & \cdots & \widetilde{\mathcal{M}}_{cc,\theta} \end{pmatrix}. \tag{13}$$

Then

$$\left(\widetilde{\mathcal{D}}_\theta\widetilde{\mathcal{D}}_\theta^T\right)^{-1} = \begin{pmatrix} (\widetilde{\mathcal{M}}_{11,\theta})^{-1} & \cdots & \mathbf{0} \\ \vdots & \ddots & \vdots \\ \mathbf{0} & \cdots & (\widetilde{\mathcal{M}}_{cc,\theta})^{-1} \end{pmatrix}. \tag{14}$$

With the new products, Eqn. (2) is simplified now as:

$$\mathcal{R} = \mathbf{P}\big((\widetilde{\mathcal{D}}_1)^T \dots (\widetilde{\mathcal{D}}_c)^T\big) \begin{pmatrix} (\widetilde{\mathcal{M}}_{11,\theta})^{-1} & \cdots & \mathbf{0} \\ \vdots & \ddots & \vdots \\ \mathbf{0} & \cdots & (\widetilde{\mathcal{M}}_{cc,\theta})^{-1} \end{pmatrix}. \tag{15}$$

After some linear algebra, it can be seen that

$$\mathcal{R} = \big(\mathcal{R}_1 \dots \mathcal{R}_c\big), \tag{16}$$

where

$$\mathcal{R}_l = \mathbf{P}(\widetilde{\mathcal{D}}_l)^T (\widetilde{\mathcal{M}}_{ll,\theta})^{-1}. \tag{17}$$

That is[2]:

$$\mathcal{R}_l = \mathbf{P}(\widetilde{\mathcal{D}}_l)^T \big((\widetilde{\mathcal{D}}_{l,\theta})(\widetilde{\mathcal{D}}_{l,\theta})^T\big)^{-1}. \tag{18}$$

Let's have a closer look to Eqn. (18). We have reached a solution that is similar to the original presented in Eqn. (2). However, instead of taking into account all the training data, we have removed the mean data, we have clustered them, and removed all the weakly correlated products. The solution obtained is similar to training a linear regressor for a modified version of each cluster. With this solution, we are considering a more informative linear regression that leads to a more accurate solution with the same data. Also, each cluster will have, in most cases, less dimensions than the original problem providing a better generalization. Algorithm 2 summarizes the proposed algorithm for training.

Algorithm 2 Blockwise Linear Regression Training

Require: $\mathbf{D}$, $\mathbf{P}$, θ, t patches
1: Compute $\widetilde{\mathbf{D}}^i = \mathbf{D}^i - \mathbf{d}_0^i \mathbf{1}_{NM}^T$, for each $i = 1,\dots,t$
2: Compute γ^{ij} (Eqn. (5)), for each $i, j = 1,\dots,t$
3: Compute $\mathbf{Z}$ (Eqn. (8))
4: Obtain clusters $\{\mathcal{C}_l\}_{l=1,\dots,c}$ (Algorithm 1).
5: Obtain $\widetilde{\mathbf{M}}_\theta^{ij} = z^{ij}\widetilde{\mathbf{M}}^{ij} = z^{ij}\widetilde{\mathbf{D}}^i\widetilde{\mathbf{D}}^j$
6: **for all** $\mathcal{C}_l$ **do**
7: Compute $\widetilde{\mathcal{M}}_{ll,\theta} = \{\widetilde{\mathbf{M}}_\theta^{ij}, P^i, P^j \in \mathcal{C}_l\}$
8: Compute $\widetilde{\mathcal{D}}_l$
9: Compute $\mathcal{R}_l$ (Eqn. (17))
10: **end for**
11: **return** $\{\mathcal{C}_l, \mathcal{R}_l, \mathbf{d}_{l0}\}_{l=1\dots c}$

Fitting is now calculated as follows. Consider the input data $\mathbf{d}$ and the mean feature vector $\mathbf{d}_0 = (\mathbf{d}_0^1 \dots \mathbf{d}_0^t) = \mathbf{L}^T(\mathbf{d}_{10} \dots \mathbf{d}_{c0})$. Shape parameters are now computed as

$$\begin{aligned} \delta\mathbf{p} &= \mathcal{R}\mathbf{L}(\mathbf{d} - \mathbf{d}_0) \\ &= \sum_{l=1}^{c} \mathcal{R}_l(\mathbf{d}_l - \mathbf{d}_{l0}), \end{aligned} \tag{19}$$

[2]It is worth noting that we do not need to replace the first term $\widetilde{\mathcal{D}}_l$ by the modified $\widetilde{\mathcal{D}}_{l,\theta}$ because the zero terms in $\widetilde{\mathcal{M}}_{ll,\theta}$ play the filtering roll. This way, also, the formulation is coherent, because $\widetilde{\mathcal{D}}_{l,\theta}$ is not completely defined, and it is not trivial to obtain it from $\widetilde{\mathcal{M}}_{ll,\theta}$

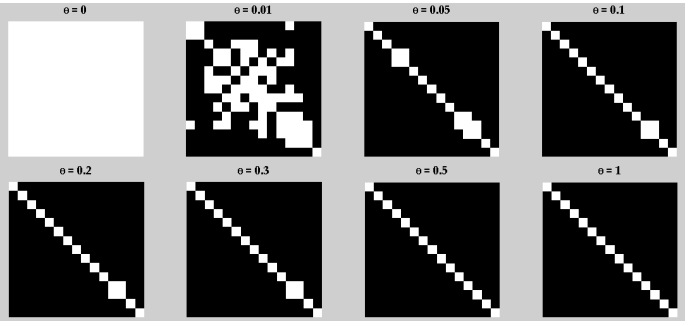

Figure 2: $\mathbf{Z}$ matrices for different values of θ. White elements correspond to 1 and black elements to 0. As can be seen, when θ is higher, less relations are found around the non-diagonal elements.

where $\mathbf{d}_l$ are the features of the input data belonging to the l-th cluster, rearranged as a column vector, and $\mathbf{d}_{l0}$ is the rearranged column vector containing the learned mean features of l-th cluster. As said above, each $\mathcal{R}_l$ has been trained only with the information associated to the l-th cluster. Summarizing, we have first obtained the *modified Pearson correlation coefficient*, and we have divided data into clusters. Then, we have set to zero all cross-products corresponding to weakly correlated patches. Then, a regressor was trained for each cluster. It is important to remark the role of θ. It is trivial to see that, when $\theta = 0$, Eqn. (18) is the same as Eqn. (2), but removing the mean, since there is only one cluster containing all the patches, and all the patches are related. When $\theta = 1$, $c = t$, and each patch is itself a cluster.

5 Experimental Evaluation

5.1 Training

We have trained our algorithm using the Multi-PIE database [9]. The shape model was learned using 2500 frontal and profile images, and consists of 25 non-rigid shape basis. Procrustes analysis was used to extract rigid information from training data. Our Regression Model consists of three levels, two for rigid registration (coarse and fine), and one for non-rigid alignment. That is, $Model = \{\mathcal{R}_{coarse}, \mathcal{R}_{fine}, \mathcal{R}_{non-rigid}\}$. All these three levels were trained using Algorithm 2. The input data $\mathbf{D}$ was obtained by perturbing the ground-truth coefficients, using 800 images from the Multi-PIE database and 200 images from the LFPW database [1]. The number of pertubations was set to 100. This perturbation was obtained applying a gaussian distribution with standard deviation equal to 2 times the variance of each mode. We have used normalized graylevel values as feature vectors, and selected $t = 15$ points, which correspond to the typical internal points of the face, excluding the center of the eyes. These points, jointly with these centers, shape the typical me_{17} for measuring the fitting error. For each point, we have extracted its corresponding patch of pixels. We have trained models for $\theta = \{0, 0.01, 0.05, 0.1, 0.2, 0.3, 0.5, 1\}$. Finally, we have also trained the discrete regression described by Eqn. (2). Figure 2 shows the binary matrices $\mathbf{Z}$ for each model at the non-rigid stage. These matrices show directly how data must be clustered.

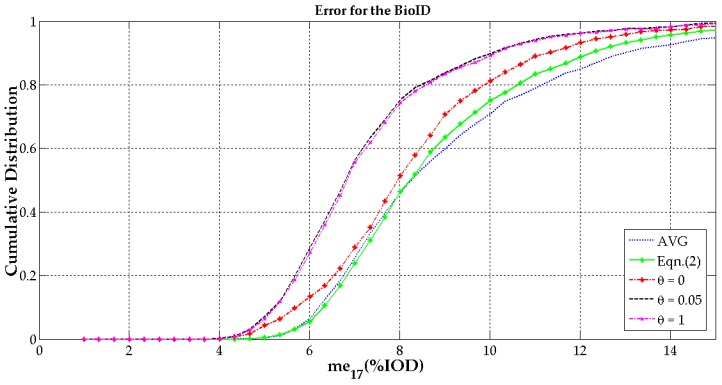

Figure 3: Results obtained for the BioID database. Blue line corresponds to the average set of points, green line corresponds to Eqn. (2), and others corresponds to $\theta = 0$ (red line), $\theta = 0.05$ (black line) and $\theta = 1$ (pink line).

5.2 Testing

We have tested our models in the BioID database [11], which consists of 1521 images, landmarked with 20 points. The process of fitting was as follows: We first detected the face using the Viola-Jones detector [22]. Then, the eyes were detected using ASEF filters [4]. With the eyes location, we have as initial points the average set. Then, the other 15 points (for measuring the me_{17}) were fitted using our models. We have compared three of our models $\theta = \{0, 0.05, 1\}$ with the original model described by Eqn. (2). Figure 3 shows the cumulative me_{17} error (as a percentage of the interocular distance) for these models, for those images where the eyes detection was successful ($\sim$ 1400 images). The best results are obtained with $\theta = 0.05$, although they may vary depending on the point distribution and the data set. This suggests that there must be taken into account that some patches provide more accuracy when they are considered together. As can be seen, there is a significative improvement in our method respect to the original one, suggesting that blockwise clustered linear regression is able to provide more accuracy, keeping a fast fitting (only sampling, permuting, and matrix multiplications are needed).

6 Conclusions

We have presented a blockwise linear regression method for efficiently training a mapping matrix. This method makes use of a correlation coefficient between data structures to detect weakly correlated image-feature patches and to filter their product out from the regression matrix. A clustering algorithm sorts then the remaining regression elements to format the matrix blockwise with many off-diagonal null blocks. It has been demonstrated that, with a fixed threshold, training the matrix is mathematically equal to training one regressor for each cluster, which leads to work with less dimensions than in the original problem, improving, thus, the accuracy of the regression. Results illustrates that considering only relevant second-order statistics of the data-features is more accurate than not preprocessing the data at all. However, there is not an optimized method for selecting the threshold. Future work will include a method for selecting it, as well as online regression for adapting an universal model

to each user. The work presented in this paper is quite generalizable because we have not made any assumption about the type of image data, features, patches, patch-wise relationship measurements, or even regressors. In short, this preliminary study shows that a blockwise clustered and noise-reduced linear regression matrix is able to improve accuracy and speed of fitting with respect to the standard approach, and pave the path for a new set of algorithms that pay more attention to the information contained in the regression matrix.

7 Acknowledgments

This work was partially supported by the Galician Government under Grant 10TIC008CT, Research contract CN2011/019 (Modalidade: Grupos de Referencia Competitiva 2007), and the project CN 2012/260 "Consolidation of Research Units: AtlantTIC"; by the Spanish Ministry of Economy and Competitivity under the project TEC2012-38939-C03-01; and by the European Commission's Seventh Framework Programme (FP7 - Capacities) under grant agreement no. 285901 (LIFTGATE project).

References

[1] P.N. Belhumeur, D. Jacobs, D.J. Kriegman, and N. Kumar. Localizing parts of faces using a consensus of exemplars. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'11)*, 2011.

[2] D.S. Bolme, B.A. Draper, and J.R. Beveridge. Average of synthetic exact filters. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'09)*, 2009.

[3] X. Cao, Y. Wei, F. Wen, and J. Sun. Face alignment by explicit shape regression. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'12)*, 2012.

[4] T. F. Cootes and C. J. Taylor. Statistical models of appearance for computer vision. *Technical Report*, 2004.

[5] T. F. Cootes, M. C. Ionita, C. Lindner, and P. Sauer. Robust and accurate shape model fitting using random fortes regression voting. In *European Conference on Computer Vision (ECCV'12)*, volume VII, pages 278–291, 2012.

[6] T.F. Cootes and C.J. Taylor. Active appearance models. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 23(6):680–689, 2001.

[7] F. De la Torre and M.H. Nguyen. Parameterized kernel principal component analysis: Theory and applications to supervised and unsupervised image alignment. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'08)*, 2008.

[8] R. Donner, M. Reiter, G. Langs, P. Pelloscheck, and H. Bischof. Fast active appearance model search using canonical correlation analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 28(10):1690–1694, 2006.

[9] R. Gross, I. Matthews, J.F. Cohn, T. Kanade, and S. Baker. Multi-pie. In *IEEE International Conference on Automatic Face and Gesture Recognition (FG'08)*, 2008.

[10] W. Hao, X. Liu, and G. Doretto. Face alignment via boosted ranking model. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'08)*, 2008.

[11] O. Jesorsky, K. Kirchberg, and R. Frischholz. Robust face detection using the hausdorff distance. In *Int'l Conference on Audio- and Video-Based Biometric Person Authentication (AVBPA'01)*, pages 90–95, 2001.

[12] X. Liu. Optimal gradient pursuit for face alignment. In *IEEE International Conference on Automatic Face and Gesture Recognition (FG'11)*, 2011.

[13] P. Martins, R. Caseiro, J. F. Henriques, and J. Batista. Discriminative bayesian active shape models. In *European Conference on Computer Vision (ECCV'12)*, volume III, pages 57–70, 2012.

[14] M.H. Nguyen and F. De la Torre. Metric learning for image alignment. *International Journal of Computer Vision*, 88(1):69–84, 2010.

[15] S. Rivera and A.M. Martinez. Learning deformable shape manifolds. *Pattern Recognition*, 45(4):1792–1801, 2012.

[16] E. Sánchez-Lozano, F. De la Torre, and D. González-Jiménez. Continuous regression for non-rigid image alignment. In *European Conference on Computer Vision (ECCV'12)*, volume VII, pages 250–263, 2012.

[17] J. Saragih and R. Göcke. Learning aam fitting through simulation. *Pattern Recognition*, 42(11):2628–2636, 2009.

[18] J. Saragih, S. Lucey, and J. F. Cohn. Deformable model fitting by regularized landmark mean-shift. *International Journal of Computer Vision*, 91(2):200–215, 2011.

[19] P. Sauer, T. F. Cootes, and C. J. Taylor. Accurate regression procedures for active appearance models. In *British Machine and Vision Conference (BMVC'11)*, 2011.

[20] P. Tresadern, P. Sauer, and T. F. Cootes. Aditive update predictors in active appearance models. In *British Machine and Vision Conference (BMVC'10)*, 2010.

[21] P. A. Tresadern, M. C. Ionita, and T. F. Cootes. Real-time facial feature tracking on a mobile device. *International Journal of Computer Vision*, 96:280–289, 2012.

[22] P. Viola and M.J. Jones. Robust real-time face detection. *International Journal of Computer Vision*, 57(2):137–154, 2004.