Int J Comput Vis
DOI 10.1007/s11263-013-0667-3

Face Alignment by Explicit Shape Regression

Xudong Cao · Yichen Wei · Fang Wen · Jian Sun

Received: 28 December 2012 / Accepted: 7 October 2013

Abstract We present a very efficient, highly accurate, "Explicit Shape Regression" approach for face alignment. Unlike previous regression-based approaches, we directly learn a vectorial regression function to infer the whole facial shape (a set of facial landmarks) from the image and *explicitly* minimize the alignment errors over the training data. The inherent shape constraint is naturally encoded into the regressor in a cascaded learning framework and applied from coarse to fine during the test, without using a fixed parametric shape model as in most previous methods. To make the regression more effective and efficient, we design a two-level boosted regression, shape indexed features and a correlation-based feature selection method. This combination enables us to learn accurate models from large training data in a short time (20 min for 2,000 training images), and run regression extremely fast in test (15 ms for a 87 landmarks shape). Experiments on challenging data show that our approach significantly outperforms the state-of-the-art in terms of both accuracy and efficiency.

Keywords Face alignment · Shape indexed feature · Correlation based feature selection · Non-parametric shape constraint · Tow-level boosted regression

X. Cao (✉) · Y. Wei · F. Wen · J. Sun
Visual Computing Group, Microsoft Research Asia, Beijing, People's Republic of China
e-mail: xudongca@microsoft.com

Y. Wei
e-mail: yichenw@microsoft.com

F. Wen
e-mail: fangwen@microsoft.com

J. Sun
e-mail: jiansun@microsoft.com

1 Introduction

Face alignment or locating semantic *facial landmarks* such as eyes, nose, mouth and chin, is essential for tasks like face recognition, face tracking, face animation and 3D face modeling. With the explosive increase in personal and web photos nowadays, a fully automatic, highly efficient and robust face alignment method is in demand. Such requirements are still challenging for current approaches in unconstrained environments, due to large variations on facial appearance, illumination, and partial occlusions.

A face shape $S = [x_1, y_1, ..., x_{N_{\text{fp}}}, y_{N_{\text{fp}}}]^T$ consists of N_{fp} facial landmarks. Given a face image, the goal of face alignment is to estimate a shape S that is as close as possible to the true shape $\hat{S}$, i.e., minimizing

$$||S - \hat{S}||_2. \tag{1}$$

The alignment error in Eq. (1) is usually used to guide the training and evaluate the performance. However, during testing, we cannot directly minimize it as $\hat{S}$ is unknown. According to how S is estimated, most alignment approaches can be classified into two categories: *optimization-based* and *regression-based*.

Optimization-based methods minimize another error function that is correlated to (1) instead. Such methods depend on the goodness of the error function and whether it can be optimized well. For example, the AAM approach (Matthews and Baker 2004; Sauer and Cootes 2011; Saragih and Goecke 2007; Cootes et al. 2001) reconstructs the entire face using an appearance model and estimates the shape by minimizing the texture residual. Because the learned appearance models have limited expressive power to capture complex and subtle face image variations in pose, expression, and illumination, it may not work well on unseen faces. It is also

Published online: 13 December 2013

well known that AAM is sensitive to the initialization due to the gradient descent optimization.

Regression-based methods learn a regression function that directly maps image appearance to the target output. The complex variations are learnt from large training data and testing is usually efficient. However, previous such methods (Cristinacce and Cootes 2007; Valstar et al. 2010; Dollar et al. 2010; Sauer and Cootes 2011; Saragih and Goecke 2007) have certain drawbacks in attaining the goal of minimizing Eq. (1). Approaches in (Dollar et al. 2010; Sauer and Cootes 2011; Saragih and Goecke 2007) rely on a parametric model (e.g., AAM) and minimize model parameter errors in the training. This is indirect and sub-optimal because smaller parameter errors are not necessarily equivalent to smaller alignment errors. Approaches in (Cristinacce and Cootes 2007; Valstar et al. 2010) learn regressors for individual landmarks, effectively using (1) as their loss functions. However, because only local image patches are used in training and appearance correlation between landmarks is not exploited, such learned regressors are usually weak and cannot handle large pose variation and partial occlusion.

We notice that the *shape constraint* is essential in all methods. Only a few salient landmarks (e.g., eye centers, mouth corners) can be reliably characterized by their image appearances. Many other non-salient landmarks (e.g., points along face contour) need help from the shape constraint—the correlation between landmarks. Most previous works use a parametric shape model to enforce such a constraint, such as PCA model in AAM (Cootes et al. 2001; Matthews and Baker 2004) and ASM (Cootes et al. 1995; Cristinacce and Cootes 2007).

Despite of the success of parametric shape models, the model flexibility (e.g., PCA dimension) is often heuristically determined. Furthermore, using a fixed shape model in an iterative alignment process (as most methods do) may also be suboptimal. For example, in initial stages (the shape is far from the true target), it is favorable to use a restricted model for fast convergence and better regularization; in late stages (the shape has been roughly aligned), we may want to use a more flexible shape model with more subtle variations for refinement. To our knowledge, adapting such shape model flexibility is rarely exploited in the literature.

In this paper, we present a novel regression-based approach without using any parametric shape models. The regressor is trained by explicitly minimizing the alignment error over training data in a holistic manner—all facial landmarks are regressed jointly. The general idea of regressing non-parametric shape has also been explored by Zhou and Comaniciu (2007).

Our regressor realizes the shape constraint in an non-parametric manner: *the regressed shape is always a linear combination of all training shapes*. Also, using features across the image for all landmarks is more discriminative than using only local patches for individual landmarks. These properties enable us to learn a flexible model with strong expressive power from large training data. We call our approach "Explicit Shape Regression".

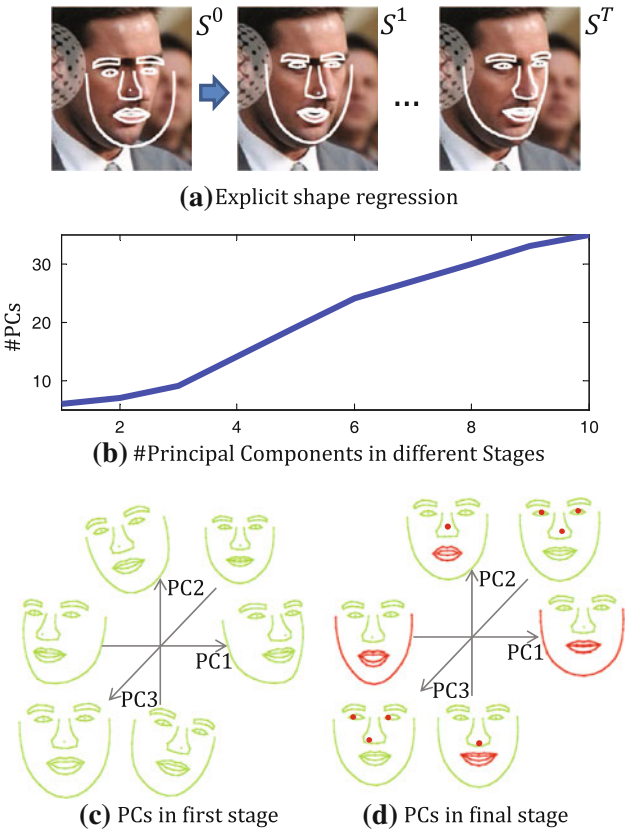


Fig. 1 Shape constraint is preserved and adaptively learned in a coarse to fine manner in our boosted regressor. **a** The shape is progressively refined by the shape increments learnt by the boosted regressors in different stages. **b** Intrinsic dimensions of learnt shape increments in a 10-stage boosted regressor, using 87 facial landmarks. **c**, **d** The first three principal components (PCs) of shape increments in the first and final stage, respectively

Jointly regressing the entire shape is challenging in the presence of large image appearance variations. We design a boosted regressor to *progressively* infer the shape—the early regressors handle large shape variations and guarantee robustness, while the later regressors handle small shape variations and ensure accuracy. Thus, the shape constraint is adaptively enforced from coarse to fine, in an automatic manner. This is illustrated in Fig. 1 and elaborated in Sect. 2.3.

Our explicit shape regression framework is inspired by the cascaded pose regression proposed by Dollar et al. (2010). In their work, a sequence of random fern regressors are learnt to predict the object pose parameters progressively. In each iteration, image features not only depend on the image content, but also depend on the predicted pose parameter from last iteration. Such *pose-indexed* features provide better geometric invariance and greatly enhance the regressor's

performance. In their experiment, this method has also been used to estimate face shape which is modeled by a simple parametric ellipse (Dollar et al. 2010).

Our method improves the cascaded pose regression framework in several important aspects and works better for face alignment problem. We adopt a non-parametric representation, directly estimate the facial landmarks by minimizing the alignment error instead of parameter error. Consequently, the underlying shape constraint is preserved automatically. To address the very challenging high-dimensional regression problem, we further propose several improvements: *a two-level boosted regression*, effective *shape indexed features*, a fast *correlation-based feature selection method* and sparse coding based model compression so that: (1) we can quickly learn accurate models from large training data (20 min on 2,000 training samples); (2) the resulting regressor is extremely efficient in the test (15 ms for 87 facial landmarks); (3) the model size is reasonably small (a few megabytes) and applicable in many scenarios. We show superior results on several challenging datasets.

2 Face Alignment by Shape Regression

2.1 The Shape Regression Framework

In this section, we describe our shape regression framework and discuss how it fit to the face alignment task.

We cast our solution to shape regression task into the gradient boosting regression (Friedman 2001; Duffy and Helmbold 2002) framework, which is a representative approach of ensemble learning. In training, it sequentially learns a series of weak learners to greedily minimize the regression loss function. In testing, it simply combine the pre-learnt weak learners in an additive manner to give the final prediction.

Before specifying how to resolve face alignment task in gradient boosting framework, we first clarify a simple and basic term, i.e. *the normalized shape*. Provided the predefined mean shape $\bar{S}$, the normalized shape of an input shape S is obtained by a similarity transform which aligns the input shape to the mean shape[1] to minimizes their $L2$ distance,

$$M_S = \underset{M}{\operatorname{argmin}} ||\bar{S} - M \circ S||_2, \tag{2}$$

where $M_S \circ S$ is the normalized shape. Now we are ready to describe our shape regression framework.

In training, given N training samples $\{I_i, \hat{S}_i, S_i^0\}_{i=1}^N$, the stage regressors $(R^1, ..., R^T)$ are sequentially learnt to reduce the alignment errors on training set. In stage each t, the stage regressor R^t is formally learnt as follows,

$$R^t = \underset{R}{\operatorname{argmin}} \sum_{i=1}^{N} ||y_i - R(I_i, S_i^{t-1})||_2 \tag{3}$$
$$y_i = M_{S_i^{t-1}} \circ (\hat{S}_i - S_i^{t-1}),$$

where S_i^{t-1} is the estimated shape in previous stage $t-1$, $M_{S_i^{t-1}} \circ (\hat{S}_i - S_i^{t-1})$ is the normalized regression target which will be discussed later. Please note herein we only apply the scale and rotation transformation to normalize the regression target.

In testing, given a facial image I and an initial shape S^0, the stage regressor computes a normalized shape increment from image features and then updates the face shape, in a cascaded manner:

$$S_i^t = S_i^{t-1} + M_{S_i^{t-1}}^{-1} \circ R^t(I_i, S_i^{t-1}), \tag{4}$$

where the stage regressor R^t updates the previous shape S^{t-1} to the new shape S^t in stage t. Note we only scale and rotate (without translating) the output of the stage regression according to the angle and the scale of the previous shape.

2.1.1 Discussion

Although our basic face alignment framework follows gradient boosting framework, there are three components specifically designed for the shape regression tasks. As these components are generic, regardless of the specific forms of the weak learner and the feature used for learning, it is worth clarifying and discussing them here.

Shape Indexed Feature The feature for learning the weak learner R^t depends on both image I and previous estimated shape S^{t-1}. It improves the performance by achieving geometric invariance. In other words, the feature is extracted relative to S^{t-1} to eliminate two kinds of variations: the variations due to scale, rotation and translation and the variations due to identity, pose and expression (i.e. the similarity transform $M_{S^{t-1}}^{-1}$ and the normalized shape $M_{S^{t-1}} \circ S^{t-1}$). It is worth mentioning that although shape indexed feature sounds more complex, our designed feature is extremely cheap to be computed and does not require image warping. The details of our feature will be described in Sect. 2.4.

Regressing the Normalized Target Instead of learning the mapping from shape indexed feature to $\hat{S}_i - S_i^{t-1}$, we argue the regression task will be simplified if we regress the normalized target $M_{S_i^{t-1}} \circ (\hat{S}_i - S_i^{t-1})$. Because the normalized target is invariant to similarity transform. To better understand its advantage, imagine two identical facial images with the estimated shape. One of them is changed by similarity transform while the other is kept unchanged. Due to the transform, the regression target (shape increments) of the transformed

[1] It is also interesting to know that the mean shape is defined as the average of the normalized training shapes. Although it sounds like a circular definition, we still can compute the mean shape in an iterative way. Readers are recommended to Active Shape Model (Cootes et al. 1995) method for details.

image are different from that of the unchanged one. Hence the regression task is complicated. In contrast, the regression task will keep simple if we regress the normalized targets, which are still the same for both samples.

Data Augmentation and Multiple Initialization Unlike typical regression task, the sample of our shape regression task is a triple which is defined by the facial image, the groundtruth shape and the initial shape i.e. $\{I_i, \hat{S}_i, S_i^0\}$. So we can augment the samples by generating multiple initial shapes for one image. In fact, it turns out such simple operation not only effectively improves generalization of training, but also reduces the variation of the final prediction by bagging the results obtained by multiple initializations. See Sect. 3 for experimental validations and further discussions.

To make the training and testing our framework clear, we format them in the pseudo codes style.

Algorithm 1 Explicit Shape Regression (ESR)

Variables: Training images and labeled shapes $\{I_l, \hat{S}_l\}_{l=1}^{L}$; ESR model $\{R^t\}_{t=1}^{T}$; Testing image I; Predicted shape S; *TrainParams*{times of data augment N_{aug}, number of stages T};
TestParams{number of multiple initializations N_{int}};
InitSet which contains exemplar shapes for initialization

ESRTraining$(\{I_l, \hat{S}_l\}_{l=1}^{L}, \mathit{TrainParams}, \mathit{InitSet})$
// augment training data
$\{I_i, \hat{S}_i, S_i^0\}_{i=1}^{N} \leftarrow \mathit{Initialization}\ (\{I_l, \hat{S}_l\}_{l=1}^{L}, N_{\text{aug}}, \mathit{InitSet})$
for t from 1 to T
  $Y \leftarrow \{M_{S_i^{t-1}} \circ (\hat{S}_i - S_i^{t-1})\}_{i=1}^{N}$ // compute normalized targets
  $R^t \leftarrow \mathit{LearnStageRegressor}(Y, \{I_i, S_i^{t-1}\}_{i=1}^{N})$ // using Eq. (3)
  for i from 1 to N
    $S_i^t \leftarrow S_i^{t-1} + M^{-1}_{S_i^{t-1}} \circ R^t(I_i, S_i^{t-1})$
return $\{R^t\}_{t=1}^{T}$

ESRTesting$(I, \{R^t\}_{t=1}^{T}, \mathit{TestParams}, \mathit{InitSet})$
// multiple initializations
$\{I_i, *, S_i^0\}_{i=1}^{N_{\text{int}}} \leftarrow \mathit{Initialization}\ (\{I, *\}, N_{\text{int}}, \mathit{InitSet})$
for t from 1 to T
  for i from 1 to N_{int}
    $S_i^t \leftarrow S_i^{t-1} + M^{-1}_{S_i^{t-1}} \circ R^t(I_i, S_i^{t-1})$
$S \leftarrow \mathit{CombineMultipleResutls}(\{S_i^T\}_{i=1}^{N_{\text{int}}})$
return S

Initialization$(\{I_c, \hat{S}_c\}_{c=1}^{C}, D, \mathit{InitSet})$
$i \leftarrow 1$
for c from 1 to C
  for d from 1 to D
    $S_i^o \leftarrow$ sampling an exemplar shape from *InitSet*
    $\{I_i^o, \hat{S}_i^o\} \leftarrow \{I_c, \hat{S}_c\}$
    $i \leftarrow i + 1$
return $\{I_i^o, \hat{S}_i^o, S_i^o\}_{i=1}^{CD}$

The details about combining multiple results and initialization (including constructing *InitSet* and sampling from *InitSet*) will be discussed later in Sect. 3.

As the stage regressor plays a vital role in our shape regression framework, we will focus on it next. We will discuss what kinds of regressors are suitable for our framework and present a series of methods for effectively learning the stage regressor.

2.2 Two-Level Boosted Regression

Conventionally, the stage regressor R^t, is a quite weak regressor such as a decision stump (Cristinacce and Cootes 2007) or a fern (Dollar et al. 2010). However, in our early experiments, we found that such regressors result in slow convergence in training and poor performance in the testing.

We conjecture this is due to two reasons: first, regressing the entire shape (as large as dozens of landmarks) is too difficult to be handled by a weak regressor in each stage; second, the shape indexed feature will be unreliable if the previous regressors are too weak to provide fairly reliable shape estimation, so will the regressor based on the shape indexed feature. Therefore it is crucial to learn a strong regressor that can rapidly reduce the alignment error of all training samples in each stage.

Generally speaking, any kinds of regressors with strong fitting capacity will be desirable. In our case, we again investigate boosted regression as the stage regressor R^t. Therefore threre are two-levels boosted regression in our method, i.e. the basic face alignment framework (external-level) and the stage regressor R^t (internal-level). To avoid terminology confusion, we term the weak learner of the internal-level boosted regression, as *primitive regressor*.

It is worth mentioning that other strong regressors have also been investigated by two notable works. Sun et al. (2013) investigated the regressor based on convolutional neural network. Xiong and De la Torre (2013) investigated the linear regression with strong hand-craft feature i.e. SIFT.

Although the external- and internal- level boosted regression bear some similarities, there is a key differences. The difference is that the shape indexed image features are fixed in the internal-level, i.e., they are indexed only relative to the previous estimated shape S^{t-1} and no longer change when those primitive regressor are being learnt.[2] This is important, as each primitive regressor is rather weak, allowing feature indexing will frequently change the features, leading to unstable consequences. Also the fixed features can lead to much faster training, as will be described later. In our experiments, we found using two-level boosted regression is more accurate than one level under the same training effort, e.g., $T = 10, K = 500$ is better than one level of T =5,000, as shown in Table 4.

[2] Otherwise this degenerates to a one level boosted regression.

2.3 Primitive Regressor

We use a *fern* as our primitive regressor. The fern was firstly introduced for classification by Ozuysal et al. (2010) and later used for regression by Dollar et al. (2010).

A fern is a composition of F (5 in our implementation) features and thresholds that divide the feature space and all training samples into 2^F bins. Each bin b is associated with a regression output y_b. Learning a fern involves a simple task and a hard task. The simple task refers to learning the outputs in bins. The hard task refers to learning the structure (the features and the splitting thresholds). We will handle the simple task first, and resolve the hard task in the Sect. 2.5 later.

Let the targets of all training samples be $\{\hat{y}_i\}_{i=1}^N$. The prediction/output of a bin should minimize its mean square distance from the targets for all training samples falling into this bin.

$$y_b = \underset{y}{\text{argmin}} \sum_{i \in \Omega_b} ||\hat{y}_i - y||_2, \tag{5}$$

where the set Ω_b indicates the samples in the bth bin. It is easy to see that the optimal solution is the average over all targets in this bin.

$$y_b = \frac{\sum_{i \in \Omega_b} \hat{y}_i}{|\Omega_b|}. \tag{6}$$

To overcome over-fitting in the case of insufficient training data in the bin, a shrinkage is performed (Friedman 2001; Ozuysal et al. 2010) as

$$y_b = \frac{1}{1 + \beta/|\Omega_b|} \frac{\sum_{i \in \Omega_b} \hat{y}_i}{|\Omega_b|}, \tag{7}$$

where β is a free shrinkage parameter. When the bin has sufficient training samples, β makes little effect; otherwise, it adaptively reduces the magnitude of the estimation.

2.3.1 Non-parametric Shape Constraint

By directly regressing the entire shape and explicitly minimizing the shape alignment error in Eq. (1), the correlation between the shape coordinates is preserved. Because each shape update is additive as in Eqs. (4), (6) and (7), it can be shown that the final regressed shape S is the sum of initial shape S^0 and the linear combination of all training shapes:

$$S = S^0 + \sum_{i=1}^{N} w_i \hat{S}_i. \tag{8}$$

Therefore, as long as the initial shape S^0 satisfies the shape constraint, *the regressed shape is always constrained to reside in the linear subspace constructed by all training shapes*. In fact, any intermediate shape in the regression also satisfies the constraint. Compare to the pre-fixed PCA shape model, the non-parametric shape constraint is adaptively determined during the learning.

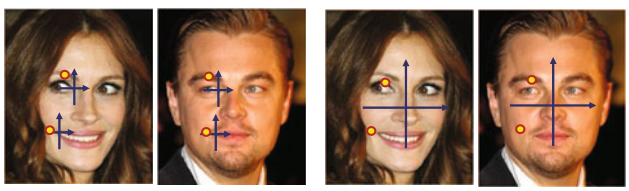
To illustrate the adaptive shape constraint, we perform PCA on all the shape increments stored in K ferns ($2^F \times K$ in total) in each stage t. As shown in Fig. 1, the intrinsic dimension (by retaining 95 % energy) of such shape spaces increases during the learning. Therefore, *the shape constraint is automatically encoded in the regressors in a coarse to fine manner*. Figure 1 also shows the first three principal components of the learnt shape increments (plus a mean shape) in first and final stage. As shown in Fig. 1c, d, the shape updates learned by the first stage regressor are dominated by global rough shape changes such as yaw, roll and scaling. In contrast, the shape updates of the final stage regressor are dominated by the subtle variations such as face contour, and motions in the mouth, nose and eyes.

2.4 Shape Indexed (Image) Features

For efficient regression, we use simple pixel-difference features, i.e., the intensity difference of two pixels in the image. Such features are extremely cheap to compute and powerful enough given sufficient training data (Ozuysal et al. 2010; Shotton et al. 2011; Dollar et al. 2010).

To let the pixel-difference achieve geometric invariance, we need to make the extracted raw pixel invariant to two kinds of variations: the variations due to similarity transform (scale, rotation and translation) and the variations due to the normalized shape (3D-poses, expressions and identities).

In this work, we propose to index a pixel by the estimated shape. Specifically, we index a pixel by the *local coordinate* $\Delta^l = (\Delta x^l, \Delta y^l)$ with respect to a landmark in the normalized face. The superscript l indicates which landmark this pixel is relative to. As Fig. 2 shows, such indexing holds invariance against the variations mentioned above and make the algorithm robust. In addition, it also enable us sampling more useful candidate features distributed around salient landmarks (e.g., a good pixel difference feature could be "eye center is darker than nose tip" or "two eye centers are similar").



(a) **(b)**

Fig. 2 Pixels indexed by the same local coordinates have the same semantic meaning (**a**), but pixels indexed by the same global coordinates have different semantic meanings due to the face shape variation (**b**)

In practical implementation, unlike previous works (Cristinacce and Cootes 2007; Valstar et al. 2010; Sauer and Cootes 2011) which requires image warping, we instead transform the local coordinate back to the global coordinates on the original image to extract the shape indexed pixels and then compute the pixel-difference features. This leads to much faster testing speed.

Let the estimated shape of a sample be S. The location of the mth landmark is obtained by $\pi_l \circ S$, where the operator π_l gets the x and y coordinates of mth landmark from the shape vector. Provided the local coordinate Δ^l, the corresponding global coordinates on the original image can formally represented as follows.[3]

$$\pi_l \circ S + M_S^{-1} \circ \Delta^l \tag{9}$$

Note that although the Δ^l is identical for different samples, the global coordinates for extracting raw pixels are adaptively adjusted for different samples to ensure geometric invariance. Herein we only scale and rotate (without translating) the local coordinates according to the angle and the scale of the shape.

For each stage regressor R^t in the external-level, we randomly generate P local coordinates $\{\Delta_\alpha^{l_\alpha}\}_{\alpha=1}^P$ which define P shape indexed pixels. Each local coordinate is generated by first randomly selecting a landmark (e.g. l_αth landmark) and then draw random x- and y-offset from uniform distribution. The P pixels result in P^2 pixel-difference features. Now, the new challenge is how to quickly select effective features from such a large pool.

2.5 Correlation-Based Feature Selection

To form a good fern regressor, F out of P^2 features are selected. Usually, this is done by randomly generating a pool of ferns and selecting the one with minimum regression error as in (5) (Ozuysal et al. 2010; Dollar et al. 2010). We denote this method as *best-of-n*, where n is the size of the pool. Due to the combinatorial explosion, it is unfeasible to evaluate (5) for all of the compositional features. As illustrated in Table 5, the error is only slightly reduced by increasing n from 1 to 1024, but the training time is significantly longer.

To better explore the huge feature space in a short time and generate good candidate ferns, we exploit the *correlation* between features and the regression target. We expect that a good fern should satisfy two properties: (1) each feature in the fern should be highly correlated to the regression target; (2) correlation between features should be low so they are complementary when composed.

To find features satisfying such properties, we propose a correlation-based feature selection method. Let Y be the regression target with N (the number of samples) rows and $2N_{\text{fp}}$ columns. Let X be pixel-difference features matrix with N rows and P^2 columns. Each column X_j of the feature matrix represents a pixel-difference feature. We want to select F columns of X which are highly correlated with Y. Since Y is a matrix, we use a projection v, which is a column vector drawn from unit Gaussian, to project Y into a column vector $Y_{\text{prob}} = Yv$. The feature which maximizes its correlation (Pearson Correlation) with the projected target is selected.

$$j_{\text{opt}} = \underset{j}{\text{argmin}} \ \text{corr}(Y_{\text{prob}}, X_j) \tag{10}$$

By repeating this procedure F times, with different random projections, we obtain F desirable features.

The random projection serves two purposes: it can preserve proximity (Bingham and Mannila 2001) such that the features correlated to the projection are also discriminative to delta shape; the multiple projections have low correlations with a high probability and the selected features are likely to be complementary. As shown in Table 5, the proposed correlation based method can select good features in a short time and is much better than the best-of-n method.

2.6 Fast Correlation Computation

At first glance, we need to compute the correlation for all candidate features to select a feature. The complexity is linear to the number of training samples and the number of pixel-difference features, i.e. $O(NP^2)$. As the size of feature pool scales square to the number of sampled pixels, the computa-

Algorithm 2 Shape indexed features

Variables: images and corresponding estimated shapes $\{I_i, S_i\}_{i=1}^N$; number of shape indexed pixel features P; number of facial points N_{fp}; the range of local coordinate κ; local coordinates $\{\Delta_\alpha^{l_\alpha}\}_{\alpha=1}^P$; shape indexed pixel features $\rho \in \Re^{N\times P}$; shape indexed pixel-difference features $X \in \Re^{N\times P^2}$;

GenerateShapeIndexedFeatures$(\{I_i, S_i\}_{i=1}^N, N_{\text{fp}}, P, \kappa)$
$\{\Delta_\alpha^{l_\alpha}\}_{\alpha=1}^P \leftarrow$ *GenerateLocalCoordinates*(*FeatureParams*)
$\rho \leftarrow$ *ExtractShapeIndexedPixels*$(\{I_i, S_i\}_{i=1}^N, \{\Delta_\alpha^{l_\alpha}\}_{\alpha=1}^P)$
$X \leftarrow$ pairwise difference of all columns of ρ
return $\{\Delta_\alpha^{l_\alpha}\}_{\alpha=1}^P, \rho, X$

GenerateLocalCoordinates$(N_{\text{fp}}, P, \kappa)$
for α from 1 to P
  $l_\alpha \leftarrow$ randomly drawn a integer in $[1, N_{\text{fp}}]$
  $\Delta_\alpha^{l_\alpha} \leftarrow$ randomly drawn two floats in $[-\kappa, \kappa]$
return $\{\Delta_\alpha^{l_\alpha}\}_{\alpha=1}^P$

ExtractShapeIndexedPixels$(\{I_i, S_i\}_{i=1}^N, \{\Delta_\alpha^{l_\alpha}\}_{\alpha=1}^P)$
for i from 1 to N
  for α from 1 to P
    $\mu_\alpha \leftarrow \pi_{l_\alpha} \circ S_i + M_{S_i}^{-1} \circ \Delta^{l_\alpha}$
    $\rho_{i\alpha} \leftarrow I_i(\mu_\alpha)$
return ρ

[3] According to aforementioned definition, the global coordinates are computed via $M_S^{-1} \circ (\pi_l \circ M_S^{-1} \circ S + \Delta^l)$. By simplifying this formula, we get Eq. (9)

tion will be very expensive, even with moderate number of pixels (e.g. 400 pixels leads to 160,000 candidate features!).

Fortunately the computational complexity can be reduced from $O(NP^2)$ to $O(NP)$ by the following facts: The correlation between the regression target and a pixel-difference feature $(\rho_m - \rho_n)$ can be represented as follows.

$$\begin{aligned} \mathrm{corr}(Y_{\mathrm{proj}}, \rho_m - \rho_n) &= \frac{\mathrm{cov}(Y_{\mathrm{proj}}, \rho_m) - \mathrm{cov}(Y_{\mathrm{proj}}, \rho_n)}{\sqrt{\sigma(Y_{\mathrm{proj}})\sigma(\rho_m - \rho_n)}} \\ \sigma(\rho_m - \rho_n) &= \mathrm{cov}(\rho_m, \rho_m) \\ &\quad + \mathrm{cov}(\rho_n, \rho_n) - 2\mathrm{cov}(\rho_m, \rho_n) \end{aligned} \tag{11}$$

We can see that the correlation is composed by two categories of covariances: the target-pixel covariance and the pixel–pixel covariance. The target-pixel covariance refers to the covariance between the projected target and pixel feature, e.g., $\mathrm{cov}(Y_{\mathrm{proj}}, \rho_m)$ and $\mathrm{cov}(Y_{\mathrm{proj}}, \rho_n)$. The pixel–pixel covariance refers to the covariance among different pixel features, e.g., $\mathrm{cov}(\rho_m, \rho_m)$, $\mathrm{cov}(\rho_n, \rho_n)$ and $\mathrm{cov}(\rho_m, \rho_n)$. As the shape indexed pixels are fixed in the internal-level boosted regression, the pixel–pixel covariances can be pre-computed and reused within each internal-level boosted regression. For each primitive regressor, we only need to compute all target-pixel covariances to compose the correlations, which scales linear to the number of pixel features. Therefore the complexity is reduced from $O(NP^2)$ to $O(NP)$.

Algorithm 3 Correlation-based feature selection

Input: regression targets $Y \in \Re^{N\times 2N_{\mathrm{fp}}}$; shape indexed pixel features $\rho \in \Re^{N\times P}$; pixel-pixel covariance $\mathrm{cov}(\rho) \in \Re^{P\times P}$; number of features of a fern F;
Output: The selected pixel-difference features $\{\rho_{m_\mathrm{f}} - \rho_{n_\mathrm{f}}\}_{\mathrm{f}=1}^{F}$ and the corresponding indices $\{m_\mathrm{f}, n_\mathrm{f}\}_{\mathrm{f}=1}^{F}$;

CorrelationBasedFeatureSelection$(Y, \mathrm{cov}(\rho), F)$
for f from 1 to F
  $v \leftarrow \mathrm{randn}(2N_{\mathrm{fp}}, 1)$ // draw a random projection from unit Gaussian
  $Y_{\mathrm{prob}} \leftarrow Yv$ // random projection
  $\mathrm{cov}(Y_{\mathrm{prob}}, \rho) \in \Re^{1\times P} \leftarrow$ compute target-pixel covariance
  $\sigma(Y_{\mathrm{prob}}) \leftarrow$ compute sample variance of Y_{prob}
  $m_\mathrm{f} = 1; n_\mathrm{f} = 1;$
  for m from 1 to P
    for n from 1 to P
      $\mathrm{corr}(Y_{\mathrm{prob}}, \rho_m - \rho_n) \leftarrow$ compute correlation using Eq. (11)
      if $\mathrm{corr}(Y_{\mathrm{prob}}, \rho_m - \rho_n) > \mathrm{corr}(Y_{\mathrm{prob}}, \rho_{m_\mathrm{f}} - \rho_{n_\mathrm{f}})$
        $m_\mathrm{f} = m; n_\mathrm{f} = n;$
return $\{\rho_{m_\mathrm{f}} - \rho_{n_\mathrm{f}}\}_{\mathrm{f}=1}^{F}, \{m_\mathrm{f}, n_\mathrm{f}\}_{\mathrm{f}=1}^{F}$

2.7 Internal-Level Boosted Regression

As aforementioned, at each stage t, we need to learn to a stage regressor to predict the normalized shape increments from the shape indexed features. Since strong fitting capacity is vital to the final performance, we again exploit the boosted regression to form the stage regressor. To distinguish it from the external-level shape regression framework, we term it as internal-level boosted regression. With the prepared ingredients in previous three sections, we are ready to describe how to learn the internal-level boosted regression.

The internal-boosted regression consists of K primitive regressors $\{r_1, ..., r_K\}$, which are in fact ferns. In testing, we combine them in an additive manner to predict the output. In training, the primitive regressors are sequentially learnt to greedily fit the regression targets, in other words, each primitive regressor handles the residues left by previous regressors. In each iteration, the residues are used as the new targets for learning a new primitive regressor. The learning procedure are essentially identical for all primitive regressors, which can be describe as follows.

- *Features* Selecting F pixel-difference features using correlation based feature selection method.
- *Thresholds* Randomly sampling F i.i.d. thresholds from an uniform distribution.[4]
- *Outputs* Partitioning all training samples into different bins using the learnt features and thresholds. Then, learning the outputs of the bins using Eq. (7).

Algorithm 4 Internal-level boosted regression

Variables: regression targets $Y \in \Re^{N\times 2N_{\mathrm{fp}}}$; training images and corresponding estimated shapes $\{I_i, S_i\}_{i=1}^{N}$; training parameters $TrainParams\{N_{\mathrm{fp}}, P, \kappa, F, K\}$; the stage regressor R; testing image and corresponding estimated shape $\{I, S\}$;

LearnStageRegressor$(Y, \{I_i, S_i\}_{i=1}^{N}, TrainParams)$
$\{\Delta_\alpha^{l_\alpha}\}_{\alpha=1}^{P} \leftarrow GenerateLocalCoordinates(N_{\mathrm{fp}}, P, \kappa)$
$\rho \leftarrow ExtractShapeIndexedPixels(\{I_i, S_i\}_{i=1}^{N}, \{\Delta_\alpha^{l_\alpha}\}_{\alpha=1}^{P})$
$\mathrm{cov}(\rho) \leftarrow$ pre-compute pixel-pixel covariance
$Y^0 \leftarrow Y$ // initialization
for k from 1 to K
  $\{\rho_{m_\mathrm{f}} - \rho_{n_\mathrm{f}}\}_{\mathrm{f}=1}^{F}, \{m_\mathrm{f}, n_\mathrm{f}\}_{\mathrm{f}=1}^{F} \leftarrow$
    $CorrelationBasedFeatureSelection(Y^{k-1}, \mathrm{cov}(\rho), F)$
  $\{\theta_\mathrm{f}\}_{\mathrm{f}=1}^{F} \leftarrow$ sample F thresholds from an uniform distribution
  $\{\Omega_b\}_{b=1}^{2^F} \leftarrow$ partition training samples into 2^F bins
  $\{y_b\}_{b=1}^{2^F} \leftarrow$ compute the outputs of all bins using Eq. (7)
  $r_k \leftarrow \{\{m_\mathrm{f}, n_\mathrm{f}\}_{\mathrm{f}=1}^{F}, \{\theta_\mathrm{f}\}_{\mathrm{f}=1}^{F}, \{y_b\}_{b=1}^{2^F}\}$ // construct a fern
  $Y^k \leftarrow Y^{k-1} - r^k(\{\rho_{m_\mathrm{f}} - \rho_{n_\mathrm{f}}\}_{\mathrm{f}=1}^{F})$ // update the targets
$R \leftarrow \{\{r^k\}_{k=1}^{K}, \{\Delta_\alpha^{l_\alpha}\}_{\alpha=1}^{P}\}$ // construct stage regressor
return R

ApplyStageRegressor(I, S, R) // i.e. $R(I, S)$
$\rho \leftarrow ExtractShapeIndexedPixels(\{I, S\}, \{\Delta_\alpha^{l_\alpha}\}_{\alpha=1}^{P})$
$\delta S \leftarrow \mathbf{0}$
for k from 1 to K
  $\delta S \leftarrow \delta S + r^k(\{\rho_{m_\mathrm{f}} - \rho_{n_\mathrm{f}}\}_{\mathrm{f}=1}^{F})$
return δS

[4] Provided the range of pixel difference feature is $[-c, c]$, the range of the uniform distribution is $[-0.2c, 0.2c]$.

2.8 Model Compression

By memorizing the fitting procedures in the regression model, our method gains very efficient testing speed. However, comparing with optimization based approaches, our regression based method leads higher storage cost. The stored delta shapes in the ferns' leaves contributes to the main cost in the total storage, which can be quantitatively computed via $8N_{\text{fp}}TK2^F$, where $TK2^F$ gives the number of all leaves of TK random ferns in T stages, $8N_{\text{fp}}$ is the storage of a delta shape in a single leaf. For example, provided $T = 10$, $K = 500$, $F = 5$ and $N_{\text{fp}} = 194$, the storage is around 240mb, which is unaffordable in many real scenarios such as mobile phones and embedded devices.

A straight forward way to compress model size is via principal component analysis (Jolliffe 2005). However, since PCA only exploits orthogonal undercompleted basis in coding, its compression capability is limited. To further compress the model size, we adopt sparse coding which not only takes the advantage of overcompleted basis but also enforces sparse regularization in its objective.

Formally, the objective of sparse coding in a certain leave of fern can be expressed as follows:

$$\min_{x} \sum_{i \in \Omega_b} \|\hat{y}_i - Bx\|_2^2, \qquad s.t. \|x\|_0 \leq Q, \tag{12}$$

where the $\hat{y}_i$ is the regression target; B is the basis for sparse coding; Q the is the upper bound of the number of non-zero codes ($Q = 5$ in our implementation).

The learning procedure of our model compression method consists of two steps in each stage. In the first step, the basis B of this stage is constructed. To learn the basis, we first run the non-sparse version of our method to obtain K ferns of this stage as well as the outputs stored in their leaves. Then the basis is constructed by random sampling[5] from the outputs in the leaves.

In the second step, we use the same method in Sect. 2.5 to learn the structure of a fern. For each leaf, sparse codes are computed by optimizing the objective in Eq. (12) using the orthogonal matching pursuit method (Tropp and Gilbert 2007). Comparing with the shape increment, sparse codes require much less storage in leaves, especially when the dimension of the shape is high.

In testing, for K ferns in one stage, their non-zero codes in the leaves which are visited by the testing sample are linearly summed up to form the final coefficient x, then the output is computed via Bx, which contributes to the main computation. It is worth noting although the training of the compressed version is more time-consuming, the testing is still as efficient as the original version.

[5] We use random sampling for basis construction due to its simplicity and effectiveness. We also tried more sophisticated K-SVD method (Elad and Aharon 2006) for learning basis. It yields similar performance comparing with random sampling.

3 Implementation Details

We discuss more implementation details, including the shape initialization in training and testing, parameter setting and running performance.

3.1 Initialization

As aforementioned, we generate a initial shape by sampling from an *InitSet* which contains exemplar shapes. The exemplar shapes could be representative shapes selected from the training data, or the groundtruth shapes of the training data $\{\hat{S}_i\}_{i=1}^N$. In training, we choose the later as the *InitSet*. While, in testing, we use the first for its storage efficiency. For one image, we draw the initial shapes without replacement to avoid duplicated samples. The scale and translation of the sampled shape should be adjusted to ensure that the corresponding face rectangle is the same with the face rectangle of the input facial image.

3.2 Training Data Augmentation

Each training sample consists of a training image, an initial shape and a ground truth shape. To achieve better generalization ability, we augment the training data by sampling multiple initializations (20 in our implementation) for each training image. This is found to be very effective in obtaining robustness against large pose variation and rough initial shapes during the testing.

3.3 Multiple Initializations in Testing

The regressor can give reasonable results with different initial shapes for a test image and the distribution of multiple results indicates the confidence of estimation. As shown in Fig. 3,

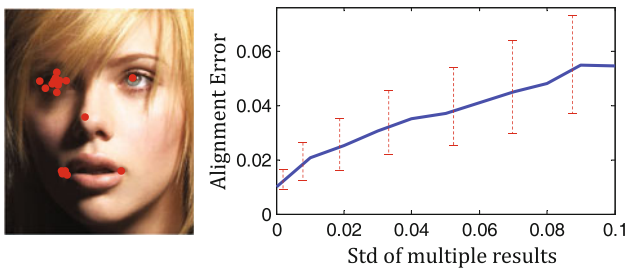


Fig. 3 *Left* results of 5 facial landmarks from multiple runs with different initial shapes. The distribution indicates the estimation confidence: *left* eye and *left* mouth corner estimations are widely scattered and less stable, due to the local appearance noises. *Right* the average alignment error increases as the standard deviation of multiple results increases

Table 1 Training and testing times of our approach, measured on an Intel Core i7 2.93GHz CPU with C++ implementation

Landmarks	5	29	87
Training (min)	5	10	21
Testing (ms)	0.32	0.91	2.9

when multiple landmark estimations are tightly clustered, the result is accurate, and vice versa. In the test, we run the regressor several times (5 in our implementation) and take the median result[6] as the final estimation. Each time the initial shape is randomly sampled from the training shapes. This further improves the accuracy.

3.4 Running Time Performance

Table 1 summarizes the computational time of training (with 2,000 training images) and testing for different number of landmarks. Our training is very efficient due to the fast feature selection method. It takes minutes with 40,000 training samples (20 initial shapes per image), The shape regression in the test is extremely efficient because most computation is pixel comparison, table look up and vector addition. It takes only 15 ms for predicting a shape with 87 landmarks (3 ms $\times$ 5 initializations).

3.5 Parameter Settings

The number of features in a fern F and the shrinkage parameter β adjust the trade off between fitting power in training and generalization ability in testing. They are set as $F = 5$, β =1,000 by cross validation.

Algorithm accuracy consistently increases as the number of stages in the two-level boosted regression (T, K) and number of candidate features P^2 increases. Such parameters are empirically chosen as $T = 10, K = 500, P = 400$ for a good tradeoff between computational cost and accuracy.

The parameter κ is used for generating the local coordinates relative to landmarks. We set κ equal to 0.3 times of the distance between two pupils on the mean shape.

4 Experiments

The experiments are performed in two parts. The first part compares our approach with previous works. The second part validates the proposed approach and presents some interesting discussions.

We briefly introduce the datasets used in the experiments. They present different challenges, due to different numbers of annotated landmarks and image variations.

BioID The dataset was proposed by Jesorsky et al. (2001) and widely used by previous methods. It consists of 1,521 near frontal face images captured in a lab environment, and is therefore less challenging. We report our result on it for completeness.

LFPW The dataset was created by Belhumeur et al. (2011). Its full name is Labeled Face Parts in the Wild. The images are downloaded from internet and contain large variations in pose, illumination, expression and occlusion. It is intended to test the face alignment methods in unconstraint conditions. This dataset shares only web image URLs, but some URLs are no longer valid. We only downloaded 812 of the 1,100 training images and 249 of the 300 test images. To acquire enough training data, we augment the training images to 2,000 in the same way as Belhumeur et al. (2011) did and use the available testing images.

LFW87 The dataset was created by Liang et al. (2008). The images mainly come from the Labeled Face in the Wild (LFW) dataset (Huang et al. 2008), which is acquired from uncontrolled conditions and is widely used in face recognition. In addition, it has 87 annotated landmarks, much more than that in BioID and LFPW, therefore, the performance of an algorithm relies more on its shape constraint. We use the same setting in Liang et al. (2008)'s work: the training set contains 4,002 images mainly from LFW, and the testing set contains 1,716 images which are all from LFW.

Helen The dataset was proposed by Le et al. (2012). It consists of 2,330 *high resolution* web images with 194 annotated landmarks. The average size of face is 550 pixels. Even the smallest face in the dataset is larger than 150 pixels. It serves as a new benchmark which provides richer and more detailed information for accurate face alignment.

4.1 Comparison with previous works

For comparisons, we use the alignment error in Eq. (1) as the evaluation metric. To make it invariant to face size, the error is not in pixels but normalized by the distance between the two pupils, similar to most previous works.

The following comparison shows that our approach outperforms the state of the art methods in both accuracy and efficiency, especially on the challenging LFPW and LFW87 datasets. Figures 4, 5, 6 and 7 show our results on challenging examples with large variations in pose, expression, illumination and occlusion from the four datasets.

[6] The median operation is performed on x and y coordinates of all landmarks individually. Although this may violate the shape constraint mentioned before, the resulting median shape is mostly correct as in most cases the multiple results are tightly clustered. We found such a simple median based fusion is comparable to more sophisticated strategies such as weighted combination of input shapes.

Fig. 4 Selected results from LFPW

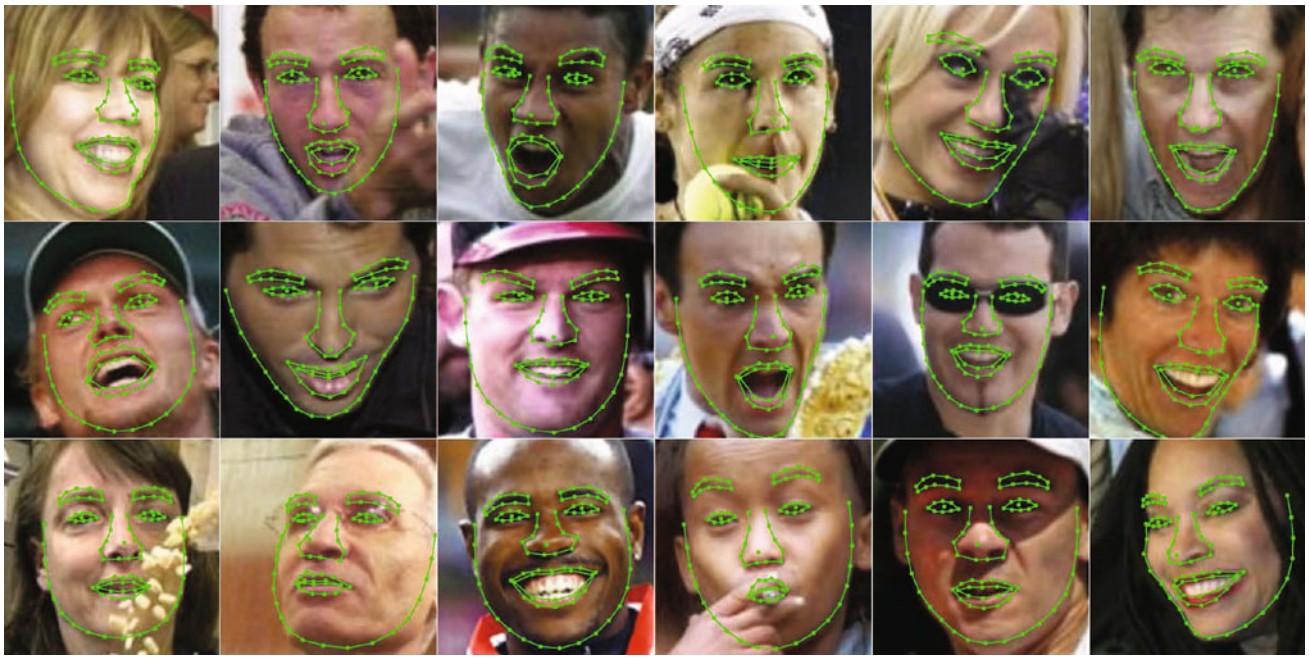


Fig. 5 Selected results from LFW87

4.1.1 Comparison on LFPW

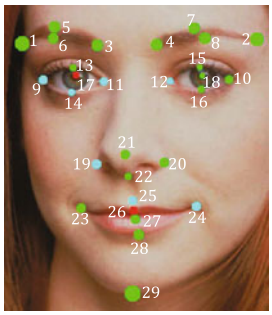
The consensus exemplar approach proposed by Belhumeur et al. (2011) is one of the state of the art methods. It was the best on BioID when published, and obtained good results on LFPW.

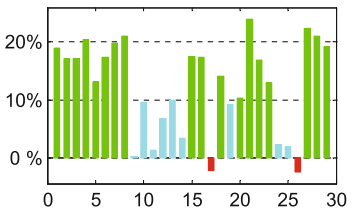
Comparison in Fig. 8 shows that most landmarks estimated by our approach are more than 10 % accurate[7] than

[7] The relative improvement is the ratio between the error reduction and the original error.

Fig. 6 Selected results from BioID

Fig. 7 Selected results from Helen dataset





Mean Error($\times 10^{-2}$)	
CE	Our Method
3.99	**3**.43

Fig. 8 Results on the LFPW dataset. *Left* 29 facial landmarks. The *circle* radius is the average error of our approach. *Point* color represents relative accuracy improvement over the results of the consensus exemplars(CE) method proposed by Belhumeur et al. (2011). *Green* more than 10 % more accurate. *Cyan* 0 to 10 % more accurate. *Red* less accurate. *Right top* relative accuracy improvement of all landmarks over the results of CE method. *Right bottom* average error of all landmarks (Color figure online)

the method proposed by Belhumeur et al. (2011) and our overall error is smaller.

In addition, our method is *thousands of times faster*. It takes around 5ms per image (0.91 × 5 initializations for 29 landmarks). The method proposed by Belhumeur et al. (2011) uses expensive local landmark detectors (SIFT+SVM) and it takes more than 10 s[8] to run 29 detectors over the entire image.

4.1.2 Comparison on LFW87

Liang et al. (2008) proposed a component-based discriminative search (CDS) method which trains a set of direction classifiers for pre-defined facial components to guide the ASM search direction. Their algorithm outperform previous ASM and AAM based works by a large margin.

We use the same root mean square error (RMSE) used in CDS (Liang et al. 2008) as the evaluation metric. Table 2 shows our method is significantly better. For the strict error threshold (5 pixels), the error rate is reduced nearly by half, from 25.3 to 13.9 %. The superior performance on a large number of landmarks verifies the effectiveness of proposed holistic shape regression and the encoded adaptive shape constraint.

[8] Belhumeur et al. (2011) discussed in their work: "The localizer requires less than 1 s per fiducial on an Intel Core i7 3.06GHz machine". We conjecture that it takes more than 10 s to locate 29 landmarks.

Table 2 Percentages of test images with root mean square error (RMSE) less than given thresholds on the LFW87 dataset

RMSE	<5 Pixels	<7.5 Pixels	<10 Pixels
CDS (%)	74.7	93.5	97.8
Our method (%)	**86.1**	**95.2**	**98.2**

Bold values represent the best results under certain settings

Table 3 Comparison on Helen dataset

Method	Mean	Median	Min	Max
STASM	0.111	0.094	0.037	0.411
CompASM	0.091	0.073	0.035	0.402
Our method	**0.057**	**0.048**	**0.024**	**0.16**

The error of each sample is first individually computed by averaging the errors of 194 landmarks, and then the mean error across all testing samples is computed
Bold values represent the best results under certain settings

4.1.3 Comparison on Helen

We adopt the same training and testing protocol as well as the same error metric used by Le et al. (2012). Specifically, we divide the Helen dataset into training set of 2,000 images and testing set of 330 images. As the pupils are not labeled in the Helen dataset, the distance between the centroids of two eyes are used to normalize the deviations from groundtruth.

We compare our method with STASM (Milborrow and Nicolls 2008) and recently proposed CompASM (Le et al. 2012). As shown in Table 3, our method outperforms them by a large margin. Comparing with STASM and CompASM, our method reduces the mean error by 50 and 40 % respectively, meanwhile, the testing speed is even faster.

4.1.4 Comparison to Previous Methods on BioID

Our model is trained on augmented LFPW training set and tested on the entire BioID dataset.

Figure 9 compares our method with previous methods (Vukadinovic and Pantic 2005; Cristinacce and Cootes 2006; Milborrow and Nicolls 2008; Valstar et al. 2010; Belhumeur et al. 2011). Our result is the best but the improvement is marginal. We believe this is because the performance on BioID is nearly maximized due to its simplicity. Note that our method is thousands of times faster than the second best method (Belhumeur et al. 2011).

4.2 Algorithm Validation and Discussions

We verify the effectiveness of different components of the proposed approach. Such experiments are performed on the our augmented LFPW dataset. The dataset is split into two parts for training and testing. The training set contains 1,500 images and the testing set contains 500 images. Parameters are fixed as in Sect. 3, unless otherwise noted.

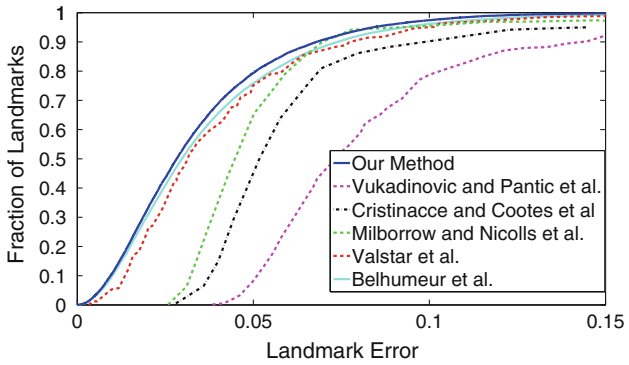


Fig. 9 Cumulative error curves on the BioID dataset. For comparison with previous results, only 17 landmarks are used (Cristinacce and Cootes 2006). As our model is trained on LFPW images, for those landmarks with different definitions between the two datasets, a fixed offset is applied in the same way in Belhumeur et al. (2011)

Table 4 Tradeoffs between two levels boosted regression

♯ Stage regressors (T)	1	5	10	100	5000
♯ Primitive regressors (K)	5000	1000	500	50	1
Mean error ($\times 10^{-2}$)	15	6.2	**3.3**	4.5	5.2

Bold value represents the best results under certain settings

4.2.1 Two-Level Boosted Regression

As discussed in Sect. 2, the stage regressor exploits shape indexed features to obtain geometric invariance and decompose the original difficult problem into easier sub-tasks. The shape indexed features are fixed within the internal-level boosted regression to avoid instability.

Different tradeoffs between two-level boosted regression are presented in Table 4, using the same number of ferns. On one extreme, regressing the whole shape in a single stage ($T = 1, K = 5000$) is clearly the worst. On the other extreme, using a single fern as the stage regressor ($T = 5000, K = 1$) also has poor generalization ability in the test. The optimal tradeoff ($T = 10, K = 500$) is found in between via cross validation.

4.2.2 Shape Indexed Feature

We compare the global and local methods of shape indexed features. The mean error of local index method is 0.033, which is much smaller than the mean error of global index method 0.059. The superior accuracy supports the proposed local index method.

Table 5 Comparison between correlation based feature selection (CBFS) method and best-of-*n* feature selection methods

Best-of-*n*	$n = 1$	$n = 32$	$n = 1024$	CBFS
Error ($\times 10^{-2}$)	5.01	4.92	4.83	**3.32**
Time (s)	0.1	3.0	100.3	0.12

The training time is for one primitive regressor
Bold value represents the best results under certain settings

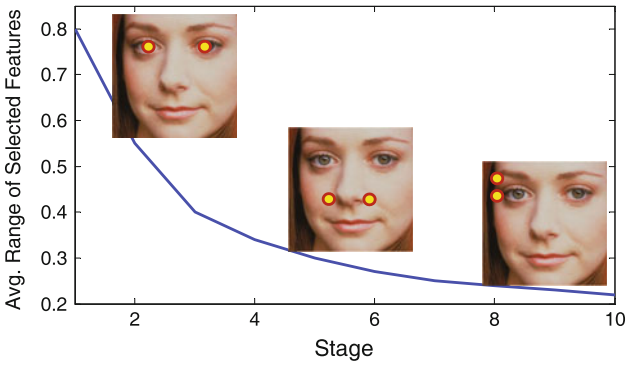


Fig. 10 Average ranges of selected features in different stages. In stage 1, 5 and 10, an exemplar feature (a pixel pair) is displayed on an image

4.2.3 Feature Selection

The proposed correlation based feature selection method (CBFS) is compared with the commonly used *best-of-n* method (Ozuysal et al. 2010; Dollar et al. 2010) in Table 5. CBFS can select good features rapidly and this is crucial to learn good models from large training data.

4.2.4 Feature Range

The *range* of a feature is the distance between the pair of pixels normalized by the distance between the two pupils. Figure 10 shows the average ranges of selected features in the 10 stages. As observed, the selected features are adaptive to the different regression tasks. At first, long range features (e.g., one pixel on the mouth and the other on the nose) are often selected for rough shape adjustment. Later, short range features (e.g., pixels around the eye center) are often selected for fine tuning.

4.2.5 Model Compression

We conduct experiments on both LFW87 and Helen datasets to compare the sparse coding(SC) based compression method with PCA based compression method.

For sparse coding based method, the number of non-zero codes is 5 and the number of basis is 512. For PCA based method, The principle components are computed by preserving 95 % energy.

Table 6 Model compression experiment

	Dataset	Raw	PCA	SC
Mean error ($\times 10^{-2}$)	LFW87	4.23	4.35	4.34
Model size (mb)	LFW87	118	30	8
Comp. ratio	LFW87	–	4	15
Mean error ($\times 10^{-2}$)	Helen194	5.70	5.83	5.79
Model size (mb)	Helen194	240	42	12
Comp. ratio	Helen194	–	6	20

The suffix of the name of the dataset means the number of annotated landmarks

As shown in Table 6, the sparse coding based method outperforms the PCA based method both in the sense of compression ratio and accruacy. For example, on Helen dataset, the sparse coding based method archives 20 times compression. In contrast, the PCA based method achieves only 6 times compression at the cost of even lager mean error.

5 Discussion and Conclusion

We have presented the explicit shape regression method for face alignment. By jointly regressing the entire shape and minimizing the alignment error, the shape constraint is automatically encoded. The resulting method is highly accurate, efficient, and can be used in real time applications such as face tracking. The explicit shape regression framework can also be applied to other problems like articulated object pose estimation and anatomic structure segmentation in medical images.

References

Belhumeur, P., Jacobs, D., Kriegman, D., & Kumar, N. (2011). Localizing parts of faces using a concensus of exemplars. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Bingham, E., & Mannila, H. (2001). *Random projection in dimensionality reduction: Applications to image and text data*. In *ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)* .

Cootes, T., Edwards, G., & Taylor, C. (2001). Active appearance models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *23*(6), 681–685.

Cootes, T., Taylor, C., Cooper, D., Graham, J., et al. (1995). Active shape models-their training and application. *Computer Vision and Image Understanding*, *61*(1), 38–59.

Cristinacce, D., & Cootes, T. (2006). Feature detection and tracking with constrained local models. In *British Machine Vision Conference (BMVC)*.

Cristinacce, D., & Cootes, T. (2007). Boosted regression active shape models. In *British Machine Vision Conference (BMVC)*.

Dollar, P., Welinder, P., & Perona, P. (2010). Cascaded pose regression. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Duffy, N., & Helmbold, D. P. (2002). Boosting methods for regression. *Machine Learning*, *47*(2–3), 153–200.

Elad, M., & Aharon, M. (2006). Image denoising via sparse and redundant representations over learned dictionaries. *IEEE Transactions on Image Processing*, *15*(12), 3736–3745.

Friedman, J. H. (2001). Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, *29*(5), 1189–1232.

Huang, G., Mattar, M., Berg, T., Learned-Miller, E. et al. (2008) Labeled faces in the wild: A database forstudying face recognition in unconstrained environments. In *Workshop on Faces in'Real-Life'Images: Detection, Alignment, and Recognition*.

Jesorsky, O., Kirchberg, K. J., & Frischholz, R. W. (2001). *Robust face detection using the hausdorff distance* (pp. 90–95). New York: Springer.

Jolliffe, I. (2005). *Principal component analysis*. Wiley Online Library.

Le, V., Brandt, J., Lin, Z., Bourdev, L., & Huang, T. (2012). Interactive facial feature localization. In *European Conference on Computer Vision*.

Liang, L., Xiao, R., Wen, F., & Sun, J. (2008). Face alignment via component-based discriminative search. In *European Conference on Computer Vision (ECCV)*.

Matthews, I., & Baker, S. (2004). Active appearance models revisited. *International Journal of Computer Vision*, *60*(2), 135–164.

Milborrow, S., & Nicolls, F. (2008). Locating facial features with an extended active shape model. In *European Conference on Computer Vision (ECCV)*.

Ozuysal, M., Calonder, M., Lepetit, V., & Fua, P. (2010). Fast keypoint recognition using random ferns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *32*(3), 448–461.

Saragih, J., & Goecke, R. (2007). A nonlinear discriminative approach to aam fitting. In *International Conference on Computer Vision (ICCV)* .

Sauer, P., & Cootes, C. T. T. (2011). Accurate regression procedures for active appearance models. In *British Machine Vision Conference (BMVC)*.

Shotton, J., Fitzgibbon, A., Cook, M., Sharp, T., Finocchio, M., Moore, R., et al. (2011). Real-time human pose recognition in parts from single depth images. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Sun, Y., Wang, X., & Tang, X. (2013). Deep convolutional network cascade for facial point detection. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Tropp, J., & Gilbert, A. (2007). Signal recovery from random measurements via orthogonal matching pursuit. *IEEE Transactions on Information Theory*, *53*(12), 4655–4666.

Valstar, M., Martinez, B., Binefa, X., & Pantic, M. (2010). Facial point detection using boosted regression and graph models. In *IEEE Conference on Computeer Vision and Pattern Recognition (CVPR)*.

Vukadinovic, D., & Pantic, M. (2005). Fully automatic facial feature point detection using gabor feature based boosted classifiers. *International Conference on Systems, Man and Cybernetics*, *2*, 1692–1698.

Xiong, X., De la Torre, F. (2013) Supervised descent method and its applications to face alignment. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Zhou, S. K., & Comaniciu, D. (2007). Shape regression machine. In *Information Processing in Medical Imaging*, (pp. 13–25). Heidelberg: Springer.