Learning Deformable Shape Manifolds

Samuel Rivera and Aleix Martinez

Abstract

We propose an approach to shape detection of highly deformable shapes in images via manifold learning with regression. Our method does not require shape key points be defined at high contrast image regions, nor do we need an initial estimate of the shape. We only require sufficient representative training data and a rough initial estimate of the object position and scale. We demonstrate the method for face shape learning, and provide a comparison to nonlinear Active Appearance Model. Our method is extremely accurate, to nearly pixel precision and is capable of accurately detecting the shape of faces undergoing extreme expression changes. The technique is robust to occlusions such as glasses and gives reasonable results for extremely degraded image resolutions.

Keywords: Shape modeling, detailed face shape detection, face detection, nonlinear regression, face recognition, manifold learning.

1. Introduction

Shape detection is an important problem in computer vision because of its utility in object recognition, classification, and segmentation among others [1, 2, 3, 4]. The general shape detection problem can be stated as follows: Given an image, can we delineate the shape of a specific object in the image? The problem becomes difficult when the shape is not rigid, but can deform, translate, rotate, change scale, or become occluded in the image. In this paper, we are interested in this scenario.

We develop a new method for deformable shape detection based on manifold learning through regression. We illustrate the concept applied to face shape detection in Fig. 1. Consider this simplified illustration of the model where the u and v axes correspond to the face *image* space, while the z axis corresponds to the face *shape* space. Given a finite set of face image samples

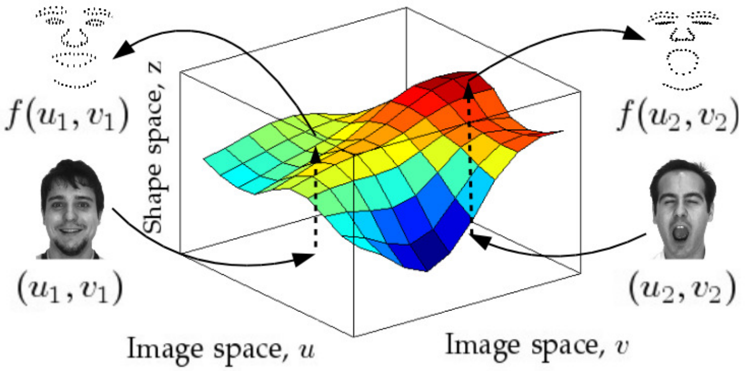


Figure 1: Conceptual illustration of the face shape manifold and the method presented. The u and v axes correspond to the face *image* space, while the z axis corresponds to the face *shape* space. A finite set of face image samples and their associated shape parameters are used to estimate the nonlinear manifold. This manifold defines a mapping $f(\cdot)$ from a face image sample to the associated shape parameters.

and their associated shape parameters, we wish to estimate the nonlinear face shape manifold so that we can interpolate the shape of new samples in the face image space. Once learned, the manifold provides a direct mapping $f(\cdot)$ from a new image $\mathbf{x} \in \mathbb{R}^p$ to the shape space $\mathbf{y} \in \mathbb{R}^d$, where p and d are the number of image features and the number of shape parameters, respectively. Hence, in contrast to most methods in shape detection and modeling which iteratively fit a model to an image until convergence, the shape estimate is given in a single step.

One of the first successful shape detection algorithms requiring an iterative approach was developed by Kass *et al.* [1] who utilized energy minimization to deform active contour models, called *snakes*, to fit salient image features. Thus, this method requires shapes be defined by high contrast regions. Additional constraints of how much the shape can deform (e.g., based on smoothness) are incorporated, and with a reasonable initialization of the shape, the model can deform to extract the shape of the object. A logical extension of snakes was to change the smoothness constraint of the shape for one that defines the variabilities of the object we want to model. Cootes *et al.* [3] developed on this idea with models that could only deform in ways specific to a given shape class. The shape variability was modeled using a probability density function (pdf) learned by manually delineating shapes in sample images of the object. This model is called the Active Shape

Model (ASM). ASM works well but still requires high contrast regions such as edges for fitting the active contour. To overcome this drawback, Cootes *et al.* defined the Active Appearance Model (AAM) [5], where the density of the texture is also learned from sample images. The algorithm finds the shape which best fits to the set of possible textures given by the learned pdf. This method was enhanced by using boosting to learn the shape parameter update and confidence score [6]. Other authors take a Bayesian approach to shape modeling, learning the conditional density of the shape parameters given the object image, and iterating to the maximum a posteriori (MAP) estimate of the shape parameters [7, 8, 9]. Liang *et al.* [10] improve on the idea by using regularization at accurately aligned points to reduce the occurence of local minima favored by the global shape model. Zhang *et al.* [11] further develop on this Bayesian approach by using regression to learn a sequence of unimodal conditional density functions which guide the shape estimate toward the correct solution.

Another alternative is to train a set of classifiers to detect various face fiducials and inter-connect them to estimate the shape [12]. In this case, a sliding window approach is used, where the classifiers are evaluated at all positions and scales of interest followed by pruning and voting for the final detection. This approach can provide very accurate results if high resolution images are available, but the sliding window method is computationally demanding.

Zhou and Comaniciu changed direction with Shape Regression Machine (SRM) and utilized nonlinear regression to segment the left ventricular endocardium in highly structured images [13, 14]. SRM employs boosting with an over-complete feature bank to train a strong learner which associates an image with a shape. A strong learner is a linear combination of weak learners which coorespond to the outputs of local feature extractors. Detecting shape using such an approach is advantageous because it avoids the initialization and iteration posed by the above methods. Regression has also been used by Cristinacce and Cootes [15] to model imprecisely detected fiducials in faces. Imprecisely detected fiducials had previously been modeled using a pdf [16].

Our approach is related to SRM in that both use nonlinear regression to relate an image with a shape, but there are fundamental differences. Our model uses kernel regression with global object appearance while SRM uses boosted regression with local features. Our experiments show that the local approach is not as effective in the low resolution setting as the holistic approach. Furthermore, SRM is proposed for medical image segmentation so objects are normalized according to an estimated scale and rotation parameter. In this work, images are normalized according to the estimated eye positions which is more appropriate in the context of faces.

The approach proposed in the present paper and illustrated in Fig. 1 has the positive aspects of the above approaches while eliminating some of the drawbacks:

1. As the discriminative approaches, the method is non-iterative.
2. As the generative approaches, the method does not require a sliding window.
3. The method does not require strong shape contours so the shape manifold can be learned at extremely low resolutions.
4. The manifold is based on a specific shape model, so it will give a reasonable shape estimate even in the case of large occlusions, deformations, or other image changes.

In the current work, we apply the method to face shapes, but the ideas can be generalized to other deformable shapes.

The remainder of this paper is organized as follows. In Section 2 we summarize regression and in particular, the methods Kernel Ridge Regression and Support Vector Regression. Section 3 describes our methodology in detail. Section 4 describes several experiments which demonstrate the method's ability to detect the shape of realistic face images with occlusions, and at extremely low resolution. We close the article with our conclusions in Section 5.

2. Regression

Regression allows us to find the functional relationship between some predictor variables $\mathbf{x} \in \mathbb{R}^p$ and an associated output $\mathbf{y} \in \mathbb{R}^d$ [17]. Given $\mathbf{x}$ and $\mathbf{y}$ from unknown distributions, we want to find the mapping function $f : \mathbf{x} \to \mathbf{y}$ which minimizes the expected risk,

$$\mathbf{E}[L(f(\mathbf{x}), \mathbf{y})],$$

where $L(f(\mathbf{x}), \mathbf{y})$ is an appropriate cost function which penalizes the deviations between $f(\mathbf{x})$ and $\mathbf{y}$. Since we do not know the underlying distribution of the independent and dependent variables, we generally minimize the empirical risk. Given a training set $(\mathbf{x}_i, \mathbf{y}_i), i = 1, 2, \ldots, n$, the empirical risk is given by

$$\frac{1}{n}\sum_{i=1}^{n} L(f(\mathbf{x}_i), \mathbf{y}_i).$$

Preliminary experiments showed that a simple linear model was insufficient for representing the manifold illustrated in Fig. 1 accurately. We thus now turn our attention to nonlinear regression.

2.1. Kernel Ridge Regression

Kernel Ridge Regression (KRR) is the kernel extension of Ridge Regression (RR), which is a penalized version of linear least squares regression [18]. RR minimizes the cost function,

$$L(\mathbf{W}) = \frac{1}{n}\sum_{i=1}^{n} \|\mathbf{y}_i - \mathbf{W}^T\mathbf{x}_i\|_F^2 + \lambda\|\mathbf{W}\|_F^2,$$

where λ is a user determined regularization parameter, $\|.\|_F$ denotes the Frobenius norm, and $\mathbf{W} \in \mathbb{R}^{p\times d}$.

If we create a matrix $\mathbf{X} \in \mathbb{R}^{n\times p}$ where each row is one of the vectors from the training input and a matrix $\mathbf{Y} \in \mathbb{R}^{n\times d}$ with the associated output values, the solution is given by

$$\arg\min_{\mathbf{W}} L(\mathbf{W}) = (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I}_p)^{-1}\mathbf{X}^T\mathbf{Y}, \tag{1}$$

where $\mathbf{I}_p$ is the $p \times p$ identity matrix. Our regressed function is then $f(\mathbf{x}) = \mathbf{W}^T\mathbf{x}$, where $\mathbf{x}$ is the input vector.

It has been shown [19] that we can extend this method to the nonlinear case through the use of kernels. The solution for the regressed function is given by

$$f(\mathbf{x}) = \mathbf{Y}^T(\mathbf{K} + \lambda\mathbf{I}_n)^{-1}\kappa(\mathbf{x}), \tag{2}$$

where $\mathbf{K} \in \mathbb{R}^{n\times n}$ is the Gram matrix of the training data. The entries of the Gram matrix are given by $\mathbf{K}_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$, where $k(\cdot,\cdot)$ a Mercer kernel [20] and $\kappa(\mathbf{x}) = (k(\mathbf{x}, \mathbf{x}_1),\ k(\mathbf{x}, \mathbf{x}_2),\ \ldots,\ k(\mathbf{x}, \mathbf{x}_n))^T$. We use the popular Radial Basis Function (RBF) kernel given by

$$k_\sigma(\mathbf{x}, \mathbf{x}_i) = \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}_i\|_2^2}{2\sigma^2}\right),$$

where σ is a parameter to tune and $\|.\|_2$ denotes the *Euclidean*-norm. We denote the σ used in KRR by σ_K. Note that multiple KRR assumes the output values, the shape parameters, are uncorrelated. Section 3.3 describes the shape model used to achieve this property.

2.2. ε-Support Vector Regression

ε-Support Vector Regression (ε-SVR) is another linear regression method which finds a linear function

$$f(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b,$$

where $\mathbf{w} \in \mathbb{R}^p$ and b is a scalar, such that the difference between the regressed output and the true output is below ε for all the training data while keeping $f(\cdot)$ as smooth as possible. Smoothness is achieved through regularization by penalizing $\|\mathbf{w}\|_2^2$ [21].

The underlying assumption of the algorithm is that there exists a function $f(\cdot)$ which can correctly predict training data with ε precision. To mitigate this assumption, one can introduce real valued scalar slack variables ξ_i and ξ_i^* to the above definition.

It has been shown [22] that this problem can be formulated as the following convex optimization problem,

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}\|\mathbf{w}\|_2^2 + C\sum_{i=1}^{n}(\xi_i + \xi_i^*) \\ \text{subject to} & \left\{ \begin{array}{lcl} y_i - \mathbf{w}^T\mathbf{x}_i - b & \leq & \varepsilon + \xi_i \\ \mathbf{w}^T\mathbf{x}_i + b - y_i & \leq & \varepsilon + \xi_i^* \\ \xi_i, \xi_i^* & \geq & 0, \end{array} \right. \end{array} \tag{3}$$

where $C > 0$ is a constant which controls the trade-off between the smoothness of $f(\cdot)$ and the allowable error greater than ε, and y_i is the scalar output associated with $\mathbf{x}_i$. Instead of solving (3) directly, it is easier to solve the dual problem:

$$\begin{array}{ll} \text{maximize} & \left\{ \begin{array}{l} -\frac{1}{2}\sum_{i,j=1}^{n}(\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*)\mathbf{x}_i^T\mathbf{x}_j \\ -\varepsilon\sum_{i=1}^{n}(\alpha_i - \alpha_i^*) + \sum_{i=1}^{n} y_i(\alpha_i + \alpha_i^*) \end{array} \right. \\ \text{subject to} & \sum_{i=1}^{n}(\alpha_i - \alpha_i^*) = 0 \text{ and } \alpha_i, \alpha_i^* \in [0, C], \end{array} \tag{4}$$

where α_i and α_i^* are the dual variables [23].

The dual problem can be solved using quadratic programming, yielding

$$\begin{aligned} \mathbf{w} &= \sum_{i=1}^{n}(\alpha_i - \alpha_i^*)\mathbf{x}_i, \\ f(\mathbf{x}) &= \sum_{i=1}^{n}(\alpha_i - \alpha_i^*)\mathbf{x}_i^T\mathbf{x} + b. \end{aligned} \tag{5}$$

The value of b follows since it must satisfy the Karush-Kuhn-Tucker (KKT) conditions [23]:

$$\begin{aligned} \alpha_i(\varepsilon + \xi_i - y_i + \mathbf{w}^T\mathbf{x}_i + b) &= 0, \\ \alpha_i^*(\varepsilon + \xi_i^* - y_i + \mathbf{w}^T\mathbf{x}_i - b) &= 0. \end{aligned}$$

Notice that in (4) and (5), the function input and training samples, $\mathbf{x}$ and $\mathbf{x}_i$, only appear as inner products in the original feature space. Therefore, the method can be extended to the nonlinear case through the use of a Mercer kernel, $k(\cdot, \cdot)$. The dual problem then becomes

$$\begin{aligned} &\text{maximize} && \begin{cases} -\dfrac{1}{2}\displaystyle\sum_{i,j=1}^{n}(\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*)k(\mathbf{x}_i, \mathbf{x}_j) \\ -\varepsilon\displaystyle\sum_{i=1}^{n}(\alpha_i - \alpha_i^*) + \sum_{i=1}^{n} y_i(\alpha_i + \alpha_i^*) \end{cases} \\ &\text{subject to} && \sum_{i=1}^{n}(\alpha_i - \alpha_i^*) = 0 \text{ and } \alpha_i, \alpha_i^* \in [0, C], \end{aligned} \tag{6}$$

and

$$f(\mathbf{x}) = \sum_{i=1}^{n}(\alpha_i - \alpha_i^*)k(\mathbf{x}_i, \mathbf{x}) + b. \tag{7}$$

As above, we use the RBF kernel. We denote the σ used in ε-SVR by σ_S.

ε-SVR is advantageous because it yields a sparse solution [21], but a good multiple output formulation is yet to be defined. Typically, the multiple output ε-SVR is defined by first uncorrelating the output values (shape parameters), then regressing each separately. We use this approach in our implementation along with the single output ε-SVR implementation LIBSVM [24].

3. Methodology

The method requires learning a function $f(\mathbf{x}) \rightarrow \mathbf{y}$ which relates an image feature vector $\mathbf{x} \in \mathbb{R}^p$ to a set of shape parameters $\mathbf{y} \in \mathbb{R}^d$ associated with shape coordinates $\mathbf{s} \in \mathbb{R}^k$. Once the shape parameters are regressed, we can obtain the associated shape coordinates by applying a mapping g from $\mathbf{y}$ to $\mathbf{s}, g(\mathbf{y}) = \mathbf{s}$. The shape parameterization and mapping functions are described in section 3.3. Shapes are defined by the two dimensional coordinates of 130 points (also known as *landmarks*) delineating the major facial fiducials (eyes, eyebrows, nose, mouth, and jaw) in a face image. This set of $\frac{k}{2}$ two-dimensional coordinates, where k is an even integer equal to twice the number of shape landmarks, defines the face shape $\mathbf{s}$ for the image feature vector $\mathbf{x}$. Throughout the discussion, when we refer to the *shape* we mean the shape landmark coordinate vector $\mathbf{s}$, while *shape parameters* correspond to the associated parameters $\mathbf{y}$. The following describes the image normalization, shape parameterization, image representation, and function learning methodology in detail.

3.1. General Algorithm

Our first step is to normalize all faces to facilitate subsequent modeling. This involves detecting and cropping the faces, then scaling them to a standard size. Then we detect the eye positions using an approach similar to [25], where regression is used to learn the function which maps the scaled face image to the eye positions. After detection, we rotate the images to an upright view and standard inter-eye distance. Normalizing the images restricts the range of face deformations, concentrating the samples in the image space and the associated shape space.

To model imprecise eye detections in *test* faces, we normalize the *training* faces according to their perturbed eye positions following a Normal distribution. More formally, let the true eye positions for all the training data be vectorized as $\textit{true position} = (L_x, L_y, R_x, R_y)^T$, where L and R correspond to the left and right eye, respectively, and x and y correspond to the horizontal and vertical coordinates, respectively. The perturbed positions are given by

$$\textit{perturbed position} = \textit{true position} + \varepsilon, \tag{8}$$

where $\varepsilon \sim N(\mu_e, \Sigma_e)$, and μ_e and Σ_e are the sample mean and covariance of the eye detection error. Note that we perturb the eye positions using the

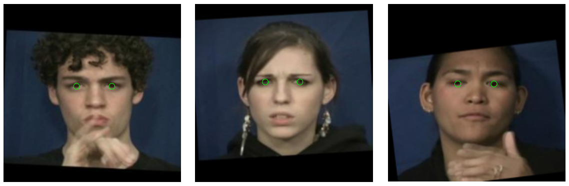


Figure 2: Normalized faces with automatic eye detection coordinates highlighted.

joint distribution of the error since we expect the errors to be correlated. Fig. 2 shows some example normalized faces.

Next, we use regression to learn the face shape manifold defined by $f(\cdot)$. The input to the regressor is a cropped image region $\mathbf{x}$ centered at the mean training face position while the outputs are the associated shape parameters $\mathbf{y}$. The features used for regression can either be the pixels themselves or some pre-processing of the image. In our experiments we evaluate the pixel intensities and the C1 features of Serre *et al.* [26].

3.2. Feature Spaces

We experiment with two image feature spaces. Our first image representation consists of the pixel intensities which are vectorized in a raster scan fashion and normalized to unit length. The unit length normalization is done to reduce the effect of intensity by the lighting source. The second image representation uses the C1 features of Serre *et al.* [26] which are reduced in dimensionality via PCA. The C1 features are the output of the 2^{nd} layer of a $4-$layer hierarchical filter architecture modeling the hierarchy of the visual cortex. The first layer, S1, comprises of Gabor filters at various positions, scales, and orientations. The second layer, C1, comprises of *maximum* pooling operations of filter responses for the S1 layer in the same image regions with the same orientation and within the same scale band. This pooling operation reduces sensitivity to small image perturbations. These image features are useful in our approach because these gradient based features reduce sensitivity to illumination changes and skin color.

3.3. Shape Modeling

Shape corresponds to the position of a discrete set of landmarks delineating the face features (eyes, eyebrows, nose, mouth, and jawline). The shape is modeled using a linear combination of a discrete set of d basis shapes usually referred to as shape modes [3]. The shape parameters are the coefficients of the shape modes. The modes correspond to directions in the shape space preserving most of the shape variance, where the shape space is defined as the span of all shape coordinate vectors. This model allows us to enforce reasonable limits on the possible shape deformation. For example, varying the parameter of the first shape mode may correspond to scaling the shape vertically. If we know the vertical range of the shape, then we could enforce a constraint on the contribution of the first mode in the final shape description. More modes can be included in the model to capture more of the possible shape variance. Following on our previous example, adding a second mode may allow us to represent vertical and horizontal shape scale changes. Representing shape using a number of modes equal to the dimensionality of the shape coordinate vector would correspond to a rotation of the shape coordinate vector in the shape space, or a change of basis.

Modes are derived using PCA. Specifically, the shape modes are defined by the primary eigenvectors of the shape coordinate covariance matrix $\mathbf{\Sigma}_{\mathbf{x}} \in \mathbb{R}^{k \times k}$ for a shape defined by $\frac{k}{2}$ landmarks in $\mathbb{R}^2$. These eigenvectors $\mathbf{p}_i, i = 1, 2, \ldots k$, are the ones associated with the largest eigenvalues, λ_i, with $\lambda_1 \geq \lambda_2 \geq \ldots \lambda_k \geq 0$.

The eigenvectors form an orthonormal basis for the shape space. This is important because this yields uncorrelated shape parameters, and recall that multiple KRR and our formulation of multiple ε-SVR assume the output variables are uncorrelated. Since most of the shape variance will be preserved by just a few of the shape modes, or eigenvectors, we can approximate shapes using a small number of parameters. The percentage of shape variance preserved by each shape mode $\mathbf{p}_i$ is given by the ratio $\frac{\lambda_i}{\sum_{j=1}^{k} \lambda_j}$. Thus, we can keep the amount of modes that preserves a desired amount of the total shape variance.

If we arrange the principal eigenvectors into the columns of a matrix $\mathbf{P} \in \mathbb{R}^{k \times d} = (\mathbf{p}_1, \mathbf{p}_2, \ldots \mathbf{p}_d)$ with $d \leq k$, and the associated eigenvalues into a diagonal matrix $\mathbf{\Lambda} \in \mathbb{R}^{d \times d}$, the shape parameter vector $\mathbf{y}$ associated with shape coordinate vector $\mathbf{s}$ is given by

$$\mathbf{y} = \mathbf{\Lambda}^{-\frac{1}{2}} \mathbf{P}^T (\mathbf{s} - \hat{\mu}_s), \tag{9}$$

where $\hat{\mu}_s$ is the sample mean shape coordinate vector. Shape parameters are converted back to the original space with

$$\mathbf{s} = \mathbf{P} \mathbf{\Lambda}^{\frac{1}{2}} \mathbf{y} + \hat{\mu}_s. \tag{10}$$

The term $\mathbf{\Lambda}$ serves as a variance normalization which ensures that the objective function favors each shape parameter appropriately.

3.4. Training the Regressor

The function $f(\cdot)$ mapping $\mathbf{x}$ to $\mathbf{y}$ for KRR is obtained using (2), and for ε-SVR using (6) and (7). Note that KRR requires tuning the parameters λ and σ_K while ε-SVR requires tuning the parameters C, ε, and σ_S. These parameters and the crop size for $\mathbf{x}$ are optimized using a grid search to minimize the 5-fold cross-validation error on the training set. Details are given in Section 4.3. For training, the face shape of the original face image is manually annotated in each of our databases. When a test image is presented to our algorithm, we detect the face position and scale using the Viola and Jones Detector [27], center and scale the image to a standard face size, then use our method to detect the face shape.

The manual shape annotations allow us to determine our final shape detection error. Error is defined by the average Euclidean distance from each landmark estimate to the corresponding manually annotated landmark over all test shapes. More formally, the error e for estimating N shapes $\mathbf{s}_j, j = 1 \ldots N$ by $\hat{\mathbf{s}}_j, j = 1 \ldots N$ is defined by

$$e = \frac{2}{Nk} \sum_{i=1}^{\frac{k}{2}} \sum_{j=1}^{N} \sqrt{(u_{ij} - \hat{u}_{ij})^2 + (v_{ij} - \hat{v}_{ij})^2}, \qquad (11)$$

where u_{ij} and v_{ij} are vertical and horizontal components of the i^{th} coordinate of the shape j, and $\hat{u}_{ij}$ and $\hat{v}_{ij}$ their estimates.

4. Experiments

We evaluated 6 shape detection algorithms on the task of face shape detection. The algorithms consisted of KRR or ε-SVR with pixel intensities or C1 features, the nonlinear AAM of [9], and Adaboost regression with Haar fetures as in SRM [13]. The goals were to evaluate how well each algorithm: generalizes across several identities, manages occlusions and extreme expression changes, handles extreme degradation in resolution, and performs in a real world setting. The specific databases used were geared toward evaluating generalization ability, robustness to occlusions, and performance in the real world. The training and testing partitions within a database were fixed across all algorithms to give a fair comparison.

4.1. Databases

The American Sign Language (ASL) database of [28] includes video sequences of 7 ASL signers. We manually annotated the face shape of $2,437$ images in the video sequence at the original resolution of 480×720 pixels. The face images in this data-set show large *variations in expression and self-occlusions* (hands can occlude facial regions when signing). Our goal with this database was to determine how well the algorithm handles these two problems.

The AR face (AR) database [29] contains frontal faces from over 100 subjects with largely varying facial expressions. We manually annotated 885 of the images at the original resolution of 576×768 pixels. Our goal with this database was to show that the learned manifold (Fig. 1) generalizes across several identities, i.e., it is *not subject-specific*, and can deal with extreme deformations such as a screaming face.

The Faces in the Wild (LFW) database [30] contains faces with varying facial expressions, pose changes, and occlusions at different resolutions, in different contexts, and in different photographic settings . Our goal with this database was to test the algorithm's performance in a real world setting. All faces in this database have already been detected, cropped, and scaled to a standard size of 250×250 pixels. The face coordinates of $2,610$ images were manually annotated at this scale of 250×250 pixels.

Different training percentages were used for each database according to the level of difficulty in estimating the manifold. In general, difficulty increases as the amount of subjects and variability in the database increases. Variability corresponds to differences in illumination, pose, occlusions, and other imaging artifacts. Therefore, we used a random partition of 60% of the annotated images for training and the remaining 40% for testing in the ASL database. We used 80% for training and the remaining 20% for testing in the AR databse. The LFW database is by far the most challenging, requiring many training samples for reasonable manifold estimation. We used a random partition of 90% of the annotated images for training and the remaining 10% for testing. Since the training set was so large in the LFW experiments, we found it necessary to reduce the dimensionality of the pixel features for computational tractability in the $\varepsilon-$SVR experiments. PCA was used, where 99% of the variance was kept to preserve as much information as possible.

4.2. Different Resolution Analysis

To simulate the effects of low resolution we detected shape in images of different sizes. We localized all faces except those from the LFW database (already localized) in their original image using an off the shelf face detector, then cropped them in a square region much larger than the detected face following the approach of Huang *et al.* [30]. The cropped region was then scaled from 250 × 250 pixels to 50 × 50 pixels in decrements of 50 pixels. Details about the face detector and crop size are given in Section 4.3. All of these images were normalized to a 42 pixels inter-eye distance. 42 pixels was the mean eye distance of a random subset of images from the LFW database which were originally 250 × 250 pixels. To further challenge the algorithms and emphasize the performance in extremely degraded image conditions, we used images with a starting size of 250×250 pixels which were normalized to a 42, 20, 10, and 5 pixel inter-eye distance. The original annotated coordinates were scaled as necessary to serve as the face shape coordinates for each new image size. In each experiment and for each scale, we estimated the face shape over 10 trials using a different training and testing partition in each case. Example detections are shown in Figs. 3, 4, and 5 for the ASL, AR, and LFW databases, respectively.

Quantitative results of detection error are graphed in Fig. 6 as standard error of equation (11) in the first row, and the normalized detection error defined by,

$$e_{\text{normalized}} = \frac{e}{\text{Inter-Eye Distance}}, \tag{12}$$

in the second row. The normalization standardizes the error rates to account for the range of inter-eye distances. All other figures and tables report the standard error of equation (11). Additional results are tabulated in Table 1. A more detailed view of the trends can be seen in the cumulative error histogram plots of Figs. 7, 8, and 9.

Some noticeable trends are evident from the results in Table 1. First, the ASL database containing 7 subjects shows that KRR with pixel features performed best over a variety of resolutions when detecting shape over a limited range of subjects. Although the C1 features did not perform as well, it was important to evaluate their performance within the manifold learning framework because it has been argued that these features facilitate recognition of different classes of objects including shape and texture based objects, and provide invariance to illumination [26]. While pixels perform favorably for faces, other classes of objects may prefer the C1 representation.

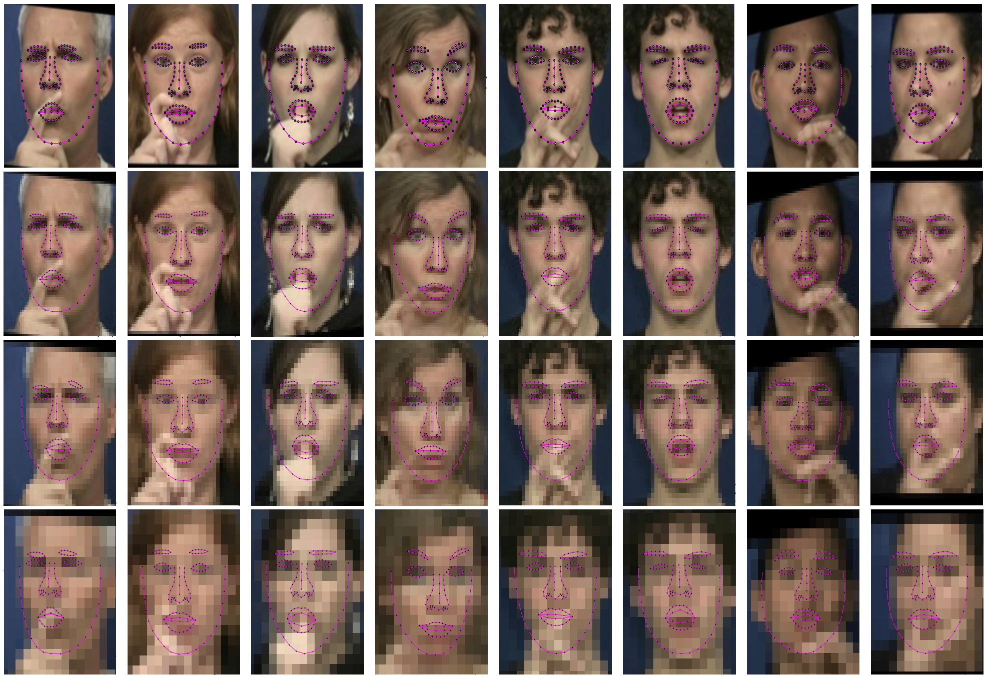


Figure 3: Example face shape detections for the ASL database using KRR with pixel features. Starting from the top row we display results for 250×250 pixel face images which have been normalized to a $42,\ 20,\ 10$, and 5 pixel inter-eye distance.

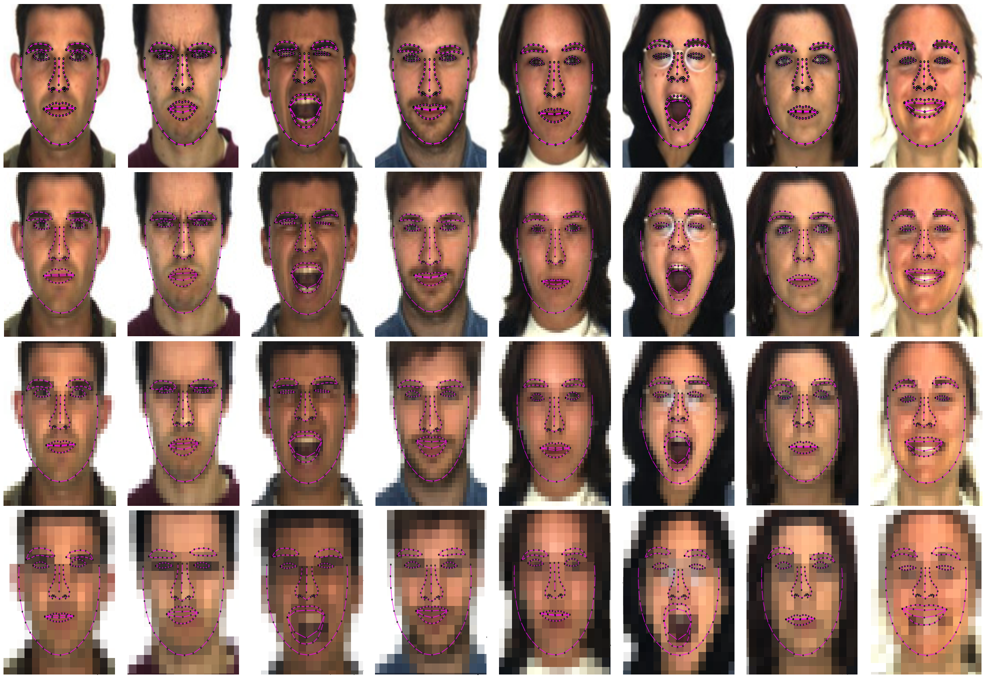


Figure 4: Example face shape detections for the AR face database using KRR with pixel features. Starting from the top row we display results for 250×250 pixel face images which have been normalized to a 42, 20, 10, and 5 pixel inter-eye distance.

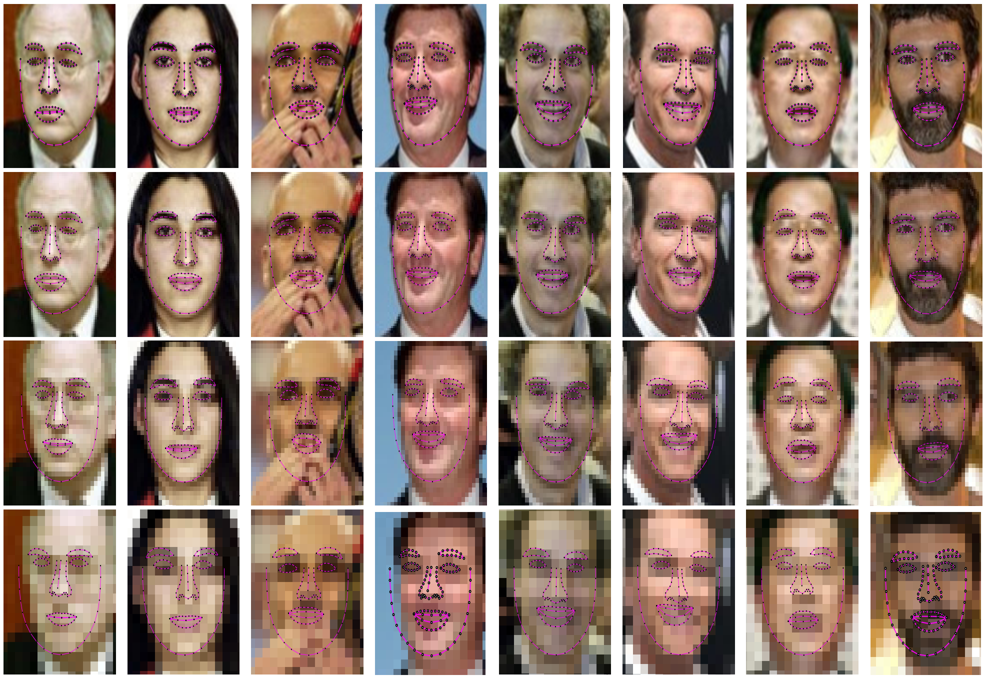
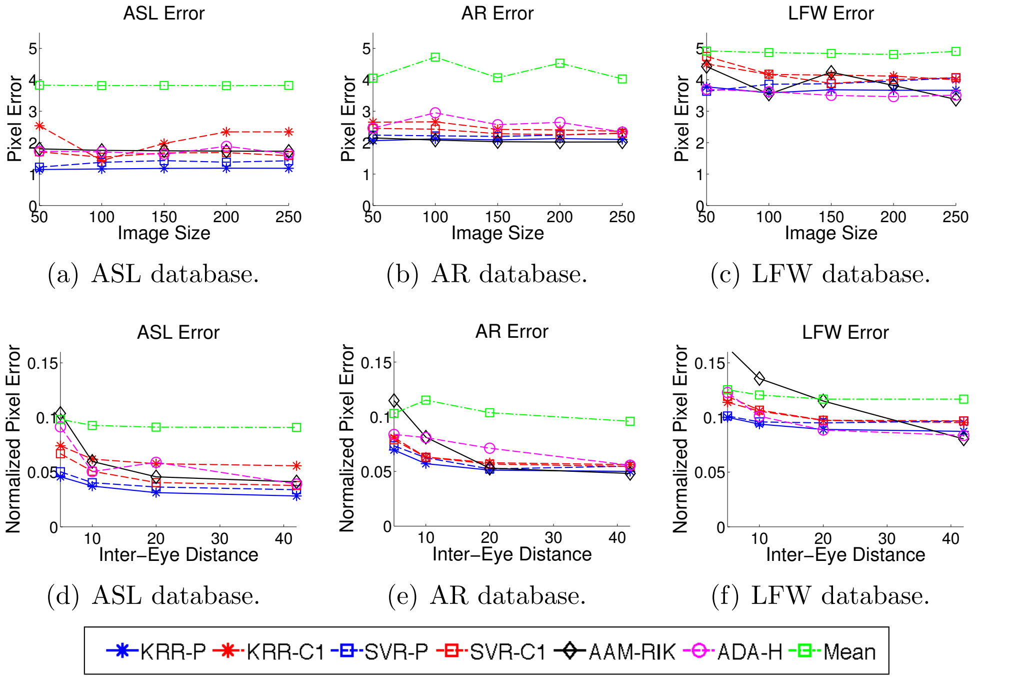


Figure 5: Example face shape detections for the LFW database using KRR with pixel features. Starting from the top row we display results for 250×250 pixel face images which have been normalized to a 42, 20, 10, and 5 pixel inter-eye distance.



(a) ASL database.

(b) AR database.

(c) LFW database.

(d) ASL database.

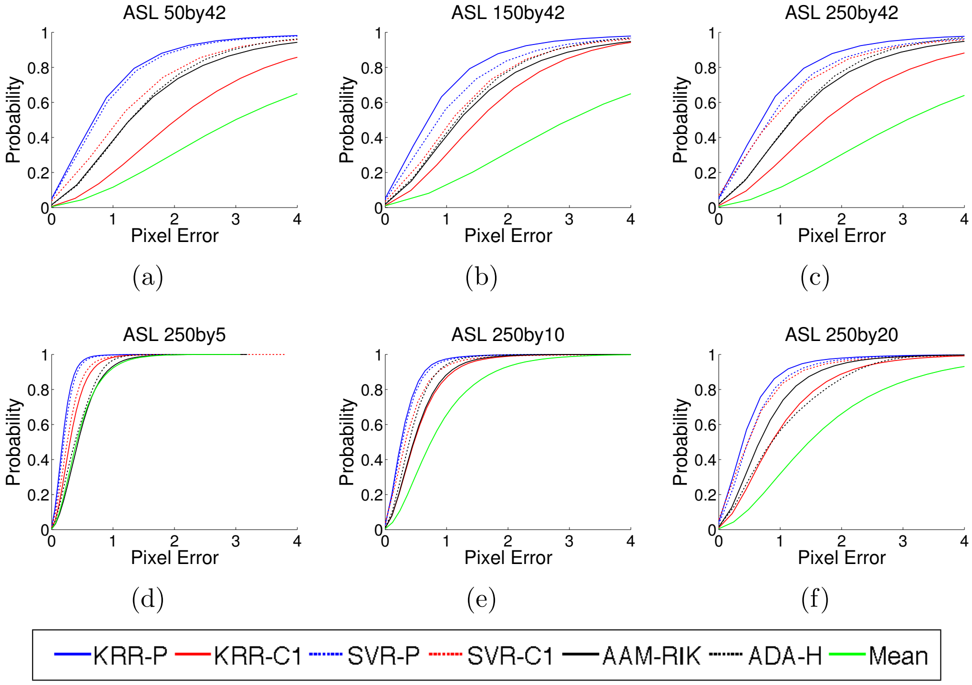
(e) AR database.

(f) LFW database.

Figure 6: (a) - (c) show the mean Euclidean pixel error for each database as a function of image size in pixels. Image size corresponds to the height and width of the face image before rotating and scaling to a standard inter-eye distance of 42 pixels. (d) - (f) show the normalized mean Euclidean pixel error for each database as a function of the inter-eye distance. P and C1 denote pixel and C1 features of [26], respectively. AAM-RIK denotes the AAM with Rotation Invariant Kernels of [9], ADA-H denotes the Adaboost regression [13] with Haar-like features of [27], and Mean denotes taking the mean of the training shapes as the estimate in every case.

Table 1: Mean Euclidean landmark error of equation (11) and standard deviation over all test images in pixels for images at 250 × 250 pixels which are normalized to 42, 20, 10, and 5 pixel inter-eye distances. (a) - (c) correspond to the ASL, AR, and LFW databases, respectively. The left column entries are defined in Fig. 6.

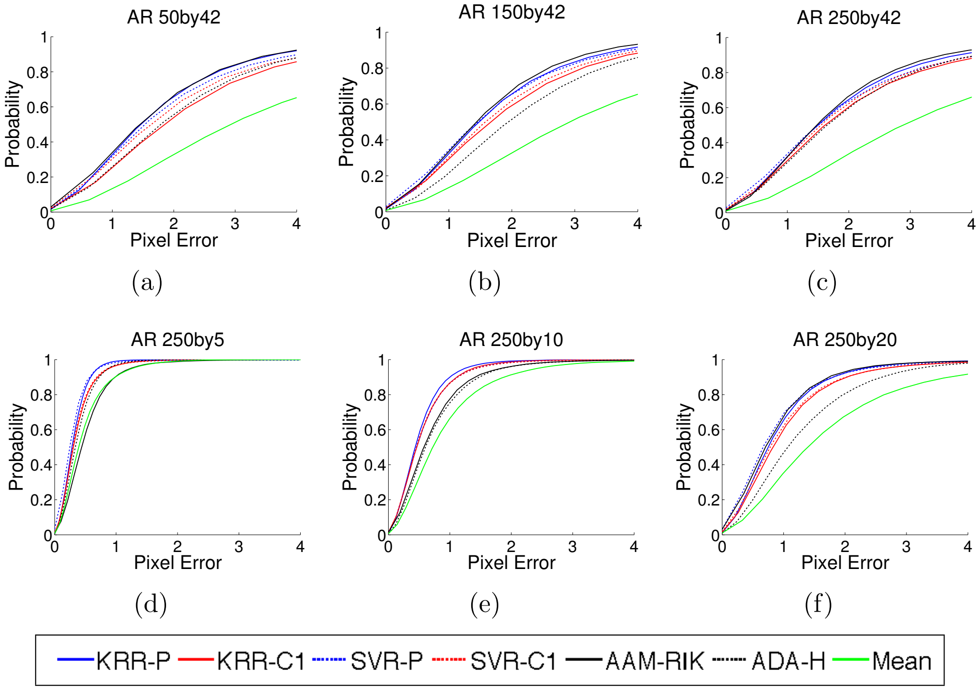
		Inter-eye Distance			
		5 pixels	10 pixels	20 pixels	42 pixels
(a) ASL	KRR-P	**.23 ± .15**	**.37 ± .28**	**.62 ± .53**	**1.18 ± 1.10**
	KRR-C1	.37 ± .23	.62 ± .42	1.15 ± .81	2.34 ± 1.68
	SVR-P	.25 ± .16	.40 ± .31	.73 ± .61	1.42 ± 1.40
	SVR-C1	.33 ± .23	.51 ± .42	.81 ± .76	1.59 ± 1.61
	AAM-RIK	.52 ± .32	.60 ± .40	.91 ± .66	1.73 ± 1.30
	ADA-H	.46 ± .28	.50 ± .32	1.18 ± .80	1.64 ± 1.12
(b) AR	KRR-P	**.35 ± .21**	**.57 ± .40**	**1.02 ± .77**	2.10 ± 1.61
	KRR-C1	.41 ± .28	.63 ± .45	1.16 ± .89	2.36 ± 1.92
	SVR-P	.37 ± .36	.63 ± .46	1.05 ± .88	2.30 ± 1.96
	SVR-C1	.39 ± .29	.63 ± .48	1.14 ± .92	2.29 ± 1.89
	AAM-RIK	.57 ± .41	.81 ± .63	1.06 ± .86	**2.02 ± 1.54**
	ADA-H	.42 ± .27	.81 ± .59	1.43 ± 1.02	2.34 ± 1.83
(c) LFW	KRR-P	**.50 ± .34**	**.94 ± .68**	1.78 ± 1.33	3.67 ± 2.80
	KRR-C1	.57 ± .37	1.06 ± .74	1.95 ± 1.37	4.00 ± 2.88
	SVR-P	.51 ± .35	.96 ± .72	1.90 ± 1.44	4.06 ± 3.09
	SVR-C1	.60 ± .40	1.07 ± .75	1.95 ± 1.41	4.07 ± 3.01
	AAM-RIK	.82 ± .60	1.36 ± .92	2.30 ± 2.01	**3.38 ± 3.04**
	ADA-H	.61 ± .39	1.01 ± .67	**1.77 ± 1.29**	3.51 ± 2.58



(a) (b) (c)

(d) (e) (f)

Figure 7: Cumulative pixel error histograms are plotted for the ASL database. The plots in (a) - (c) show error rates for images which are first scaled to 50 × 50, 150 × 150, and 250 × 250 pixels, then normalized to a 42 pixel inter-eye distance. (d) - (f) show error rates for images which are first scaled to 250 × 250 pixels, then normalized to a 5, 10, and 20 pixel inter-eye distance. Legend entries are defined in Fig. 6.



(a) (b) (c)

(d) (e) (f)

Figure 8: Cumulative pixel error histograms are plotted for the AR database. As above, the errors in (a) - (c) are for images scaled to 50 × 50, 150 × 150, and 250 × 250 pixels, then normalized to a 42 pixel inter-eye distance. (d) - (f) show error rates for images which are first scaled to 250 × 250 pixels, then normalized to a 5, 10, and 20 pixel inter-eye distance. Legend entries are defined in Fig. 6.

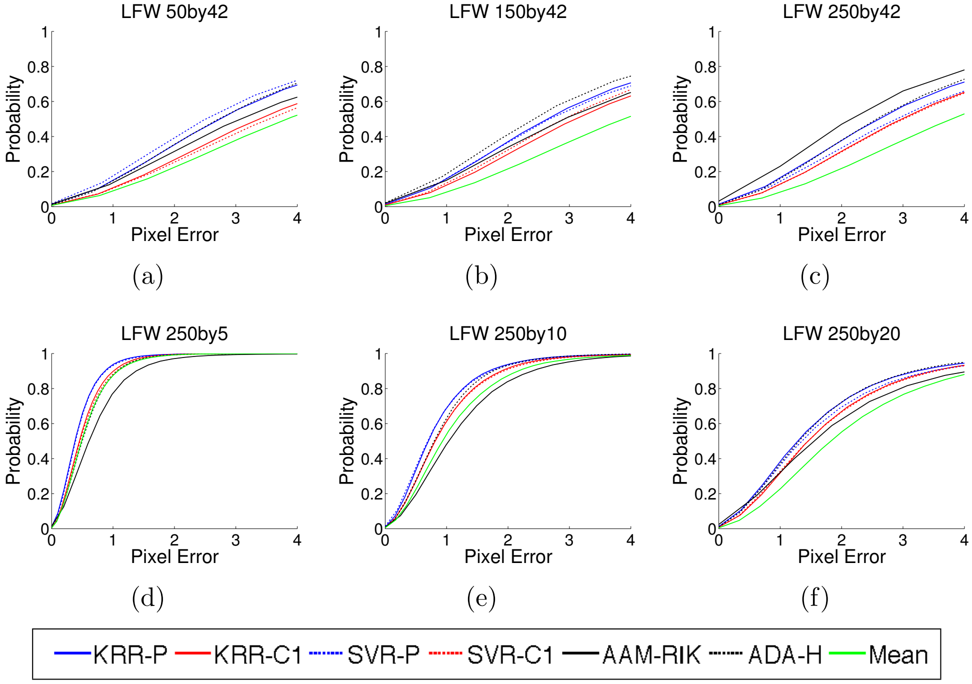


Figure 9: Cumulative pixel error histograms are plotted for the LFW database. The plots in (a) - (c) show error rates for images of 50 × 50, 150 × 150, and 250 × 250 pixels, then normalized to a 42 pixel inter-eye distance. (d) - (f) show error rates for images which are first scaled to 250 × 250 pixels, then normalized to a 5, 10, and 20 pixel inter-eye distance. Legend entries are defined in Fig. 6.

The 5 and 10 pixel inter-eye distance results over all databases show that AAM-RIK and adaboost with Haar features suffer when the resolution is degraded significantly. This can be explained for the AAM-RIK by the lack of precision in synthesizing a face that is degraded to that magnitude. Similarly, the Haar features are unable to precisely describe local image cues at very low resolutions. The kernel regression based algorithms did not suffer from the degradation in resolution because the test images are not synthesized as in the AAM-RIK, and a holistic (not local) image representation is used. However, the AAM-RIK and adaboost methods yielded very accurate results at higher resolutions, as supported by the AR and LFW results in Table 1.

To provide a comparison to the state of the art in detailed face shape detection, the experiments were repeated for the ASL and AR database as in the work of Ding and Martinez [12]. Their algorithm relies heavily on color and edge cues, so higher resolution images are required. Therefore, we performed our eye detection in the images which were kept at the original resolution. After eye detection, the faces were scaled to an inter-eye distance of 120 pixels for the ASL database, and 111 pixels for the AR Face database, corresponding to the average inter-eye distances within the databases at the original resolution. In [12], the authors report an average pixel error of 6.9 pixels with a standard deviation of 1.5 for the ASL database, and an average pixel error of 8.4 pixels with a standard deviation of 1.2 for a combination of 400 images from the AR database and 800 images from the XM2VTS database [31]. We achieved an average error rate of 4.30 pixels with a standard deviation of 3.80 for the ASL database, and an average error rate of 6.97 pixels with a standard deviation of 5.61 for the AR database.

The error rates favor the algorithm presented, but the comparison is not easy to make for a few reasons. Namely, the algorithm in [12] is not based on regression, so subsets of the database were not sequestered for testing. Therefore, evaluation was not performed using the same images. This is especially the case for the AR results which Ding and Martinez pool with a database not used here. However, the comparison along with the others presented demonstrates that we have achieved performances above the state of the art.

To illustrate the performance with occlusions, we refer the reader to Fig. 3 which contains sample results from the ASL database. These are particular examples where occlusions are prominent in some of the fiducials. The unoccluded fiducials are mostly unaffected, which is expected since these fiducial shapes are regressed with the local texture information. In largely occluded regions such as the mouth in the 5^{th} column, there is a good shape guess. We can achieve this because of the shape model learned from the training samples.

To emphasize the effect of degrading the image resolution, results for each database are shown for images which are normalized to an inter-eye distance of 42, 20, 10, and 5 pixels before doing the shape detection. By comparing the results for the degraded resolutions to the original resolution, we can see that the accuracy degrades, but marginally compared to the reduction in resolution. This is not the case for the AAM-RIK and ADA-H, which break down as the inter-eye distance gets too small.

In some cases, the images with a 5 pixel inter-eye distance appear to have a large error (such as for the eye shape). However, this results from the image interpolation which produce the appearance of eyes which sag under the true eye positions. You can see this effect if you compare the 5 pixel inter-eye distance images to the 42 pixel inter-eye distance images.

To illustrate the performance in a more challenging setting, where the manifold must generalize across subjects with more extreme facial expressions, we show sample results from the AR face database in Fig. 4. We can see from some examples that the results are not as accurate as the previous database. This is also clear from the results in Fig. 6 and Table 1. The mouth and jawline are not aligned perfectly, but the estimate is very close.

For the realistic setting where there is little control over the pose, illumination, photographic setting, or even identity of the subject, we show results from the LFW database in Fig. 5. Much like the results in Fig. 3, the detection is not as precise as in the ASL database (zooming into the image may illustrate this better). However, the estimate is still very accurate, as seen in Fig. 6 and Table 1. Of particular interest is the 3^{rd} column, which has very large occlusions. Much like in the ASL database, a good estimate is given.

4.3. Implementation Details

Faces are first detected using the openCV implementation [32] of the Viola and Jones face detector [27]. The faces are scaled to a standard size following the approach of Huang *et al.* [30]. A region 2.2 times the detected face size is cropped around the detected face position and scaled to a standard square size, then normalized as described in Section 3.1. We define the face shape by a set of 130 landmarks delineating the eyes, eyebrows, nose, mouth, and jaw. We use 5-fold cross-validation with a grid search on the training data to tune the crop size for $\mathbf{x}$, and the regression parameters. We choose the parameters which minimize the mean squared error of estimating the shape parameters y over the 5 validation sets. The parameters are obtained in the first experimental trial, and used for the remaining trials. The joint distribution of the eye position detection error is then estimated using the estimation errors from the cross-validation trial corresponding to the optimal parameters.

To simplify the computation of the proposed approach we proceed as follows. Shape modes are stored for faster function evaluation since they only depend on the training data. In addition, the $\kappa(\mathbf{x})$ in the KRR solution given in (2) is arranged into a matrix for all training samples, allowing batch shape detection in all images. Our MATLAB implementation which utilizes this batch procedure and stores the learned shape modes can detect shapes at .084 seconds per image, or 11.9 frames per second on a 2.4 GHz Intel Core 2 Duo with 4 GB RAM. This timing includes the eye detection and image rotation, and was conducted on the ASL dataset with 150×150 pixel images which are normalized to a 20 pixel inter-eye distance. 60% of the images were used for training, with the detection being performed on the remaining 40%. Faster running times can be achieved by storing the inverted matrix and Gram matrix of the KRR solution since they only depend on the training data.

$\varepsilon-$SVR yields a sparser model than KRR which can also be stored for fast function evaluation. Unfortunately, it requires we train a separate model for each shape mode.

The nonlinear AAM implementations is based on [9] which employs nonlinear shape and texture models through the use of Rotation Invariant Kernels (RIK) [33]. The model is trained separately for the different scales and resolution, using the training image subsets which are normalized to the size and resolution of the testing set. The kernel parameters are selected as suggested by the authors. In testing, we first detect and normalize the images using the approach described in Section 3.1, where KRR with pixel features is employed since it consistently yields very accurate results. Then, we initialize the AAM with 10 random training face shapes which are aligned in position and scale with the test image, and update the shape estimate until convergence or a maximum of 30 iterations. Careful checks were made to prevent divergence. The final estimate is given by the shape estimate which gives the lowest texture estimation error over the different initializations.

The Adaboost implementation is based on the work of [13], where piecewise functions of single Haar feature outputs are selected in a greedy manner to build a strong learner for an image based regression task. At each round of boosting, the weak learner which most reduces the residual error is added to the strong learner. Shrinkage is used to prevent over-fitting. Specifically, each weak learner is scaled by a real positive value less than 1 before being added to the strong learner. The strong learner has several parameters: the type and number of weak learners to use, and the shrinkage value. We choose a strong learner comprising of 400 weak learners with a shrinkage factor of .1. Piecewise linear functions are employed as weak learners. For a given Haar feature, the piecewise functions are split into evenly spaced bins by 40 knots according to the maximum range of that Haar feature output from the training data. Each piecewise linear function for each bin is obtained by fitting the output from a single Haar feature to the residual error using (1) with regularization parameter $\lambda = .1/n$.

The training and testing images are first normalized to un upright view and standard scale using the approach described in Section 3.1, where KRR with pixel features is employed since it consistently yields very accurate results. Then, a strong learner is learned which maps these normalized face images to their associated shape modes. Similarly to the $\varepsilon-$SVR implementation, the strong learner is an ensemble of single output strong learners for each shape mode. Shape modes are selected to preserve 90% of the shape variance, as described in Section 3.3.

5. Conclusion

We have presented a new algorithm for deformable shape detection that is based on manifold learning through nonlinear regression. By taking this approach, the algorithm has the benefits of generative and discriminative methods while avoiding the major drawbacks associated with generative, energy minimization based, and sliding window methods. Our algorithm is non-iterative and does not require shapes be defined by salient edges. Therefore, as we have demonstrated experimentally, the algorithm can be made to work at extremely coarse resolutions. Additionally, we learn a shape model based on training data which ensures a reasonable shape estimate even in the case of large occlusions.

Although we presented certain feature spaces and regression techniques, these are only particular choices for a general shape detection framework based on manifold learning with supervised training data. The learned manifold relates the image feature space to the corresponding shape space. What is important about the feature space used is that it preserves the local image information. As we have demonstrated, the framework presented is flexible enough to model deformable faces, and can be trained reliably given the constraints on the training data.

Acknowledgments
This research was partially supported by NIH grants R01-EY-020834 and R21-DC-011081.

References

[1] M. Kass, A. Witkins, D. Terzopoulos, Snakes: Active contour models, International Journal of Computer Vision 1 (1988) 321–331.

[2] R. Malladi, J. Sethian, B. Vemuri, Shape modeling with front propagation: A level set approach, IEEE Trans. Pattern Analysis and Machine Intelligence 17 (1995) 158–175.

[3] T. F. Cootes, C. J. Taylor, D. H. Cooper, J. Graham, Active shape models - their training and application, Computer Vision and Image Understanding 61 (1995) 38–59.

[4] G. Mori, S. Belongie, J. Malik, Efficient shape matching using shape contexts, IEEE Trans. Pattern Analysis and Machine Intelligence 27 (2005) 517–530.

[5] T. F. Cootes, G. J. Edwards, C. J. Taylor, Active appearance models, In Proc. European Conference on Computer Vision (1998) 484–498.

[6] X. Liu, Discriminative face alignment, IEEE Trans. Pattern Analysis and Machine Intelligence 99 (2009) 1941–1954.

[7] L. Gu, T. Kanade, A generative shape regularization model for robust face alignment, In Proc. European Conference on Computer Vision (2008) 413–426.

[8] Y. Zhou, L. Gu, H.-J. Zhang, Bayesian tangent shape model: estimating shape and pose parameters via bayesian inference, In Proc. IEEE Computer Vision and Pattern Recognition (2003) I–109–I–116.

[9] O. C. Hamsici, A. M. Martinez, Active appearance models with rotation invariant kernels, In Proc. IEEE International Conference on Computer Vision (2009) 1003–1009.

[10] L. Liang, F. Wen, X. Tang, Y. Xu, An integrated model for accurate shape alignment, In Proc. European Conference on Computer Vision (2006) 333–346.

[11] J. Zhang, S. K. Zhou, D. Comaniciu, L. McMillan, Conditional density learning via regression with application to deformable shape segmentation, In Proc. IEEE Computer Vision and Pattern Recognition (2008) 1–8.

[12] L. Ding, A. Martinez, Features versus context: An approach for precise and detailed detection and delineation of faces and facial features, IEEE Trans. Pattern Analysis and Machine Intelligence (2010) 2022–2038.

[13] S. Zhou, D. Comaniciu, Shape regression machine, Information Processing in Medical Imaging (2007) 13–25.

[14] S. K. Zhou, Shape regression machine and efficient segmentation of left ventricle endocardium from 2D b-mode echocardiogram, Medical Image Analysis (2010).

[15] D. Cristinacce, T. F. Cootes, Boosted regression active shape models, In Proc. British Machine Vision Conference (2007) 880–889.

[16] A. M. Martinez, Recognizing imprecisely localized, partially occluded, and expression variant faces from a single sample per class, IEEE Trans. Pattern Analysis and Machine Intelligence 24 (2002) 748–763.

[17] G. Seber, C. Wild, Nonlinear Regression, Wiley-Interscience, 2003.

[18] A. E. Hoerl, Application of ridge analysis to regression problems, Chemical Engineering Progress 58 (1962) 54–59.

[19] T. Hastie, R. Tibshirani, J. Friedman, The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Springer-Verlag, New York, 2001.

[20] G. Wahba, Spline Models for Observational Data, Society for Industrial and Applied Mathematics, 1990.

[21] A. J. Smola, B. Schlkopf, A tutorial on support vector regression, Statistics and Computing 14 (2004) 199–222.

[22] V. N. Vapnik, The Nature of Statistical Learning Theory, Springer, 1999.

[23] D. P. Bertsekas, Nonlinear Programming, Athena Scientific, 1999.

[24] C.-C. Chang, C.-J. Lin, LIBSVM: a library for support vector machines, 2001. Software available at http://www.csie.ntu.edu.tw/ cjlin/libsvm.

[25] M. Everingham, A. Zisserman, Regression and classification approaches to eye localization in face images, In Proc. IEEE Automatic Face and Gesture Recognition (2006) 441–446.

[26] T. Serre, L. Wolf, S. Bileschi, M. Riesenhuber, T. Poggio, Robust object recognition with cortex-like mechanisms, IEEE Trans. Pattern Analysis and Machine Intelligence 29 (2007) 411–426.

[27] P. Viola, M. Jones, Rapid object detection using a boosted cascade of simple features, In Proc. IEEE Computer Vision and Pattern Recognition (2001) I–511–I–518.

[28] L. Ding, A. Martinez, Precise detailed detection of faces and facial features, In Proc. IEEE Computer Vision and Pattern Recognition (2008) 1–7.

[29] A. M. Martinez, R. Benavente, The AR face database, CVC Technical Report No. 24 (1998).

[30] G. B. Huang, M. Ramesh, T. Berg, E. Learned-Miller, Labeled faces in the wild: A database for studying face recognition in unconstrained environments, University of Massachusetts, Amherst, Technical Report 07-49, October (2007).

[31] K. Messer, J. Matas, J. Kittler, J. Lüttin, G. Maitre, Xm2vtsdb: The extended m2vts database, In Proc. International Conference on Audio and Video-based Biometric Person Authentication (1999) 72–77.

[32] G. Bradski, The OpenCV Library, Dr. Dobb's Journal of Software Tools (2000).

[33] O. C. Hamsici, A. M. Martinez, Rotation invariant kernels and their application to shape analysis, IEEE Trans. Pattern Analysis and Machine Intelligence 31 (2009) 1985–1999.