Local Minima Free Parameterized Appearance Models

Minh Hoai Nguyen Fernando De la Torre
Robotics Institute, Carnegie Mellon University
Pittsburgh, PA 15213, USA.
minhhoai@cmu.edu ftorre@cs.cmu.edu

Abstract

Parameterized Appearance Models (PAMs) (e.g. Eigentracking, Active Appearance Models, Morphable Models) are commonly used to model the appearance and shape variation of objects in images. While PAMs have numerous advantages relative to alternate approaches, they have at least two drawbacks. First, they are especially prone to local minima in the fitting process. Second, often few if any of the local minima of the cost function correspond to acceptable solutions. To solve these problems, this paper proposes a method to learn a cost function by explicitly optimizing that the local minima occur at and only at the places corresponding to the correct fitting parameters. To the best of our knowledge, this is the first paper to address the problem of learning a cost function to explicitly model local properties of the error surface to fit PAMs. Synthetic and real examples show improvement in alignment performance in comparison with traditional approaches.

1. Introduction

Since the early work of Sirovich and Kirby [21] parameterizing the human face using Principal Component Analysis (PCA) and the successful eigenfaces of Turk and Pentland [23], many computer vision researchers have used PCA techniques to construct linear models of optical flow, shape or graylevel [3, 10, 4, 6, 19, 14, 5, 11]. In particular, Parameterized Appearance Models (PAMs) (e.g. eigentracking [4], active appearance models [6, 10, 17, 9, 11], morphable models [5, 14]) have proven to be an appropriate statistical tool for modeling shape and appearance variation of objects in images. In PAMs, the appearance/shape models of objects are built by performing PCA on training data. Once the models have been constructed, finding the location/configuration of an object of interest in a testing image is achieved by minimizing a cost function w.r.t. some transformation (motion) parameters; this is referred to as the fitting process.

Although widely used, PAMs suffer from two problems in the fitting process. First, they are especially prone to local minima. Second, often few, if any, of the local minima of the cost function correspond to acceptable solutions. Figures 1a,d,f illustrate these problems. Fig. 1d plots the error surface constructed by translating the testing image (Fig. 1c) around the ground truth landmarks (Fig. 1c) and computing the values of the cost function. The cost function is based on a PCA model constructed from labeled training data (Fig. 1a). Fig. 1f shows the contour plot of this error surface. As can be observed, any gradient-based optimization method is likely to get stuck at local minima, and will

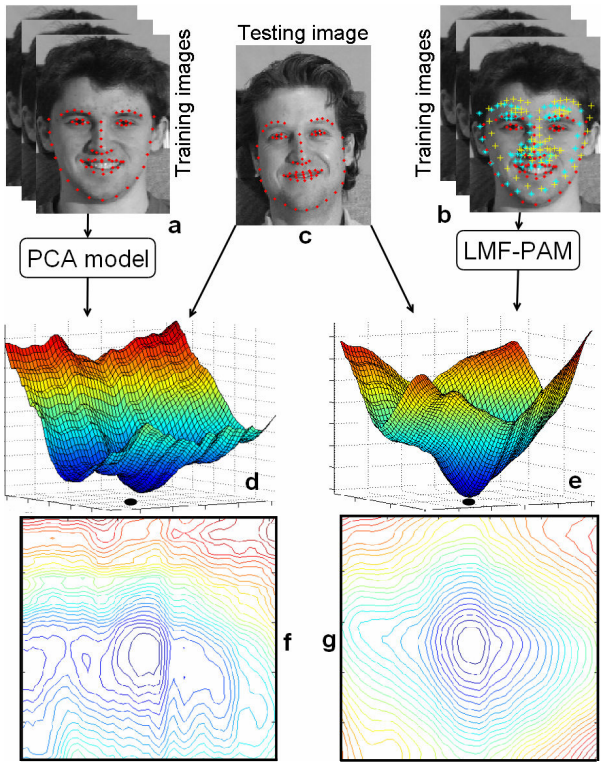

Figure 1. Learning a better model for image alignment. (d,f): surface and contour plot of the PCA model. It has many local minima; (e, g): Local Minima Free PAM (LMF-PAM) method learns a better error surface to fit PAMs. This figure is best seen in color.

not converge to the global minimum. Moreover, the global minimum of this cost function is not at the desired position, the black dot of Fig. 1d, which corresponds to the correct landmarks' locations. These problems occur because the PCA model is constructed without considering the neighborhoods of the correct motion parameters (parameters that correspond to ground truth landmarks of training data). The neighborhoods determine the local minima properties of the error surface, and should be taken into account while constructing the models.

In this paper, we propose to learn the cost function (i.e. appearance model) that has a local minimum at the "expected" location and no other local minima in its neighborhood. This is done by enforcing constraints on the gradients of the cost function at the desired location and its neighborhood. Fig. 1e,g plot the error surface and contours of the learned cost function. This cost function has a local minimum in the expected place (black dot of Fig. 1e), and no other local minima near by.

2. Previous work

Over the last decade, appearance models have become increasingly important in computer vision and graphics. In particular, PAMs have been proven useful for alignment, detection, tracking, and face synthesis [5, 4, 10, 6, 17, 19, 14, 11, 24]. This section reviews PAMs and gradient-based methods for the efficient alignment of high dimensional deformation models.

2.1. PAMs

PAMs [4, 10, 6, 19, 14, 5, 24] build the objects' appearance/shape representation from the principal components of training data. Let $\mathbf{d}_i \in \Re^{m\times 1}$ (see notation [1]) be the i^{th} sample of a training set $\mathbf{D} \in \Re^{m\times n}$ and $\mathbf{U} \in \Re^{m\times k}$ the first k principal components [13]. Once the model has been constructed (i.e. $\mathbf{U}$ is known), tracking/alignment is achieved by finding the motion parameter $\mathbf{p}$ that best aligns the data w.r.t. the subspace $\mathbf{U}$, i.e.

$$\min_{\mathbf{c},\mathbf{p}} ||\mathbf{d}(\mathbf{f}(\mathbf{x},\mathbf{p})) - \mathbf{U}\mathbf{c}||_2^2 \qquad (1)$$

Here $\mathbf{x} = [x_1, y_1, ...x_l, y_l]^T$ is the vector containing the coordinates of the pixels to track. $\mathbf{f}(\mathbf{x},\mathbf{p})$ is the function for geometric transformation; denote $\mathbf{f}(\mathbf{x},\mathbf{p})$ by $[u_1, v_1, ..., u_l, v_l]^T$. $\mathbf{d}$ is the image frame in consideration, and $\mathbf{d}(\mathbf{f}(\mathbf{x},\mathbf{p}))$ is the appearance vector of which the i^{th} entry is the intensity of image $\mathbf{d}$ at pixel (u_i, v_i). For affine and non-rigid transformations, (u_i, v_i) relates to (x_i, y_i) by:

$$\begin{bmatrix} u_i \\ v_i \end{bmatrix} = \begin{bmatrix} a_1 & a_2 \\ a_4 & a_5 \end{bmatrix} \begin{bmatrix} x_i^s \\ y_i^s \end{bmatrix} + \begin{bmatrix} a_3 \\ a_6 \end{bmatrix} \qquad (2)$$

with $[x_1^s, y_1^s, ...x_l^s, y_l^s]^T = \mathbf{x} + \mathbf{U}^s\mathbf{c}^s$, where $\mathbf{U}^s$ is the non-rigid shape model learned by performing PCA on a set of registered shapes [7]. $\mathbf{a}, \mathbf{c}^s$ are affine and non-rigid motion parameters respectively, and $\mathbf{p} = [\mathbf{a}; \mathbf{c}^s]$.

2.2. Optimization for PAMs

Given an image $\mathbf{d}$, PAM tracking/alignment algorithms optimize (1). Due to the high dimensionality of the motion space, a standard approach to efficiently search over the parameter space is to use gradient-based methods [1, 7, 17, 4, 8, 25]. To compute the gradient of the cost function given in (1), it is common to use Taylor series expansion to approximate $\mathbf{d}(\mathbf{f}(\mathbf{x},\mathbf{p}+\delta\mathbf{p}))$ by $\mathbf{d}(\mathbf{f}(\mathbf{x},\mathbf{p})) + \mathbf{J}^{\mathbf{d}}(\mathbf{p})\delta\mathbf{p}$, where $\mathbf{J}^{\mathbf{d}}(\mathbf{p}) = \frac{\partial \mathbf{d}(\mathbf{f}(\mathbf{x},\mathbf{p}))}{\partial \mathbf{p}}$ is the Jacobian of the image $\mathbf{d}$ w.r.t. to the motion parameter $\mathbf{p}$ [16]. Once linearized, a standard approach is to use the Gauss-Newton method for optimization [2, 4]. Other approaches learn an approximation of the Jacobian matrix with linear [7] or non-linear [20, 15] regression.

Over the last few years, several strategies for improving the fitting performance have been proposed. For examples, Black & Anandan [4] and Cootes & Taylor [7] proposed using multi-resolution schemes, Xiao *et al* [26] proposed using 3D models to constrain 2D solutions, de la Torre *et al* proposed learning filters to achieve robustness to local minima, de la Torre & Black [8], and Baker & Matthews [1] learned a PCA model invariant to rigid and non-rigid transformations. Although these methods show significant performance improvement, they do not directly address the problem of learning a cost function with no local minima. In this paper, we deliberately learn a cost function which has local minima at and only at the desired places.

3. Learning parameters of the cost functions

Gradient-based algorithms, such as the ones discussed in the previous section, might not converge to the correct location (i.e. correct motion parameters) for several reasons. First, gradient-based methods are susceptible to being stuck at local minima. Second, even when the optimizer converges to a global minimum, the global minimum might not correspond to the correct motion parameters. These two problems occur primarily because PCA has limited generalization capabilities to model appearance variation. This section proposes a method to learn cost functions that do not exhibit these two problems in training data.

[1] Bold uppercase letters denote matrices (e.g. $\mathbf{D}$), bold lowercase letters denote column vectors (e.g. $\mathbf{d}$). $\mathbf{d}_j$ represents the j^{th} column of the matrix $\mathbf{D}$. d_{ij} denotes the scalar in the row i^{th} and column j^{th} of the matrix $\mathbf{D}$. Non-bold letters represent scalar variables. $\mathbf{1}_k \in \Re^{k\times 1}$ is a column vector of ones. $\mathbf{0}_k \in \Re^{k\times 1}$ is a column vector of zeros. $\mathbf{I}_k \in \Re^{k\times k}$ is the identity matrix. $tr(\mathbf{D}) = \sum_i d_{ii}$ is the trace of square matrix $\mathbf{D}$. $||\mathbf{d}||_2 = \sqrt{\mathbf{d}^T\mathbf{d}}$ designates Euclidean norm of $\mathbf{d}$. $||\mathbf{D}||_F = tr(\mathbf{D}^T\mathbf{D})$ is the Frobenious norm of $\mathbf{D}$. $diag(\cdot)$ is the operator that extracts the diagonal of a square matrix or constructs a diagonal matrix from a vector.

3.1. A generic cost function for alignment

This section proposes a generic quadratic error function where many PAMs can be cast. The quadratic error function has the form:

$$E(\mathbf{d},\mathbf{p}) = \mathbf{d}(\mathbf{f}(\mathbf{x},\mathbf{p}))^T\mathbf{A}\mathbf{d}(\mathbf{f}(\mathbf{x},\mathbf{p})) + 2\mathbf{b}^T\mathbf{d}(\mathbf{f}(\mathbf{x},\mathbf{p})) \tag{3}$$

Here $\mathbf{A} \in \Re^{m\times m}$ and $\mathbf{b} \in \Re^{m\times 1}$ are the fixed parameters of the function, and $\mathbf{A}$ is symmetric. This function is the general form of many cost functions used in the literature including Active Appearance Models [6], Eigentracking [4], and template tracking [16, 18]. For instance, consider the cost function given in (1). If $\mathbf{p}$ is fixed, the optimal $\mathbf{c}$ that minimizes (1) can be obtained using $\mathbf{c} = \mathbf{U}^T\mathbf{d}(\mathbf{f}(\mathbf{x},\mathbf{p}))$. Substituting this back into (1) and performing some basic algebra, (1) is equivalent to: $\min_{\mathbf{p}} \mathbf{d}(\mathbf{f}(\mathbf{x},\mathbf{p}))^T(\mathbf{I}_m - \mathbf{U}\mathbf{U}^T)\mathbf{d}(\mathbf{f}(\mathbf{x},\mathbf{p}))$. Thus (1) is a special case of (3), with $\mathbf{A} = \mathbf{I}_m - \mathbf{U}\mathbf{U}^T$, and $\mathbf{b} = \mathbf{0}_m$.

For template tracking, the cost function is typically the sum of squared differences: $||\mathbf{d}(\mathbf{f}(\mathbf{x},\mathbf{p})) - \mathbf{d}_{ref}||_2^2$, where $\mathbf{d}_{ref}$ is the reference template. This cost function is equivalent to: $\mathbf{d}(\mathbf{f}(\mathbf{x},\mathbf{p}))^T\mathbf{d}(\mathbf{f}(\mathbf{x},\mathbf{p})) - 2\mathbf{d}_{ref}^T\mathbf{d}(\mathbf{f}(\mathbf{x},\mathbf{p}))$. Thus the cost function used in template tracking is also a special case of (3) with $\mathbf{A} = \mathbf{I}_m$ and $\mathbf{b} = -\mathbf{d}_{ref}$.

3.2. Desired properties of cost functions

As discussed previously, it is desirable that the cost function have minima at and only at the 'right' places. In this section, we deliberately address this need as an optimization problem over $\mathbf{A}$ and $\mathbf{b}$.

Let $\{\mathbf{d}_i\}_1^n$ be a set of training images containing the objects of interest (e.g. faces), and assume the landmarks for the object shapes are available (e.g. manually labeled facial landmarks as in Fig. 5a). Let $\mathbf{s}_i$ be the vector containing the landmark coordinates of image $\mathbf{d}_i$. Given $\{\mathbf{s}_i\}_1^n$, we perform Procrustes analysis [7] and build the shape model as follows. First, the mean shape $\bar{\mathbf{s}} = \frac{1}{n}\sum_i \mathbf{s}_i$ is calculated. Second, we compute $\mathbf{a}_i$ the affine parameter that best transforms $\bar{\mathbf{s}}$ to $\mathbf{s}_i$, and let $\mathbf{a}_i^{-1}$ be the inverse affine transformation of $\mathbf{a}_i$. Third, $\hat{\mathbf{s}}_i$ is obtained by applying the inverse affine transformation $\mathbf{a}_i^{-1}$ on $\mathbf{s}_i$ (warping toward the mean shape). Next, we perform PCA on $\{\hat{\mathbf{s}}_i - \bar{\mathbf{s}}\}_{i=1}^n$ to construct $\mathbf{U}^s$, a basis for non-rigid shape variation. We then compute $\mathbf{c}_i^s$, the coefficients of $\hat{\mathbf{s}}_i - \bar{\mathbf{s}}$ w.r.t. the the basis $\mathbf{U}^s$. Finally, let $\mathbf{p}_i = [\mathbf{a}^i; \mathbf{c}_i^s]$, $\mathbf{p}_i$ is the parameter of image $\mathbf{d}_i$ w.r.t. to our shape model. Notably, the shape model and $\{\mathbf{p}_i\}_1^n$ are derived independently of the appearance model. The appearance model (i.e. the cost function $E(\mathbf{d},\mathbf{p})$) is what needs to be learned.

For $E(\mathbf{d}_i,\mathbf{p})$ to have a local minimum at the right place, $\mathbf{p}_i$ must be a local minimum of $E(\mathbf{d}_i,\mathbf{p})$. Theoretically, this

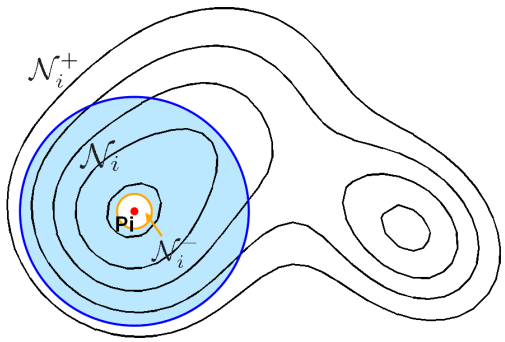

Figure 2. Neighborhoods around the ground truth motion parameter $\mathbf{p}_i$ (ret dot). $\mathcal{N}_i^-$: region inside the orange circle; it is satisfactory for fitting algorithms to converge to this region. $\mathcal{N}_i^+$: region outside the blue circle; alignment algorithm will not be initialized in this region. $\mathcal{N}_i$: shaded region, region to enforce constraints on gradients.

requires $\frac{\partial E(\mathbf{d}_i,\mathbf{p})}{\partial \mathbf{p}}\Big|_{\mathbf{p}_i}$ to vanish, i.e.

$$\frac{\partial E(\mathbf{d}_i,\mathbf{p})}{\partial \mathbf{p}}\Big|_{\mathbf{p}_i} = \mathbf{0} \quad \forall i \tag{4}$$

To learn a cost function that has few local minima, it is necessary to consider $\mathbf{p}_i$'s neighborhoods. Let $\mathcal{N}_i = \{\mathbf{p} : lb \le ||\mathbf{p}-\mathbf{p}_i||_2 \le ub\}$, $\mathcal{N}_i^- = \{\mathbf{p} : ||\mathbf{p}-\mathbf{p}_i||_2 < lb\}$, $\mathcal{N}_i^+ = \{\mathbf{p} : ||\mathbf{p}-\mathbf{p}_i||_2 > ub\}$. Here lb is chosen such that $\mathcal{N}_i^-$ is a set of neighbor parameters that are very close to $\mathbf{p}_i$; it is satisfactory for a fitting algorithm to converge to a point in $\mathcal{N}_i^-$. ub is chosen so that the fitting algorithm is guaranteed to be initialized at a point in $\mathcal{N}_i$ or $\mathcal{N}_i^-$. In most applications, such ub exists. For example, for tracking problems, ub can be set to the maximum movement of the object being tracked between two consecutive frames. Fig. 2 depicts the relationship between $\mathcal{N}_i^-, \mathcal{N}_i, and \mathcal{N}_i^+$.

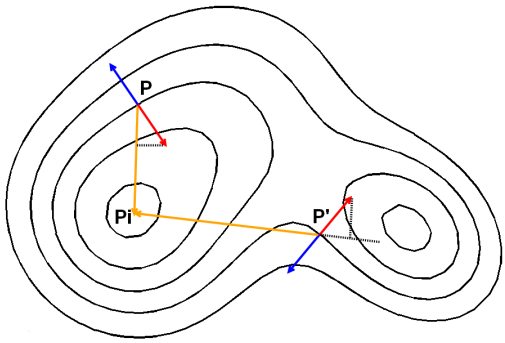

Figure 3. $\mathbf{p}_i$: desired convergence location. Blue arrows: gradient vectors, red arrows: walking directions of gradient descent algorithm, orange arrows: optimal directions to the desired location. Performing gradient descent at $\mathbf{p}$ advances closer to $\mathbf{p}_i$ while performing gradient descent at $\mathbf{p}'$ moves away from $\mathbf{p}_i$.

For a gradient descent algorithm to converge to $\mathbf{p}_i$ or a

point close enough to $\mathbf{p}_i$, it is necessary that $E(\mathbf{d}_i, .)$ have no local minima in $\mathcal{N}_i$. This implies that $\frac{\partial E(\mathbf{d}_i,\mathbf{p})}{\partial \mathbf{p}}$ does not vanish for $\mathbf{p} \in \mathcal{N}_i$. Notably, it is not necessary to enforce similar constraints for $\mathbf{p} \in \mathcal{N}_i^- \cup \mathcal{N}_i^+$ because of the way lb, ub are chosen. Another desirable property is that each iteration of gradient descent advances closer to the correct position. Because gradient descent walks against the gradient direction at every iteration, we would like the opposite direction of the gradient at point $\mathbf{p} \in \mathcal{N}_i$ to be similar to the optimal walking direction $\mathbf{p}_i - \mathbf{p}$. This quantity can be measured as the projection of the walking direction onto the optimal direction. Fig. 3 illustrates the rationale of this requirement. This requirement leads to the constraints:

$$\left\langle -\left(\frac{\partial E(\mathbf{d}_i,\mathbf{p})}{\partial \mathbf{p}}\right)^T, \frac{\mathbf{p}_i - \mathbf{p}}{||\mathbf{p}_i - \mathbf{p}||_2} \right\rangle > 0 \;\; \forall \mathbf{p} \in \mathcal{N}_i \quad (5)$$

Equations (4) and (5) specify the constraints for the ideal cost function. However, these constraints might be too stringent. Therefore, we propose to relax the constraints to get the optimization problem:

$$\begin{aligned} \min_{\mathbf{A},\mathbf{b},\boldsymbol{\xi}} \;& \frac{1}{2}\sum_i \left|\left| \frac{\partial E(\mathbf{d}_i,\mathbf{p})}{\partial \mathbf{p}}\Big|_{\mathbf{p}_i} \right|\right|_2^2 + C\sum_i \xi_i \quad (6) \\ \text{s.t.} \;& \left\langle -\left(\frac{\partial E(\mathbf{d}_i,\mathbf{p})}{\partial \mathbf{p}}\right)^T, \frac{\mathbf{p}_i - \mathbf{p}}{||\mathbf{p}_i - \mathbf{p}||_2} \right\rangle > -\xi_i \;\; \forall i, p \in \mathcal{N}_i \\ & \xi_i \geq 0 \quad \forall i \end{aligned}$$

Here $\left|\left| \frac{\partial E(\mathbf{d}_i,\mathbf{p})}{\partial \mathbf{p}}\Big|_{\mathbf{p}_i} \right|\right|_2^2$ is required to be small instead of strictly zero. ξ_i's are slack variables for constraints in (5) which allows for penalized constraint violation. C is the parameter controlling the trade-off between having few local minima and having local minima at the right places.

The gradient of the function $E(\mathbf{d}, \mathbf{p})$ plays a fundamental role in the above optimization problem. To compute the gradient $\frac{\partial E(\mathbf{d},\mathbf{p})}{\partial \mathbf{p}}$, it is common to use first order Taylor series expansion to approximate $\mathbf{d}(\mathbf{f}(\mathbf{x}, \mathbf{p} + \delta\mathbf{p}))$ by $\mathbf{d}(\mathbf{f}(\mathbf{x}, \mathbf{p})) + \mathbf{J}^{\mathbf{d}}(\mathbf{p})\delta\mathbf{p}$, where $\mathbf{J}^{\mathbf{d}}(\mathbf{p}) = \frac{\partial \mathbf{d}(\mathbf{f}(\mathbf{x},\mathbf{p}))}{\partial \mathbf{p}}$ is the spatial intensity gradient of the image $\mathbf{d}$ w.r.t. to the motion parameter $\mathbf{p}$ [16]. This yields:

$$\left(\frac{\partial E(\mathbf{d},\mathbf{p})}{\partial \mathbf{p}}\right)^T \approx 2(\mathbf{J}^{\mathbf{d}}(\mathbf{p}))^T(\mathbf{A}\mathbf{d}(\mathbf{f}(\mathbf{x},\mathbf{p})) + \mathbf{b}) \quad (7)$$

Substituting (7) into (6), we obtain a quadratic optimization problem with linear constraints over $\mathbf{A}$ and $\mathbf{b}$.

3.3. Practical issues and alternative fitting methods

In practice, there is an issue regarding the optimization of (6): the small components of $\frac{\partial E(\mathbf{d},\mathbf{p})}{\partial \mathbf{p}}$ tend to be neglected when optimizing (6). This occurs due to the magnitude difference between some columns of $\mathbf{J}^{\mathbf{d}}(\mathbf{p})$. For example, in (2), the magnitudes of the Jacobians of $\mathbf{d}(\mathbf{f}(\mathbf{x}, \mathbf{p}))$ w.r.t. to a_1, a_2, a_4, a_5 can be much larger than the magnitudes of the Jacobians of $\mathbf{d}(\mathbf{f}(\mathbf{x}, \mathbf{p}))$ w.r.t. to a_3, a_6.

To address this concern, we consider an alternative optimization strategy where the update rule at iteration k^{th} is:

$$\begin{aligned} & \mathbf{p}^{k+1} = \mathbf{p}^k + \Delta^{\mathbf{d}}(\mathbf{p}^k) \quad (8) \\ \text{with } & \Delta^{\mathbf{d}}(\mathbf{p}^k) = -\frac{1}{2}\mathbf{H}^{\mathbf{d}}(\mathbf{p}^k)^{-1}\left(\frac{\partial E(\mathbf{d},\mathbf{p})}{\partial \mathbf{p}}\Big|_{\mathbf{p}^k}\right)^T \\ & \mathbf{H}^{\mathbf{d}}(\mathbf{p}^k) = \mathbf{J}^{\mathbf{d}}(\mathbf{p}^k)^T\mathbf{J}^{\mathbf{d}}(\mathbf{p}^k) \end{aligned}$$

The update rule of the above algorithm is a variant of Newton iteration. Intuitively, $\mathbf{H}^{\mathbf{d}}(\mathbf{p}^k)$ is similar to the Hessian of $E(\mathbf{d}, \mathbf{p})$ at $\mathbf{p}^k$, and it acts as a normalization matrix for the gradient. This algorithm is indeed a reasonable optimization scheme for cost functions in which $\mathbf{A}$ is symmetric positive semidefinite with all eigenvalues less than or equal to 1. See Theorem 1 in the Appendix for the proof.

Similar to the case of gradient descent, requiring the incremental updates to vanish at only at the places corresponding to acceptable solutions yields the following optimization problem:

$$\begin{aligned} \min_{\mathbf{A},\mathbf{b},\boldsymbol{\xi}} \;& \frac{1}{2}\sum_i \left|\left|\Delta^{\mathbf{d}_i}(\mathbf{p}_i)\right|\right|_2^2 + C\sum_i \xi_i \quad (9) \\ \text{s.t.} \;& \left\langle \Delta^{\mathbf{d}_i}(\mathbf{p}), \frac{\mathbf{p}_i - \mathbf{p}}{||\mathbf{p}_i - \mathbf{p}||_2} \right\rangle > -\xi_i \;\; \forall i, \forall \mathbf{p} \in \mathcal{N}_i \\ & \xi_i \geq 0 \quad \forall i. \end{aligned}$$

$\mathbf{A}$ is also constrained to be a symmetric positive semidefinite matrix where eigenvalues are less than or equal one. By incorporating the ideas of maximal margin and regularization, we obtain:

$$\begin{aligned} \min_{\mathbf{A},\mathbf{b},\boldsymbol{\xi}} \;& \frac{1}{2}\sum_i \left|\left|\Delta^{\mathbf{d}_i}(\mathbf{p}_i)\right|\right|_2^2 + C\sum_i \xi_i + C_2\Omega(\mathbf{A},\mathbf{b}) \quad (10) \\ \text{s.t.} \;& \left\langle \Delta^{\mathbf{d}_i}(\mathbf{p}), \frac{\mathbf{p}_i - \mathbf{p}}{||\mathbf{p}_i - \mathbf{p}||_2} \right\rangle \geq C_3 - \xi_i \;\; \forall i, \forall \mathbf{p} \in \mathcal{N}_i \\ & \xi_i \geq 0 \quad \forall i \;\; \& \;\; \mathbf{A} \in \mathcal{H}_m, \end{aligned}$$

where $\mathcal{H}_m$ denotes the set of all $m \times m$ symmetric matrices of which all eigenvalues are non-negative and less than or equal to one. $\Omega(\mathbf{A}, \mathbf{b})$ is the regularization term for $\mathbf{A}$ and $\mathbf{b}$, C_2 is the weight for the regularization term, and C_3 is the user-defined margin size. Since $\Delta^{\mathbf{d}_i}(\mathbf{p}_i)$ is linear in terms of $\mathbf{A}$ and $\mathbf{b}$, this is a quadratic programming problem with linear constraints, provided the requirement $\mathbf{A} \in \mathcal{H}_m$ can be described by linear constraints.

Of course, one can derive a similar learning problem for $\mathbf{A}$ and $\mathbf{b}$ where the Newton method is the optimizer of choice. The incremental update in Newton iteration is:

$$-\frac{1}{2}\left[\mathbf{J}^{\mathbf{d}}(\mathbf{p}^k)^T\mathbf{A}\mathbf{J}^{\mathbf{d}}(\mathbf{p}^k)\right]^{-1}\left(\frac{\partial E(\mathbf{d},\mathbf{p})}{\partial \mathbf{p}}\bigg|_{\mathbf{p}^k}\right)^T \quad (11)$$

However, each Newton iteration has to invert $\mathbf{J}^{\mathbf{d}}(\mathbf{p}^k)^T\mathbf{A}\mathbf{J}^{\mathbf{d}}(\mathbf{p}^k)$. As a result, learning $\mathbf{A}$ and $\mathbf{b}$ becomes much harder because the optimization problem is no longer quadratic with linear constraints.

4. Special cases and experiments

Sec. 3.3 proposes a method for learning generic $\mathbf{A}$ and $\mathbf{b}$. However, in specific situations, $\mathbf{A}$ and $\mathbf{b}$ can be further parameterized. The benefits of further parameterization are threefold. First, the number of parameters to learn can be reduced. Second, the relationship between $\mathbf{A}$ and $\mathbf{b}$ can be established. Third, the constraint that $\mathbf{A} \in \mathcal{H}_m$ can be replaced by a set of linear constraints. This section provides the formulation for two special cases, namely weighted template alignment and weighted-basis AAM alignment. Experimental results on synthetic and real data are included.

4.1. Weighted template alignment

As shown in Sec. 3.1, template alignment is a special case of (3) in which $\mathbf{A} = \mathbf{I}_m$, and $\mathbf{b} = -\mathbf{d}_{ref}$. In template alignment, pixels of the template are weighed equally; however, there is no reason why this is optimal. Here, we propose learning the weights of template pixels to avoid local minima in template matching.

Consider the weighted sum of squared differences: $(\mathbf{d}(\mathbf{f}(\mathbf{x},\mathbf{p})) - \mathbf{d}_{ref})^T diag(\mathbf{w})(\mathbf{d}(\mathbf{f}(\mathbf{x},\mathbf{p})) - \mathbf{d}_{ref})$, where, $\mathbf{w}$ is the weight vector for the template's pixels. This cost function is equivalent to (3) with $\mathbf{A} = diag(\mathbf{w})$ and $\mathbf{b} = -diag(\mathbf{w})\mathbf{d}_{ref}$. The constraint $\mathbf{A} \in \mathcal{H}_m$ can be imposed by requiring $0 \leq w_i \leq 1$. Furthermore, in this setting, $\left|\left|\Delta^{\mathbf{d}_i}(\mathbf{p}_i)\right|\right|_2^2 = 0\ \forall i$. Thus (10) becomes a linear programming problem with linear constraints over $\mathbf{w}$.

To demonstrate this idea, we create a synthetic template of an isotropic Gaussian (Fig. 4a). Suppose the task is to locate the template inside an image containing the template (Fig. 4c), starting at an arbitrary location. Fig. 4d plots the error surface of the naive cost function (sum of squared differences). The value of this error surface at a particular pixel (x, y) is calculated by computing the sum of squared differences between the template and the circular patch centered at (x, y). Similarly, the error surface of the learned cost function (weighted sum of squared differences) is calculated and displayed in Fig. 4e. The learned template weights are shown in Fig. 4b; brighter pixels mean higher weights. As can be seen, the naive cost function has a fence of local maxima surrounding the template location. This prevents alignment algorithms from converging to the desired location. The learned cost function is convex, and therefore, is more suitable for this particular template.

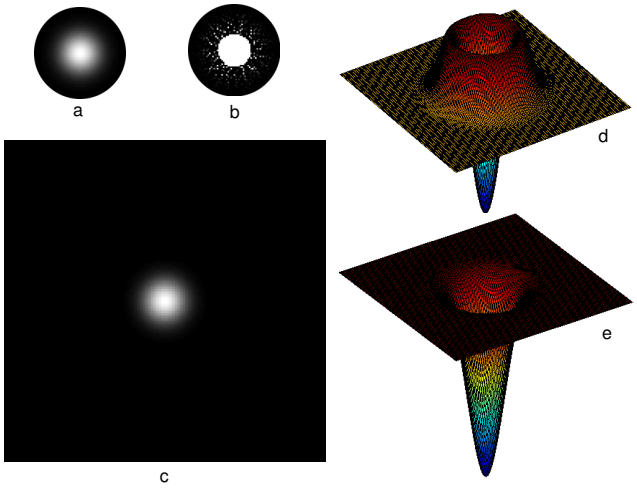

Figure 4. Learning to weight template's pixels. (a) synthetic template of an isotropic Gaussian. (b) the learned weights, brighter pixels mean higher weights. (c) an image containing the template. (d) error surface of the sum of squared differences. (e) error surface of the weighted sum of squared differences with the learned weights given in (b).

The template's weights given in Fig. 4b are learned by optimizing (10) with the following parameter settings: $\Omega(\mathbf{A},\mathbf{b}) = 0, C_2 = 0, C_3 = 10^{-2}, C = 1$. The linear constraints are reduced to a set of 5000 constraints obtained by random sampling. How to deal with infinitely many constraints is discussed in more detail in Sec. 4.2.

4.2. Weighted-basis for AAM alignment

As shown in Sec. 3.1, AAM alignment is a special case of (3) in which $\mathbf{A} = \mathbf{I}_m - \mathbf{U}\mathbf{U}^T = \mathbf{I}_m - \sum_1^k \mathbf{u}_i\mathbf{u}_i^T$, and $\mathbf{b} = \mathbf{0}$. $\mathbf{U}$ is the set of k first eigenvectors from the total of K PCA basis of the training data subspace. $k\ (\leq K)$ is usually chosen experimentally. In this section, we propose to use all K eigenvectors, but weigh them differently. Specifically, we learn $\mathbf{A}$ which has the form: $\mathbf{A} = \mathbf{I}_m - \sum_1^K \lambda_i\mathbf{u}_i\mathbf{u}_i^T$. To ensure that $\mathbf{A} \in \mathcal{H}_m$, we require $0 \leq \lambda_i \leq 1$. Let $\mathbf{w} = [\boldsymbol{\lambda}^T\ \mathbf{b}^T]^T$. Substituting this into (10) we get a quadratic programming problem with linear constraints on $\mathbf{w}$.

To demonstrate this idea, we perform experiments on the Multi-PIE database [12]. This database consists of facial images of 337 subjects taken under different illuminations, expressions and poses. We only make use of the directly-illuminated frontal face images under five expressions (smile, disgust, squint, surprise and scream). Our dataset contains 1100 images, 400 are selected for training, 200 are used for validation (parameter tuning), and the rest

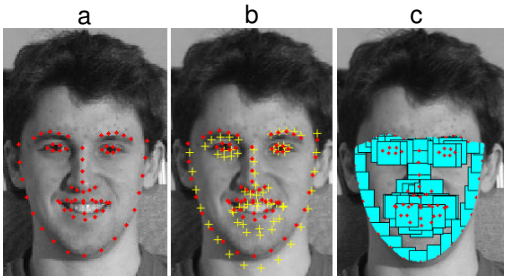

Figure 5. (a) example of landmarks associated with each face (red dots), (b) example of shape distortion (yellow pluses), (c) example of patches for appearance modeling.

are reserved for testing. Each face is manually labeled with 68 landmarks, as shown in Fig. 5a. Images are down sampled to 120×160 pixels.

The shape model is built as described in Sec. 3.2. The final shape model requires 10 coefficients (6 affine + 4 non-rigid) to describe a shape. For object appearance, we extract intensity values of pixels inside the patches located at the landmarks (Fig. 5c).

The training data is further divided into two subsets, one contains 300 images and the other contains 100 images. $\mathbf{U}$ is obtained by performing PCA on the subset of 300 images. The second subset is used to set up the optimization problem (10). For better generalization, (10) is constructed without using images in the first training subset. To avoid $\mathcal{N}_i$ being of infinite size, we restrict our attention to a set of 200 random samples from $\mathcal{N}_i$. The random samples are drawn by introducing random Gaussian perturbation to the correct shape parameter $\mathbf{p}_i$.

Following the approach by Tsochantarisdis *et al* [22] for minimizing a quadratic function with an exponentially large number of linear constraints, we maintain a smaller subset of active constraints $\mathbf{S}$ and optimize (10) iteratively. We repeat the following steps for 10 iterations: (i) empty $\mathbf{S}$; (ii) randomly choose 20 training images; (iii) for each chosen training image $\mathbf{d}_i$, find the 100 most violated constraints from $\mathcal{N}_i$ and include them in $\mathbf{S}$; (iv) run quadratic programming with the reduced set of constraints.

Testing data are generated by randomly perturbing the components of $\mathbf{p}_i$, the correct shape parameters of test image $\mathbf{d}_i$. Perturbation amounts are generated from a zero mean Gaussian distribution with standard deviation $PerMag \times [0.05\ 0.05\ 1\ 0.05\ 0.05\ 1\ 2\ 2\ 2\ 2]^T$. $PerMag$ controls the overall difficulty of the testing data. The relative perturbation amounts of shape coefficients are determined to simulate possible motion in tracking, and this is estimated visually. Fig. 5b shows an example of shape perturbation, the ground truth landmarks are marked in red (circles), while the perturbed shape is shown in yellow (pluses).

Table 1 describes the experimental results with four difficulty levels of testing data (controlled by $PerMag$). The performance of the learned cost function is compared with four other cost functions constructed using PCA with popular energy settings (70%, 80%, 90%, and 100%). As can be observed, when the amount of perturbation is small, PCA models with higher energy levels perform better. However, as the amount of pertubation increases, PCA models with lower energy levels perform better. This suggests that cost functions using fewer basis vectors have less local minima while cost functions using more basis vectors are more likely to have local minima at the 'right' places. Thus it is unclear what the energy for the PCA model should be. On the other hand, the learned cost function performs significantly better than the PCA models for most difficulty levels. In this experiment, we use $\Omega(\mathbf{A}, \mathbf{b}) = ||\mathbf{b}||_2^2, C = 2, C_2 = 0.1$, and $C_3 = 0.01$. The parameters are tuned using the validation set.

Table 1. Alignment results of different methods for four different difficulty levels of testing data (PerMag). *Initial* is the initial amount of perturbation before running any alignment algorithm. PCA $e\%$ is the cost function constructed using PCA preserving $e\%$ of energy. The table shows the means and standard deviations of mis-alignment (average over 68 landmarks and over testing data). The unit for measurement is pixel.

PerMag	0.75	1.00	1.25	1.5
Initial	0.75±.25	1.08±.38	1.37±.52	1.54±.54
PCA 100%	0.37±.18	0.41±.25	0.55±.45	0.60±.51
PCA 90%	**0.36±.20**	0.43±.33	0.47±.36	0.60±.65
PCA 80%	0.40±.23	0.43±.34	0.49±.37	0.57±.50
PCA 70%	0.41±.20	0.43±.25	0.47±.30	0.55±.46
Ours	0.37±.19	**0.40±.25**	**0.43±.29**	**0.48±.39**

5. Conclusion

In this paper, we have proposed a method for learning the cost functions for PAMs. We directly address the problem of learning cost functions that have local minima at and only at the desired places. The task of learning a cost function is formulated as optimizing a quadratic function under some linear constraints. To the best of our knowledge, this is the first paper that addresses this problem. Encouraging results have been achieved in the context of template matching and AAM fitting. Further work needs to address how to select the most interesting points in the error surface to reduce the number of constraints in the optimization.

Acknowledgments: This material is based upon work supported by the U.S. Naval Research Laboratory under Contract No. N00173-07-C-2040 and National Institute of Health Grant R01 MH 051435. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the U.S. Naval Research Laboratory.

Appendix

This section states and proves a theorem used to justify the optimization algorithm given in (8).

Theorem 1: Consider an m-dimensional function $f(\mathbf{x})$ of p-dimensional variable $\mathbf{x}$, and suppose we have to minimize the function: $E(\mathbf{x}) = f(\mathbf{x})^T\mathbf{A}f(\mathbf{x}) + 2\mathbf{b}^T f(\mathbf{x})$, where $\mathbf{A} \in \mathcal{H}_m$. Consider an iterative optimization method which has the following update rule:

$$\begin{aligned} & \mathbf{x}^{new} = \mathbf{x}^{old} + \delta\mathbf{x} \\ \text{with} \quad & \delta\mathbf{x} = -\mathbf{H}^{-1}\mathbf{J}^T\left(\mathbf{A}f(\mathbf{x}) + \mathbf{b}\right) \\ \text{and} \quad & \mathbf{J} = \left.\frac{\partial f}{\partial \mathbf{x}}\right|_{\mathbf{x}^{old}}, \mathbf{H} = \mathbf{J}^T\mathbf{J} \end{aligned} \tag{12}$$

The above optimization method, when started sufficiently close to a regular local minimum, will converge to that local minimum. Here, a point $\mathbf{x}_0$ is said to be regular if $\mathbf{H}$ is not singular and the Taylor series of $f(\cdot)$ converges for every point in the neighborhood of $\mathbf{x}_0$.

Proving Theorem 1 requires two lemmas. We now state and prove those two lemmas.

Lemma 1: $\mathbf{A} \in \mathcal{H}_m$ if and only if $\mathbf{I}_m - \mathbf{A} \in \mathcal{H}_m$.

Proof: This lemma can be proven easily, based on:

$$0 \leq \frac{\mathbf{u}^T\mathbf{A}\mathbf{u}}{\mathbf{u}^T\mathbf{u}} \leq 1 \Leftrightarrow 0 \leq \frac{\mathbf{u}^T(\mathbf{I}_m - \mathbf{A})\mathbf{u}}{\mathbf{u}^T\mathbf{u}} \leq 1 \ \forall \mathbf{u} \tag{13}$$

Lemma 2: $\mathbf{A} \in \mathcal{H}_m$ if and only if there exists a positive integer k, scalars α_i's, and matrices $\mathbf{B}_i$'s such that:

i. $\mathbf{B}_i^T\mathbf{B}_i$ is invertible $\forall i = \overline{1,k}$.
ii. $\alpha_i \geq 0 \ \forall i = \overline{1,k}$, and $\sum_{i=1}^k \alpha_i \leq 1$
iii. $\mathbf{A} = \sum_{i=1}^k \alpha_i \mathbf{B}_i(\mathbf{B}_i^T\mathbf{B}_i)^{-1}\mathbf{B}_i^T$

Proof for sufficiency conditions: Suppose there exist k, α_i's, and $\mathbf{B}_i$'s that satisfy all all three conditions above. Because $\mathbf{A}$ is a linear combination of symmetric matrices, $\mathbf{A}$ is also symmetric. We only need to prove that $\mathbf{A}$ is positive semidefinite of which all eigenvalues are less than or equal to 1. Consider $\mathbf{v}^T\mathbf{A}\mathbf{v}$ for an arbitrarily vector $\mathbf{v} \in \Re^m$:

$$\begin{aligned} \mathbf{v}^T\mathbf{A}\mathbf{v} &= \sum_{i=1}^k \alpha_i \mathbf{v}^T\mathbf{B}_i(\mathbf{B}_i^T\mathbf{B}_i)^{-1}\mathbf{B}_i^T\mathbf{v} \\ &= \sum_{i=1}^k \alpha_i \mathbf{v}^T\mathbf{B}_i(\mathbf{B}_i^T\mathbf{B}_i)^{-1}\mathbf{B}_i^T\mathbf{B}_i(\mathbf{B}_i^T\mathbf{B}_i)^{-1}\mathbf{B}_i^T\mathbf{v} \\ &= \sum_{i=1}^k \alpha_i ||\mathbf{B}_i(\mathbf{B}_i^T\mathbf{B}_i)^{-1}\mathbf{B}_i^T\mathbf{v}||_2^2 \end{aligned} \tag{14}$$

We know that $\mathbf{B}_i(\mathbf{B}_i^T\mathbf{B}_i)^{-1}\mathbf{B}_i^T$ is a projection matrix and $\mathbf{B}_i(\mathbf{B}_i^T\mathbf{B}_i)^{-1}\mathbf{B}_i^T\mathbf{v}$ is the projection of $\mathbf{v}$ in the subspace $\mathbf{B}_i$. Thus we have $||\mathbf{B}_i(\mathbf{B}_i^T\mathbf{B}_i)^{-1}\mathbf{B}_i^T\mathbf{v}||_2^2 \leq ||\mathbf{v}||_2^2 \ \forall i$. Therefore:

$$\mathbf{v}^T\mathbf{A}\mathbf{v} \leq (\sum_{i=1}^k \alpha_i)||\mathbf{v}||_2^2 \leq ||\mathbf{v}||_2^2 \tag{15}$$

Furthermore, we have $\mathbf{v}^T\mathbf{A}\mathbf{v} \geq 0$ because $||\mathbf{B}_i(\mathbf{B}_i^T\mathbf{B}_i)^{-1}\mathbf{B}_i^T\mathbf{v}||_2^2 \geq 0$, and $\alpha_i \geq 0 \ \forall i$. Combining this with the inequality in (15), we have: $0 \leq \mathbf{v}^T\mathbf{A}\mathbf{v} \leq \mathbf{v}^T\mathbf{v}$ Since these inequalities hold for arbitrary vector $\mathbf{v} \in \Re^m$, $\mathbf{A}$ must be an element of $\mathcal{H}_m$.

Proof for necessary conditions: Suppose $\mathbf{A} \in \mathcal{H}_m$. Consider the singular value decomposition of $\mathbf{A}, \mathbf{A} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^T$. Here, the columns of $\mathbf{U}$ are orthonormal vectors. $\mathbf{\Lambda}$ is a diagonal matrix, $\mathbf{\Lambda} = \text{diag}([\lambda_1, ..., \lambda_m])$ with $0 \leq \lambda_i \leq 1 \forall i$. Without loss of generality, suppose $\lambda_1 \geq \lambda_2 \geq ... \geq \lambda_m$. We have:

$$\begin{aligned} \mathbf{A} &= \mathbf{U}\mathbf{\Lambda}\mathbf{U}^T = \sum_{i=1}^m \lambda_i \mathbf{u}_i\mathbf{u}_i^T \\ &= \sum_{i=1}^{m-1}(\lambda_i - \lambda_{i+1})(\sum_{j=1}^i \mathbf{u}_j\mathbf{u}_j^T) + \lambda_m(\sum_{j=1}^m \mathbf{u}_j\mathbf{u}_j^T) \end{aligned} \tag{16}$$

Let $\alpha_i = \lambda_i - \lambda_{i+1}$ for $i = 1, ..., m-1$, and $\alpha_m = \lambda_m$. Let $\mathbf{B}_i = [\mathbf{u}_1 ... \mathbf{u}_i]$ for $i = \overline{1,m}$. Since $\{u_i\}_1^m$ is a set of orthonormal vectors, $\mathbf{B}_i^T\mathbf{B}_i = \mathbf{I}_i$ an identity matrix. Therefore, $\mathbf{B}_i(\mathbf{B}_i^T\mathbf{B}_i)^{-1}\mathbf{B}_i^T = \mathbf{B}_i\mathbf{B}_i^T = \sum_{j=1}^i \mathbf{u}_j\mathbf{u}_j^T$. Hence:

$$\mathbf{A} = \sum_{i=1}^m \alpha_i \mathbf{B}_i(\mathbf{B}_i^T\mathbf{B}_i)^{-1}\mathbf{B}_i^T \tag{17}$$

Finally, we have $\alpha_i \geq 0 \ \forall i$ and $\sum_{i=1}^m \alpha_i = \lambda_1 \leq 1$. This completes our proof for Lemma 1 $\square$.

Proof of Theorem 1: From Lemmas 1 and 2 we know that $\exists \alpha_i \geq 0, \exists \mathbf{B}_i$: $\mathbf{I}_m - \mathbf{A} = \sum_{i=1}^k \alpha_i \mathbf{B}_i(\mathbf{B}_i^T\mathbf{B}_i)^{-1}\mathbf{B}_i^T$ and $\sum_1^k \alpha_i \leq 1$. To prove Theorem 1, let us first consider the optimization of the following function:

$$\begin{aligned} E_2(\mathbf{x}, \{\mathbf{c}_i\}) &= \sum_{i=1}^k \alpha_i ||f(\mathbf{x}) - \mathbf{B}_i\mathbf{c}_i||_2^2 \\ &\quad + \alpha_0 ||f(\mathbf{x})||_2^2 + 2\mathbf{b}^T f(\mathbf{x}) \end{aligned} \tag{18}$$

with $\alpha_0 = 1 - \sum_{i=1}^k \alpha_i$. One way to optimize this function is using coordinate descent, alternating between:

i. minimizing E_2 w.r.t. $\mathbf{x}$ while fixing $\{\mathbf{c}_i\}$.
ii. minimizing E_2 w.r.t. $\{\mathbf{c}_i\}$ while fixing $\mathbf{x}$.

To minimize E_2 w.r.t. $\mathbf{x}$ while fixing $\{\mathbf{c}_i\}$, we can use the Newton method:

$$\mathbf{x}^{new} = \mathbf{x}^{old} - \left(\frac{\partial^2 E_2}{\partial \mathbf{x}^2}\right)^{-1}\left(\frac{\partial E_2}{\partial \mathbf{x}}\right)^T$$

Using the first order Taylor approximation, we have $f(\mathbf{x} + \delta\mathbf{x}) \approx f(\mathbf{x}) + \mathbf{J}\delta\mathbf{x}$ with $\mathbf{J} = \frac{\partial f}{\partial \mathbf{x}}$

$$\begin{aligned} \text{Thus} \quad & E_2(\mathbf{x} + \delta\mathbf{x}, \{\mathbf{c}_i\}) \approx E_2(\mathbf{x}, \{\mathbf{c}_i\}) + \delta\mathbf{x}^T\mathbf{J}^T\mathbf{J}\delta\mathbf{x} \\ & + 2\delta\mathbf{x}^T\mathbf{J}^T(f(x) - \sum_{i=1}^k \alpha_i\mathbf{B}_i\mathbf{c}_i + \mathbf{b}) \end{aligned} \tag{19}$$

$$\text{Hence} \quad \frac{\partial E_2}{\partial \mathbf{x}} \approx 2(f(x) - \sum_{i=1}^k \alpha_i\mathbf{B}_i\mathbf{c}_i + \mathbf{b})^T\mathbf{J} \tag{20}$$

$$\frac{\partial^2 E_2}{\partial \mathbf{x}^2} \approx 2\mathbf{J}^T\mathbf{J} \tag{21}$$

Therefore, we have the Newton update rule:

$$\mathbf{x}^{new} = \mathbf{x}^{old} - (\mathbf{J}^T\mathbf{J})^{-1}\mathbf{J}^T(f(x) - \sum_{i=1}^{k}\alpha_i\mathbf{B}_i\mathbf{c}_i + \mathbf{b}) \quad (22)$$

When $\mathbf{x}$ is fixed, $\{\mathbf{c}_i^*(\mathbf{x})\}$ that globally minimize E_2 are:

$$\mathbf{c}_i^*(\mathbf{x}) = (\mathbf{B}_i^T\mathbf{B}_i)^{-1}\mathbf{B}_i^T f(\mathbf{x}) \quad (23)$$

Combining (22) and (23), we have the update rule for minimizing E_2: $\mathbf{x}^{new} = \mathbf{x}^{old} - (\mathbf{J}^T\mathbf{J})^{-1}\mathbf{J}^T[\mathbf{A}f(\mathbf{x}) + \mathbf{b}]$ This update rule is exactly the same as the update rule given in (12). As a result, (12) will always lead us to a local minimum of E_2.

We now prove that a local minimum of E_2 obtained by (12) will be a local minimum of E. Suppose $(\mathbf{x}_0, \{\mathbf{c}_i^*(\mathbf{x}_0)\})$ is a local minimum of E_2, we have $\exists\epsilon_1 > 0$ such that $E_2(\mathbf{x}_0, \{\mathbf{c}_i^*(\mathbf{x}_0)\}) \leq E_2(\mathbf{x}_0+\delta\mathbf{x}, \{\mathbf{c}_i^*(\mathbf{x}_0)+\delta\mathbf{c}_i)\})\ \forall\delta\mathbf{x}, \delta\mathbf{c}_i : ||\delta\mathbf{x}||_2^2 + \sum_i ||\delta\mathbf{c}_i||_2^2 < \epsilon_1$. Because $\mathbf{c}_i^*(\mathbf{x})$ is a continuous function in terms of $\mathbf{x}$, we can always find $\epsilon_2 > 0$ small enough such that $\forall\delta\mathbf{x}$ if $||\delta\mathbf{x}||_2^2 < \epsilon_2$ then $||\delta\mathbf{x}||_2^2 + \sum_i ||\mathbf{c}_i^*(\mathbf{x}_0 + \delta\mathbf{x}) - \mathbf{c}_i^*(\mathbf{x}_0)||_2^2 < \epsilon_1$. Thus $\exists\epsilon_2$ such that $E_2(\mathbf{x}_0, \{\mathbf{c}_i^*(\mathbf{x}_0)\}) \leq E_2(\mathbf{x}_0 + \delta\mathbf{x}, \{\mathbf{c}_i^*(\mathbf{x}_0 + \delta\mathbf{x})\})\ \forall\delta\mathbf{x} : ||\delta\mathbf{x}||_2^2 < \epsilon_2$. On the other hand, one can easily verify that $E_2(\mathbf{x}, \{\mathbf{c}_i\}) \geq E_2(\mathbf{x}, \{\mathbf{c}_i^*(\mathbf{x})\}) = E(\mathbf{x})\ \forall\mathbf{x}$ Therefore, we have $\exists\epsilon_2 > 0$ such that $E(\mathbf{x}_0) \leq E(\mathbf{x}_0 + \delta\mathbf{x})\ \forall\delta\mathbf{x} : ||\delta\mathbf{x}||_2^2 < \epsilon_2$. Hence, $\mathbf{x}_0$ must be a local minimum of E.

To sum up, we have shown that (12) will converge to a local minimum of E_2. Furthermore, a local minimum of E_2 found by (12) is also a local minimum of E. Thus the update rule given in (12) is guaranteed to converge to a local minimum of E. This concludes our proof for Theorem 1 □.

References

[1] S. Baker and I. Matthews. Lucas-Kanade 20 years on: a unifying framework. *International Journal of Computer Vision*, 56(3):221–255, March 2004.

[2] J. R. Bergen, P. Anandan, K. J. Hanna, and R. Hingorani. Hierarchical model-based motion estimation. *European Conference on Computer Vision*, pages 237–252, 1992.

[3] M. J. Black, D. J. Fleet, and Y. Yacoob. Robustly estimating changes in image appearance. *Computer Vision and Image Understanding*, 78(1):8–31, 2000.

[4] M. J. Black and A. D. Jepson. Eigentracking: Robust matching and tracking of objects using view-based representation. *International Journal of Computer Vision*, 26(1):63–84, 1998.

[5] V. Blanz and T. Vetter. A morphable model for the synthesis of 3D faces. In *ACM SIGGRAPH*, 1999.

[6] T. Cootes, G. Edwards, and C. Taylor. Active appearance models. *PAMI*, 23(6):681–685, 2001.

[7] T. F. Cootes and C. Taylor. Statistical models of appearance for computer vision. Technical report, University of Manchester., 2001.

[8] F. de la Torre and M. J. Black. Robust parameterized component analysis: theory and applications to 2D facial appearance models. *Computer Vision and Image Understanding*, 91:53 – 71, 2003.

[9] F. de la Torre, A. Collet, J. Cohn, and T. Kanade. Filtered component analysis to increase robustness to local minima in appearance models. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2007.

[10] F. de la Torre, J. Vitrià, P. Radeva, and J. Melenchón. Eigenfiltering for flexible eigentracking. In *International Conference on Pattern Recognition*, pages 1118–1121, 2000.

[11] S. Gong, S. Mckenna, and A. Psarrou. *Dynamic Vision: From Images to Face Recognition*. Imperial College Press, 2000.

[12] R. Gross, I. Matthews, J. Cohn, T. Kanade, and S. Baker. The CMU multi-pose, illumination, and expression (Multi-PIE) face database. Technical report, Robotics Institute, Carnegie Mellon University, 2007. TR-07-08.

[13] I. Jolliffe. *Principal Component Analysis*. Springer-Verlag, New York, 1986.

[14] M. J. Jones and T. Poggio. Multidimensional morphable models. In *International Conference on Computer Vision*, pages 683–688, 1998.

[15] X. Liu. Generic face alignment using boosted appearance model. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2007.

[16] B. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *Proceedings of Imaging Understanding Workshop*, 1981.

[17] I. Matthews and S. Baker. Active appearance models revisited. *International Journal of Computer Vision*, 60(2):135–164, Nov. 2004.

[18] I. Matthews, T. Ishikawa, and S. Baker. The template update problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26:810–815, 2004.

[19] S. K. Nayar and T. Poggio. *Early Visual Learning*. Oxford University Press, 1996.

[20] J. Saragih and R. Goecke. A nonlinear discriminative approach to AAM fitting. In *International Conference on Computer Vision*, 2007.

[21] L. Sirovich and M. Kirby. Low-dimensional procedure for the characterization of human faces. *Journal of the Optical Society of America A: Optics, Image Science, and Vision*, 4(3):519–524, March 1987.

[22] I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun. Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research*, 6:1453–1484, 2005.

[23] M. Turk and A. Pentland. Eigenfaces for recognition. *Journal Cognitive Neuroscience*, 3(1):71–86, 1991.

[24] T.Vetter. Learning novel views to a single face image. In *International Conference on Automatic Face and Gesture Recognition*, pages 22–27, 1997.

[25] M. Wimmer, F. Stulp, S. J. Tschechne, and B. Radig. Learning robust objective functions for model fitting in image understanding applications. In *Proceedings of British Machine Vision Conference*, 2006.

[26] J. Xiao, S. Baker, I. Matthews, and T. Kanade. Real-time combined 2D+3D active appearance models. In *Conference on Computer Vision and Pattern Recognition*, volume II, pages 535–542, 2004.