Int J Comput Vis (2010) 88: 69–84
DOI 10.1007/s11263-009-0299-9

Metric Learning for Image Alignment

Minh Hoai Nguyen · Fernando de la Torre

Received: 17 February 2009 / Accepted: 15 September 2009 / Published online: 23 September 2009

Abstract Image alignment has been a long standing problem in computer vision. Parameterized Appearance Models (PAMs) such as the Lucas-Kanade method, Eigentracking, and Active Appearance Models are commonly used to align images with respect to a template or to a previously learned model. 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 registration process. Second, often few, if any, of the local minima of the cost function correspond to acceptable solutions. To overcome these problems, this paper proposes a method to learn a metric for PAMs that explicitly optimizes that 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 metric to explicitly model local properties of the PAMs' error surface. Synthetic and real examples show improvement in alignment performance in comparison with traditional approaches. In addition, we show how the proposed criteria for a good metric can be used to select good features to track.

Keywords Image alignment · Metric learning · Template matching · Active appearance models

M.H. Nguyen (✉) · F. de la Torre
Robotics Institute, Carnegie Mellon University, 5000 Forbes Ave, Pittsburgh, PA 15213, USA
e-mail: minhhoai@cmu.edu

F. de la Torre
e-mail: ftorre@cs.cmu.edu

1 Introduction

Image alignment is a fundamental building block of many computer vision based systems ranging from robotics applications to medical diagnosis. Because of its importance, image alignment has long been an important research topic in computer vision. In particular, Parameterized Appearance Models (PAMs) such as the Lucas-Kanade method (Lucas and Kanade 1981), Eigentracking (Black and Jepson 1998), Active Appearance Models (Cootes et al. 2001; de la Torre et al. 2000, 2007; Matthews and Baker 2004; Gong et al. 2000), and Morphable Models (Blanz and Vetter 1999; Jones and Poggio 1998) are among the most popular methods for aligning a new image w.r.t. another image or a previously learned model. Typically, appearance and/or shape variation of a class of objects are modeled by performing Principal Component Analysis (PCA) on training samples. In the Lucas-Kanade tracker, the model is typically the first or the previous image. Once the model has been built, finding the correspondence between the model and an image containing the object of interest is achieved by minimizing a cost function w.r.t. some geometric transformation parameters; this is referred to as the fitting, registration, or alignment process.

Although widely used, PAMs suffer from two major problems. 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 in the case of Active Appearance Models (AAMs). Figure 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). Figure 1f shows the contour plot of this error surface. As

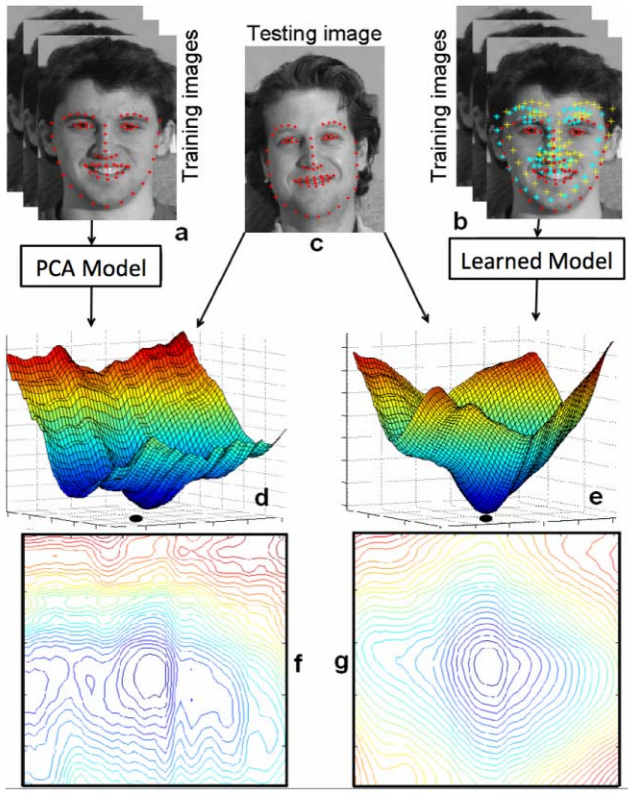

Fig. 1 (Color online) Learning a metric for image alignment. (**d**, **f**): Surface and contour plot of the PCA model. It has many local minima; (**e**, **g**): Our method learns a better error surface to fit PAMs. It has a global minimum in the expected location and no local minima in a given neighborhood

can be observed, any gradient-based optimization method is likely to get stuck in local minima and will 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. In the case of AAMs, these problems occur mainly 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 they should be taken into account while constructing the models.

On the other hand, in recent years distance metric learning techniques (see Yang 2006 for a review) have demonstrated both empirically and theoretically that a learned metric can significantly improve the performance in classification, clustering, and retrieval tasks. The aim of this paper is to define and learn a metric for image alignment. We propose to learn a distance metric (i.e., parameters of a cost function) that has a local minimum at the "expected" location and no other local minima in a defined neighborhood. Figure 1e, g plot the error surface and the contours of the learned cost function. This cost function has a local minimum at the expected place (black dot of Fig. 1e) and no other local minima near by. Moreover, we will show how the proposed criteria for an optimal metric can be used to select good features or pixels to track.

The rest of the paper is organized as follows. The next section reviews the previous work on image alignment especially PAMs. Section 3 discusses two desired properties of an optimal metric for image alignment and proposes a data driven approach for learning such a metric. Experiments on synthetic and Multi-PIE database (Gross et al. 2007) are provided in Sect. 4. Section 5 shows how the proposed criteria also can be used to select good features to track. Section 6 summarizes our findings and discusses several directions for future work. Appendix A states and proves a theorem that lays theoretical foundations for the learning formulation proposed in Sect. 3. Strictly speaking, our method learns a pseudometric rather than a metric. The difference between pseudometrics and metrics is subtle, and it is discussed in Appendix B.

2 Previous Work

Over the last decade, image alignment algorithms have become increasingly important in computer vision and graphics. In particular, PAMs have proven useful for image alignment, detection, tracking, and face synthesis (Lucas and Kanade 1981; Blanz and Vetter 1999; Black and Jepson 1998; de la Torre et al. 2000; Cootes et al. 2001; Matthews and Baker 2004; Nayar and Poggio 1996; Jones and Poggio 1998; Gong et al. 2000; Vetter 1997; de la Torre and Nguyen 2008). This section reviews PAMs and gradient-based methods for efficient alignment of high dimensional deformation models.

2.1 PAMs

PAMs (Lucas and Kanade 1981; Black and Jepson 1998; de la Torre et al. 2000; Cootes et al. 2001; Nayar and Poggio 1996; Jones and Poggio 1998; Blanz and Vetter 1999; Vetter 1997; de la Torre and Nguyen 2008) build object appearance and shape representation from the principal components of training data. Let $\mathbf{d}_i \in \Re^{m\times 1}$ (see Footnote 1 for an explanation of the notation[1]) be the ith sample of a train-

[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 jth column of the matrix $\mathbf{D}$. d_{ij} denotes the scalar in the row ith and column jth 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. $\mathrm{tr}(\mathbf{D}) = \sum_i d_{ii}$ is the trace of square matrix $\mathbf{D}$. $\|\mathbf{d}\|_2 = \sqrt{\mathbf{d}^T\mathbf{d}}$ designates Euclidean norm of $\mathbf{d}$. $\|\mathbf{D}\|_F = \sqrt{\mathrm{tr}(\mathbf{D}^T\mathbf{D})}$ is the Frobenious norm of $\mathbf{D}$. $\mathrm{diag}(\cdot)$ is the operator that extracts the diagonal of a square matrix or constructs a diagonal matrix from a vector.

ing set $\mathbf{D} \in \Re^{m\times n}$ and $\mathbf{U} \in \Re^{m\times k}$ the first k principal components (Jolliffe 1986). Once the appearance model has been constructed (i.e., $\mathbf{U}$ is known), alignment is achieved by finding the motion parameter $\mathbf{p}$ that best aligns the image w.r.t. the subspace $\mathbf{U}$, i.e.,

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

where $\mathbf{c}$ is the vector for the appearance coefficients that also are optimized. $\mathbf{x} = [x_1, y_1, \dots, x_l, y_l]^T$ is the vector containing the coordinates of the pixels to track. $\mathbf{f}(\mathbf{x},\mathbf{p})$ is the function for geometric transformation; the value of $\mathbf{f}(\mathbf{x},\mathbf{p})$ is a vector denoted by $[u_1, v_1, \dots, u_l, v_l]^T$. $\mathbf{d}$ is the image frame in consideration, and $\mathbf{d}(\mathbf{f}(\mathbf{x},\mathbf{p}))$ is the appearance vector of which the ith 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:

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

Here $[x_1^s, y_1^s, \dots, x_l^s, y_l^s]^T = \mathbf{x} + \mathbf{U}^s\mathbf{c}^s$, where $\mathbf{U}^s$ is the non-rigid shape model learned by performing PCA on a set of registered shapes (Cootes and Taylor 2001). $\mathbf{a}, \mathbf{c}^s$ are affine and non-rigid motion parameters respectively and $\mathbf{p} = [\mathbf{a}; \mathbf{c}^s]$, a combination of both affine and non-rigid motion parameters.

In the case of the (Lucas and Kanade 1981) tracker, $\mathbf{c}$ is fixed to be one and $\mathbf{U}$ is the subspace that contains a single vector, the reference template is the appearance of the tracked object in the initial/previous frame.

2.2 Optimization for PAMs

Given an image $\mathbf{d}$, PAM 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 (Bergen et al. 1992; Black and Jepson 1998; Baker and Matthews 2004; Cootes and Taylor 2001; Matthews and Baker 2004; de la Torre and Black 2003). To compute the gradient of the cost function given in (1), it is common to use Taylor series expansion to approximate:

$$\mathbf{d}(\mathbf{f}(\mathbf{x},\mathbf{p}+\delta\mathbf{p})) \approx \mathbf{d}(\mathbf{f}(\mathbf{x},\mathbf{p})) + \mathbf{J}^{\mathbf{d}}(\mathbf{p})\delta\mathbf{p}, \tag{3}$$

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 $\mathbf{p}$ (Lucas and Kanade 1981). There also exist other approximations that take into account more general noise models (Matei and Meer 2006; Kanatani 1996). Once linearized, a standard approach is to use the Gauss-Newton method for optimization (Bergen et al. 1992; Black and Jepson 1998). Other approaches learn an approximation of the Jacobian matrix with linear (Cootes and Taylor 2001) or non-linear (Saragih and Goecke 2007; Liu 2007) regression.

Over the last few years, several strategies to avoid local minima in the fitting process have been proposed. For example, Black and Jepson (1998) and Cootes and Taylor (2001) used multi-resolution schemes, Xiao et al. (2004) proposed to constrain the 2D shape with a 3D model, de la Torre et al. (2007) learned a multi-band representation robust to local minima, de la Torre and Black (2003) and Baker and Matthews (2004) learned a better PCA model invariant to rigid and non-rigid transformations. Recently, de la Torre and Nguyen (2008) proposed a kernel extension of AAMs, and showed some improved generalization in the fitting process. Although these methods show significant performance improvement, they do not directly address the problem of learning a metric for image alignment to explicitly minimize the number of local minima. In this paper, we deliberately learn a cost function which has local minima at and only at the desired places.

Recently and independently, there have been a couple of papers that pursue a goal similar to ours. Wimmer et al. (2006) proposed to improve Active Shape Models by learning separate one dimensional convex cost functions for individual landmark points. Wu et al. (2008) proposed a method to learn a discriminative appearance model for alignment using rank constraints. Our work differs from these methods in several aspects. We directly learn a metric for PAMs using convex quadratic programming. Our method jointly optimizes over rigid and non-rigid motion parameters. Furthermore, the proposed criteria for an optimal metric also can be used to select good features to track in feature-based tracking. Preliminary versions of this work have been presented in Nguyen and de la Torre (2008a, 2008b).

3 Learning Parameters of the Cost Function

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.

3.1 A Generic Cost Function for Alignment

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

$$E(\mathbf{d},\mathbf{p}) = \mathbf{d}(\mathbf{f}(\mathbf{x},\mathbf{p}))^T \mathbf{A}\mathbf{d}(\mathbf{f}(\mathbf{x},\mathbf{p})) + 2\mathbf{b}^T \mathbf{d}(\mathbf{f}(\mathbf{x},\mathbf{p})). \qquad (4)$$

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 that include Active Appearance Models (Cootes et al. 2001), Eigentracking (Black and Jepson 1998), and template tracking (Lucas and Kanade 1981; Matthews et al. 2004). 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:

$$\min_{\mathbf{p}} \quad \mathbf{d}(\mathbf{f}(\mathbf{x},\mathbf{p}))^T(\mathbf{I}_m - \mathbf{U}\mathbf{U}^T)\mathbf{d}(\mathbf{f}(\mathbf{x},\mathbf{p})). \qquad (5)$$

Thus (1) is a special case of (4), with $\mathbf{A} = \mathbf{I}_m - \mathbf{U}\mathbf{U}^T$ and $\mathbf{b} = \mathbf{0}_m$. Here, $\mathbf{I}_m$ denotes the $m \times m$ identity matrix.

For template alignment, the cost function is typically the Sum of Squared Differences (SSD):

$$\|\mathbf{d}(\mathbf{f}(\mathbf{x},\mathbf{p})) - \mathbf{d}_{ref}\|_2^2, \qquad (6)$$

where $\mathbf{d}_{ref}$ is the reference template. This cost function is equivalent to:

$$\mathbf{d}(\mathbf{f}(\mathbf{x},\mathbf{p}))^T\mathbf{d}(\mathbf{f}(\mathbf{x},\mathbf{p})) - 2\mathbf{d}_{ref}^T\mathbf{d}(\mathbf{f}(\mathbf{x},\mathbf{p})). \qquad (7)$$

Thus the cost function used in template tracking is also a special case of (4) with $\mathbf{A} = \mathbf{I}_m$ and $\mathbf{b} = -\mathbf{d}_{ref}$.

3.2 Desired Properties of Optimal Cost Functions

In this section we show how to learn the parameters of the cost function ($\mathbf{A}$ and $\mathbf{b}$) to have minima at and only at the 'right' places.

Let $\{\mathbf{d}_i\}_1^n$ be a set of training images containing the object 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 (Cootes and Taylor 2001) 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 towards 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 basis $\mathbf{U}^s$. Finally, let $\mathbf{p}_i = [\mathbf{a}_i; \mathbf{c}_i^s]$; $\mathbf{p}_i$ is the parameter for the image $\mathbf{d}_i$ w.r.t. to our shape model. Notably, the shape model and $\{\mathbf{p}_i\}_1^n$ are

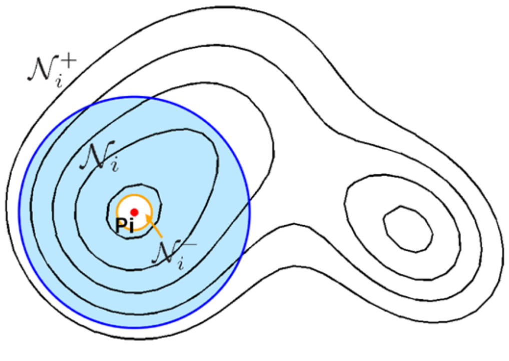

Fig. 2 (Color online) Neighborhoods around the ground truth motion parameter $\mathbf{p}_i$ (*red dot*). $\mathcal{N}_i^-$: region inside the orange circle; it is satisfactory for alignment algorithms to converge to this region. $\mathcal{N}_i^+$: region outside the blue circle; alignment algorithm will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

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

$$\left.\frac{\partial E(\mathbf{d}_i,\mathbf{p})}{\partial \mathbf{p}}\right|_{\mathbf{p}=\mathbf{p}_i} = \mathbf{0} \quad \forall i. \qquad (8)$$

To learn a cost function that has no 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. Figure 2 depicts the relationship between $\mathcal{N}_i^-$, $\mathcal{N}_i$, and $\mathcal{N}_i^+$.

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 and 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. Figure 3 illustrates the rationale of this requirement,

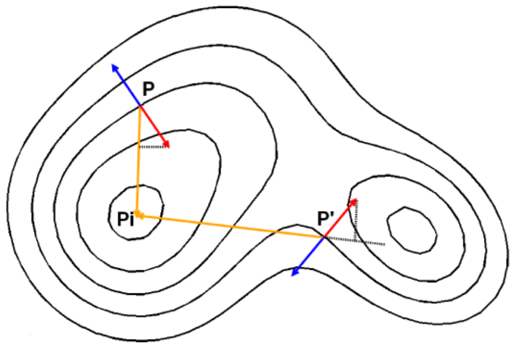

Fig. 3 (Color online) $\mathbf{p}_i$: desired convergence location. *Blue arrows*: gradient vectors, *red arrows*: walking directions of gradient descent algorithm, *orange arrows*: optimal directions to the desired location. Performing gradient descent at $\mathbf{p}$ advances closer to $\mathbf{p}_i$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

which leads to:

$$\left\langle -\left(\frac{\partial E(\mathbf{d}_i,\mathbf{p})}{\partial \mathbf{p}}\right)^T, \frac{\mathbf{p}_i-\mathbf{p}}{\|\mathbf{p}_i-\mathbf{p}\|_2}\right\rangle > 0 \quad \forall \mathbf{p}\in\mathcal{N}_i. \tag{9}$$

Equations (8) and (9) specify the constraints for the ideal cost function. This cost function can be obtained if its parameters can be chosen to satisfy these constraints. However, these inequalities might be too stringent to provide any feasible set of parameters; therefore, we focus on minimizing the constraint violation.

Equation (8) is equivalent to:

$$\left\| \left.\frac{\partial E(\mathbf{d}_i,\mathbf{p})}{\partial \mathbf{p}}\right|_{\mathbf{p}=\mathbf{p}_i}\right\|_2^2 = 0 \quad \forall i, \tag{10}$$

which can be relaxed by requiring the left hand side of (10) to be small instead of strictly zero. The constraint violation can be penalized by minimizing:

$$\min_{\mathbf{A},\mathbf{b}} \quad \frac{1}{2}\sum_i \left\| \left.\frac{\partial E(\mathbf{d}_i,\mathbf{p})}{\partial \mathbf{p}}\right|_{\mathbf{p}=\mathbf{p}_i}\right\|_2^2. \tag{11}$$

The set of constraints (9) can be handled similarly to the case of support vector machines (Vapnik 1998). First, instead of requiring the left hand side of (9) to be strictly positive, we will require it to be greater than or equal to a positive user-defined margin, μ:

$$\left\langle -\left(\frac{\partial E(\mathbf{d}_i,\mathbf{p})}{\partial \mathbf{p}}\right)^T, \frac{\mathbf{p}_i-\mathbf{p}}{\|\mathbf{p}_i-\mathbf{p}\|_2}\right\rangle \geq \mu \quad \forall \mathbf{p}\in\mathcal{N}_i. \tag{12}$$

The idea of requiring the constraints to be well satisfied by a margin was introduced previously, see Taskar et al. (2003) for an example. Unfortunately, the family of parameters for the cost function often are not rich enough to satisfy all constraints. In such cases, we need to introduce slack variables, ξ_i's, to allow for some constraints to be violated.

Combining the relaxation of both sets of constraints (8) and (9), we obtain the following optimization problem:

$$\begin{aligned} \min_{\mathbf{A},\mathbf{b},\boldsymbol{\xi}} \quad & \frac{1}{2}\sum_i \left\| \left.\frac{\partial E(\mathbf{d}_i,\mathbf{p})}{\partial \mathbf{p}}\right|_{\mathbf{p}=\mathbf{p}_i}\right\|_2^2 + C\sum_i \xi_i, \\ \text{s.t.} \quad & \left\langle -\left(\frac{\partial E(\mathbf{d}_i,\mathbf{p})}{\partial \mathbf{p}}\right)^T, \frac{\mathbf{p}_i-\mathbf{p}}{\|\mathbf{p}_i-\mathbf{p}\|_2}\right\rangle \geq \underset{\forall i,p\in\mathcal{N}_i}{\mu-\xi_i}, \\ & \xi_i \geq 0 \quad \forall i. \end{aligned} \tag{13}$$

C is the parameter controlling the trade-off between two types of constraint violation: (8) versus (9). It is the trade-off between having fewer 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 the first order Taylor series expansion to approximate:

$$\mathbf{d}(\mathbf{f}(\mathbf{x},\mathbf{p}+\delta\mathbf{p})) \approx \mathbf{d}(\mathbf{f}(\mathbf{x},\mathbf{p})) + \mathbf{J}^{\mathbf{d}}(\mathbf{p})\delta\mathbf{p}, \tag{14}$$

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 $\mathbf{p}$ (Lucas and Kanade 1981). This yields:

$$\left(\frac{\partial E(\mathbf{d},\mathbf{p})}{\partial \mathbf{p}}\right)^T \approx 2(\mathbf{J}^{\mathbf{d}}(\mathbf{p}))^T(\mathbf{A}\mathbf{d}(\mathbf{f}(\mathbf{x},\mathbf{p}))+\mathbf{b}). \tag{15}$$

Substituting (15) into (13), we obtain a quadratic program 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 (13): the small components of $\frac{\partial E(\mathbf{d},\mathbf{p})}{\partial \mathbf{p}}$ tend to be neglected when optimizing (13). 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 Jacobian of $\mathbf{d}(\mathbf{f}(\mathbf{x},\mathbf{p}))$ w.r.t. a_1, a_2, a_4, a_5 can be much larger than the magnitudes of the Jacobian of $\mathbf{d}(\mathbf{f}(\mathbf{x},\mathbf{p}))$ w.r.t. a_3, a_6.

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

$$\mathbf{p}^{k+1} = \mathbf{p}^k + \Delta^{\mathbf{d}}(\mathbf{p}^k), \tag{16}$$

with

$$\Delta^{\mathbf{d}}(\mathbf{p}^k) = -\frac{1}{2}\mathbf{H}^{\mathbf{d}}(\mathbf{p}^k)^{-1}\left(\left.\frac{\partial E(\mathbf{d},\mathbf{p})}{\partial \mathbf{p}}\right|_{\mathbf{p}=\mathbf{p}^k}\right)^T,$$

$$\mathbf{H}^{\mathbf{d}}(\mathbf{p}^k) = \mathbf{J}^{\mathbf{d}}(\mathbf{p}^k)^T\mathbf{J}^{\mathbf{d}}(\mathbf{p}^k).$$

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 especially well suited to optimize a cost function in which $\mathbf{A}$ is symmetric positive semidefinite with all eigenvalues less than or equal to one. Under these assumptions, the above optimization scheme is guaranteed to converge to a local minimum. This is proved in Appendix A.

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

$$\min_{\mathbf{A},\mathbf{b},\boldsymbol{\xi}} \quad \frac{1}{2}\sum_i \left\| \Delta^{\mathbf{d}_i}(\mathbf{p}_i) \right\|_2^2 + C\sum_i \xi_i, \tag{17}$$
$$\text{s.t.} \quad \left\langle \Delta^{\mathbf{d}_i}(\mathbf{p}), \frac{\mathbf{p}_i - \mathbf{p}}{\|\mathbf{p}_i - \mathbf{p}\|_2} \right\rangle \geq \mu - \xi_i \quad \forall i, \forall \mathbf{p} \in \mathcal{N}_i,$$
$$\xi_i \geq 0 \quad \forall i.$$

For the cost function to be a metric and for the theoretical guarantee of the above optimization scheme, $\mathbf{A}$ is constrained to be a symmetric positive semidefinite matrix where eigenvalues are less than or equal to one. Let $\mathcal{H}_m$ denote the set of all $m \times m$ symmetric matrices of which all eigenvalues are non-negative and less than or equal to one. The learning formulation becomes:

$$\min_{\mathbf{A},\mathbf{b},\boldsymbol{\xi}} \quad \frac{1}{2}\sum_i \left\| \Delta^{\mathbf{d}_i}(\mathbf{p}_i) \right\|_2^2 + C\sum_i \xi_i, \tag{18}$$
$$\text{s.t.} \quad \left\langle \Delta^{\mathbf{d}_i}(\mathbf{p}), \frac{\mathbf{p}_i - \mathbf{p}}{\|\mathbf{p}_i - \mathbf{p}\|_2} \right\rangle \geq \mu - \xi_i \quad \forall i, \forall \mathbf{p} \in \mathcal{N}_i,$$
$$\xi_i \geq 0 \quad \forall i \;\&\; \mathbf{A} \in \mathcal{H}_m.$$

Since $\Delta^{\mathbf{d}_i}(\mathbf{p}_i)$ is linear in terms of $\mathbf{A}$ and $\mathbf{b}$, this is a quadratic program with linear constraints, provided the requirement $\mathbf{A} \in \mathcal{H}_m$ can be expressed by a set of linear constraints.

One can derive a similar learning formulation for $\mathbf{A}$ and $\mathbf{b}$ where Newton's method is the optimizer of choice for (4). The update rule for iteration k of Newton's method is:

$$\mathbf{p}^{k+1} = \mathbf{p}^k + \Delta_{nt}^{\mathbf{d}}(\mathbf{p}^k), \tag{19}$$
$$\Delta_{nt}^{\mathbf{d}}(\mathbf{p}^k) = -\frac{1}{2}\mathbf{H}_{nt}^{\mathbf{d}}(\mathbf{p}^k)^{-1} \left(\left. \frac{\partial E(\mathbf{d}, \mathbf{p})}{\partial \mathbf{p}} \right|_{\mathbf{p}=\mathbf{p}^k} \right)^T$$

with $\mathbf{H}_{nt}^{\mathbf{d}}(\mathbf{p}^k) = \mathbf{J}^{\mathbf{d}}(\mathbf{p}^k)^T \mathbf{A} \mathbf{J}^{\mathbf{d}}(\mathbf{p}^k)$.

Similar to the case of gradient descent and the optimization scheme in (16), reasoning about the incremental update of Newton's method leads to the following learning formulation for $\mathbf{A}$ and $\mathbf{b}$:

$$\min_{\mathbf{A},\mathbf{b},\boldsymbol{\xi}} \quad \frac{1}{2}\sum_i \left\| \Delta_{nt}^{\mathbf{d}_i}(\mathbf{p}_i) \right\|_2^2 + C\sum_i \xi_i, \tag{20}$$
$$\text{s.t.} \quad \left\langle \Delta_{nt}^{\mathbf{d}_i}(\mathbf{p}), \frac{\mathbf{p}_i - \mathbf{p}}{\|\mathbf{p}_i - \mathbf{p}\|_2} \right\rangle \geq \mu - \xi_i \quad \forall i, \forall \mathbf{p} \in \mathcal{N}_i,$$
$$\xi_i \geq 0 \quad \forall i.$$

Equation (20) is very similar to (17); the only difference is that (17) uses the incremental update $\Delta^{\mathbf{d}_i}(\mathbf{p})$ while (20) uses $\Delta_{nt}^{\mathbf{d}_i}(\mathbf{p})$. Compare $\Delta^{\mathbf{d}_i}(\mathbf{p})$ and $\Delta_{nt}^{\mathbf{d}_i}(\mathbf{p})$, the former is linear in terms of $\mathbf{A}$ and $\mathbf{b}$ while the latter is not. $\Delta_{nt}^{\mathbf{d}_i}(\mathbf{p})$ is not linear in terms of $\mathbf{A}$ and $\mathbf{b}$ because it involves the inversion of $\mathbf{H}_{nt}^{\mathbf{d}_i}(\mathbf{p}) = \mathbf{J}^{\mathbf{d}_i}(\mathbf{p})^T \mathbf{A} \mathbf{J}^{\mathbf{d}_i}(\mathbf{p})$ which depends on $\mathbf{A}$. Meanwhile, the normalization matrix of $\Delta^{\mathbf{d}_i}(\mathbf{p})$, $\mathbf{H}^{\mathbf{d}_i}(\mathbf{p}) = \mathbf{J}^{\mathbf{d}_i}(\mathbf{p})^T \mathbf{J}^{\mathbf{d}_i}(\mathbf{p})$, does not depend on $\mathbf{A}$.

Because $\Delta_{nt}^{\mathbf{d}_i}(\mathbf{p})$ is not linear in terms of $\mathbf{A}$ and $\mathbf{b}$, (20) is not a quadratic program. As a result, learning $\mathbf{A}$ and $\mathbf{b}$ would be harder if the learning formulation was derived from the incremental update of Newton's method.

It is necessary to distinguish between two different optimization problems. One problem is to optimize the learning formulations (see (13), (18), or (20)) to learn $\mathbf{A}$ and $\mathbf{b}$, the parameters of the cost function. Another problem is to optimize the learned cost function $E(\mathbf{d}, \mathbf{p})$ w.r.t. to the motion parameter $\mathbf{p}$ for image alignment. The latter problem, once the cost function has been learned, can be optimized using any gradient-based algorithms including gradient descent, the optimization scheme in (16), and Newton's method. The connection between a learning formulation and an optimization scheme is considered from a different perspective. The learning formulation is derived by reasoning about the walking direction of an optimization scheme. Different optimization schemes lead to different learning formulations (c.f., (13), (17), and (20)). In theory, one also can reason about the Newton direction to derive a similar learning formulation for the cost function. However, as shown above, the analytic formulae for Newton iteration includes the inversion of $\mathbf{J}^{\mathbf{d}_i}(\mathbf{p})^T \mathbf{A} \mathbf{J}^{\mathbf{d}_i}(\mathbf{p})$. This will introduce non-linear constraints (in terms of $\mathbf{A}$) into the learning formulation if it is derived using Newton's method. This would make the learning formulation a much harder problem to optimize. In short, although Newton's method can be used for alignment (i.e., minimizing $E(\mathbf{d}, \mathbf{p})$), it is not preferable to be used for deriving the learning formulation. This is why we introduce (16), a novel optimization scheme that addresses the issues of gradient descent and Newton's method.

4 Special Cases and Experiments

Section 3.3 proposes a method for learning a generic $\mathbf{A}$ and $\mathbf{b}$. However, in many cases of interest, $\mathbf{A}$ and $\mathbf{b}$ can be further parameterized. The benefits of further parameterization are fourfold. 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. Fourth, the metric property of the obtained cost function can be guaranteed. 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 Sect. 3.1, template alignment is a special case of (4) 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 to believe that all pixels should contribute equally to construct optimal fitting surfaces. Here, we propose learning the weights of template pixels to avoid local minima in template matching.

Consider the weighted SSD:

$$(\mathbf{d}(\mathbf{f}(\mathbf{x},\mathbf{p})) - \mathbf{d}_{ref})^T \operatorname{diag}(\mathbf{w})(\mathbf{d}(\mathbf{f}(\mathbf{x},\mathbf{p})) - \mathbf{d}_{ref}), \qquad (21)$$

where $\mathbf{w}$ is the weight vector for the template's pixels. This cost function is equivalent to (4) with $\mathbf{A} = \operatorname{diag}(\mathbf{w})$ and $\mathbf{b} = -\operatorname{diag}(\mathbf{w})\mathbf{d}_{ref}$. The constraint $\mathbf{A} \in \mathcal{H}_m$ can be imposed by requiring $0 \le w_i \le 1$. Furthermore, if the template $\mathbf{d}_{ref}$ is part of the images $\mathbf{d}_i$'s, then $\|\Delta^{\mathbf{d}_i}(\mathbf{p}_i)\|_2^2 = 0\ \forall i$. In this case, (18) becomes a linear program with linear constraints on $\mathbf{w}$.

4.1.1 Experiments on Synthetic Data

To demonstrate this idea, we create a synthetic template of an isotropic Gaussian (Fig. 4b). Suppose that the task is to locate the template inside an image containing the template (Fig. 4c), starting at an arbitrary location. Figure 4d plots the error surface of the naive cost function SSD. The value of this error surface at a particular pixel (x, y) is calculated by computing the SSD between the template and the circular patch centered at (x, y). Similarly, the error surface of the learned cost function (weighted SSD) is calculated and displayed in Fig. 4e. The learned template weights are shown in Fig. 4f; 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 quasi-convex, and therefore, is more suitable for this particular template.

The template's weights given in Fig. 4f are learned by optimizing (18) with $\mu = 0.01$ and $C = 1$. The linear constraints are reduced to a set of 5000 constraints obtained by random sampling. How to deal with large or even infinitely many constraints is discussed in more detail in Sect. 4.2.

4.1.2 Experiments on the Multi-PIE Database

The second experiment is on the Multi-PIE database (Gross et al. 2007). This database contains the face images of 337

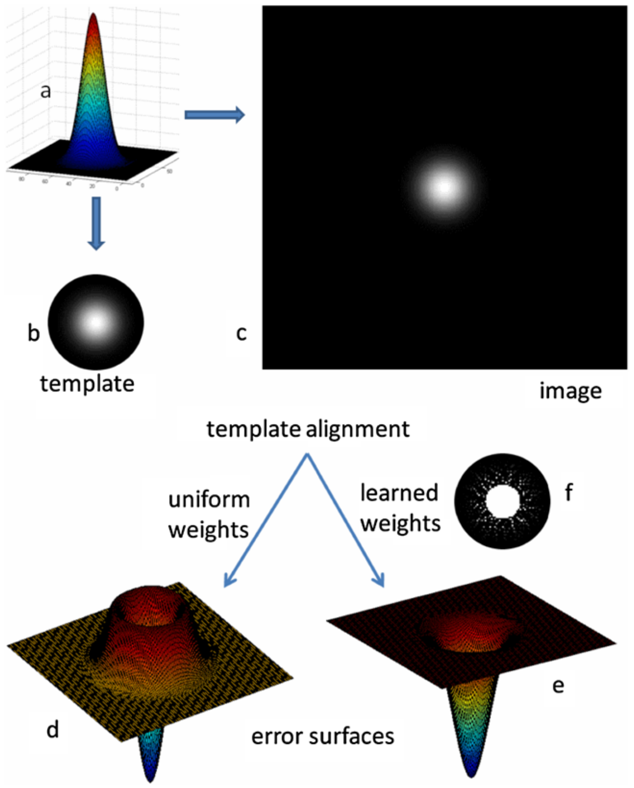

Fig. 4 (Color online) Learning to weight template's pixels. (**a**) An isotropic Gaussian. (**b**) A synthetic template of which the 3D representation is given in (**a**). (**c**) An image containing the template. (**d**) SSD error surface. (**e**) Weighted SSD error surface, and learned weights are given in (**f**)

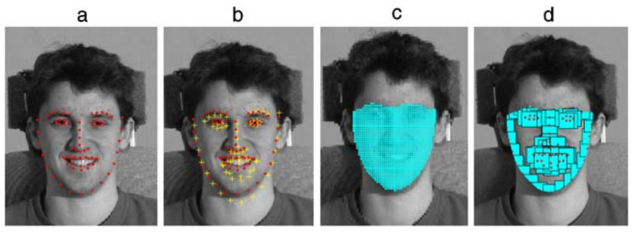

Fig. 5 (Color online) (**a**) Example of hand labeled landmarks associated with each face (*red dots*), (**b**) example of shape distortion (*yellow pluses*), (**c**) location of pixels for appearance modeling for the experiment in Sect. 4.1.2, (**d**) example of patches for appearance modeling for the experiment in Sect. 4.2

subjects taken under different illuminations, expressions, and poses. Each face is manually labeled with 68 landmarks, as shown in Fig. 5a. Images are down-sampled to 120 × 160 pixels

For this experiment, we only use directly-illuminated frontal neutral faces. Each subject in the Multi-PIE database might have more than one image. To ensure the testing data is completely independent of the training data, we restrict our attention to at most one image per subject. Our dataset contains 217 images, 10 are selected for training, 65 are used

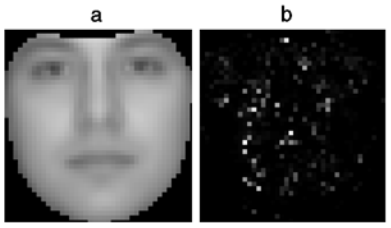

Fig. 6 (**a**) The template (mean image) used in weighted template experiment. (**b**) The learned weights, brighter pixels mean higher weights. Interestingly, the eye and mouth regions do not receive high weights

Table 1 Template alignment experiments: results of different methods for three different difficulty levels of testing data (PerMag). *Initial* is the initial amount of perturbation before running any alignment algorithm. *SSD* is the method which gives uniform weights for the pixels. Weighted SSD is the cost function learned by our method. The table shows the means and standard errors of misalignment (average over 68 landmarks and over testing data). The unit for measurement is pixel

PerMag	0.75	1.25	1.75
Initial	0.99 ± 0.02	1.37 ± 0.04	1.93 ± 0.06
SSD	1.13 ± 0.11	1.22 ± 0.15	1.38 ± 0.23
Weighted SSD	$\mathbf{0.98} \pm \mathbf{0.05}$	$\mathbf{1.05} \pm \mathbf{0.06}$	$\mathbf{1.20} \pm \mathbf{0.07}$

for validation (parameter tuning), and the rest are reserved for testing.

The shape model is built by aligning the training images using Procrustes analysis, as explained in Sect. 3.2. In this experiment we only consider affine transformation (six parameters). For the object appearance, we extract intensity values of pixels inside the region formed by the landmarks (Fig. 5c).

The template is the mean image of all aligned training images (Fig. 6a). Thus the task is to do template alignment between an arbitrary image with the mean image. The template's weights are learned by optimizing (18) with $\mu = 0.01$ and $C = 1$. To avoid $\mathcal{N}_i$ being of infinite size, we restrict our attention to a set of 150 random samples from $\mathcal{N}_i$. The random samples are drawn by introducing random Gaussian perturbation to the correct shape parameter $\mathbf{p}_i$. Figure 6b displays the learned weights; brighter pixels mean higher weights. Notably, the pixels in the eye and mouth regions do not receive high weights. This is consistent with the intuition that a cost function using high weights in areas with high variability are more susceptible to local minima.

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 $PerMag \times [0.05\ 0.05\ 1\ 0.05\ 0.05\ 1]^T$. *PerMag* controls the overall difficulty of the testing data. The relative perturbation amounts of shape coefficients are determined to simulate possible motion in tracking. Figure 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 shows the experimental results with three difficulty levels of testing data (controlled by *PerMag*). The performance of the learned cost function, weighted SSD, is compared with SSD, the cost function giving uniform weights for all the template's pixels. The learned cost function outperforms SSD in all levels of perturbation.

4.2 Weighted-Basis for AAM Alignment

As shown in Sect. 3.1, AAM alignment is a special case of (4) 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 $0 \leq \lambda_i \leq 1$. To ensure the resulted cost function is a metric, we also enforce $\mathbf{b} = \mathbf{0}$.

From the Multi-PIE database, 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 are reserved for testing.

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

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 (18). For better generalization, (18) 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 1000 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 Tsochantaridis et al. (2005) for minimizing a quadratic function with an exponentially large number of linear constraints, we maintain a smaller subset of active constraints $\mathcal{S}$ and optimize (18) iteratively. We repeat the following steps for 50 iterations: (i) empty $\mathcal{S}$; (ii) for each training image $\mathbf{d}_i$, find 25 most violated constraints from $\mathcal{N}_i$ and include them in $\mathcal{S}$; (iii) run quadratic programming with the reduced set of constraints.

Table 2 AAM alignment experiments: 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 errors of misalignment (average over 68 landmarks and over testing data). The unit for measurement is pixel

PerMag	0.75	1.00	1.25	1.5	1.75
Initial	$0.78 \pm .01$	$1.06 \pm .02$	$1.31 \pm .02$	$1.61 \pm .02$	$1.82 \pm .03$
PCA 90%	$\mathbf{0.40 \pm .01}$	$0.45 \pm .01$	$0.48 \pm .02$	$0.61 \pm .02$	$0.66 \pm .03$
PCA 80%	$0.42 \pm .01$	$0.44 \pm .01$	$0.48 \pm .024$	$0.59 \pm .02$	$0.66 \pm .03$
PCA 70%	$0.45 \pm .01$	$0.48 \pm .01$	$0.51 \pm .02$	$0.57 \pm .02$	$0.63 \pm .03$
Ours	$\mathbf{0.40 \pm .01}$	$\mathbf{0.42 \pm .01}$	$\mathbf{0.45 \pm .01}$	$\mathbf{0.52 \pm .02}$	$\mathbf{0.58 \pm .03}$

Similar to the case of weighted template alignment, 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 $PerMag \times [0.05\ 0.05\ 1\ \ 0.05\ 0.05\ 1\ 2\ 2\ 2\ 2]^T$.

Table 2 describes the experimental results with four difficulty levels of testing data (controlled by *PerMag*). The performance of the learned cost function is compared with three other cost functions constructed using PCA with popular energy settings (70%, 80%, and 90%). As can be observed, when the amount of perturbation is small, PCA models with higher energy levels perform better. However, as the amount of perturbation 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. To some extent, our method learns a Pareto optimal tradeoff between having less local minima and having local minima at the right places. In this experiment we used $\mu = 0.01$ and $C = 0.5$. The parameters were tuned using the validation set.

4.3 More Implementation Details and Discussion

This section describes several implementation details including parameter tuning. Learning and alignment speeds also are reported and discussed.

Section 3.2 defines $\mathcal{N}_i^-$ and $\mathcal{N}_i^+$ as spherical neighborhoods around $\mathbf{p}_i$ to simplify the presentation. However, in our implementation, $\mathcal{N}_i^-$ and $\mathcal{N}_i^+$ are ellipsoid neighborhoods instead of being spherical. Mathematically, $\mathcal{N}_i^-$, $\mathcal{N}_i^+$, and $\mathcal{N}_i$ are defined as follows:

$$\mathcal{N}_i^- = \{\mathbf{p} : (\mathbf{p} - \mathbf{p}_i)^T \operatorname{diag}(\boldsymbol{\omega})(\mathbf{p} - \mathbf{p}_i) < lb\}, \tag{22}$$

$$\mathcal{N}_i^+ = \{\mathbf{p} : (\mathbf{p} - \mathbf{p}_i)^T \operatorname{diag}(\boldsymbol{\omega})(\mathbf{p} - \mathbf{p}_i) > ub\}, \tag{23}$$

$$\mathcal{N}_i \ = \{\mathbf{p} : lb \le (\mathbf{p} - \mathbf{p}_i)^T \operatorname{diag}(\boldsymbol{\omega})(\mathbf{p} - \mathbf{p}_i) \le ub\}. \tag{24}$$

This modification is necessary because the shape model is more sensitive to some parameters than it is to others. In our experiments, a reasonable setting is determined by examining the training data. In particular, we set $\omega_i = 0.05$ for the parameters corresponding to rotation, scale, and shear (a_1, a_2, a_4, a_5 in (2)), $\omega_i = 1$ for the translational parameters (a_3, a_6 in (2)), and $\omega_i = 2$ for non-rigid parameters.

The parameter lb is needed in the definition of $\mathcal{N}_i$ for a practical purpose. It is to prevent two desired criteria from contradicting each other. The first criterion expects the gradient at the ground truth position $\mathbf{p}_i$ to vanish, while the other criterion requires the gradients at other locations not to vanish. In practice, to prevent these two types of constraints from contradicting each other, the latter set of constraints should not be enforced at locations that are too close to the ground truth $\mathbf{p}_i$. In other words, the second set of constraints should not be enforced on $\mathcal{N}_i^-$. The parameter lb can be determined empirically by experimenting with the training data. In our experiments, this value is not too sensitive and can be set to any small positive value. This value is 0.2 in all of our experiments.

Other tunable parameters of our method are C and μ. These parameters can be picked using validation data or by cross validation. In our experiments, we found that the performance of our method is not too sensitive to the choice of μ. Usually, μ simply can be set to a small positive number (0.01 in our experiments).

The training phase of our algorithm takes several hours. For example, on a 2.4 GHz Pentium Core 2 Duo machine with 4 GB of RAM, it took almost two hours to learn a cost function for the experiment in Sect. 4.2. This amount of training time is high because producing efficient code was not the main focus of the paper. Our main programing language was MATLAB. For optimization, we used CVX, a package for specifying and solving convex programs (Grant and Boyd 2008a, 2008b). This is a generic convex program solver which does not have special support for constraint addition and for iterative procedures.

Regarding the alignment speed for AAMs (Sect. 4.2), fitting the AAM with the learned cost function tends to be slower than PCA-based cost functions. This is because the learned cost function involves many more principal components ($\mathbf{u}_i$'s). In the experiment in Sect. 4.2, the average alignment time for an image when using our learned cost function is 5.3 seconds. The average alignment time when using PCA 90%, PCA 80%, and PCA 70% are 3.6, 2.5, and 2.0 seconds respectively. Though the learned cost function does not take too much longer than PCA-based cost functions do, one interesting possible direction for future work would be to investigate a compromise between computational complexity and performance. This possibly can be done by adding a L_1 regularization term on $\boldsymbol{\lambda}$ to encourage the sparsity of $\boldsymbol{\lambda}$, the weight vector for the principal components.

5 Good Features to Track

Previous sections have addressed the problem of learning a metric for alignment given a fixed template. In this section we address the reverse question: which templates can be aligned well using a given fixed metric? We will show how the criteria for an optimal metric can be used to select good templates/features to track. In particular, we will use SSD as the cost function and find the templates based on which the criteria of an optimal metric are satisfied. This leads to a novel method for selecting good features to track. Experimental comparison with Shi and Tomasi (1994) shows that our method extracts more reliable features to track.

5.1 Selection Criterion for Feature Points

Feature-based tracking (Tomasi and Kanade 1991) is a key component of many vision systems. Some systems identify a set of feature points once and track them through an entire video sequence. Some redraw a new set of feature points at every frame. Some others replace lost features by new ones. In any situation, the ability to select feature points that can be tracked reliably between consecutive frames is critical to the success of the tracker.

Typically, a feature point can be tracked by finding the displacement between consecutive frames. Because of image noise, the displacement of the feature point is taken to be the displacement of a small window around the feature point. Therefore, feature point tracking from one frame to the next is essentially template alignment. Here, the template is the window around the feature point in the current frame, and the successive frame is the image that needs to be aligned with the template.

Consider a particular feature point at a given frame, let $\mathbf{x}$ be the set of pixels that corresponds to the window around the feature point. Let $\mathbf{d}$ and $\mathbf{d}_s$ denote the current frame and the successive frame respectively. In this case, the reference template is $\mathbf{d}_{ref} = \mathbf{d}(\mathbf{x})$. Suppose the correct displacement is $\hat{\mathbf{p}}$. As in Sect. 3.1, the displacement $\hat{\mathbf{p}}$ of the feature point is estimated by optimizing:

$$\underset{\mathbf{p}}{\text{minimize}} \quad \|\mathbf{d}_s(\mathbf{f}(\mathbf{x},\mathbf{p})) - \mathbf{d}(\mathbf{x})\|_2^2. \tag{25}$$

Here, $\mathbf{f}$ is the function for geometric warping. For feature-based tracking, it is generally good enough to just consider translational or affine motions. In both cases, there exists a geometric transformation function $\mathbf{f}$ that is additive in parameter $\mathbf{p}$ (e.g., see Learned-Miller 2006 for derivation), i.e.,

$$\mathbf{f}(\mathbf{f}(\mathbf{x},\mathbf{p}_1),\mathbf{p}_2) = \mathbf{f}(\mathbf{x},\mathbf{p}_1+\mathbf{p}_2). \tag{26}$$

For the sake of clear presentation and a computational efficiency reason that will become clearer later on, let us assume (26) and consider the inverse additive image alignment (Hager and Belhumeur 1998; Baker and Matthews 2001). The idea is to switch the role of the image and the template. Let $\mathbf{z} = \mathbf{f}(\mathbf{x},\mathbf{p})$, then $\mathbf{x} = \mathbf{f}(\mathbf{f}(\mathbf{x},\mathbf{p}),-\mathbf{p}) = \mathbf{f}(\mathbf{z},-\mathbf{p})$, (25) is equivalent to:

$$\underset{\mathbf{p}}{\text{minimize}} \quad \|\mathbf{d}_s(\mathbf{z}) - \mathbf{d}(\mathbf{f}(\mathbf{z},-\mathbf{p}))\|_2^2. \tag{27}$$

Our cost function now is $E(\mathbf{d},\mathbf{p}) = \|\mathbf{d}_s(\mathbf{z}) - \mathbf{d}(\mathbf{f}(\mathbf{z},-\mathbf{p}))\|_2^2$. Let us now revisit the two desired criteria (8) and (9) for an optimal cost function $E(\mathbf{d},\mathbf{p})$:

$$\left.\frac{\partial E(\mathbf{d},\mathbf{p})}{\partial \mathbf{p}}\right|_{\mathbf{p}=\hat{\mathbf{p}}} = \mathbf{0}, \tag{28}$$

$$\left\langle -\left(\frac{\partial E(\mathbf{d},\mathbf{p})}{\partial \mathbf{p}}\right)^T, \frac{\hat{\mathbf{p}}-\mathbf{p}}{\|\hat{\mathbf{p}}-\mathbf{p}\|_2}\right\rangle > 0 \tag{29}$$

$\forall \mathbf{p} : lb \le \|\mathbf{p}-\hat{\mathbf{p}}\|_2 \le ub.$

Here lb, ub are the lower bound and upper bound of the motion which are defined in Sect. 3.2. As in (14) and (15), using Taylor series approximation, we have:

$$\begin{aligned}
&\left(\frac{\partial E(\mathbf{d},\mathbf{p})}{\partial \mathbf{p}}\right)^T \\
&\quad\approx 2\left(\frac{\partial \mathbf{d}(\mathbf{f}(\mathbf{z},-\mathbf{p}))}{\partial \mathbf{p}}\right)^T (\mathbf{d}(\mathbf{f}(\mathbf{z},-\mathbf{p})) - \mathbf{d}_s(\mathbf{z})) \\
&\quad\approx 2\left(\frac{\partial \mathbf{d}(\mathbf{f}(\mathbf{z},-\mathbf{p}))}{\partial \mathbf{p}}\right)^T (\mathbf{d}(\mathbf{x}) - \mathbf{d}_s(\mathbf{z})),
\end{aligned} \tag{30}$$

where

$$\frac{\partial \mathbf{d}(\mathbf{f}(\mathbf{z},-\mathbf{p}))}{\partial \mathbf{p}} = -\left.\frac{\partial \mathbf{d}(\mathbf{f}(\mathbf{z},-\mathbf{p}+\epsilon))}{\partial \epsilon}\right|_{\epsilon=\mathbf{0}} \tag{31}$$

$$= -\left.\frac{\partial \mathbf{d}(\mathbf{f}(\mathbf{f}(\mathbf{z},-\mathbf{p}),\epsilon))}{\partial \epsilon}\right|_{\epsilon=\mathbf{0}}$$

$$= -\left.\frac{\partial \mathbf{d}(\mathbf{f}(\mathbf{x},\epsilon))}{\partial \epsilon}\right|_{\epsilon=\mathbf{0}}. \tag{32}$$

The quantity $\frac{\partial \mathbf{d}(\mathbf{f}(\mathbf{x},\epsilon))}{\partial \epsilon}|_{\epsilon=\mathbf{0}}$ does not depend on $\mathbf{p}$. Let us denote it by $\mathbf{J}^{\mathbf{d}(\mathbf{x})}$. On the other side, using the brightness conservation assumption, we have $\mathbf{d}_s(\mathbf{f}(\mathbf{x},\hat{\mathbf{p}})) = \mathbf{d}(\mathbf{x})$. Applying the transformation $\mathbf{f}$ with the amount $\mathbf{p}-\hat{\mathbf{p}}$ on both sides of the above equation, we get:

$$\mathbf{d}_s(\mathbf{f}(\mathbf{f}(\mathbf{x},\hat{\mathbf{p}}),\mathbf{p}-\hat{\mathbf{p}})) = \mathbf{d}(\mathbf{f}(\mathbf{x},\mathbf{p}-\hat{\mathbf{p}})) \tag{33}$$

$$\Rightarrow \mathbf{d}_s(\mathbf{f}(\mathbf{x},\mathbf{p})) = \mathbf{d}(\mathbf{f}(\mathbf{x},\mathbf{p}-\hat{\mathbf{p}})) \quad (34)$$

$$\Rightarrow \mathbf{d}_s(\mathbf{z}) = \mathbf{d}(\mathbf{f}(\mathbf{x},\mathbf{p}-\hat{\mathbf{p}})). \quad (35)$$

From (30), (32) & (35) we get:

$$\left(\frac{\partial E(\mathbf{d},\mathbf{p})}{\partial \mathbf{p}}\right)^T \approx 2\left(\mathbf{J}^{\mathbf{d}(\mathbf{x})}\right)^T \left(\mathbf{d}(\mathbf{f}(\mathbf{x},\mathbf{p}-\hat{\mathbf{p}})) - \mathbf{d}(\mathbf{x})\right). \quad (36)$$

Equation (36) leads to $\frac{\partial E(\mathbf{d},\mathbf{p})}{\partial \mathbf{p}}|_{\mathbf{p}=\hat{\mathbf{p}}} \approx \mathbf{0}$; therefore, the constraint (28) is always satisfied. Let $\mathbf{q} = \mathbf{p} - \hat{\mathbf{p}}$, the constraint (29) becomes:

$$\left\langle \left(\mathbf{J}^{\mathbf{d}(\mathbf{x})}\right)^T (\mathbf{d}(\mathbf{f}(\mathbf{x},\mathbf{q})) - \mathbf{d}(\mathbf{x})), \frac{\mathbf{q}}{\|\mathbf{q}\|_2} \right\rangle > 0 \quad (37)$$

$\forall \mathbf{q} : lb \le \|\mathbf{q}\|_2 \le ub.$

In short, (37) is the criterion for the template defined by $\mathbf{x}$ to combine well with the SSD cost function. As a result, (37) establishes a criterion for good features to track.

In many situations, it is desirable to extract a certain number of feature points. The number of required feature points might be more or less than the number of feature points that satisfy (37). Thus, it is necessary to provide a ranking for feature points. Let us consider a ranking score which is based on how much the constraint (37) is satisfied or violated, i.e.,

$$\min_{lb \le \|\mathbf{q}\|_2 \le ub} \left\langle \left(\mathbf{J}^{\mathbf{d}(\mathbf{x})}\right)^T (\mathbf{d}(\mathbf{f}(\mathbf{x},\mathbf{q})) - \mathbf{d}(\mathbf{x})), \frac{\mathbf{q}}{\|\mathbf{q}\|_2} \right\rangle. \quad (38)$$

In the next section we will describe experimental results of using (38) as the selection criterion for good features to track. A nice property of (38) is that $\mathbf{J}^{\mathbf{d}(\mathbf{x})}$ does not depend on $\mathbf{q}$; this makes the method computationally efficient. If efficiency is not a concern, there is no need to consider the inverse additive image alignment. Using forward additive image alignment instead, one can derive a similar criterion to (38). In this case, the assumption (26) is not necessary.

5.2 Experiments

This section describes an experiment to compare several selection methods for feature tracking. We compare tracking results of feature points selecting the criterion using (38) and using the standard criterion proposed by Shi and Tomasi (1994).

5.2.1 *Software, Data, and Tuning*

We use a public implementation[2] of Lucas-Kanade-Tomasi tracker. This software also includes an implementation of Shi & Tomasi's method for feature point selection. The software comes with a sequence of ten video frames (Fig. 7) and demo code that tracks feature points detected using Shi & Tomasi's criteria. After tracking, it outputs the number of feature points successfully tracked as an evaluation criterion.

[2]http://www.ces.clemson.edu/~stb/klt/.

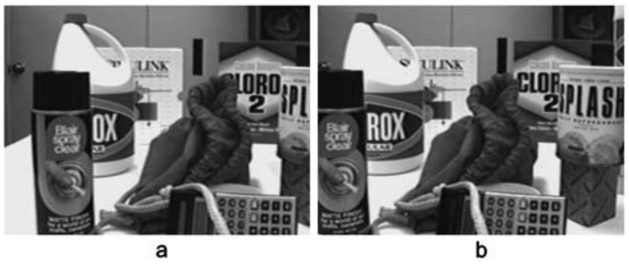

Fig. 7 Two frames from the data sequence. (**a**) The first frame; (**b**) the 10th frame. This figure shows the amount of motion between frames

In our experiments, we use the sequence of video frames that come with the software. We write our own code for selecting feature points but use the provided Lucas-Kanade-Tomasi tracker for evaluation. We accept all the default parameters of the software. These include the size of the square templates for feature points, the number of feature points retained, the minimum distance between feature points, how the images are smoothed, and how the image gradients are computed. Our method requires little tuning which involves selecting the bounds (lb, ub) for the perturb parameter $\mathbf{q}$ (see (38)). We set $lb = 1$ and ub to the size of the template.

Two frames from the image sequence bundled with the software are given in Fig. 7. The resolution is 240×320. The size of the square templates is 7×7 pixels.

5.2.2 *Experiments and Results*

The aforementioned software tracks feature points by recovering translational displacements. Because of this reason, and for consistency with the assumption of Shi and Tomasi (1994), the geometric warp $\mathbf{f}$ is taken as the translation function.

Figure 8 displays two sets of selected features points. The cyan circles are 100 points selected using Shi & Tomasi's method. The red stars are 100 points selected using our method. The procedure of both methods for selecting feature points is as follows. First, the goodness of each pixel is measured using an appropriate criterion ((38) in the case of our method). After all the pixels have been considered, they are sorted in descending order according to goodness. Then, one by one, the top pixel of the list is selected, ensuring that each new feature point is at least 10 pixels away from all the other features.

To quantitatively compare two methods, we performed two types of experiments. In the first experiment, for each frame in the image sequence, we detect 100 feature points

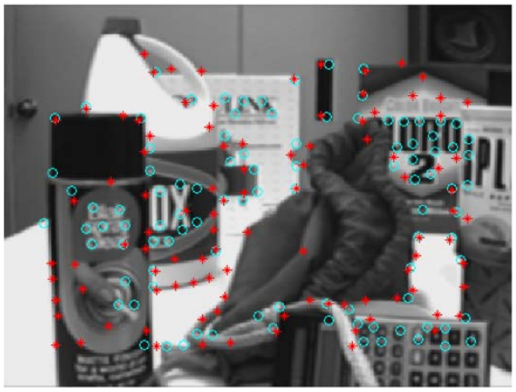

Fig. 8 (Color online) Selected feature points based on two different criteria. Cyan circles: 100 points selected using Shi & Tomasi criterion. *Red stars*: 100 points selected using our criterion. Selected feature points of each method are at least 10 pixels apart

Table 3 Results of alignment between all frames with the first frame, starting with a set of 100 feature points, either detected using Shi & Tomasi's method or our method. This table reports the numbers of points that are successfully aligned. As the frame number increases, the amount of motion between the frame and the first frame increases; as a result, the alignment problem gets harder. The set of feature points extracted by our method is more reliable than that of Shi & Tomasi's method for all levels of difficulty

frame #	2	3	4	5	6	7	8	9	10
Shi&Tomasi	92	85	81	73	62	56	44	32	25
Ours	95	91	83	77	65	60	54	42	38

and count the number of points that are successfully tracked in the successive frame. The means and standard deviations of this statistic for Shi & Tomasi's method and ours are: 93.22 ± 1.56 and 96.44 ± 2.07 respectively.

In the second experiment, we compare two sets of selected feature points with increasing amounts of motion. We align every frame in the image sequence with the first frame (not between consecutive frames). The sets of 100 feature points are detected from the first frame. Alignment between the first frame and all other frames are performed, and the numbers of successfully tracked points are recorded. The results are reported in Table 3. As can be observed, the set of feature points extracted by our method is more reliable than that of Shi & Tomasi's method for all amounts of motion.

6 Conclusion

Gradient-based methods for image alignment such as Lucas-Kanade, Eigentracking, and AAMs are a key component of many computer vision systems. A major problem of current gradient-based image alignment algorithms is the sensitivity to local minima. Local minima in PAMs mainly occur because an appearance model (e.g. template or PCA subspace) is constructed without considering the neighborhoods of the correct motion parameters, the parameters corresponding to ground truth annotation of training data. These neighborhoods determine the local minima properties of the error surface and should be taken into account while constructing the model. In this paper, we have proposed a data driven approach to learn a metric for image alignment that reduces the effect of local minima. Metric learning was posed as a convex program optimizing the parameters of a quadratic cost function. This cost function is very general; it subsumes the cost functions of many image alignment algorithms including template matching and AAMs. Given training samples, the metric was learned by requiring: (i) there is a local minimum in the expected location, and (ii) there are no local minima in a specified neighborhood. However, it was typically not possible to satisfy both criteria; therefore, our method learned a Pareto optimal tradeoff between having fewer local minima and having local minima at the desired places. The advantages of the proposed method to template alignment and AAM fitting have been demonstrated with several synthetic and real experiments. To the best of our knowledge, this is the first paper that explicitly learns a metric with no local minima for PAMs. In addition, we showed how the proposed criteria to learn a metric can be used to select good features to track in feature-based tracking.

The proposed criteria for an ideal error surface can be used to learn a cost function that is more general than the quadratic one used in this paper. The quadratic cost function was chosen because it has two major benefits: (i) the cost functions used in many important alignment algorithms can be cast in this form, and (ii) the metric learning formulation is convex. Convexity of the learning formulation is achieved thanks to a novel optimization scheme. In this paper, we stated and proved the validity of this scheme as well as its advantages over alternate optimization algorithms.

Although encouraging results have been achieved, there are several issues that remain unsolved and can be considered in future work. A bottleneck of our algorithm is the need to incorporate many constraints (theoretically infinite) to satisfy the criterion that there are no local minima in a specified neighborhood. Because it is computationally expensive to include many constraints in optimization, there is a need for further research that addresses the question: what are the most critical points for generating constraints and how can they be sampled?

In this paper we have shown performance improvement in the cases of template alignment and AAMs. In general, however, it is unclear for which type of alignment problems a learned metric, using training data, would guarantee improvement in registering unseen images. Unfortunately, there is no mathematically grounded theory to answer this question. In future work we plan to research and to develop a general theory for learning image alignment.

Acknowledgements 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. The authors would like to thank General Motors Corporation for their continued support of this research.

Appendix A

This section states and proves a theorem used to justify the optimization algorithm given in (16).

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:*

$$\mathbf{x}^{new} = \mathbf{x}^{old} + \delta\mathbf{x}$$

$$\text{with} \quad \delta\mathbf{x} = -\mathbf{H}^{-1}\mathbf{J}^T(\mathbf{A} f(\mathbf{x}) + \mathbf{b}) \tag{39}$$

$$\text{and} \quad \mathbf{J} = \left.\frac{\partial f}{\partial \mathbf{x}}\right|_{\mathbf{x}^{old}}, \qquad \mathbf{H} = \mathbf{J}^T\mathbf{J}.$$

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:

$$0 \le \frac{\mathbf{u}^T\mathbf{A}\mathbf{u}}{\mathbf{u}^T\mathbf{u}} \le 1 \Leftrightarrow 0 \le \frac{\mathbf{u}^T(\mathbf{I}_m - \mathbf{A})\mathbf{u}}{\mathbf{u}^T\mathbf{u}} \le 1 \quad \forall \mathbf{u}. \tag{40}$$

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) $\alpha_i \ge 0$ $\forall i = \overline{1,k}$, *and* $\sum_{i=1}^k \alpha_i \le 1$,
(iii) $\mathbf{A} = \sum_{i=1}^k \alpha_i \mathbf{B}_i(\mathbf{B}_i^T\mathbf{B}_i)^{-1}\mathbf{B}_i^T$.

Note: the number of rows of $\mathbf{B}_i$'s must always be m but the number of their columns can differ.

Proof for sufficiency conditions Suppose there exist k, α_i's, and $\mathbf{B}_i$'s that satisfy 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 $\mathbf{v} \in \Re^m$:

$$\begin{aligned}\mathbf{v}^T\mathbf{A}\mathbf{v} &= \sum_{i=1}^k \alpha_i \mathbf{v}^T\mathbf{B}_i(\mathbf{B}_i^T\mathbf{B}_i)^{-1}\mathbf{B}_i^T\mathbf{v} \\ &= \sum_{i=1}^k \alpha_i \mathbf{v}^T\mathbf{B}_i(\mathbf{B}_i^T\mathbf{B}_i)^{-1}\mathbf{B}_i^T\mathbf{B}_i(\mathbf{B}_i^T\mathbf{B}_i)^{-1}\mathbf{B}_i^T\mathbf{v} \\ &= \sum_{i=1}^k \alpha_i \|\mathbf{B}_i(\mathbf{B}_i^T\mathbf{B}_i)^{-1}\mathbf{B}_i^T\mathbf{v}\|_2^2.\end{aligned} \tag{41}$$

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 $\|\mathbf{B}_i(\mathbf{B}_i^T\mathbf{B}_i)^{-1}\mathbf{B}_i^T\mathbf{v}\|_2^2 \le \|\mathbf{v}\|_2^2$ $\forall i$. Therefore:

$$\mathbf{v}^T\mathbf{A}\mathbf{v} \le \left(\sum_{i=1}^k \alpha_i\right)\|\mathbf{v}\|_2^2 \le \|\mathbf{v}\|_2^2. \tag{42}$$

Furthermore, because $\|\mathbf{B}_i(\mathbf{B}_i^T\mathbf{B}_i)^{-1}\mathbf{B}_i^T\mathbf{v}\|_2^2 \ge 0$ and $\alpha_i \ge 0$ $\forall i$, we have $\mathbf{v}^T\mathbf{A}\mathbf{v} \ge 0$. Combining this with the inequality in (42), we have:

$$0 \le \mathbf{v}^T\mathbf{A}\mathbf{v} \le \mathbf{v}^T\mathbf{v}. \tag{43}$$

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 $\mathbf{A}$, $\mathbf{A} = \mathbf{U}\boldsymbol{\Lambda}\mathbf{U}^T$. Here, the columns of $\mathbf{U}$ are orthonormal vectors. $\boldsymbol{\Lambda}$ is a diagonal matrix, $\boldsymbol{\Lambda} = \mathrm{diag}([\lambda_1, \ldots, \lambda_m])$ with $0 \le \lambda_i \le 1$ $\forall i$. Without loss of generality, suppose $\lambda_1 \ge \lambda_2 \ge \cdots \ge \lambda_m$. We have:

$$\begin{aligned}\mathbf{A} = \mathbf{U}\boldsymbol{\Lambda}\mathbf{U}^T &= \sum_{i=1}^m \lambda_i \mathbf{u}_i\mathbf{u}_i^T \\ &= \sum_{i=1}^{m-1}(\lambda_i - \lambda_{i+1})\left(\sum_{j=1}^i \mathbf{u}_j\mathbf{u}_j^T\right) + \lambda_m\left(\sum_{j=1}^m \mathbf{u}_j\mathbf{u}_j^T\right).\end{aligned} \tag{44}$$

Let $\alpha_i = \lambda_i - \lambda_{i+1}$ for $i = 1, \ldots, m-1$, and $\alpha_m = \lambda_m$. Let $\mathbf{B}_i = [\mathbf{u}_1 \ldots \mathbf{u}_i]$ 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:

$$\mathbf{A} = \sum_{i=1}^m \alpha_i\mathbf{B}_i(\mathbf{B}_i^T\mathbf{B}_i)^{-1}\mathbf{B}_i^T. \tag{45}$$

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

Proof of Theorem 1 From Lemmas 1 and 2 we know that $\exists \alpha_i \ge 0, \exists \mathbf{B}_i$: $\mathbf{I}_m - \mathbf{A} = \sum_{i=1}^k \alpha_i\mathbf{B}_i(\mathbf{B}_i^T\mathbf{B}_i)^{-1}\mathbf{B}_i^T$ and

$\sum_1^k \alpha_i \leq 1$. To prove Theorem 1, let us first consider the optimization of the following function:

$$E_2(\mathbf{x}, \{\mathbf{c}_i\}) = \sum_{i=1}^k \alpha_i \| f(\mathbf{x}) - \mathbf{B}_i \mathbf{c}_i \|_2^2 + \alpha_0 \| f(\mathbf{x}) \|_2^2 + 2\mathbf{b}^T f(\mathbf{x}) \tag{46}$$

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 Newton's method:

$$\mathbf{x}^{new} = \mathbf{x}^{old} - \left(\frac{\partial^2 E_2}{\partial \mathbf{x}^2}\right)^{-1} \left(\frac{\partial E_2}{\partial \mathbf{x}}\right)^T. \tag{47}$$

Using the first order Taylor approximation, we have

$$f(\mathbf{x} + \delta\mathbf{x}) \approx f(\mathbf{x}) + \mathbf{J}\delta\mathbf{x} \quad \text{with } \mathbf{J} = \frac{\partial f}{\partial \mathbf{x}}. \tag{48}$$

Thus

$$\begin{aligned} E_2(\mathbf{x} + \delta\mathbf{x}, \{\mathbf{c}_i\}) &\approx E_2(\mathbf{x}, \{\mathbf{c}_i\}) + \delta\mathbf{x}^T \mathbf{J}^T \mathbf{J} \delta\mathbf{x} \\ &\quad + 2\delta\mathbf{x}^T \mathbf{J}^T \left(f(\mathbf{x}) - \sum_{i=1}^k \alpha_i \mathbf{B}_i \mathbf{c}_i + \mathbf{b} \right). \end{aligned} \tag{49}$$

Hence

$$\frac{\partial E_2}{\partial \mathbf{x}} \approx 2\left(f(\mathbf{x}) - \sum_{i=1}^k \alpha_i \mathbf{B}_i \mathbf{c}_i + \mathbf{b} \right)^T \mathbf{J} \tag{50}$$

$$\frac{\partial^2 E_2}{\partial \mathbf{x}^2} \approx 2\mathbf{J}^T \mathbf{J}. \tag{51}$$

Therefore, we have the Newton update rule:

$$\mathbf{x}^{new} = \mathbf{x}^{old} - (\mathbf{J}^T \mathbf{J})^{-1} \mathbf{J}^T \left(f(\mathbf{x}) - \sum_{i=1}^k \alpha_i \mathbf{B}_i \mathbf{c}_i + \mathbf{b} \right). \tag{52}$$

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

$$\mathbf{c}_i^*(\mathbf{x}) = (\mathbf{B}_i^T \mathbf{B}_i)^{-1} \mathbf{B}_i^T f(\mathbf{x}). \tag{53}$$

Combining (52) and (53), we have the update rule for minimizing E_2:

$$\mathbf{x}^{new} = \mathbf{x}^{old} - (\mathbf{J}^T \mathbf{J})^{-1} \mathbf{J}^T [\mathbf{A} f(\mathbf{x}) + \mathbf{b}]. \tag{54}$$

This update rule is exactly the same as the update rule given in (39). As a result, (39) will always lead us to a local minimum of E_2.

We now prove that a local minimum of E_2 obtained by (39) 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:

$$E_2(\mathbf{x}_0, \{\mathbf{c}_i^*(\mathbf{x}_0)\}) \leq E_2(\mathbf{x}_0 + \delta\mathbf{x}, \{\mathbf{c}_i^*(\mathbf{x}_0) + \delta\mathbf{c}_i\}) \tag{55}$$

for all $\delta\mathbf{x}, \delta\mathbf{c}_i : \|\delta\mathbf{x}\|_2^2 + \sum_i \|\delta\mathbf{c}_i\|_2^2 < \epsilon_1$.

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

$$E_2(\mathbf{x}_0, \{\mathbf{c}_i^*(\mathbf{x}_0)\}) \leq E_2(\mathbf{x}_0 + \delta\mathbf{x}, \{\mathbf{c}_i^*(\mathbf{x}_0 + \delta\mathbf{x})\}) \tag{56}$$

for all $\delta\mathbf{x} : \|\delta\mathbf{x}\|_2^2 < \epsilon_2$.

On the other hand, one can easily verify that:

$$E_2(\mathbf{x}, \{\mathbf{c}_i^*(\mathbf{x})\}) = E(\mathbf{x}) \quad \forall \mathbf{x}. \tag{57}$$

From (56) and (57), we have $\exists \epsilon_2 > 0$ such that

$$E(\mathbf{x}_0) \leq E(\mathbf{x}_0 + \delta\mathbf{x}) \quad \forall \delta\mathbf{x} : \|\delta\mathbf{x}\|_2^2 < \epsilon_2. \tag{58}$$

Hence, $\mathbf{x}_0$ must be a local minimum of E.

To summarize, we have shown that (39) will converge to a local minimum of E_2. Furthermore, a local minimum of E_2 found by (39) is also a local minimum of E. Thus the update rule given in (39) is guaranteed to converge to a local minimum of E. This concludes our proof for Theorem 1. □

Appendix B

A metric is a function measuring distance between elements of a set. But, what exactly is the distance that image alignment cost functions measure? This section explains the metric properties of the learned cost functions. Strictly speaking, these functions define a pseudometric rather than a metric. The difference is very subtle. This section discusses this difference and formally shows how a pseudometric can be derived from the learned cost functions. First, we state the definitions of metric and pseudometric.

Definition of metric (Rudin 1976) A metric on a set $\mathcal{X}$ is a function (also called distance function or distance) $\mathcal{D} : \mathcal{X} \times \mathcal{X} \to \Re$ that satisfies:

(i) $\mathcal{D}(\mathbf{x}, \mathbf{y}) \geq 0$ (non-negativity)
(ii) $\mathcal{D}(\mathbf{x}, \mathbf{y}) = 0$ if and only if $\mathbf{x} = \mathbf{y}$
(iii) $\mathcal{D}(\mathbf{x}, \mathbf{y}) = \mathcal{D}(\mathbf{y}, \mathbf{x})$ (symmetry)
(iv) $\mathcal{D}(\mathbf{x}, \mathbf{y}) \leq \mathcal{D}(\mathbf{x}, \mathbf{z}) + \mathcal{D}(\mathbf{z}, \mathbf{y})$ (subadditivity)

Definition of pseudometric (Rudin 1976) A pseudometric on a set $\mathcal{X}$ is a function $\mathcal{D} : \mathcal{X} \times \mathcal{X} \to \Re$ that satisfies:

(i) $\mathcal{D}(\mathbf{x}, \mathbf{y}) \geq 0$ (non-negativity)

(ii) $\mathcal{D}(\mathbf{x},\mathbf{x}) = 0$
(iii) $\mathcal{D}(\mathbf{x},\mathbf{y}) = \mathcal{D}(\mathbf{y},\mathbf{x})$ (symmetry)
(iv) $\mathcal{D}(\mathbf{x},\mathbf{y}) \le \mathcal{D}(\mathbf{x},\mathbf{z}) + \mathcal{D}(\mathbf{z},\mathbf{y})$ (subadditivity)

A pseudometric only differs from a metric in the second requirement: a pseudometric does not require $\mathcal{D}(\mathbf{x},\mathbf{y})$ to be strictly positive when $\mathbf{x} \neq \mathbf{y}$.

Consider the weighted SSD cost function in Sect. 4.1:

$$(\mathbf{d}(\mathbf{f}(\mathbf{x},\mathbf{p})) - \mathbf{d}_{ref})^T \mathrm{diag}(\mathbf{w})(\mathbf{d}(\mathbf{f}(\mathbf{x},\mathbf{p})) - \mathbf{d}_{ref}), \tag{59}$$

with

$$0 \le w_i \le 1 \quad \forall i. \tag{60}$$

The above cost function clearly induces a pseudometric on $\Re^l$ where l is the dimension of $\mathbf{d}(\mathbf{f}(\mathbf{x},\mathbf{p}))$ and $\mathbf{d}_{ref}$. Indeed, it can be easily verified that the function $\mathcal{D} : \Re^l \times \Re^l \to \Re$ defined below is a pseudometric:

$$\mathcal{D}(\mathbf{x},\mathbf{y}) = \sqrt{(\mathbf{x}-\mathbf{y})^T \mathrm{diag}(\mathbf{w})(\mathbf{x}-\mathbf{y})}. \tag{61}$$

The weighted basis AAM cost function in Sect. 4.2 also induces a pseudometric. To see this, consider the function $\mathcal{D} : (\Re^l \cup \{\mathbf{U}\}) \times (\Re^l \cup \{\mathbf{U}\}) \to \Re$ defined as follows:

$$\mathcal{D}(\mathbf{x},\mathbf{y}) = \begin{cases} 0 & \text{if } \mathbf{x} = \mathbf{U}, \mathbf{y} = \mathbf{U}, \\ \mathbf{y}^T\mathbf{A}\mathbf{y} & \text{if } \mathbf{x} = \mathbf{U}, \mathbf{y} \neq \mathbf{U}, \\ \mathbf{x}^T\mathbf{A}\mathbf{x} & \text{if } \mathbf{x} \neq \mathbf{U}, \mathbf{y} = \mathbf{U}, \\ |\mathbf{x}^T\mathbf{A}\mathbf{x} - \mathbf{y}^T\mathbf{A}\mathbf{y}| & \text{if } \mathbf{x} \neq \mathbf{U}, \mathbf{y} \neq \mathbf{U}. \end{cases} \tag{62}$$

Here, $\mathbf{U} = [\mathbf{u}_1 \cdots \mathbf{u}_K]$ are the eigenvectors of the training data subspace and $\mathbf{A} = \mathbf{I}_m - \sum_{i=1}^K \lambda_i \mathbf{u}_i \mathbf{u}_i^T$ with $0 \le \lambda_i \le 1$.

From the above definition, together with the positive semi-definiteness of $\mathbf{A}$, one can easily verify that:

$$\mathcal{D}(\mathbf{x},\mathbf{y}) = |\mathcal{D}(\mathbf{x},\mathbf{U}) - \mathcal{D}(\mathbf{y},\mathbf{U})| \quad \forall \mathbf{x},\mathbf{y}. \tag{63}$$

Based on this observation, it is obvious that $\mathcal{D}$ is nonnegative, symmetric, and subadditive. Furthermore, $\mathcal{D}(\mathbf{x},\mathbf{x}) = |\mathcal{D}(\mathbf{x},\mathbf{U}) - \mathcal{D}(\mathbf{x},\mathbf{U})| = 0$. Thus $\mathcal{D}$ satisfies all the requirements of a pseudometric. This pseudometric defines a distance measurement between elements of $\Re^l \cup \{\mathbf{U}\}$. In practice, the only type of distance measurement that matters for AAM image alignment is the distance between an element of $\Re^l$ and the training data subspace $\mathbf{U}$. This distance is exactly the value of the cost function learned in Sect. 4.2.

References

Baker, S., & Matthews, I. (2001). Equivalence and efficiency of image alignment algorithms. In *Proceedings of IEEE conference on computer vision and pattern recognition*.

Baker, S., & Matthews, I. (2004). Lucas-Kanade 20 years on: A unifying framework. *International Journal of Computer Vision*, *56*(3), 221–255.

Bergen, J. R., Anandan, P., Hanna, K. J., & Hingorani, R. (1992). Hierarchical model-based motion estimation. In *European conference on computer vision* (pp. 237–252).

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

Blanz, V., & Vetter, T. (1999). A morphable model for the synthesis of 3D faces. In *ACM SIGGRAPH*.

Cootes, T., Edwards, G., & Taylor, C. (2001). Active appearance models. *Pattern Analysis and Machine Intelligence*, *23*(6).

Cootes, T. F., & Taylor, C. (2001). *Statistical models of appearance for computer vision* (Tech. rep.). University of Manchester.

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

de la Torre, F., & Nguyen, M. H. (2008). Parameterized kernel principal component analysis: Theory and applications to supervised and unsupervised image alignment. In *Proceedings of IEEE conference on computer vision and pattern recognition*.

de la Torre, F., Vitrià, J., Radeva, P., & Melenchón, J. (2000). Eigenfiltering for flexible eigentracking. In *International conference on pattern recognition* (pp. 1118–1121).

de la Torre, F., Collet, A., Cohn, J., & Kanade, T. (2007). Filtered component analysis to increase robustness to local minima in appearance models. In *IEEE conference on computer vision and pattern recognition*.

Gong, S., Mckenna, S., & Psarrou, A. (2000). *Dynamic vision: from images to face recognition*. Imperial College Press.

Grant, M., & Boyd, S. (2008a). CVX: Matlab software for disciplined convex programming (web page & software). http://stanford.edu/~boyd/cvx.

Grant, M., & Boyd, S. (2008b). Graph implementations for nonsmooth convex programs. In V. Blondel, S. Boyd, & H. Kimura (Eds.), *Lecture notes in control and information sciences*: *Recent advances in learning and control (a tribute to M. Vidyasagar)* (pp. 95–110). Berlin: Springer.

Gross, R., Matthews, I., Cohn, J., Kanade, T., & Baker, S. (2007). *The CMU multi-pose, illumination, and expression (Multi-PIE) face database* (Tech. rep. tR-07-08). Carnegie Mellon University.

Hager, G., & Belhumeur, P. (1998). Efficient region tracking with parametric models of geometry and illumination. *Pattern Analysis and Machine Intelligence*, *20*, 1025–1039.

Jolliffe, I. (1986). *Principal component analysis*. New York: Springer.

Jones, M. J., & Poggio, T. (1998). Multidimensional morphable models. In *International conference on computer vision* (pp. 683–688).

Kanatani, K. (1996). *Statistical optimization for geometric computations: theory and practice*. New York: Elsevier Science.

Learned-Miller, E. G. (2006). Data driven image models through continuous joint alignment. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *28*(2), 236–250.

Liu, X. (2007). Generic face alignment using boosted appearance model. In *IEEE conference on computer vision and pattern recognition*.

Lucas, B., & Kanade, T. (1981). An iterative image registration technique with an application to stereo vision. In *Proceedings of imaging understanding workshop*.

Matei, B. C., & Meer, P. (2006). Estimation of nonlinear errors-in-variables models for computer vision applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *28*(10), 1537–1552.

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

Matthews, I., Ishikawa, T., & Baker, S. (2004). The template update problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *26*, 810–815.

Nayar, S. K., & Poggio, T. (1996). *Early visual learning*. Oxford: Oxford University Press.

Nguyen, M. H., & de la Torre, F. (2008a). Learning image alignment without local minima for face detection and tracking. In *8th IEEE international conference on automatic face and gesture recognition*.

Nguyen, M. H., & de la Torre, F. (2008b). Local minima free parameterized appearance models. In *Proceedings of IEEE conference on computer vision and pattern recognition*.

Rudin, W. (1976). *Principles of mathematical analysis* (3rd ed.). New York: McGraw-Hill.

Saragih, J., & Goecke, R. (2007). A nonlinear discriminative approach to AAM fitting. In *International conference on computer vision*.

Shi, J., & Tomasi, C. (1994). Good features to track. In *IEEE conference on computer vision and pattern recognition*.

Taskar, B., Guestrin, C., & Koller, D. (2003). Max-margin Markov networks. In *Advances in neural information processing systems*.

Tomasi, C., & Kanade, T. (1991). *Detection and tracking of point features* (Tech. Rep. CMU-CS-91-132). Carnegie Mellon University.

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

Vapnik, V. (1998). *Statistical learning theory*. New York: Wiley.

Vetter, T. (1997). Learning novel views to a single face image. In *International conference on automatic face and gesture recognition*.

Wimmer, M., Stulp, F., Tschechne, S. J., & Radig, B. (2006). Learning robust objective functions for model fitting in image understanding applications. In *Proceedings of British machine vision conference*.

Wu, H., Liu, X., & Doretto, G. (2008). Face alignment via boosted ranking model. In *Proceedings of IEEE conference on computer vision and pattern recognition*.

Xiao, J., Baker, S., Matthews, I., & Kanade, T. (2004). Real-time combined 2D+3D active appearance models. In *Conference on computer vision and pattern recognition* (Vol. II, pp. 535–542).

Yang, L. (2006). Distance metric learning: A comprehensive survey. http://www.cse.msu.edu/~yangliu1/frame_survey_v2.pdf.