Face Image Retrieval by Shape Manipulation

Brandon M. Smith Shengqi Zhu Li Zhang
University of Wisconsin – Madison

http://www.cs.wisc.edu/~lizhang/projects/finding-faces/

Abstract

Current face image retrieval methods achieve impressive results, but lack efficient ways to refine the search, particularly for geometric face attributes. Users cannot easily find faces with slightly more furrowed brows or specific leftward pose shifts, for example. To address this problem, we propose a new face search technique based on shape manipulation that is complementary to current search engines. Users drag one or a small number of contour points, like the bottom of the chin or the corner of an eyebrow, to search for faces similar in shape to the current face, but with updated geometric attributes specific to their edits. For example, the user can drag a mouth corner to find faces with wider smiles, or the tip of the nose to find faces with a specific pose. As part of our system, we propose (1) a novel confidence score for face alignment results that automatically constructs a contour-aligned face database with reasonable alignment accuracy, (2) a simple and straightforward extension of PCA with missing data to tensor analysis, and (3) a new regularized tensor model to compute shape feature vectors for each aligned face, all built upon previous work. To the best of our knowledge, our system demonstrates the first face retrieval approach based chiefly on shape manipulation. We show compelling results on a sizable database of over 10,000 face images captured in uncontrolled environments.

1. Introduction

Retrieving one or several desired face images from a large collection has been recently studied in several contexts [1, 6, 7, 21]. These works can be roughly grouped into two categories: *example based* (given a query face image, find similar face images) and *attribute based* (given some natural language description, *e.g.*, black hair, find faces with the desired attributes). While these methods achieve impressive results, they lack efficient ways to refine the search, particularly for geometric face attributes. For example, among the search results, there is no efficient way to find a face with a specific type of grin, or a slightly leftward gaze.

In this paper, we propose a new face search technique based on shape manipulation that is complementary to current search engines. For example, by clicking on the tip of the nose and dragging it to the left, our goal is to find faces similar in shape to the current face, but with leftward pose, as shown in Figure 1; by dragging the corner of the mouth, we hope to find smiling faces, *etc*. Our approach is particularly well suited for geometric face attributes that (1) cannot be easily expressed in natural language or otherwise supported by current face search methods, but (2) can be intuitively specified via a mouse or touchpad interface.

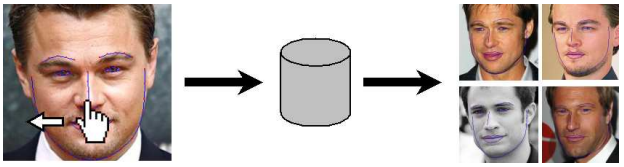

Figure 1. Illustration of our face search technique based on interactive shape manipulation. In this example, the user drags the tip of the nose leftward to search for similar faces with leftward pose in a sizable database of aligned faces.

To achieve this goal, we must address the following three challenges:

- *Face alignment.* Although well studied, accurately identifying facial shape contour features (*e.g.*, eyelid contours, mouth contours) in a large database is still a challenging problem, especially for face images captured in uncontrolled environments.
- *User input interpretation.* The user should be able to find his/her desired faces with very few shape edits. However, ambiguities exist. For example, when dragging the corner of the mouth to the right, the user may want to change the pose, but this could also be interpreted as a desire to widen the mouth.
- *Search metric.* User edits must be transformed into geometric shape features from which desired faces can be retrieved from the database.

To the best of our knowledge, our system demonstrates the *first* face retrieval approach based chiefly on shape manipulation. As part of our system, we propose three techniques that build upon previous work:

- A novel confidence score for face alignment results. This score allows us to automatically reject poor face alignment results in order to construct a sizable database of reasonably well-aligned faces.
- A simple algorithm of tensor decomposition in the presence of missing data. This algorithm is a straightforward generalization of PCA with missing data.
- A new regularized tensor model of aligned face shape. We use this model to (1) associate a tensor coordinate of shape features to each face, (2) resolve shape manipulation ambiguities so that a user can specify his/her intended face shape with few edits, and (3) find the desired faces in the database.

We next review related work before presenting our system.

2. Related Work

Our work is directly inspired by Goldman *et al*. [2], in which the user can drag points on a face in one frame of a video to retrieve desired faces from other frames. Their system tracks a single person's face in a single video for retrieval purposes, and does not differentiate facial motion induced by pose or expression. Later in [6], Kemelmacher-Shlizerman *et al*. demonstrate a system which, given a photo of person A, finds a photo of person B with similar expression for a puppetry application. In this system, the query is the appearance descriptor of the user's own face. In our system, a user often only needs to provide a few (1-3) edits to find desired faces. Each database they use is on the order of 1000 faces of a single person. Our test database is 10 times larger, and contains many different people. We have only used shape features for query; including appearance features is complementary and remains part of our future work.

In computer graphics, creating a desired 3D face model is a central challenge. Recent solutions [10, 22, 14] generate a face from a small number of user edits. Their goal is different from ours in that they seek to generate a new 3D model from example models, but we hope to find one or several desired existing images. No new images are generated in our system. Technically, they work with 3D models, which eliminates pose as a shape parameter. We work with 2D images, where pose is one parameter used to model the underlying object shape.

Face alignment is one important component of our system; however, face alignment is not our contribution. We implemented Gu and Kanade's face alignment method [4] as part of our system. Rather, we propose a novel method of measuring alignment confidence, which allows us to *automatically* construct a large database of reasonably well-aligned faces. This is important because large databases cannot be easily verified by manual inspection. Human-based computation, *e.g*., via Amazon Mechanical Turk, may be suitable for such a task, but is beyond the scope of this paper. The recent face alignment paper by Liu *et al*. [11] also points out the importance of measuring confidence. However, the confidence they use is specific to their objective function. Although their method achieves impressive results, they demonstrate performance on datasets of about 300 examples using congealing. We need to align tens of thousands of images, which would likely be a very slow process if we choose to use a congealing based approach.

Tensor analysis [9] has been successfully used in vision and graphics to model textures [18, 20], for face image recognition [17], and for 3D face transfer [19]. In this paper, we use tensor analysis to model 2D face contour shapes.

Our tensor analysis builds upon [17], with the addition of a regularization term to deal with situations in which the number of tensor coefficients is greater than the number of user edits. Our regularized tensor analysis differs from the previous probabilistic tensor analysis of [15] in that our method takes a single data tensor as input while theirs assumes multiple data tensors as input.

2.1. A Brief Review of Tensor Algebra

We provide a brief summary of tensor algebra to define the notations used in subsequent sections of this paper. Tensor analysis generalizes the widely used principal component analysis (PCA). Given a data matrix $\mathbf{X}$, PCA decomposes $\mathbf{X}$ as

$$\mathbf{X} = \mathbf{U}\mathbf{Z}\mathbf{V}^\top = \sum_{m,n} z_{m,n} u_{:,m} v_{:,n}^\top, \tag{1}$$

where $z_{m,n}$ is an element in $\mathbf{Z}$; $u_{:,m}$ and $v_{:,n}$ are columns of $\mathbf{U}$ and $\mathbf{V}$, respectively, following MATLAB notation. In the case of PCA, $\mathbf{Z}$ is diagonal, *i.e*., $z_{m,n} \neq 0$ only if $m = n$. The scalar version of Eq. (1) is

$$x_{i,j} = \sum_{m,n} z_{m,n} u_{i,m} v_{j,n}. \tag{2}$$

A matrix is a 2-dimensional array; tensor analysis more generally operates on a multidimensional array, or a *data tensor*. For example, a 3-dimensional data tensor (cube) is $\mathcal{X} = [x_{i,j,k}]$. The right hand side of the second "=" in Eq. (1) can be generalized for tensors as

$$\mathcal{X} = \sum_{l,m,n} z_{l,m,n} u_{:,l} \circ v_{:,m} \circ w_{:,n}, \tag{3}$$

where each (l, m, n)-triple $u_{:,l} \circ v_{:,m} \circ w_{:,n}$ is the outer product of the three column vectors, the result of which is a data tensor (cube) of the same size as $\mathcal{X}$. A linear combination of such tensors with $z_{l,m,n}$ as combination weights yields $\mathcal{X}$. Similar to Eq. (2), the scalar version of Eq. (3) is

$$x_{i,j,k} = \sum_{l,m,n} z_{l,m,n} u_{i,l} v_{j,m} w_{k,n}, \tag{4}$$

in which, if we fix $z_{:,:,:}$ as well as any two of the three row vectors $u_{i,:}$, $v_{j,:}$, and $w_{k,:}$, $x_{i,j,k}$ is a linear function of the remaining row vector; that is, Eq. (4) is a *tri-linear* function.

If the summation in Eq. (4) is executed in a particular order, Eq. (4) is equivalent to

$$x_{i,j,k} = \sum_n \left(\sum_m \left(\sum_l z_{l,m,n} u_{i,l} \right) v_{j,m} \right) w_{k,n}. \tag{5}$$

Each summation in Eq. (5) can be viewed as a matrix product, which leads to the matrix form of tensor product as

$$\mathcal{X} = \mathcal{Z} \times \mathbf{U} \times \mathbf{V} \times \mathbf{W}, \tag{6}$$

where $\mathcal{Z} = [z_{i,j,k}]$ is the *core tensor*, which usually has a smaller size than $\mathcal{X}$; $\mathbf{U}$, $\mathbf{V}$, and $\mathbf{W}$ are the matrices with u, v, and w in Eq. (5) as elements. Since the summation order in Eq. (5) can be arbitrarily switched, the tensor product $\times$ in Eq. (6) is *commutative*.

Finally, in Eq. (4), if we fix j, k and vary the index i, the column vector $x_{:,j,k}$ can be viewed as a degenerate cube, evaluated in the following three equivalent ways:

$$\begin{aligned} x_{:,j,k} &= \sum_l u_{:,l} \left(\sum_{m,n} z_{l,m,n} v_{j,m} w_{k,n} \right) \\ &= (\mathcal{Z} \times v_{j,:} \times w_{k,:}) \times \mathbf{U} \\ &= \mathbf{U} (\mathcal{Z} \times v_{j,:} \times w_{k,:}). \end{aligned} \tag{7}$$

3. System Overview

Our face retrieval system consists of the following four components.

- *Face database construction*. We construct a sizable database of aligned faces that exhibit a wide range of pose and facial expression variation. Unfortunately, even state-of-the-art alignment methods [4, 11, 13, 16, 23] cannot guarantee perfect results in all cases; we therefore propose a novel confidence score to filter out poor alignment results based on the face alignment method from [4].
- *Tensor model training*. From a set of 2D training face shapes of different poses, expressions, and identities (each shape is represented by a set of points), we form a 4-dimensional data tensor $\mathcal{X}$, with each of the four dimensions indexing point vertex, pose, expression, and identity. We decompose the tensor as
$$\mathcal{X} = \mathcal{Z} \times \mathbf{U}_{\texttt{vert}} \times \mathbf{U}_{\texttt{pose}} \times \mathbf{U}_{\texttt{expr}} \times \mathbf{U}_{\texttt{iden}}. \quad (8)$$
We propose a new and simple iterative algorithm to achieve this decomposition in the presence of missing data. Note that the dataset for training the model is much smaller than the database used for searching.
- *Tensor coefficient recovery*. Using the estimated $\mathcal{Z}$ and $\mathbf{U}_{\texttt{vert}}$, we associate each aligned face in the database with three coefficient vectors, $\mathbf{c}_{\texttt{pose}}$, $\mathbf{c}_{\texttt{expr}}$, and $\mathbf{c}_{\texttt{iden}}$ for pose, expression, and identity, respectively. For each search, we also estimate the three coefficient vectors $\mathbf{c}_{\texttt{pose}}$, $\mathbf{c}_{\texttt{expr}}$, and $\mathbf{c}_{\texttt{iden}}$ for the user-specified query shape, which we expect to have few known vertices (corresponding to user edits).
- *Search by tensor coefficient comparison*. The coefficient vectors associated with each face in the database are compared against the coefficient vectors estimated for the query face; the closest face images (according to this tensor coefficient comparison) are retrieved. The user can then edit one of the retrieved faces to refine the search.

More details are presented in the subsequent sections.

4. Technical Details

4.1. Tensor Decomposition with Missing Data

In order to obtain $\mathcal{Z}$, $\mathbf{U}_{\texttt{vert}}$, $\mathbf{U}_{\texttt{pose}}$, $\mathbf{U}_{\texttt{expr}}$, and $\mathbf{U}_{\texttt{iden}}$, we need a training set in which each subject's face is photographed under a *complete* set of known expressions and poses, along with ground truth shapes. Such a subset of data is difficult to obtain. Even highly structured datasets like Multi-PIE [3] have missing faces. Computing tensor decomposition in the presence of missing data is unavoidable in practice. We propose a simple algorithm for this purpose.

We separate a data tensor $\mathcal{X}$ into two parts: $[\mathcal{X}_{\texttt{known}}, \mathcal{X}_{\texttt{missing}}]$. For notational convenience, we interchangeably view $\mathcal{X}$ as a multidimensional array or as a vector consisting of all elements in $\mathcal{X}$ (*i.e.*, $\mathcal{X}(:)$ in MATLAB notation). Without loss of generality, we assume that the vectorized form is arranged such that the missing elements come after the known elements. Under this arrangement, we seek to estimate $\mathcal{X}_{\texttt{missing}}$, $\mathcal{Z}$, $\mathbf{U}_{\texttt{vert}}$, $\mathbf{U}_{\texttt{pose}}$, $\mathbf{U}_{\texttt{expr}}$, and $\mathbf{U}_{\texttt{iden}}$ by minimizing the following error function:

$$\left\| \mathcal{Z} \times \mathbf{U}_{\texttt{vert}} \times \mathbf{U}_{\texttt{pose}} \times \mathbf{U}_{\texttt{expr}} \times \mathbf{U}_{\texttt{iden}} - \begin{bmatrix} \mathcal{X}_{\texttt{known}} \\ \mathcal{X}_{\texttt{missing}} \end{bmatrix} \right\|^2. \quad (9)$$

We iterate between the following steps to minimize Eq. (9):

1. Fix $\mathcal{X}_{\texttt{missing}}$ and optimize $\mathcal{Z}$, $\mathbf{U}_{\texttt{vert}}$, $\mathbf{U}_{\texttt{pose}}$, $\mathbf{U}_{\texttt{expr}}$, and $\mathbf{U}_{\texttt{iden}}$. This step is the standard tensor decomposition, as given in [9].
2. Fix all the $\mathbf{U}$'s and optimize $\mathcal{X}_{\texttt{missing}}$ and $\mathcal{Z}$. This step is a least squared problem because $\mathcal{Z} \times \mathbf{U}_{\texttt{vert}} \times \mathbf{U}_{\texttt{pose}} \times \mathbf{U}_{\texttt{expr}} \times \mathbf{U}_{\texttt{iden}}$ is linear with respect to $\mathcal{Z}$ when all the $\mathbf{U}$'s are fixed. More specifically, we use
$$\mathbf{A}\mathcal{Z} = \begin{bmatrix} \mathbf{A}_{\texttt{known}} \\ \mathbf{A}_{\texttt{missing}} \end{bmatrix} \mathcal{Z} \quad (10)$$
to represent this linear transformation, and Eq. (9) becomes
$$\left\| \begin{bmatrix} \mathbf{A}_{\texttt{known}} & \mathbf{0} \\ \mathbf{A}_{\texttt{missing}} & -\mathbf{I} \end{bmatrix} \begin{bmatrix} \mathcal{Z} \\ \mathcal{X}_{\texttt{missing}} \end{bmatrix} - \begin{bmatrix} \mathcal{X}_{\texttt{known}} \\ \mathbf{0} \end{bmatrix} \right\|^2. \quad (11)$$
This function can be efficiently minimized using the conjugate gradient method, assuming both transformations $\mathbf{A}$ and $\mathbf{A}^\top$ can be implemented without explicitly storing the matrix elements. This is indeed the case because $\mathbf{A}\mathcal{Z}$ represents $\mathcal{Z} \times \mathbf{U}_{\texttt{vert}} \times \mathbf{U}_{\texttt{pose}} \times \mathbf{U}_{\texttt{expr}} \times \mathbf{U}_{\texttt{iden}}$ and $\mathbf{A}^\top \mathcal{X}$ represents $\mathcal{X} \times \mathbf{U}_{\texttt{vert}}^\top \times \mathbf{U}_{\texttt{pose}}^\top \times \mathbf{U}_{\texttt{expr}}^\top \times \mathbf{U}_{\texttt{iden}}^\top$.

In practice, we initialize $\mathcal{X}_{\texttt{missing}}$ using subsets of face shapes from $\mathcal{X}_{\texttt{known}}$. That is, for a needed pose i, expression j, and identity k vector $x_{:,i,j,k}$ in $\mathcal{X}_{\texttt{missing}}$, we gather all vectors in $\mathcal{X}_{\texttt{known}}$ that share pose i and expression j; the mean average of these vectors is used to initialize $x_{:,i,j,k}$.

4.2. Tensor Coefficient as Facial Feature Vector

In this subsection, we present an algorithm that takes a face shape (partial or complete) as input and computes three coefficient vectors describing the pose, expression, and identity of the face. This algorithm is used in two places in our retrieval system. First, it gives each face shape in the database pose, expression, and identity coefficient vectors, which are used as features vectors. Second, it estimates the three feature vectors from a few user constraints and retrieves faces with similar feature vectors from the database in real time.

Since a user seldom wants to edit every point on the face contour for retrieval, our algorithm needs to be able to handle partial shapes. From Eq. (8), we know that a face shape vector $\mathbf{f}$ can be expressed as

$$\mathbf{f} = \mathcal{Z} \times \mathbf{U}_{\texttt{vert}} \times \mathbf{c}_{\texttt{pose}}^\top \times \mathbf{c}_{\texttt{expr}}^\top \times \mathbf{c}_{\texttt{iden}}^\top, \quad (12)$$

where the core tensor $\mathcal{Z}$ and the vertex basis matrix $\mathbf{U}_{\texttt{vert}}$ are estimated in Section 4.1, and $\mathbf{c}_{\texttt{pose}}$, $\mathbf{c}_{\texttt{expr}}$, and $\mathbf{c}_{\texttt{iden}}$ are the

coefficient vectors we seek. We break $\mathbf{f}$ into $[\mathbf{f}_{\texttt{known}}, \mathbf{f}_{\texttt{unknown}}]$; our goal is to compute all the $\mathbf{c}$ vectors from $\mathbf{f}_{\texttt{known}}$.

In practice, the dimension of $\mathbf{f}_{\texttt{known}}$ is often much less than the total number of variables in $\mathbf{c}_{\texttt{pose}}$, $\mathbf{c}_{\texttt{expr}}$, and $\mathbf{c}_{\texttt{iden}}$, which makes the estimation under-constrained. To address this issue, we estimate all of the $\mathbf{c}$'s by minimizing a regularized objective function as follows:

$$\begin{aligned} & \phi(\mathbf{c}_{\texttt{pose}}, \mathbf{c}_{\texttt{expr}}, \mathbf{c}_{\texttt{iden}}) \\ = \; & \tfrac{1}{2\sigma^2} \| \mathbf{f}_{\texttt{known}} - \mathcal{Z} \times \mathbf{U}_{\texttt{vert}}^{\texttt{known}} \times \mathbf{c}_{\texttt{pose}}^{\top} \times \mathbf{c}_{\texttt{expr}}^{\top} \times \mathbf{c}_{\texttt{iden}}^{\top} \|^2 \\ + \; & \|\mathbf{c}_{\texttt{pose}}\|^2 + \|\mathbf{c}_{\texttt{expr}}\|^2 + \|\mathbf{c}_{\texttt{iden}}\|^2, \end{aligned} \tag{13}$$

where $\mathbf{U}_{\texttt{vert}}^{\texttt{known}}$ are the rows in $\mathbf{U}_{\texttt{vert}}$ that correspond to $\mathbf{f}_{\texttt{known}}$, and σ^2 is the variance of tensor shape fitting noise, estimated as the average of the squared residual errors after tensor model fitting by minimizing Eq. (9).

We minimize Eq. (13) iteratively. Starting with an initial estimate of $\mathbf{c}_{\texttt{pose}}$, $\mathbf{c}_{\texttt{expr}}$, and $\mathbf{c}_{\texttt{iden}}$, we iteratively hold two of them constant and update the remaining one until Eq. (13) decreases by 10^{-6} or less compared to the previous iteration. This algorithm is used both during runtime for retrieval and during the face database construction stage; the initialization of the $\mathbf{c}$'s are described in the following two subsections.

4.3. Searching for Faces Using Tensor Coefficients

Using the tensor coefficient vectors $\mathbf{c}_{\texttt{pose}}$, $\mathbf{c}_{\texttt{expr}}$, and $\mathbf{c}_{\texttt{iden}}$ in Section 4.2, our system enables a user to search images by shape manipulation. For example, starting with one face image, the user can drag the tip of the nose to find images with desired pose. Constraining the locations of multiple points on the face will further refine the search results.

In general, computing all the $\mathbf{c}$ coefficient vectors from a single user input is under-constrained even with the regularization term in Eq. (13). For example, dragging the corner of the mouth may result in a smiling face or a rotated face.

To address this ambiguity, our search interface allows the user to specify whether the pose vector or the expression vector should be adjusted to satisfy the edit. Given this user specification, only the selected coefficient vector(s) will be estimated when minimizing Eq. (13). For example, if the user selects "pose," then only the pose coefficient vector will be estimated. The non-selected coefficient vector (*e.g.*, expression in the previous example) is used to calculate the distance between each image in the database and the current query image; the nearest 1000 images form a search candidate set. This winnowing procedure helps to minimize non-selected shape variation in the search results (*e.g.*, expression will remain approximately constant if only pose is selected). Once the coefficient vectors are estimated, we use them to construct a face $\mathbf{f}$ using Eq. (12). Finally, the most similar 50 faces in the search candidate set are retrieved based on their Euclidean distance from $\mathbf{f}$.

4.4. Face Alignment Confidence Measure

To support face search by shape manipulation, our system needs automatic face alignment to establish the shape of each face in the database. We implemented the robust face alignment method of Gu and Kanade [4] for this purpose. Being among state-of-the-art methods [4, 11, 13, 16, 23], this method produces impressive results on real world images, as we demonstrate in Section 5.1. However, natural face images exhibit a wide range of shape, pose, illumination, and other appearance variation; occlusions, image noise, and motion blur further confound the problem. This method does not guarantee accuracy in all situations.

One approach might be to remain agnostic—to simply allow all aligned faces to exist in the database regardless of their accuracy. However, this reduces the quality of the query results, and it burdens users with the additional task of recognizing and ignoring poorly aligned results. Similarly, manually removing poorly aligned faces from large databases is burdensome if not impractical.

4.4.1 Identifying Poorly Aligned Faces

We instead propose a novel method of measuring alignment confidence, which allows us to *automatically* remove poorly aligned faces from the database. In a nutshell, our method computes a confidence score for each vertex in the aligned shape, filters the scores along the shape contours, and finally integrates the filtered scores to arrive at an overall confidence measure.

More precisely, the alignment confidence score s_n for the point n is computed as

$$s_n = \exp\left\{-\frac{\rho_n^2}{2d_n}\right\}. \tag{14}$$

d_n is the average distance from point n to its contour neighbors in the shape model (the canonical shape we used is approximately 160 pixels tall, from eyebrows to chin, irrespective of the target face size), ρ_n^2 is found by the alignment algorithm [4] and is the observation variance, or noise level, of landmark n at the end of the matching process. Figure 2 shows an illustration of the ρ_n values for an actual alignment result.

In Eq. (14), the landmark confidence function maps the raw observation variance to the range $[0, 1]$. It acts as a robust measure of confidence, with 1 denoting maximum confidence, and 0 denoting no confidence; landmark locations with very large observation variance are clamped to 0. Empirically, we found that if s_n is large, the alignment for point n is often reliable. However, if s_n is small, the alignment for point n *may or may not* be reliable, depending on the alignment accuracy of its neighboring points.

To deal with this phenomenon, we filter the point confidence scores using $f(\cdot)$ defined as

$$f(s_n) = \max\{s_n, g(s_n)\}, \tag{15}$$

where $g(s_n)$ is a Gaussian filter of the confidence scores along point n's contour, centered on point n, with $\sigma = 1.5$ in units of neighbor rank, *i.e.*, 1 = nearest neighbor, 2 = second nearest neighbor, *etc.* in one direction.

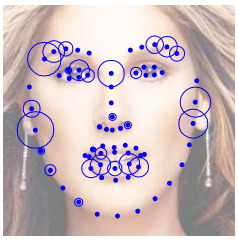

Figure 2. An illustration of the alignment confidence of each landmark. The radius of each circle is given by the variance ρ_n, which measures the noise level of the observation at landmark n; large circles indicate low confidence and small circles indicate high confidence. Each confidence measure gives a *hint* at the alignment accuracy of the corresponding landmark. Unfortunately, spurrious local image features can cause this hint to be wrong. To correct local confidence mistakes, we filter the confidence measures. Finally, we aggregate them to arrive at an overall alignment score, as shown in Section 4.4.1.

In Eq. (15), the filter aims to eliminate erroneously labeled bad landmark locations. For example, a landmark might have low confidence because of spurious local image features despite its contour neighbors exhibiting high confidence. In such a scenario, the alignment algorithm would significantly reduce the contribution of the low confidence point so that the contour will be driven by the surrounding high confidence neighbors. Assuming the contour is correct, the single low confidence landmark should also be correct, and should have a higher confidence value than it was originally given.

Finally, the overall alignment confidence for a face shape is computed as

$$\bar{\mathbf{s}} = \frac{1}{N}\sum_{n=1}^{N} f\{s_n\}, \tag{16}$$

where N is the number of landmarks.

We found that the confidence score is strongly correlated with alignment accuracy as shown in Section 5.2. By comparing the confidence score of each alignment result to a threshold, bad alignment results can be identified and removed from the database.

5. Experiments

In this section we describe in detail the construction of a sizable and varied database of aligned faces; we demonstrate experimentally that the alignment confidence score described in Section 4.4.1 is a good predictor of alignment accuracy; and we show that the tensor model described in Section 3 allows a user to find desired faces using only one or a few shape edits.

5.1. Constructing a Database of Aligned Faces

Our system is designed to search for desired face images in a large database. For experimentation purposes, we constructed a sizeable database of approximately 10,000 aligned faces from the Public Figures (PubFig) dataset [8]. In its entirety, PubFig contains roughly 50,000 images of 200 celebrities from the internet captured in uncontrolled environments.

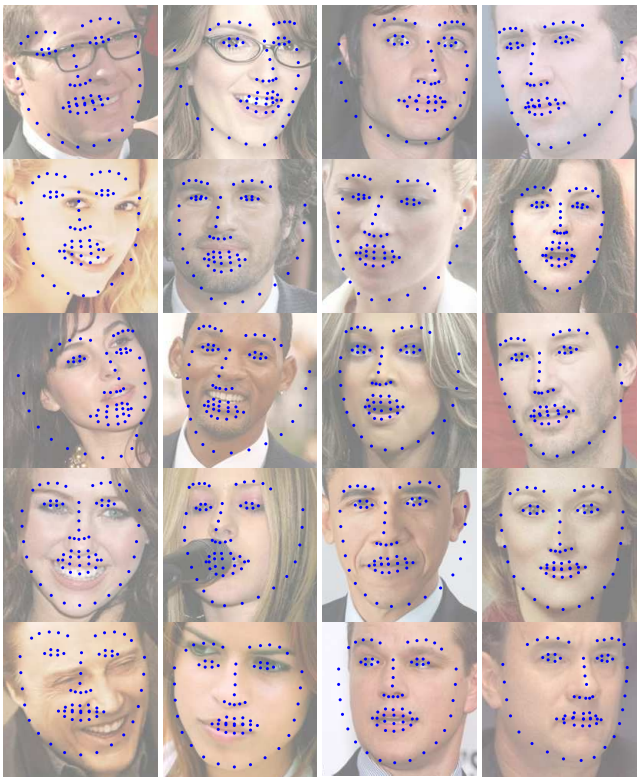

Figure 3. Each column shows a selection of alignment results on the PubFig dataset that share approximately the same alignment confidence score. The scores are ordered from left to right. Each column shows typical results for the associated score. The score is statistically a good indicator of alignment accuracy, as we show in Figure 4. **Best viewed electronically**

The faces were aligned using our implementation of [4]. We first trained shape and appearance models using ground truth landmarks provided with the Multi-PIE dataset [3], and 962 additional ground truth landmarks that we supplied (the original set of ground truth landmarks does not include faces with both non-frontal pose and non-neutral expression). Approximately 330 subjects are represented in our training set, with five different poses (± 30, ± 15, and 0 degrees relative to frontal) and six different expressions (neutral, smile, surprise, squint, disgust, and scream), although about half of the possible subject-pose-expression combinations are missing.

For each face, we first use a face detector [12] to estimate the size of the face in each image. We removed all faces with bounding boxes smaller than 120 pixels in height to ensure the database would contained few low quality images. We ran the face alignment algorithm on the remaining 21,919 images.

Our system relies on face alignment accuracy to return good query results. Therefore, we used a relatively high alignment confidence score threshold to remove all but the most confident 10,000 results from the database. In Figure 3, the results in each column are representative of results with similar align-

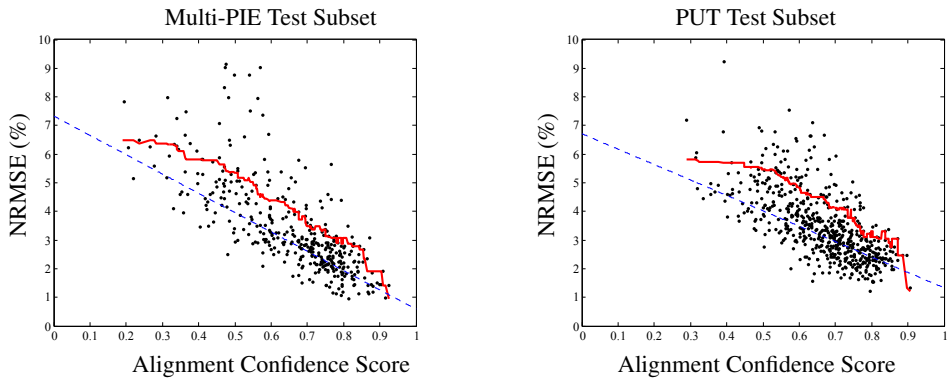

Figure 4. Each data point represents one alignment result. The y-axis gives the *normalized root-mean-squared error* (NRMSE) of each result, and the x-axis gives the alignment confidence score, according to Section 4.4.1. A good/bad score threshold can be thought of as a boundary that separates "bad" results to the left from "good" to the right. The solid red line gives a 95% error bound for any given score threshold. That is, for a given score threshold, 95% of the "good" results have a NRMSE at or below the red line. The dashed blue line shows the global trend of the best 95% NRMSE with respect to the score, found by dividing the scores into 0.05-width bins, computing the mean average NRMSE among points in each bin under the 95% error bound, and fitting a line to the averages. We do not show a linear regression fit of all the points in the plot because the tightly clustered points in the lower right corner heavily dominate, and the resulting fit does not reflect the global trend. **Best viewed electronically**

ment confidence score. The score is statistically a good indicator of the alignment accuracy. However, we note that the score is not perfect; outliers do exist. Some good results have uncharacteristically low scores and vice versa, which is quantitatively characterized in Figure 4.

5.2. Alignment Confidence Score Performance

The alignment confidence score given in Section 4.4.1 should be correlated with the alignment error. A low score should predict large alignment error and vice versa. Here we give experimental results that confirm this relationship.

We first trained shape and appearance models as in Section 5.1, but with subjects 1 - 20 omitted from the training set. The shape and appearance models were then used to align the 375 images of subjects 1 - 20. After alignment, we computed (1) the normalized root-mean-squared error (NRMSE) relative to ground truth, and (2) the alignment confidence score described in Section 4.4.1. The NRMSE is given as a percentage, computed by dividing the root mean squared (RMS) error by the height of the smallest bounding box that encompasses the ground truth landmarks in each image. A similar measure is given in [11], but they divide the RMS error by the pupillary distance; this is not invariant under pose change and so we don't use it. Figure 4(a) shows that the NRMSE decreases significantly as the alignment confidence score increases.

Although the sets of subjects used for training and testing in the previous experiment were independent, they both came from the same structured database. Similarities exist between these training and testing sets that would not occur naturally. To avoid making erroneous conclusions that might be due to these similarities, we performed the experiment again using the same training set, but a different test set—namely 583 images selected randomly from the PUT face database [5], which exhibits moderate variation in pose and facial expression.

The ground truth landmarks given in the PUT database do not exactly match those given in the Multi-PIE database. However, with few exceptions, corresponding ground truth contours exist. We therefore divide up the PUT contours to obtain a set of landmark points that very closely match the Multi-PIE ground truth. The six Multi-PIE landmarks not found on any PUT contours (four on the vertical portion of the nose and one near each ear) were omitted in computing the NRMSE. Figure 4(b) similarly shows that the NRMSE decreases significantly as the alignment confidence score increases.

5.3. Tensor Model Training

To construct the tensor model, we used 1470 faces (49 identities $\times$ 5 poses $\times$ 6 expressions) from the Multi-PIE dataset. Within the tensor model we used 4 bases for identity, 5 for pose, and 6 for expression. We specifically retained all bases for pose and expression because they constitute the two searching options used in our system. We used 120 bases to represent the point vertices, because the number of bases for point vertices in Eq. (13) should be larger than the number of unknown coefficients so that solving for the unknown coefficients is over-determined for known faces.

5.4. Face Retrieval Performance

In this section we demonstrate that users only need to edit one or a few face points for our system to find desired faces. Figure 5 shows the top user-selected results for three queries. By using a combination of multiple edits and holding expression or pose constant, users can refine their search result in an intuitive way. **Please refer to our supplemental video for a demonstration of the interactive search experience.**

We also quantitatively evaluate how many edits on a query

(a) *One edit*. Expression is held constant; the nose tip is dragged left to search for similar faces with slightly leftward pose.

(b) *Two edits*. Pose is held constant; the lips are dragged apart to search for smiling faces.

(c) *Three edits*. Pose is held constant; the lips are pulled together and the cheeck is pulled outward to search for serious expressions.

Figure 5. Top user-selected results for three queries. The query images are shown in the leftmost column, with user edits illustrated by yellow arrows. Each row shows one query. Five selected results are given to the right of each query image. **Please refer to our supplemental video for a demonstration of the interactive search experience.**

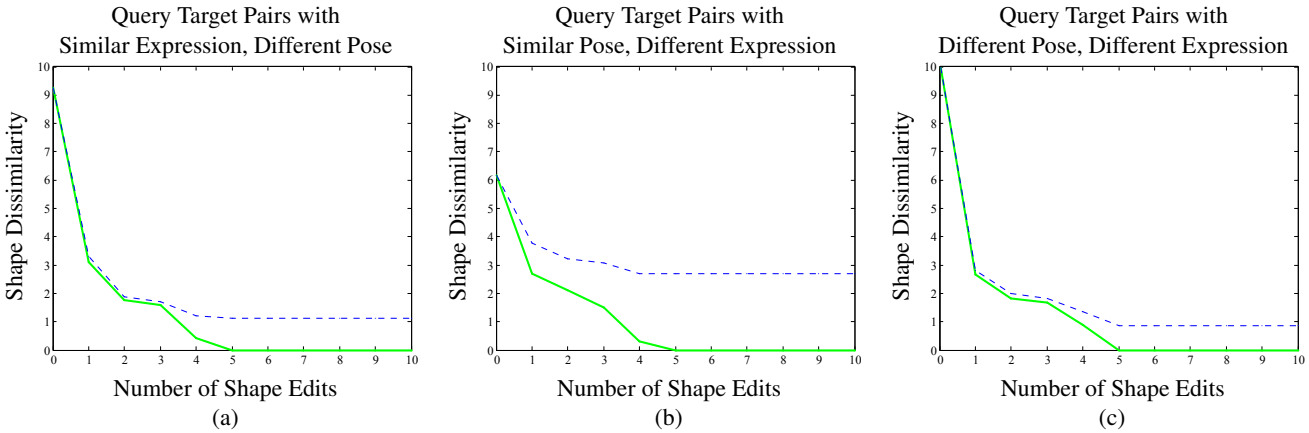

Figure 6. The number of edits needed to find desired faces. The x-axis is the number of edits performed; $x = 0$ corresponds to the original query shape before any edit is applied. The y-axis shows the shape difference between the result shapes and the target shape. The solid green line indicates the minimum Euclidean distance between the target face shape and each of the top 10 results (*i.e.*, it represents the best among the returned results). The dashed blue line indicates the average Euclidean distance. Each curve is computed by averaging over 10 query-target image pairs. In all the tested cases, with fewer than 5 edits, the target image can be retrieved. Note that the maximum number of edits in the x-axis is 10, which is only a small portion of the total number (68) of points in the face shape model. In fact, even after only 1-2 edits, the decrease in shape difference is significant, which suggests that we can retrieve images similar to the desired one within 1-2 edits.

image are needed in order to find a target image. To this end, we emulate user edits in our system as follows. We start by picking up query-target image pairs from the database. Each pair of images either share a similar pose but have different expressions, or share a similar expression but have different poses, or are dissimilar in both pose and expression.[1]

Given a pair of query and target images, our testing system randomly orders the landmarks and edits them one after another according to this order. Each edit moves one landmark from the query to the target image. After each edit, the top 10 results will be retrieved. We calculate the average Euclidean distance between the top 10 results and the desired image. We also calculate the minimum Euclidean distance between the top results and the target image. If the minimum Euclidean

[1]When selecting a query-target image pair with a similar expression but different poses, we first randomly pick the target image, then remove all the images whose Euclidean distance is among the nearest 1/3rd. In the remaining images, we select the one whose expression is most similar to the target image by comparing $\mathbf{c}_{\mathtt{pose}}$ and use it as the query image. We can select a pair with a similar pose but different expressions in a similar way. A pair with both dissimilar pose and dissimilar expressions can be selected randomly.

distance is zero, it indicates the target is among the top results.

Figure 6 shows the results, averaged over 10 randomly selected pairs. The x-axis is the number of edits operated; $x = 0$ corresponds to the original query shape before any edits are applied. The y-axis shows the shape difference between the result shapes and the target shape. The solid green line indicates the minimum Euclidean distance between the target and the returned 10 results, so it represents the best shape among the returned results. The dashed blue line indicates the average Euclidean distance. As the number of edits increases, a more accurate shape will be returned. In all tested cases, with fewer than 5 edits, the target image can be retrieved. Note that the max number of edits on the x-axis is 10, which is only a small portion of the total number (68) of points in the face shape model. In fact, even after only 1-2 edits, the decrease in shape difference is significant, which suggests that very often we can retrieve images similar to the desired one within 1-2 edits.

6. Conclusion and Future Work

In this paper, we have proposed a new face search technique that aims to address a common problem in face image search. That is, it is difficult to refine face search results based on subtle shape attributes that are easy to see, but hard to describe in text. To the best of our knowledge, our system is the first face retrieval approach based chiefly on shape manipulation. This approach is complementary to current search engines [1, 6, 7, 21], and could be used to further refine face search results. Such a system could be used by law enforcement agencies to quickly profile a suspect, or a photo editor could use it to quickly find a face with desired expression and pose for compositing, for example.

Although we have a reasonable confidence measure which helps us to automatically construct a sizable database, face alignment still needs improvement to further enhance the system performance and utility, both in terms of query accuracy and constructing a database with better face alignments.

Our database is relatively sparse compared to much larger collections [7], which reduces our ability to lock identity in refining search results. In the future we hope to demonstrate our approach on databases containing millions of face images and videos. To realize this goal, we will need to use a more efficient coefficient search algorithm than our naive linear search.

Although face image retrieval is a key challenge, we note that our approach generalizes well. As part of our future work, we hope to apply our technique to other more general types of image collections. Additionally, we hope to incorporate appearance-based attributes into our system to further improve search results.

7. Acknowledgements

This work is supported in part by NSF IIS-0845916, NSF IIS-0916441, a Sloan Research Fellowship, and a Packard Fellowship for Science and Engineering. Brandon Smith is also supported by an NSF graduate fellowship.

References

[1] D. Bitouk, N. Kumar, S. Dhillon, P. N. Belhumeur, and S. K. Nayar. Face Swapping: Automatically Replacing Faces in Photographs. *SIGGRAPH*, 2008.

[2] D. B. Goldman, C. Gonterman, B. Curless, D. Salesin, and S. M. Seitz. Video annotation, navigation, and composition. In *UIST*, 2008.

[3] R. Gross, I. Matthews, J. Cohn, T. Kanade, and S. Baker. Multi-pie. In *Proc. Int. Conf. Autom. Face Gesture Recog.*, 2010.

[4] L. Gu and T. Kanade. A generative shape regularization model for robust face alignment. In *ECCV*, 2008.

[5] A. Kasiński, A. Florek, and A. Schmidt. The put face database. *Technical report, Poznan University of Technology*, 2009.

[6] I. Kemelmacher-Shlizerman, A. Sankar, E. Shechtman, and S. M. Seitz. Being john malkovich. In *ECCV*, 2010.

[7] N. Kumar, P. Belhumeur, and S. Nayar. FaceTracer: A search engine for large collections of images with faces. In *ECCV*, 2008.

[8] N. Kumar, A. C. Berg, P. N. Belhumeur, and S. K. Nayar. Attribute and Simile Classifiers for Face Verification. In *ICCV*, 2009.

[9] L. D. Lathauwer, B. D. Moor, and J. Vandewalle. On the best rank-1 and rank-(R1,R2,...,Rn) approximation of higher-order tensors. *SIAM J. Matrix Analysis and Applications*, 2000.

[10] M. Lau, J. Chai, Y.-Q. Xu, and H. Shum. Face poser: Interactive modeling of 3d facial expressions using facial priors. In *SIGGRAPH*, 2010.

[11] X. Liu, Y. Tong, F. W. Weeler, and P. H. Tu. Facial contour labeling via congealing. In *ECCV*, 2010.

[12] H. Rowley, S. Baluja, and T. Kanade. Rotation invariant neural network-based face detection. In *CVPR*, 1998.

[13] J. Saragih, S. Lucey, and J. Cohn. Face alignment through subspace constrained mean-shifts. In *CVPR*, 2009.

[14] R. W. Sumner, M. Zwicker, C. Gotsman, and J. Popović. Mesh-based inverse kinematics. In *SIGGRAPH*, 2005.

[15] D. Tao, J. Sun, X. Wu, X. Li, J. Shen, S. J. Maybank, and C. Faloutsos. Neural information processing: Prob. tensor analysis with akaike and bayesian information criteria. 2008.

[16] Y. Tong, X. Liu, F. W. Wheeler, and P. Tu. Automatic facial landmark labeling with minimal supervision. In *CVPR*, 2009.

[17] M. A. O. Vasilescu and D. Terzopoulos. Multilinear analysis of image ensembles: TensorFaces. In *ECCV*, 2002.

[18] M. A. O. Vasilescu and D. Terzopoulos. Tensortextures: multilinear image-based rendering. *ACM Trans. Graph.*, 2004.

[19] D. Vlasic, M. Brand, H. Pfister, and J. Popović. Face transfer with multilinear models. In *SIGGRAPH*, 2005.

[20] H. Wang, Q. Wu, L. Shi, Y. Yu, and N. Ahuja. Out-of-core tensor approximation of multi-dimensional matrices of visual data. *ACM Trans. Graph.*, 2005.

[21] Z. Wu, Q. Ke, J. Sun, and H.-Y. Shum. Scalable face image retrieval with identity-based quantization and multi-reference reranking. In *CVPR*, 2010.

[22] L. Zhang, N. Snavely, B. Curless, and S. M. Seitz. Spacetime faces: High-resolution capture for modeling and animation. In *SIGGRAPH*, August 2004.

[23] J. Zhu, L. V. Gool, and S. C. H. Hoi. Unsupervised face alignment by robust nonrigid mapping. In *ICCV*, 2009.