Feature-Based Techniques for Real-Time Morphable Model Facial Image Analysis

Siddhartha Chaudhuri
Indian Institute of Technology, Kanpur, India
`sidc@iitk.ac.in`

Supervisors:

Edoardo Charbon
Randhir K. Singh
École Polytechnique Fédérale de Lausanne, Switzerland
{`edoardo.charbon, randhirkumar.singh`}`@epfl.ch`

Abstract

We present an algorithm to quickly analyse and compress facial images using a 2-dimensional morphable model. It runs in real-time on reasonable resources, and offers considerable opportunities for parallelization.

A morphable model associates a "shape vector" and a "texture vector" with each image of a sample set. The model is used to analyze a novel image by estimating the model parameters via an optimization procedure. The novel image is compressed by representing it by the set of best match parameters. For real-time performance, we separate the novel image into shape and texture components by computing correspondences between the novel image and a reference image, and match each component separately using eigenspace projection. This approach can be easily parallelized.

We improve the speed of algorithm by exploiting the fact that facial correspondence fields are smooth. By computing correspondences only at a number of "feature points" and using interpolation to approximate the dense fields, we drastically reduce the dimensionality of the vectors in the eigenspace, resulting in much smaller compression times. As an added benefit, this system reduces spurious correspondences, since weak features that may confuse the correspondence algorithm are not tracked.

Contents

1 Introduction

Portable and embedded devices typically need to communicate at high speed through low bandwidth channels. Frequently, images are transmitted or received by such devices, for example in a wireless security camera network or in a video phone. It is essential to compress these images to the smallest size possible, so that they can be transmitted quickly over slow connections. In such applications, lossless compression is not essential, since the display systems are usually of limited quality and size. It is however important to preserve strong visual features of the image, so that easy recognition is possible.

At the EPFL, Switzerland, the *MegaWatch* project aims to build a wristwatch-sized system with a range of functionalities, including image acquisition and transmission over very low bandwidth channels. One of the targetted applications of this feature is to capture and transmit images of faces – of the wearer and of other people. The system has considerable multi-processing power, so advanced techniques may be used for image compression and reconstruction. We present a real-time procedure designed to compress facial images on such a platform.

The method outlined in this report is extremely fast, reasonably robust and highly parallelizable. It takes advantage of the relative visual significances of image areas: strong features such as the mouth and the eyes are better preserved than flat features such as the cheeks and forehead. It has been tested on a large number of facial images and its various components are the subjects of much ongoing study, so we can expect to see considerable refinements and improvements in the near future.

2 Background

Image compression may be divided into two categories: **informed** and **uninformed**. Uninformed compression assumes no knowledge of the objects represented by the image. Most general image compression standards such as GIF, PNG, JPEG and JPEG2000 fall in this category. The sequence of pixel intensities is compressed using lossless (which allow exact reconstruction) or lossy (which allow only approximate reconstruction) methods. The advantage of uninformed methods is that any image whatsoever may be compressed with adequate scope for reconstruction – they are therefore suitable for general image processing applications.

Informed algorithms deal with specific classes of images, for example facial images. Much greater compression is possible because the range of images is restricted. However, a setup to compress one class of images cannot be used to compress other classes – a facial image compression system will handle faces but not furniture. In many applications, such restriction is

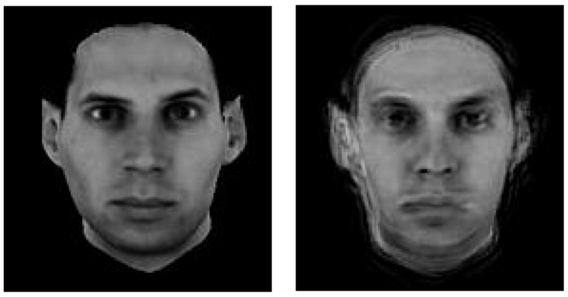

Figure 1: Matching by pixel-wise overlay. On the left is the original image, on the right the best match.

acceptable. For example, mugshots of criminals in police records are frontal or profile views of faces, all shot from the same viewpoint under similar lighting conditions.

Much effort in developing informed compression algorithms stemmed from research in face identification systems. The task was to identify a face from a photograph. It was assumed that another photograph of the same person was already in a large database of facial images. The new image was approximated as closely as possible by a linear combination of the images in the database. The person was identified by the database image that had the largest coefficient in this combination. This method worked reasonably well for identification, but the approximations were fuzzy and badly-defined because the faces were generally not of the same shape. Fig. 1 illustrates this.

To solve this problem, it was essential to somehow *normalize* the facial images so that they all had the same shape and could be overlaid accurately. **Morphable models** were developed to provide such a representation of image classes. In its conventional form, a morphable model associates a **shape** and a **texture** with each image in a sample set. The shape component (a correspondence function) describes how the sample image deviates from a reference image. The texture component encodes the colour/intensity information of the sample image, normalized to the reference shape. Fig. 2 shows an instance of this representation.

A novel image is approximated as a warped combination of the components in the morphable model. More precisely, for n sample images, if the shapes are represented by the set $\{S_i,\ i = 1 \dots n\}$ and the textures by the set $\{T_i,\ i = 1 \dots n\}$, then the novel image I_{novel} is approximated as:

$$I_{novel}(x, y) \approx T_{model} \circ S_{model}^{-1}(x, y)$$

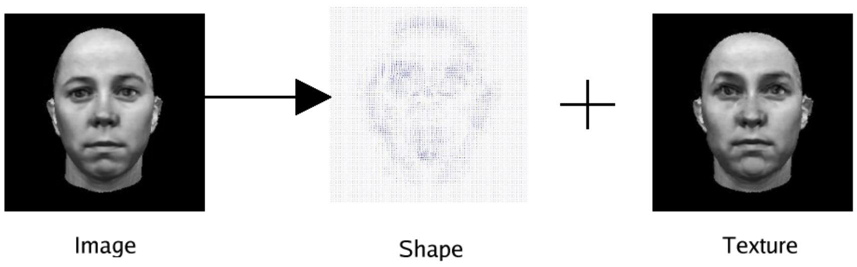

Figure 2: An image split into shape and texture components.

where

$$T_{model} = \sum_{i=1}^{n} b_i T_i$$

$$S_{model} = \sum_{i=1}^{n} c_i S_i$$

Intuitively, the novel texture is approximated by a linear combination of the sample textures, and is then warped to the approximate shape of the novel image by a linear combination of sample shapes. The scalars $\{b_i\}$ and $\{c_i\}$ are called the *best match coefficients*.

We have assumed here that our shape components S_i are defined in the direction of a "backward warp". That is, $S_i(x, y)$ is the location of the point in the ith sample image that corresponds to the point (x, y) in the reference image. To obtain the novel image from the novel texture, we must apply a "forward warp", so we use the inverse of the model shape S_{model}.

Assuming the approximation is good, drastic compression is achieved by representing the novel image only by the best match coefficients. It is easy to see that if the model is known, these coefficients are sufficient to reconstruct the closest match to the novel image.

Vetter, Jones, Poggio, Beymer et al have done a considerable amount of work in developing two and three dimensional morphable models. Their work has mainly focussed on facial images. This is influenced by the fact that the morphable model description lends itself very well to faces: a dense and smooth correspondence field can be established between features on two different faces, something which is difficult to do with, say, buildings, in which external features present in one are frequently not present in another. Of course, we assume that the two faces have similar hair growth, accessories (such as spectacles), and no missing features (van Gogh, minus one ear, might be difficult to represent with the model).

In the approach of Jones, Poggio, Vetter et al [5], the mismatch between the morphable model and novel image is represented by a single sum-of-squared-errors function, which may be written as:

$$E(\mathbf{b}, \mathbf{c}) = \sum_{x,y} \Big(I_{novel}(x, y) - T_{model} \circ S^{-1}_{model}(x, y) \Big)^2$$

where $\mathbf{b}$ represents the set of b_i's and $\mathbf{c}$ represents the set of c_i's. A numerical minimization method such as stochastic gradient descent is used to minimize the error and thus obtain the best match coefficients $\mathbf{b}$ and $\mathbf{c}$.

3 A Real-Time Approach

Our objective was to design a real-time system that would match a morphable two-dimensional face model to a novel image. We experimented with the error minimization approach of [5] but ruled it out as it took too much time, partly because it is an iterative process that must loop many times before it converges to a solution and partly because the evaluation of the error (and its derivatives) at each step requires a large number of operations.

A faster solution was to first separate the novel image into shape and texture components [8] – the construction of the model requires that sample images be split in this way, so code could be reused. The components are then matched individually: the novel shape is approximated as a linear combination of sample shapes and the novel texture as a linear combination of sample textures.

The advantage of matching components separately is that the error functions are much simpler. They do not have the implicit warping induced by the $T_{model} \circ S^{-1}_{model}$ term. For example, the texture error in the new formulation is simply:

$$E_{texture}(\mathbf{b}) = \sum_{x,y} \Big(T_{novel}(x, y) - T_{model}(x, y) \Big)^2$$

where T_{novel} is the novel texture and T_{model} is $\sum_{i=1}^{n} b_i T_i$ as before.

3.1 Shape and Texture Representation

Following the standard practice, we represent both shapes and textures as vectors indexed by image coordinates. Each element of a vector corresponds to an image pixel. The texture vector stores intensity values, and the shape vector stores the (absolute or relative) coordinates of corresponding points. We note that each element of a shape vector has two components, in the x and y directions. In our implementation, we simplify this further with two separate vectors, one storing the correspondence values in the x direction and the other in the y direction. Our three error terms are, therefore:

$$E_{x,shape}(\mathbf{c_x}) = \sum_{x,y} \Big(S_{x,novel}[x, y] - \sum_{i=1}^{n} c_{x,i} S_{x,i}[x, y] \Big)^2$$

$$E_{y,shape}(\mathbf{c_y}) = \sum_{x,y} \Big(S_{y,novel}[x,y] - \sum_{i=1}^{n} c_{y,i} S_{y,i}[x,y] \Big)^2$$

$$E_{texture}(\mathbf{b}) = \sum_{x,y} \Big(T_{novel}[x,y] - \sum_{i=1}^{n} b_i T_i[x,y] \Big)^2$$

where square brackets represent vector (or array) lookup instead of function evaluation. This representation facilitates an uniform treatment of shape and texture. The cost of the extra coefficients ($\mathbf{c_x}$ and $\mathbf{c_y}$ instead of just $\mathbf{c}$) is outweighed by the ease of implementation. It is possible to work with single shape vectors in which each element has an x and a y component, but the implementation is messier. We will assume separate x and y vectors for shape in the rest of this report: the interested reader may modify the algorithms to work with single vectors.

3.2 Matching a Component: Unconstrained Quadratic Programming

We observe that each of the errors above is a simple quadratic function of the appropriate coefficients. Unconstrained quadratic programming can be used to minimize each error. Consider the texture error. It may be minimized with respect to the vector $\mathbf{b}$ as follows:

1. Transform the error formula: (For clarity, the $[x,y]$ suffixes are not shown. Also, the vector $\{T_1, T_2, \dots T_n\}$ is written as $\mathbf{T}$.)

$$\begin{aligned}
E_{texture}(\mathbf{b}) &= \textstyle\sum_{x,y} \Big(T_{novel} - \mathbf{b}.\mathbf{T} \Big)^2 \\
&= \textstyle\sum_{x,y} \Big(T_{novel}^2 - 2T_{novel}\mathbf{b}.\mathbf{T} + (\mathbf{b}.\mathbf{T})^2 \Big) \\
&= \textstyle\sum_{x,y} T_{novel}^2 - 2\sum_{x,y} T_{novel}\mathbf{b}.\mathbf{T} + \sum_{x,y}(\mathbf{b}.\mathbf{T})^2 \\
&= \textstyle\sum_{x,y} T_{novel}^2 - 2\sum_{x,y} T_{novel}\mathbf{b}.\mathbf{T} + \sum_{x,y} \mathbf{b}(\mathbf{T}^T\mathbf{T})\mathbf{b}^T \\
&= \textstyle\sum_{x,y} T_{novel}^2 - \mathbf{b}.(2\sum_{x,y} T_{novel}\mathbf{T}) + \mathbf{b}(\sum_{x,y} \mathbf{T}^T\mathbf{T})\mathbf{b}^T
\end{aligned}$$

   The error is now in the standard form $k + \mathbf{b}.\mathbf{g} + \mathbf{b}\mathbf{H}\mathbf{b}^T$ of a quadratic programming objective function, with $\mathbf{g} = -2\sum_{x,y} T_{novel}\mathbf{T}$ and $\mathbf{H} = \sum_{x,y} \mathbf{T}^T\mathbf{T}$.

2. Compute the derivative of the error with respect to $\mathbf{b}$:

$$\frac{\mathrm{d}(k + \mathbf{b}.\mathbf{g} + \mathbf{b}\mathbf{H}\mathbf{b}^T)}{\mathrm{d}\mathbf{b}} = \mathbf{g}^T + 2\mathbf{H}\mathbf{b}^T$$

   therefore

$$\frac{\mathrm{d}E_{texture}}{\mathrm{d}\mathbf{b}} = -2\sum_{x,y} T_{novel}\mathbf{T} + \Big(2\sum_{x,y} \mathbf{T}^T\mathbf{T} \Big)\mathbf{b}^T$$

3. Set the derivative to zero. This defines an extremum of the error function. Since our error is a non-negative quadratic function, this extremum must be the minimum.

$$-2\sum_{x,y} T_{novel}\mathbf{T} + \Big(2\sum_{x,y}\mathbf{T}^T\mathbf{T}\Big)\mathbf{b}^T = 0$$

   or

$$\Big(\sum_{x,y}\mathbf{T}^T\mathbf{T}\Big)\mathbf{b}^T = \sum_{x,y} T_{novel}\mathbf{T}$$

   Evaluating the matrices $\sum_{x,y}\mathbf{T}^T\mathbf{T}$ and $\sum_{x,y} T_{novel}\mathbf{T}$ is straightforward.

4. Solve the above system for the best match coefficients $\mathbf{b}$. We used LU decomposition in our experiments.

3.3 Faster Matching: Orthonormal Bases and Projection

The approach of the last section is reasonably effective, but by preconditioning the sample set during model construction we can have much faster matching. The trick is to transform the set of sample vectors into an orthonormal basis. Adding up the projections of the novel vector onto the elements of the basis, we obtain the best match to the novel vector, measured in terms of the sum of squared errors (which is nothing but the squared L$_2$ distance between the vectors). The following lemma proves this.

Lemma 1. *Let V be an n-dimensional subspace of $\mathbb{R}^m$, $n < m$. Let $\{\mathbf{B}_i\}$ be an orthonormal basis of V. Given a vector $\mathbf{X} \in \mathbb{R}^m$, the element of V closest to $\mathbf{X}$ is*

$$\sum_{i=1}^{n}(\mathbf{X}.\mathbf{B}_i)\mathbf{B}_i$$

where the "closest" element $\mathbf{C}$ is defined as that which minimizes the squared L$_2$ *distance $\|\mathbf{X}-\mathbf{C}\|^2$.*

Proof. Let $\mathbf{C} = \sum_{i=1}^{n}(\mathbf{X}.\mathbf{B}_i)\mathbf{B}_i$, and let $\Delta\mathbf{X} = \mathbf{X} - \mathbf{C}$. Consider the dot product of this vector with the elements of the basis.

$$\begin{aligned}
\Delta\mathbf{X}.\mathbf{B}_j &= (\mathbf{X} - \textstyle\sum_{i=1}^{n}(\mathbf{X}.\mathbf{B}_i)\mathbf{B}_i).\mathbf{B}_j \\
&= \mathbf{X}.\mathbf{B}_j - (\textstyle\sum_{i=1}^{n}(\mathbf{X}.\mathbf{B}_i)\mathbf{B}_i).\mathbf{B}_j \\
&= \mathbf{X}.\mathbf{B}_j - \textstyle\sum_{i=1}^{n}\delta_{ij}\mathbf{X}.\mathbf{B}_i \quad \text{(since the basis is orthonormal)} \\
&= \mathbf{X}.\mathbf{B}_j - \mathbf{X}.\mathbf{B}_j \\
&= 0
\end{aligned}$$

(Here δ_{ij} is the Kronecker delta: $\delta_{ij} = 1$ if $i = j$, 0 otherwise.)

So $\Delta\mathbf{X}$ is orthogonal to any vector in V.

Now consider any vector $\mathbf{C}'$ other than $\mathbf{C}$ in V. Since vector spaces are closed under linear combination, $\mathbf{C}' - \mathbf{C}$ is in V. Let $\mathbf{Y} = \mathbf{C}' - \mathbf{C}$. Then,

$$\begin{aligned}
\|\mathbf{X} - \mathbf{C}'\|^2 &= \|\mathbf{X} - \mathbf{C} - \mathbf{Y}\|^2 \\
&= \|\Delta\mathbf{X} - \mathbf{Y}\|^2 \\
&= (\Delta\mathbf{X} - \mathbf{Y}).(\Delta\mathbf{X} - \mathbf{Y}) \\
&= \Delta\mathbf{X}.\Delta\mathbf{X} + \mathbf{Y}.\mathbf{Y} - 2\Delta\mathbf{X}.\mathbf{Y}
\end{aligned}$$

But $\Delta\mathbf{X}$ is orthogonal to $\mathbf{Y}$, since $\mathbf{Y} \in V$. Therefore $2\Delta\mathbf{X}.\mathbf{Y} = 0$. So,

$$\begin{aligned}
\|\mathbf{X} - \mathbf{C}'\|^2 &= \Delta\mathbf{X}.\Delta\mathbf{X} + \mathbf{Y}.\mathbf{Y} \\
&= \|\Delta\mathbf{X}\|^2 + \|\mathbf{Y}\|^2 \\
&> \|\Delta\mathbf{X}\|^2 \\
&= \|\mathbf{X} - \mathbf{C}\|^2
\end{aligned}$$

Therefore $\mathbf{C} = \sum_{i=1}^{n}(\mathbf{X}.\mathbf{B}_i)\mathbf{B}_i$ is indeed the element of V "closest" to $\mathbf{X}$.

$\square$

Each projection takes time linear in the size of the vectors. If each vector has m dimensions (which is, in our case, the number of pixels in each image), then the matching is performed in $\mathrm{O}(mn)$ time. Compare this with the equation-solving approach, which takes $\mathrm{O}(mn^2 + n^3)$ time (the usual algorithms for solving a system of n linear equations, such as LU decomposition, contribute the $\mathrm{O}(n^3)$ term). Actually, since n (number of vectors) is always assumed to be less than m (dimensionality of vectors) in our application to maintain linear independence, the latter approach takes $\mathrm{O}(mn^2)$ time in general.

In our experiments, projection gave matching times of the order of 100 milliseconds, compared to 5 seconds for equation-solving. Section 5 gives more details.

3.4 Principal Component Analysis and Eigenvectors

In practice, the transformation to an orthonormal basis is not performed on the sample vectors themselves, but on the *deviations* of the sample vectors from their mean. This complication is introduced by **Principal Component Analysis** (PCA), which has the benefit of reducing the size of the sample space by discarding "less significant" elements from the basis. PCA transforms a set of linearly independent vectors $\{\mathbf{X}_i,\ i = 1 \ldots n\}$ as follows:

1. Let $\overline{\mathbf{X}}$ be the mean of the $\mathbf{X}_i$'s.

2. Consider the set of vectors $\{\mathbf{X}'_i \mid \mathbf{X}'_i = \mathbf{X}_i - \overline{\mathbf{X}}\}$. This set has rank $n-1$, since
$$\sum_{i=1}^{n} \mathbf{X}'_i = \sum_{i=1}^{n} (\mathbf{X}_i - \overline{\mathbf{X}}) = \sum_{i=1}^{n} \mathbf{X}_i - n\frac{\sum_{i=1}^{n} \mathbf{X}_i}{n} = 0$$
3. PCA gives an orthonormal basis $\{\mathbf{E}_i\}$ of size $n-1$ for the space spanned by the set $\{\mathbf{X}'_i\}$. PCA additionally reduces the size of the basis by discarding "less significant" elements, but we will ignore this in our discussion for the moment. The $\mathbf{E}_i$'s are conventionally called *eigenvectors.*

After constructing the basis, we match a novel vector $\mathbf{X}_{novel}$ as follows:

1. Compute $\mathbf{X}'_{novel} = \mathbf{X}_{novel} - \overline{\mathbf{X}}$
2. Project $\mathbf{X}'_{novel}$ onto each eigenvector $\mathbf{E}_i$, obtaining scalars $e_i = \mathbf{E}_i.\mathbf{X}_{novel}$. Note that all eigenvectors are of unit length, since the basis is orthonormal.
3. Approximate $\mathbf{X}_{novel}$ as $\overline{\mathbf{X}} + \sum_{i=1}^{n-1} e_i\mathbf{E}_i$.

We observe that $\sum_{i=1}^{n-1} e_i\mathbf{E}_i$ is equivalent to $\sum_{i=1}^{n} k_i\mathbf{X}'_i$ for some k_i's, since the sets $\{\mathbf{E}_i\}$ and $\{\mathbf{X}'_i\}$ span the same space. So our approximation can be rewritten as
$$\mathbf{X}_{approx} = \overline{\mathbf{X}} + \sum_{i=1}^{n} k_i\mathbf{X}'_i$$

Lemma 2. *The set of all possible approximations of the above form is*
$$\left\{\sum_{i=1}^{n} b_i\mathbf{X}_i,\ \sum b_i = 1\right\}$$

Proof.
$$\begin{array}{rcl}
\mathbf{X}_{approx} & = & \overline{\mathbf{X}} + \sum_{i=1}^{n} k_i\mathbf{X}'_i \\
 & = & \frac{\sum_{i=1}^{n}\mathbf{X}_i}{n} + \sum_{i=1}^{n} k_i(\mathbf{X}_i - \frac{\sum_{j=1}^{n}\mathbf{X}_j}{n}) \\
 & = & \sum_{i=1}^{n}(k_i + \frac{1}{n} - \frac{\sum_{j=1}^{n}k_j}{n})\mathbf{X}_j \\
 & = & \sum_{i=1}^{n} b_i\mathbf{X}_i
\end{array}$$

where $b_i = k_i + \frac{1}{n} - \frac{\sum_{j=1}^{n}k_j}{n} = k_i + \frac{1}{n} - \overline{k}$.
Now,
$$\begin{array}{rcl}
\sum_{i=1}^{n} b_i & = & \sum_{i=1}^{n}(k_i + \frac{1}{n} - \frac{\sum_{j=1}^{n}k_j}{n}) \\
 & = & \sum_{i=1}^{n} k_i + 1 + n\frac{\sum_{j=1}^{n}k_j}{n} \\
 & = & \sum_{i=1}^{n} k_i + 1 - \sum_{j=1}^{n} k_j \\
 & = & 1
\end{array}$$

We must show that each possible set of b_i's summing to 1 corresponds to some set of k_i's. Let us take any such set of b_i's. Then we can write:

$$\begin{aligned} b_1 &= k_1 + \tfrac{1}{n} - \frac{\sum_{j=1}^{n} k_j}{n} \\ b_2 &= k_2 + \tfrac{1}{n} - \frac{\sum_{j=1}^{n} k_j}{n} \\ \vdots &= \vdots \\ b_n &= k_n + \tfrac{1}{n} - \frac{\sum_{j=1}^{n} k_j}{n} \end{aligned}$$

We must show that this system of equations is consistent, i.e. it has a solution for the k_i's. Let us construct a solution as follows:

Let $\sum_{j=1}^{n} k_j = K$, where K is any arbitrary real number. Solving for the k_i's:

$$\begin{aligned} k_1 &= b_1 - \tfrac{1}{n} + \tfrac{K}{n} \\ k_2 &= b_2 - \tfrac{1}{n} + \tfrac{K}{n} \\ \vdots &= \vdots \\ k_n &= b_n - \tfrac{1}{n} + \tfrac{K}{n} \end{aligned}$$

Adding all the equations and observing that $K = \sum_{j=1}^{n} k_j$, we get back our original condition $\sum_{i=1}^{n} b_i = 1$. Hence the system is consistent and the solution is valid. We note that our choice of b_i's corresponds to an infinite number of possible choices for the k_i's, differing in the sum $\sum_{j=1}^{n} k_j$.

□

Theorem 1. *Approximation by projection onto eigenvectors is equivalent to minimizing the* L_2 *distance between a novel vector* $\mathbf{X}_{novel}$ *and the vector space* $V_1 = \{\sum_{i=1}^{n} b_i \mathbf{X}_i,\ \sum b_i = 1\}$.

Proof. In the procedure outlined previously, we first construct $\mathbf{X}'_{novel} = \mathbf{X}_{novel} - \overline{\mathbf{X}}$, then project $\mathbf{X}'_{novel}$ onto each of the eigenvectors to obtain coefficients, then take the linear combination of the eigenvectors weighted by these coefficients, and finally add the mean $\overline{\mathbf{X}}$ to obtain the approximation. So the approximation may be written as:

$$\mathbf{X}_{approx} = \overline{\mathbf{X}} + \sum_{i=1}^{n-1} (\mathbf{X}'_{novel}.\mathbf{E}_i)\mathbf{E}_i$$

We note, from Lemma 2, that the space of possible approximations is precisely V_1.

Let $\mathbf{X}'_{approx} = \sum_{i=1}^{n-1} (\mathbf{X}'_{novel}.\mathbf{E}_i)\mathbf{E}_i$. From Lemma 1, $\mathbf{X}'_{approx}$ is the point in the space V' spanned by $\{\mathbf{E}_i\}$ that minimises the L_2 distance to $\mathbf{X}'_{novel}$.

Now, we observe that

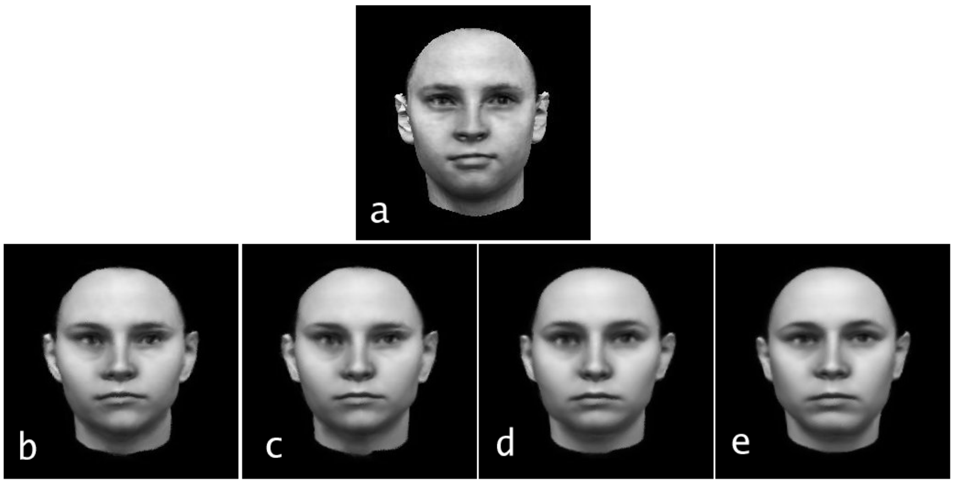

Figure 3: Matches to the novel image (a) obtained by retaining b) all 49, c) 30, d) 15 and e) 7 eigenvectors.

$$\begin{aligned}\|\mathbf{X}_{novel} - \mathbf{X}_{approx}\|^2 &= \|\mathbf{X}_{novel} - (\overline{\mathbf{X}} + \mathbf{X}'_{approx})\|^2 \\ &= \|(\mathbf{X}_{novel} - \overline{\mathbf{X}}) - \mathbf{X}'_{approx}\|^2 \\ &= \|\mathbf{X}'_{novel} - \mathbf{X}'_{approx}\|^2\end{aligned}$$

So minimizing the L_2 distance between $\mathbf{X}'_{novel}$ and $\mathbf{X}'_{approx}$ corresponds to minimizing the L_2 distance between $\mathbf{X}_{novel}$ and $\mathbf{X}_{approx}$.

Also, there is a one-to-one correspondence between V_1 and V':

$$\mathbf{X} \in V' \leftrightarrow (\mathbf{X} + \overline{\mathbf{X}}) \in V_1$$

Hence the minimization is over the whole of V_1.

Therefore $\mathbf{X}_{approx}$, as constructed above, minimizes the L$_2$ distance from V_1 to $\mathbf{X}_{novel}$.

$\square$

We thus conclude that matching by projection onto the PCA space yields exactly the same best matches as quadratic programming, with the constraint that the sum of the coefficients of the sample vectors must be 1. In our experiments, we observed that except in cases where lighting conditions, shape or overall intensity deviated drastically from the images of the sample set, this constraint was not very restrictive and the best matches from the two methods were virtually identical. Further, by retaining only the first few principal components, we reduced the number of projections required, thus speeding up the matching and generating excellent approximations to the matches produced with the full eigenspace. Fig. 3 demonstrates this.

4 Feature-Based Optimizations

The eigenvector-projection approach is fast, but has a significant associated challenge: the separation of the novel image into shape and texture components must be done as accurately as possible. This calls for a robust algorithm to set points in two facial images in correspondence.

The simplest approach is to compute the optical flow between the reference image and the novel image – the flow field is taken as the correspondence field. This approach is conceptually flawed, because optical flow algorithms are designed to track points on the *same* object across multiple frames, not locate corresponding points in images of *different* objects. However, since two faces are superficially similar, this approach works reasonably well in practice.

We experimented with the Bergen-Hingorani algorithm [1] for dense optical flow. It gave acceptable results on the whole, but spurious correspondences were frequently generated. Larger window sizes gave noticeably better results.

More sophisticated algorithms designed specifically for computing facial correspondences exist, but they are usually also more computation-intensive and not suitable for real-time processing.

To develop a suitable algorithm for our application which gave few spurious correspondences and ran in real-time, we took note of the following facts:

- Certain points in facial images are easier to track than others. Finding correspondences for a point at the corner of the mouth where contrast and texture are strong is, for instance, easier than for a point in the middle of the cheek, where the surface is smooth and unbroken.

- Correspondence fields between two facial images are smooth. This is to be expected since human faces have elastic, organic structure. This suggests that the field can be approximated fairly accurately by interpolation from a set of "defining points".

- Human vision is sensitive to strong features. When we observe a face, recognition is triggered more by the intensity variation in sharply-defined regions such as the overall outline, the eyes, the nose and the mouth than in undistinguished ones such as the cheek and forehead. Therefore, in compressing a facial image, it is important to preserve the structure of strong features using accurate correspondences at these locations, but other regions need not be rendered so exactly.

A suitable correspondence algorithm could therefore select a set of easily trackable *feature points*, compute correspondences at these points only, and interpolate from the resulting values to obtain the correspondences at every pixel in the source image. By not tracking weakly-defined points (which resemble their neighbours), spurious correspondences may be reduced.

We implemented this method in the following manner:

Feature selection: We used a Harris-type corner detector [4] to select well-defined feature points in the source image. The Harris detector is sensitive to strong gradients in the image. It ranks pixels on the basis of the invertibility of a particular matrix $\mathbf{G}$ at each pixel (p_x, p_y), defined as follows:

$$\mathbf{G} = \sum_{x=p_x-w_x}^{p_x+w_x} \sum_{y=p_y-w_y}^{p_y+w_y} \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

where I_x and I_y are the gradients in the x and y directions respectively and the sums are computed over a window of size $[-w_x, w_x] \times [-w_y, w_y]$. The invertibility of the matrix is measured by the magnitude of its minimum eigenvalue λ_m – the larger this quantity, the easier it is to invert $\mathbf{G}$. We retain pixels having λ_m greater than a threshold value, indicating that they mark strong gradients and are easy to locate and track. From these pixels, we select only those which have maximum λ_m in their immediate neighbourhoods. Finally, we prune the set further if necessary to ensure all pixels are separated by some minimum threshold distance [3]. Typically, about 300 feature points were selected.

Sparse correspondences: To calculate the correspondences at the feature points, we sampled a dense optical flow field. This field was generated using the Bergen-Hingorani algorithm [1]. Sparse sampling of a dense flow field had the advantage that the smoothing influence of neighbouring pixels on the value at each pixel was retained in the sparse set of flows. This helped prevent spurious correspondences.

Interpolation: We triangulated the set of feature points using Delaunay triangulation. To compute the flow at a pixel $\mathbf{p}$, we determined which Delaunay facet f it lay in. The flow at $\mathbf{p}$ was computed by interpolation from the vertices of f. We used generalized barycentric coordinates as interpolation coefficients: the coefficient w_j corresponding to the jth vertex was calculated as in [7]:

$$w_j = \frac{\cot(\gamma_j) + \cot(\delta_j)}{\|\mathbf{p} - \mathbf{q}_j\|^2}$$

where the vertex $\mathbf{q}_j$ and the angles γ_j and δ_j are as in Fig. 4. In practice, the coefficients at each pixel were computed during model construction and stored in a lookup table. The costs of triangulation,

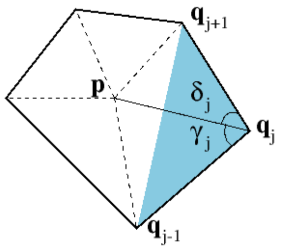

Figure 4: Barycentric coordinates in an irregular convex polygon.

point location and barycentric coordinate computation were therefore not incurred during matching. We developed this approach from original ideas, but discovered later that Kardouchi et al [6] had done similar work.

Fig. 5 illustrates the entire process.

4.1 Features for Speed

Although our feature-based approach was initially designed to generate good correspondences, we observed that it could also be used to drastically speed up the matching process. The shape components were completely defined by their values at the feature points. Hence, we could restrict the shape vectors to these values only, reducing their dimensionality from m, the total number of pixels in each image, to n_f, the number of feature points. In our experiments with 256×256 images and 300 feature points, this implied a reduction from 65536 dimensions to 300.

We compute the PCA space of the restricted shape vectors and match by projection onto this space. The projection-based approach executes in

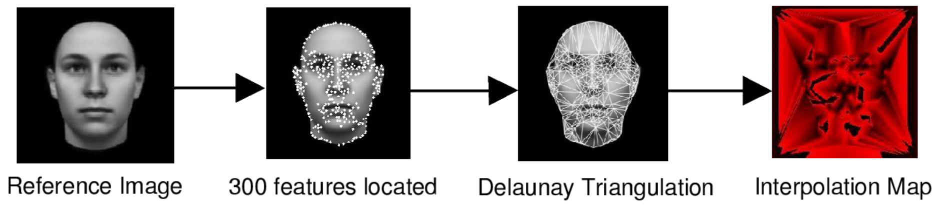

Figure 5: The feature-based interpolation process. For clarity, background triangles are not shown in the Delaunay triangulation. The interpolation map is colour-coded: each feature point is assigned an unique intensity and the intensity at each pixel is obtained by interpolation from the appropriate feature points.

time linear in the dimensionality of the vectors, so execution time is reduced from O(mn) (see Section 3.3) to O($n_f n$). The full (dense) shape component is approximated *after* matching, by interpolating from the best match restricted shape.

Ideally, we would like to show that matching with restricted vectors *first* and interpolating *afterwards* is equivalent to interpolating *first* and matching with full vectors *afterwards*.

4.1.1 Investigating Equivalence

We are given a set X of n linearly independent vectors $\{\mathbf{X}_i,\ i = 1 \dots n\}$ in $\mathbb{R}^m$, $n < m$. We are also given a set Y of "restricted vectors" $\{\mathbf{Y}_i,\ i = 1 \dots n\}$ in $\mathbb{R}^{n_f}$, $n_f < m$. These sets have the following property: for any j:

$$\begin{array}{rcl} X_{1j} & = & \sum_{k=1}^{n_f} c_{jk} Y_{1k} \\ X_{2j} & = & \sum_{k=1}^{n_f} c_{jk} Y_{2k} \\ \vdots & = & \vdots \\ X_{nj} & = & \sum_{k=1}^{n_f} c_{jk} Y_{nk} \end{array}$$

where X_{ij} denotes the jth element of $\mathbf{X}_i$ and Y_{ij} denotes the jth element of $\mathbf{Y}_i$. The c_{jk}'s are a set of scalar *interpolation coefficients* – there is one such set for each j. In simple terms, the set X may be obtained by interpolation from the set Y.

We are now given a novel vector $\mathbf{X}_{novel} \in \mathbb{R}^m$ and the corresponding restricted vector $\mathbf{Y}_{novel} \in \mathbb{R}^{n_f}$, related by the same sets of interpolation coefficients as above: for any j:

$$X_{novel,j} = \sum_{k=1}^{n_f} c_{jk} Y_{novel,k}$$

We obtain two approximations to X_{novel} as follows:

1. Minimize the L$_2$ distance from the space spanned by the elements of X to $\mathbf{X}_{novel}$, obtaining the closest match $\mathbf{X}^1_{approx}$.
2. Minimize the L$_2$ distance from the space spanned by the elements of Y to $\mathbf{Y}_{novel}$, obtaining the closest match $\mathbf{Y}_{approx}$. Now interpolate from $\mathbf{Y}_{approx}$ using the coefficients $\{c_{jk}\}$ to obtain $\mathbf{X}^2_{approx}$.

We ask if and under what conditions $\mathbf{X}^1_{approx}$ and $\mathbf{X}^2_{approx}$ are equal. We are currently working on an answer to this question.

4.1.2 Theoretical Speed Gain

We conclude this section with some explicit calculations for the speed gain from the feature-based approach, compared to the case when full vectors are projected for all components. We assume that features are used to restrict only the two shape components (x and y), not the texture component. We also assume that the platform is serial-processing, not parallel-processing.

As mentioned previously, each shape component restricted to the feature points is projected onto the set of eigenvectors in $\mathrm{O}(n_f n)$ time, compared to $\mathrm{O}(mn)$ for projecting full vectors. This results in a net speed gain by a factor of g:

$$g = \frac{3 \times \mathrm{O}(mn)}{\mathrm{O}(mn) + 2 \times \mathrm{O}(n_f n)}$$

If a sequence of one multiplication and one addition (a unit operation in projection) takes time τ, then the speed gain is:

$$g = \frac{3mn\tau}{mn\tau + 2n_f n\tau} = \frac{3m}{m + 2n_f} = 3 - \frac{6n_f}{m + 2n_f}$$

For practical purposes, $n_f \ll m$, so $g \approx 3$: we expect the feature-based approach to be three times faster in projection. (Matching in PCA space requires us to add the mean, but this is linear in the dimensions of the vectors and does not change g.) The results satisfy our expectations: as noted in Section 5, feature-based projection takes ∼20ms while full-vector projection takes ∼60ms. We exclude the (more or less identical) times taken to compute correspondences in the two approaches.

If the texture vectors could somehow be restricted as well, i.e. if we could predict an entire texture map from values at feature points, then the speedup is $g = m/n_f$, typically of the order of a few hundred.

5 Implementation and Results

We implemented our approach on a dual 2.7 GHz Pentium 4 system with 1GB RAM, running RedHat Linux 9 (Shrike) with the symmetric multi-processing (SMP) kernel 2.4.20-8smp. A `uname -a` command yielded the following:

```
Linux lappc16.epfl.ch 2.4.20-8smp #1 SMP Thu Mar 13 17:45:54 EST
2003 i686 i686 i386 GNU/Linux
```

The coding language was C. `OpenCV-0.9.5` was used as the image-processing library. No explicit multi-processing or threading instructions were included in the source code.

In our experiments, we obtained a minimum average matching time of 120ms. A slightly slower implementation which was easier to time took 140ms, of which 120ms was for computing sparse correspondences (pruning a Bergen-Hingorani optical flow field) and 20ms for projecting feature-based restricted vectors. We quote the times from the latter implementation since we are more confident that the timing was accurate.

We present the execution times for various approaches in tabular form (all measurements on the above platform).

Method	**Approximate average matching time (ms)**
Iterative error-minimization (Conjugate Gradient Descent)	3×10^4
Unconstrained quadratic programming (LU decomposition)	5×10^3
Full-image projection with Black [2] correspondences	5×10^3
Full-image projection with Bergen-Hingorani [1] correspondences	180 (Correspondence: 120 + Projection: 60)
Feature-based projection with pruned Bergen-Hingorani [1] correspondences	140 (Correspondence: 120 + Projection: 20)

Reconstruction typically took 40-80ms.

The reconstructed images in general closely resembled the original novel images. However, there were some spurious deformations in the results that are a cause for concern. It seems a better feature-tracking / correspondence / interpolation scheme is required. Incidentally, we note that interpolation with barycentric coordinates only ensures C^0 continuity, i.e. the interpolation surface is continuous but not necessarily smooth across edges of the subdivision. Spline-based interpolation schemes give C^1 (continuous first derivatives) and higher continuity, and are more suitable.

We present a selection of matches obtained, both good and bad, in Figs. 6, 7, 8 and 9. 50 images were present in the sample set. 40 principal components (out of a possible maximum of 49) were used for both shape and texture. The novel images were rendered from textured 3D models at the Max Planck Institute, Germany.

6 Future Work

We list work that needs to be done, ranked in increasing order of immediate requirement and complexity:

1. Investigate how much of the spurious deformations in reconstructed images are due to the inadequate size of the sample set and how much are due to incorrect correspondences.

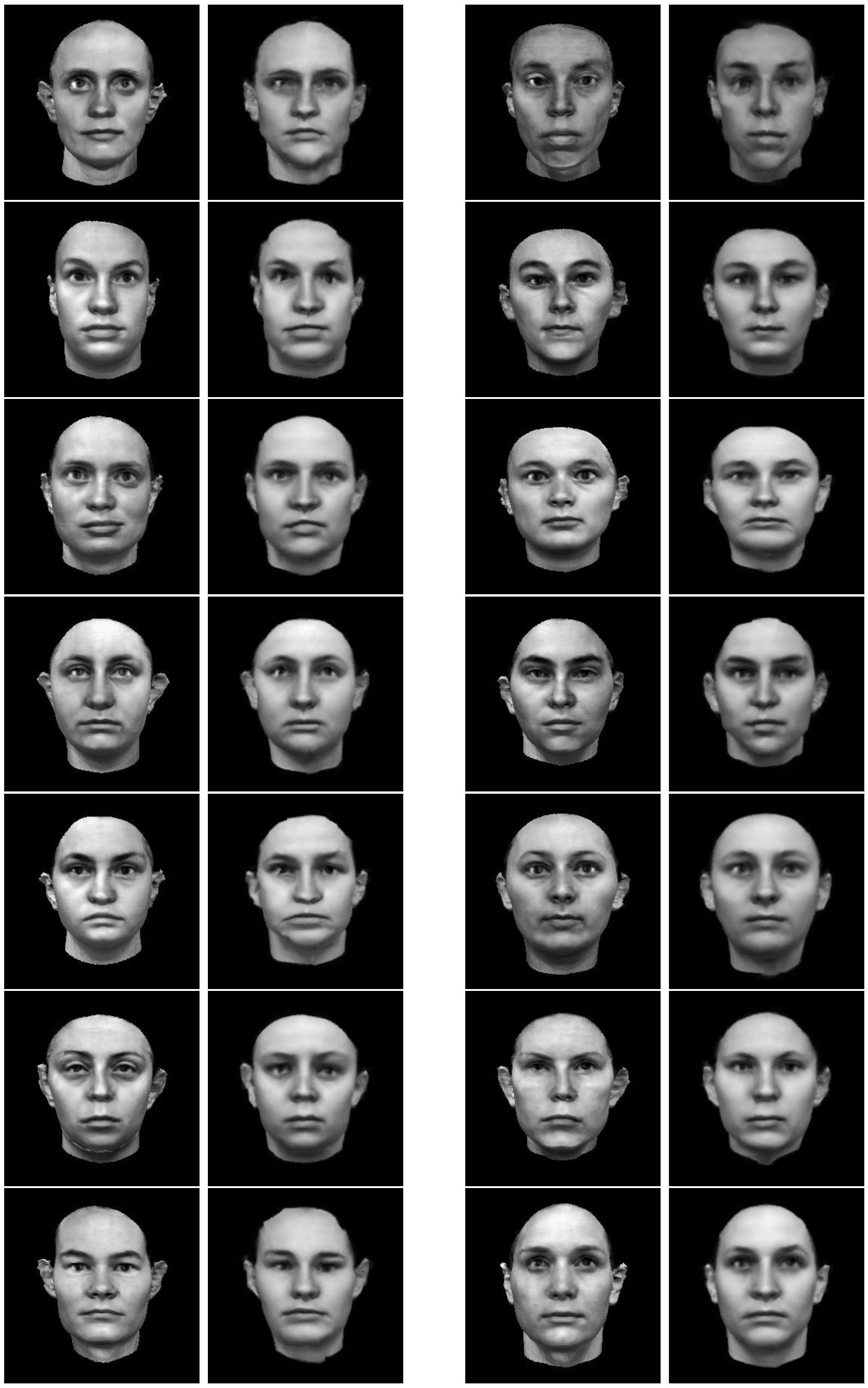

Figure 6: A selection of novel images (first and third columns) and the corresponding feature-based matches (second and fourth columns): some good, some not-so-good. Part 1.

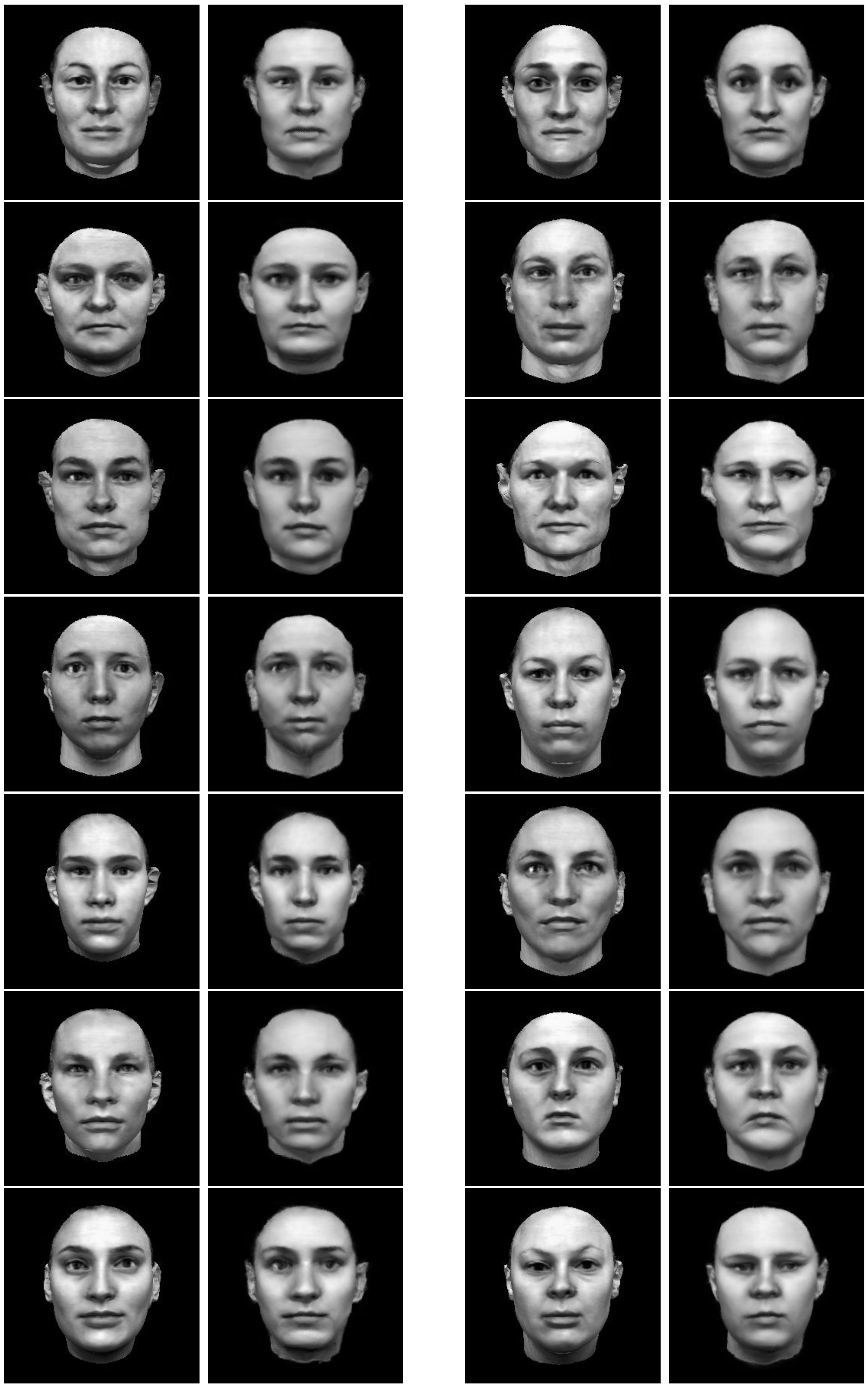

Figure 7: A selection of novel images (first and third columns) and the corresponding feature-based matches (second and fourth columns): some good, some not-so-good. Part 2.

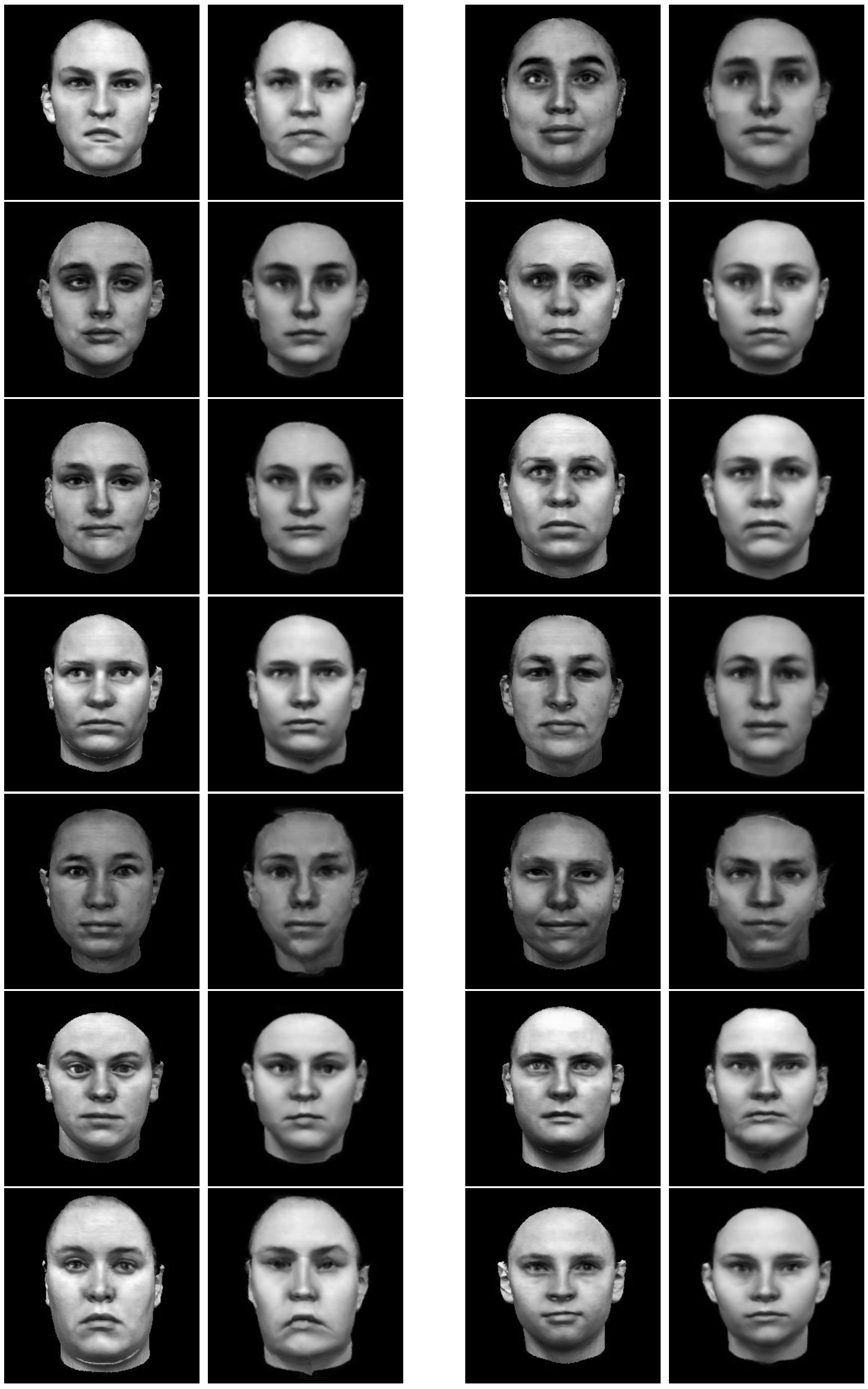

Figure 8: A selection of novel images (first and third columns) and the corresponding feature-based matches (second and fourth columns): some good, some not-so-good. Part 3.

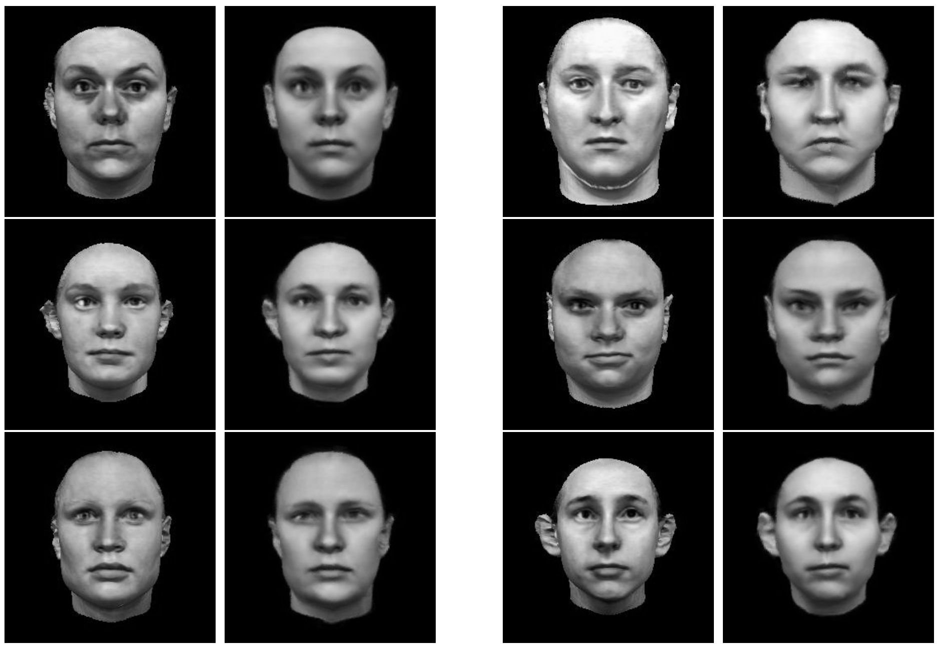
Figure 9: A selection of novel images (first and third columns) and the corresponding feature-based matches (second and fourth columns): some good, some not-so-good. Part 4.

2. Obtain substantial quantitative information about the performance of the algorithm. Measure running times on a variety of facial images, and also measure the accuracy of the approximation using multiple norms.

3. Implement a smoother interpolation scheme, such as a spline-based approach.

4. Implement the algorithm on a parallel-processing platform. The algorithm is highly parallelizable and a large speed increase could be achieved in a multi-processor environment.

5. Investigate whether preprocessing by wavelet decomposition and similar techniques can give smaller and more workable images to increase accuracy and speed.

6. Research, develop and implement a method to extend the feature-based approach to texture components as well. This will make projection a few hundred times more efficient.

7. Research, develop and implement a better correspondence scheme. The performance of the algorithm really hinges on just this step.

Acknowledgements

I am grateful to Dr. Edoardo Charbon and Randhir K. Singh of the EPFL, Switzerland, for their suggestions, encouragement and patience. Randhir wrote the code for matching morphable models by error-minimization which formed the basis for my efforts, and provided help and advice at crucial stages. He is also responsible for the idea of using wavelet decompositions for better matching, a work-in-progress. I also thank the people at the Processor Architecture Laboratory, EPFL, who tolerated me for two-and-a-half months, and the countless people in the scientific community whose ideas and publications influenced my work.

References

[1] J. Bergen and R. Hingorani, "Hierarchical Motion-Based Frame Rate Conversion," Technical Report, David Sarnoff Research Center, Princeton, 1990.

[2] M. J. Black, "Robust Incremental Optical Flow," Ph.D. Thesis, Yale, 1992.

[3] J.-Y. Bouget, "Pyramidal Implementation of the Lucas Kanade Feature Tracker: Description of the Algorithm,", Microprocessor Research Labs, Intel Corp., 2000.

[4] C. Harris and M. Stephens, "A combined corner and edge detector," *Proc. Alvey Vision Conf.*, pp.147-151, 1988.

[5] M. J. Jones and T. Poggio, "Model-Based Matching by Linear Combinations of Prototypes," unpublished AI memo, MIT, 1996.

[6] M. Kardouchi, J. Konrad and C. Vázquez, "Estimation of large-amplitude motion and disparity fields: Application to intermediate view reconstruction," *Proc. Visual Communications and Image Processing, IS&T/SPIE Symp. on Elec. Imaging*, San Jose, 2000.

[7] M. Meyer, Haeyoung L., A. Barr, M. Desbrun, "Generalized Barycentric Coordinates on Irregular Polygons,", Journal of Graphics Tools, 7(1):pp.13-22, 2002.

[8] T. Vetter and N. F. Troje, "Separation of texture and shape in images of faces for image coding and synthesis," *JOSA*, A 14:9 pp.2152-2161, 1997.