

3.2 Representation in terms of eigenfaces.

In the previous section we obtained a basis for the space spanned by the faces in our training set. It turned out that the dimension of this space is very small in comparison with that of the whole image space. Our immediate task is to represent the faces of the training set in terms of this new basis. Writing,

$$\mathbf{X}_k = y_{1k}\mathbf{u}_1 + y_{2k}\mathbf{u}_2 + \cdots + y_{Lk}\mathbf{u}_L \quad (3.8)$$

we need to find the expansion coefficients y_{jk} . If we let $U = \begin{pmatrix} \mathbf{u}_1 & \mathbf{u}_2 & \cdots & \mathbf{u}_L \end{pmatrix}$ and

$$\mathbf{y}_k = \begin{pmatrix} y_{1k} \\ y_{2k} \\ \vdots \\ y_{Lk} \end{pmatrix}$$

(3.8) can be rewritten as

$$\mathbf{X}_k = U\mathbf{y}_k. \quad (3.9)$$

Since the eigenfaces are orthonormal it follows that

$$\mathbf{y}_k = U^T \mathbf{X}_k. \quad (3.10)$$

Conversely, given the expansion coefficients one easily reconstructs the original image from,

$$\mathbf{I}_k = \|\mathbf{I}_k - A\|U\mathbf{y}_k + A \quad (3.11)$$

This mean of course that we need to save the normalization factor, $\|\mathbf{I}_k - A\|$. Although normalization is important to derive the eigenfaces, it can be ignored once the eigenfaces have been obtained.

For faces in the training set, the representation in terms of the eigenfaces is exact (assuming of course that we kept enough eigenfaces). This is not strictly true

of faces noting the training set. Faces not in the training are projected onto the eigenfaces, i.e. we compute the best eigenface representation in a L^2 norm sense. Provided that the training set is sufficiently representative of the faces that one is likely to encounter, the projection error should be small. (If not, we need to include the face into the training; we will discuss an efficient way of doing this in section ??) In any event, given an arbitrary face \mathbf{J} , its eigenface representation coefficients are calculated from,

$$\mathbf{y}_J = U^T(\mathbf{J} - \mathbf{A}) \quad (3.12)$$

and the reconstructed image is given by

$$\tilde{\mathbf{J}} = U\mathbf{y}_J + \mathbf{A}. \quad (3.13)$$

Let us now use formulas (3.12) and (3.13) to project and then reconstruct two images onto the eigenfaces. Of course, it is perfectly possible to project an arbitrary image onto the eigenfaces. Let us see what happens if we project a very nonface-like image as in Figure 3.6.



Figure 3.6: A rose is not face!

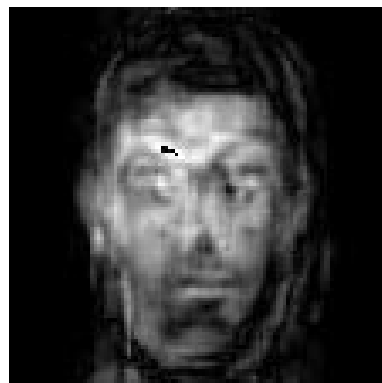


Figure 3.7: And a face is not a rose!.

If we project the rose onto the eigenfaces we get a very face-like rose as shown in Figure 3.7.

What happens in this case is that the most rose-like face is being reconstructed from the eigenfaces. In this case the projection error is large.

The next image is of a person not in the training set.

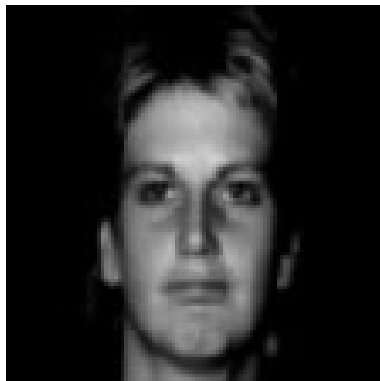


Figure 3.8: An image not in the training set.

We reconstruct this face using 10 and 40 eigenfaces as shown in Figure 3.9.



Figure 3.9: The reconstruction of Figure 3.8 using 10 and 40 eigenfaces.

Although the reconstruction is poor when only 10 eigenfaces are used, there is no discernible difference in the case of 40 eigenfaces. This confirmed by Figure 3.10 which shows the difference between the original face and the reconstruction using 40 eigenfaces.

Notice how the error is concentrated around the edges in the original image. This

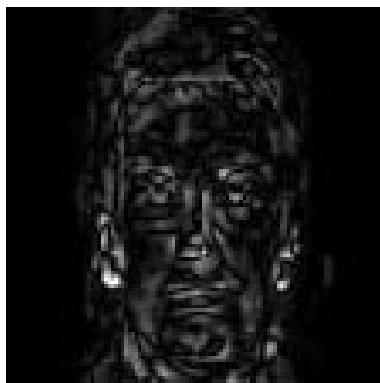


Figure 3.10: The error in the reconstruction of Figure 3.8, using 40 eigenfaces.

is a very desirable feature since the eye is not sensitive for small errors surrounding the edges. In general purpose lossy image compression algorithms such as JPEG, this type of behavior is positively encouraged by the designers of the software; they are very aware of the fact that errors along sharp edges in images are not easily detected by the eye and lead to better compression ratios.

We now turn to the question how to compare different faces, i.e. how does the computer recognize a face. The short answer is: By comparing their eigengace expansion coefficients. A more detailed answer is provided in the next section.