

OPTIMAL FACE RECONSTRUCTION USING TRAINING

D. Darian Muresan and Thomas W. Parks

School of Electrical and Computer Engineering, Cornell University
Ithaca, NY 14853
darian, parks@ece.cornell.edu

ABSTRACT

In previous work [2] we considered the problem of image interpolation from an adaptive optimal recovery point of view. We showed how a training set S determines a quadratic signal class and how to use this signal class to perform image interpolation. In [2] the training set S was taken from the low resolution version of the image we were interpolating. In this paper we continue our discussion of the method presented in [2] by looking more closely at the training set S . In particular, we will show how a training set of high resolution images can give very good interpolation results through the use of the method [2].

1. INTRODUCTION

The problem of image interpolation seems to depend heavily on the sampling rate of the decimated image. However, if we had a badly decimated image of a face, the knowledge of the fact that we are interpolating a face should help the reconstruction algorithm. In this paper we review the optimal recovery interpolation algorithm presented in [2] and show how the use of a good training set can generate much more superior results than if we only had the low resolution image. In particular, we will apply optimal recovery together with a training set to interpolate very low resolution faces. Other work in the area of image interpolation using training sets is that of [3] and [4].

2. OPTIMAL RECOVERY INTERPOLATION

We now review the interpolation method presented in [2]. Locally, at location y , we model the image as belonging to a certain ellipsoidal signal class K

$$K = \{x \in R^n : x^T Q x \leq \epsilon\} \quad (1)$$

where Q is derived from a training set or may be assumed known. Vector x is any subset of the image containing the missing pixel y . Vector x is chosen such that any L linear

functionals ($F_i, i = 1, \dots, L$) of x are assumed known. If we note the actual values of the functionals by f_i we have $F_i(x) = f_i$. In this paper we assume that the functionals are based on derivatives and/or actual pixel values of the decimated image. The optimal estimates for the missing pixels are the pixels of the optimal image \bar{u} . From [7], the optimal image \bar{u} belonging to class K and satisfying the given functionals F_i is a linear combination of the representors ϕ_i of the given functionals F_i

$$\bar{u} = \sum_{i=1}^L \alpha_i \phi_i \quad (2)$$

The representors, using the least representors theorem [6] are used to represent the linear functionals as Q inner products. This is the central role of Q in optimal recovery [1]. The challenge is then to design a Q for which the class K is a proper representation of the local signal. Assume that a given training set S is to be used for learning our matrix Q . For convenience, we further assume that S is a matrix with the training vectors as columns. These training signals lie in a particular region of the space which we assume to be representative of the signal we are trying to interpolate. Our aim is to find a Q such that the ellipsoid

$$x^T Q x = \text{constant}$$

is representative of the training set S as shown in Fig. 1. What does this mean? We want a Q for which the contour of $x^T Q x = c$ models the locus of the training set S . Intuitively, the contour $x^T Q x = c$ is an ellipsoid stretched in the direction of the eigenvectors of Q . The stretch is largest in the direction where Q has the smallest eigenvalues and smallest in the direction where Q has the largest eigenvalues. In [2] we showed that $Q = (SS^T)^{-1}$.

3. THE TRAINING SET

Selecting the correct training set S is key. In [2], when we dealt with generic images and all we had was the decimated image, the training set S came from the lower resolution

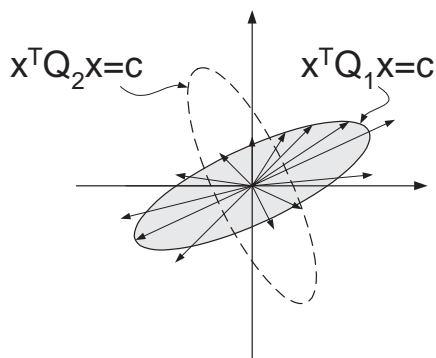


Fig. 1. Training set S and contours corresponding to $x^T \mathbf{Q}_1 x = c$ and $x^T \mathbf{Q}_2 x = c$. The training set S is better represented by the contour corresponding to $x^T \mathbf{Q}_1 x = c$ than the one corresponding to $x^T \mathbf{Q}_2 x = c$.

decimated image. When we deal with the specific problem of face reconstruction, we can select our training set from a database of high resolution faces. By using a set of high resolution faces, we hope that our algorithm will learn how to deal with an eye, a mouth, nose and other features of the human face. Then, when presenting our algorithm with a new decimated face, we hope that around the eyes, nose and mouth, the things which are specific to every human face, the decimated face can be reconstructed much better.

Next, what should the training vectors be? Should they be the entire high resolution images, or only patches of them. Using entire face images as training vectors is appealing since this would generate a single \mathbf{Q} matrix. The drawback is memory. Using 96×128 training images would generate a \mathbf{Q}^{-1} matrix with $(96 \times 128)^2 = 150,994,944$ elements. This is obviously unacceptable. The alternate solution is to break up the high resolution image into patches and perform interpolation separately for each patch. If we break the 96×128 into 16×16 patches, there will be 48 patches and for each patch the matrix \mathbf{Q}^{-1} will have only 256 elements. Even if we keep all 48 \mathbf{Q}^{-1} matrices, the memory requirements are much better. Breaking up the image into smaller blocks should also be fine considering the fact that pixels are more correlated locally. In other words, training on the entire image at once will not be any more advantageous, and may even be disadvantageous, considering that pixels around one area of the face will not be too correlated with pixels from a different area of the face. For example, we do not expect the nose pixels to tell us much about the eye or hair pixels.

Experimentally, we found that if we simply break up the image into blocks the results tend to be blocky. For example, we don't want to have a block edge in the middle of an eye. To solve this problem, instead of breaking up the image into non-overlapping blocks we allow some overlap in the

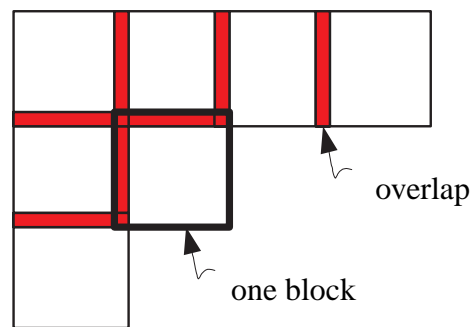


Fig. 2. Full image with the block overlap of the training sets.

neighboring blocks, as shown in Fig. 2. The overlapping regions can then be averaged out or not. Experimentally, both methods yield good results.

4. EXPERIMENTAL RESULTS

For our training set we used a subset of the images used in [3]. These images are cropped versions of the FERET dataset [5] and are 96×128 pixels. All images were aligned by hand marking the location of 3 points, the centers of the two eyes and the lower tip of the nose. (These images can be found at <http://dsplab.ece.cornell.edu>.) Each 96×128 image was sectioned into 16×16 overlapping blocks and we generated a \mathbf{Q} matrix for each of the 16×16 blocks. Unlike other training methods where patches need to be compared against patches from the training set, once the \mathbf{Q} matrices are formed, we no longer need the training data. In some sense, the \mathbf{Q} matrices now contain the information of how pixels relate to each other. This is advantageous for several reasons

1. Memory space. After the \mathbf{Q} matrices are formed, the memory space required for the training data is no longer necessary.
2. Computational speed. If patches need to be compared against patches from the training set, we can easily see that for very large data sets this can be very time consuming. On the other hand, our \mathbf{Q} matrices will always be the same size, regardless of the training set (the size of \mathbf{Q} is only dependent on the patch size). Once we have the \mathbf{Q} matrices, the estimation of the missing samples will always take a fixed amount of time, regardless of the training set size.

We applied our interpolation algorithm to four different faces. For the training set we used 300 images. The faces we used for testing were not part of the training set. In Fig. 3 the 96×128 faces were first decimated by four to 24×32 and then interpolated back. Similarly, in Fig. 4 the $96 \times$

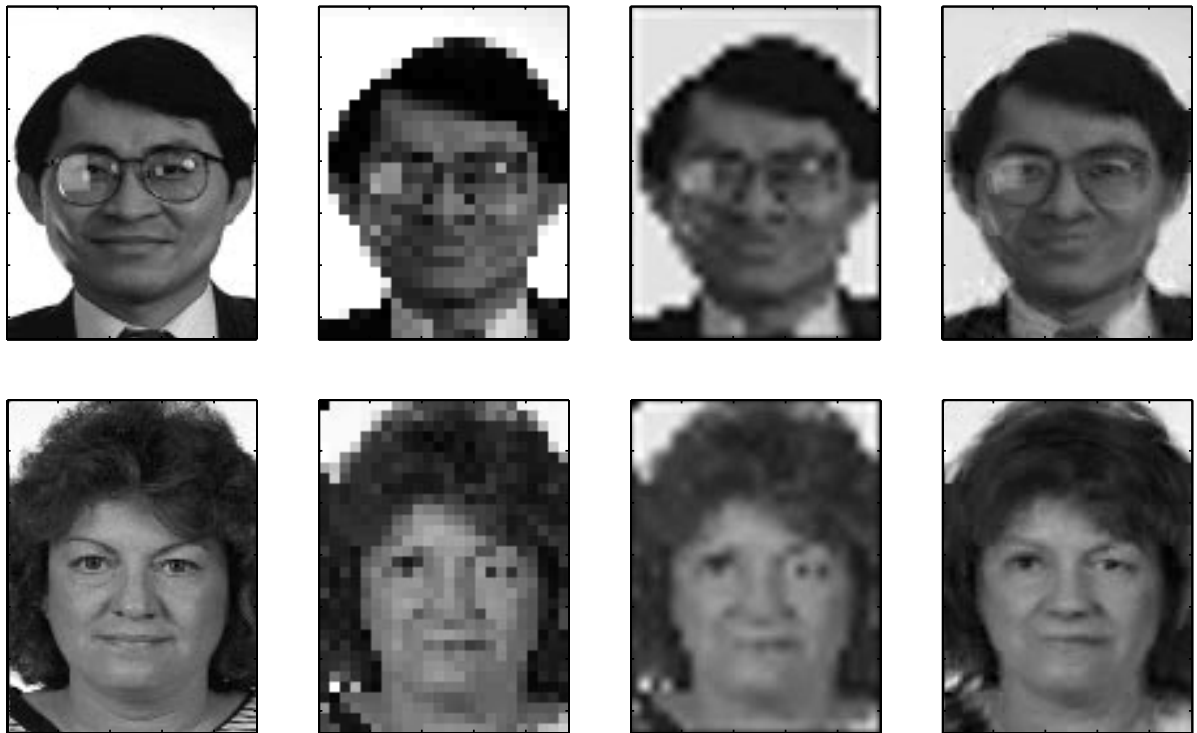


Fig. 3. $4\times$ interpolation (from left to right): original, pixel replication, cubic interpolation and optimal recovery.



Fig. 4. $8\times$ interpolation (from left to right): original, pixel replication, cubic interpolation and optimal recovery.

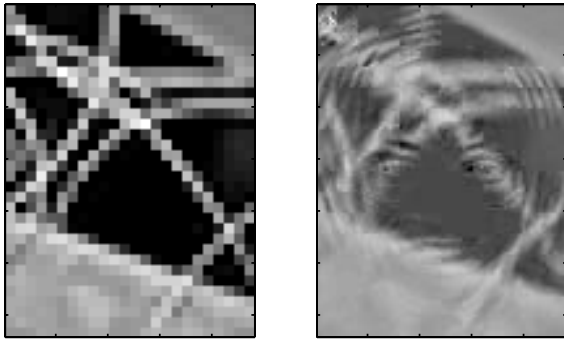


Fig. 5. $4\times$ interpolation. On the left is pixel replication and on the right is optimal recovery using a face training set. Notice how the algorithm tries to generate a face even though there is no face.

128 faces were first decimated by eight to 12×16 and then interpolated back. In both figures, from left to right, we have: the original image, pixel replication, cubic and our optimal recovery interpolation. A few things to notice:

1. The optimal recovery approach generates better results than cubic and pixel replication.
2. Notice how the interpolation algorithm learned the local behavior of the face. Around the eyes, pixels are ordered in the shape of the eye. In Fig. 3 the light reflection on the right lens of the man's glasses has been elongated along the eye-brow, instead of remaining a round reflection. On the boundary of the head, things are interpolated in a circular direction. The woman's hair in Fig. 3 is not as curly as in the original image. Rather, is arranged in a circular direction around the head.

An interesting question to ask is what would happen if we applied our interpolation, with the Q obtained from the face training set, to an image that does not contain a face? If the interpolation algorithm really learned the behavior of the face features, we would expect that the algorithm will try to generate a face, regardless of the input image.

In Fig. 5 we applied the interpolation algorithm to an image containing no face. As expected, the algorithm tried to generate a face even when there was none. This behavior is similar to the behavior observed in [3], which the authors called *hallucinating faces*, although their interpolation algorithm is quite different from ours.

5. CONCLUSION

In [2] we introduced a new method for image interpolation, which required the use of a training set to learn a quadratic

signal class. When dealing with generic images it is hard to generate a good training set and in [2] we opted for using information from the decimated image as our training data. However, under certain, more restrictive assumptions training data can be generated from other high resolution images. In this paper we looked at the specific case of face reconstruction.

Our work was motivated by the work of [3], where they also looked at the specific problem of face interpolation from a training set. We recently obtained software from [3] and [4]. While we did not have time to include any comparisons in this paper, we will do so at the conference.

Finally, the face database and the Matlab code presented in this paper can be found at <http://dsplab.ece.cornell.edu>.

6. ACKNOWLEDGMENTS

We wish to thank Simon Baker for providing us with the 96-by-128 face images database and for allowing us to share the image database. Moreover, we would also like to thank Simon Baker and Bill Freeman for providing us with software of their working algorithms.

7. REFERENCES

- [1] D. Darian Muresan, "Review of Optimal Recovery," *Cornell DSP Lab Technical Report TR-2002-10*, <http://dsplab.ece.cornell.edu>, 2002.
- [2] D. Darian Muresan and Thomas W. Parks, "Optimal Recovery Approach to Image Interpolation," *ICIP 2001, Greece*, 2001.
- [3] S. Baker and T. Kanade, "Hallucinating Faces," *Proceedings of the Fourth International Conference on Automatic Face and Gesture Recognition*, Grenoble, France, 2000.
- [4] W. T. Freeman, Thouis R. Jones, and Egon C. Pasztor. "Example-Based Super-Resolution," *MERL - Mitsubishi Electronic Research Laboratory, TR-2001-30* August 2001.
- [5] P.J. Philips, H. Moon, P. Rauss, and S.A. Rizvi. "The FERET evaluation methodology for face-recognition algorithms." *In CVPR 97*, 1997.
- [6] Walter Rudin. "Principles of Mathematical Analysis," *McGraw-Hill, Inc.* 1976.
- [7] M. Golomb and H. F. Weinberger, "Optimal approximation and Error Bounds," *On Numerical Approximation*, R. E. Langer ed., The University of Wisconsin Press, Maddison, pp. 117-190, 1959.