Volterrafaces: Discriminant Analysis using Volterra Kernels *

Ritwik Kumar, Arunava Banerjee, Baba C. Vemuri
Department of Computer and Information Science and Engineering, University of Florida
{rkkumar,arunava,vemuri}@cise.ufl.edu

Abstract

In this paper we present a novel face classification system where we represent face images as a spatial arrangement of image patches, and seek a smooth non-linear functional mapping for the corresponding patches such that in the range space, patches of the same face are close to one another, while patches from different faces are far apart, in L_2 sense. We accomplish this using Volterra kernels, which can generate successively better approximations to any smooth non-linear functional. During learning, for each set of corresponding patches we recover a Volterra kernel by minimizing a goodness functional defined over the range space of the sought functional. We show that for our definition of the goodness functional, which minimizes the ratio between intra-class distances and inter-class distances, the problem of generating Volterra approximations, to any order, can be posed as a generalized eigenvalue problem. During testing, each patch from the test image that is classified independently, casts a vote towards image classification and the class with the maximum votes is chosen as the winner. We demonstrate the effectiveness of the proposed technique in recognizing faces by extensive experiments on Yale, CMU PIE and Extended Yale B benchmark face datasets and show that our technique consistently outperforms the state-of-the-art in learning based face discrimination.

1. Prologue

World events, specially in the last decade, have lead to an increased interest in the field of biometrics based person identification. Face recognition in particular, has attracted prolific research in the computer vision and pattern recognition community. Even though impressive strides have been made towards providing an ultimate solution to this problem, significant and interesting problems remain.

If we try to organize the epitome of literature present in this field, a dichotomy of approaches emerges. The first class of these tries to capture the physical processes of image formation under various scene parameter variations like illumination (e.g. Generic ABRDF [3]), pose (e.g. Morphable Models[5]), expression(e.g. Geometry-Texture [16]) etc. In contrast, the second class of approaches invokes mathematical and statistical tools to capture the structure of the oft-invisible relations among the numbers that make up the face images. These techniques explore the intrinsic data geometry assuming images to be either vectors (e.g. Eigenfaces [17], Fisherfaces [4], Laplacianfaces [13], orthogonal Laplacianfaces (OLAP) [7], Locality Preserving Projections [11], Kernel Locality Preserving Projections with Side Information (KLPPSI) [2], MLASSO [18], Kernel Ridge Regression (KRR) [1]), or higher dimensional tensors (e.g. Tensor Subspace Analysis [12], 2-Dimensional Linear Discriminant Analysis [24], Orthogonal Rank One Tensor Projection (ORO) [14], Tensor Average Neighborhood Margin Maximization (TANMM) [23], Correlation Tensor Analysis (CTA) [10], Spectral Regression [6], Regularized Discriminant Analysis [6], Smooth LDA [8]).

A major advantage of the techniques in the first class comes from their being generative in nature. This property allows these methods to accomplish tasks like face relighting (e.g. [3]) or novel pose generation or complete 3D image reconstruction (e.g. [5]) in addition to recognition. At the same time, methods in the first class tend to demand more side information from the data as compared to the second class of methods (e.g. [3] requires illumination direction for the training set, [5] requires facial feature points for initialization etc). The second class of methods are in a sense more versatile as they can be seamlessly applied to a variety of different image sets without any significant requirement of side information.

The method that we propose in this paper loosely falls into the second category of techniques. We seek a mapping of face image patches such that in the range space, discrimination among different classes is easier. We choose Volterra kernels to accomplish this because it allows us to systematically build progressively better approximations to such a mapping. Furthermore, Volterra kernels can be learnt in a data driven fashion which relieves us from being predis-

*This work was supported in part by the UF Alumni Fellowship to RK and NIH grant NS46812 to BCV.

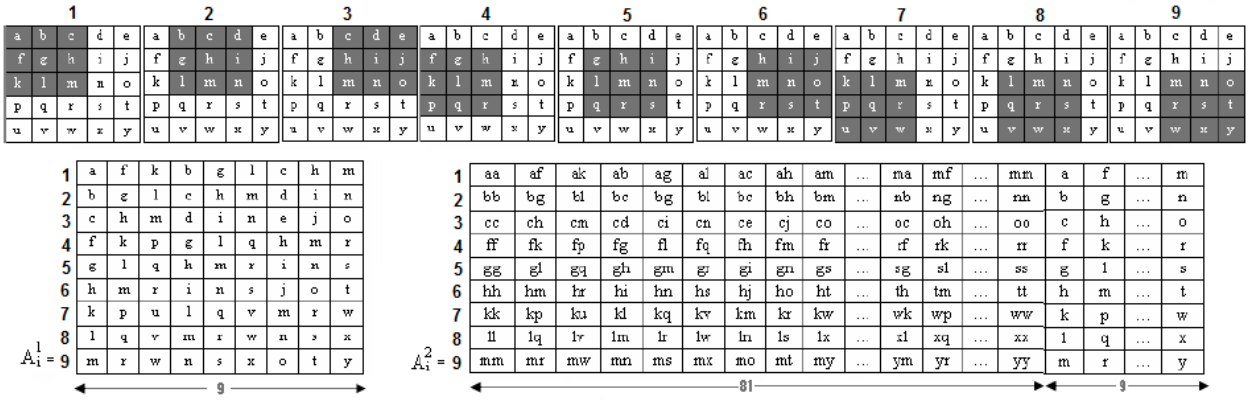

Figure 1. Structure of A_i^1 and A_i^2 for an image of size 5×5 and kernel of size 3×3. In the first row, 9 neighborhoods of the image I_i are highlighted. For first order approximation, each of of these neighborhoods become a row in A_i^1. For the second order case, we take all the second order combinations of pixel values in each neighborhood and use them as the first 81 (b^4) elements of a row in A_i^2. The remaining 9 (b^2) elements are simply the pixel values. Rows are numbered to show which neighborhood they correspond to.

posed towards any fixed kernel form (e.g. Gaussian, Radial Basis Function etc). The face images in the range space are called Volterrafaces in this paper.

2. Volterra Kernel Approximations

From signal processing theory we know that a linear translation invariant (LTI) functional $\Im : \mathbf{H} \rightarrow \mathbf{H}$, which maps the function $x(t)$ to the function $y(t)$, can be completely described by a function $h(t)$ as

$$\Im(x(t)) = y(t) = x(t) \otimes h(t) = \int_{-\infty}^{\infty} h(\tau)x(t-\tau)d\tau. \quad (1)$$

Volterra theory generalizes this concept and states that any non-linear translation invariant functional $\aleph : \mathbf{H} \rightarrow \mathbf{H}$, which maps the function $x(t)$ to the function $y(t)$, can be described by a sequence of functions $h_n(\cdot)$ as

$$\Im(x(t)) = y(t) = \sum_{n=1}^{\infty} y_n(t) \quad (2)$$

where $y_n(t) =$

$$\int_{-\infty}^{\infty} \cdots \int_{-\infty}^{\infty} h_n(\tau_1, \ldots, \tau_n)x(t-\tau_1) \ldots x(t-\tau_n)d\tau_1 \cdots d\tau_n \quad (3)$$

Here $h_n(\tau_1, \ldots, \tau_n)$ are called the Volterra Kernels of the functional. It must be noted that the above equation can be seamlessly generalized to 2 dimensional functions, $I(u, v)$, which for instance, can be images. It should be noted that eq. (1) is just a special case of the more general eq. (3) if the first order terms are the only ones taken into account.

Since we are interested in computing using this theory, we would be using the following discrete form of eq. (3).

$y_n(m) =$

$$\sum_{q_1=-\infty}^{\infty} \cdots \sum_{q_n=-\infty}^{\infty} h_n(q_1, \ldots, q_n)x(m-q_1) \ldots x(m-q_n). \quad (4)$$

The infinite series form in eq. (4) does not lend itself well for practical implementations. Further, for a given application, only the first few terms may give the desired approximation of the functional. Thus, we need a truncated form of the Volterra series, which is denoted in this paper by

$$\Im^p(x(m)) = \sum_{n=1}^{p} y_n(m) = x(m) \otimes_p h(m) \quad (5)$$

where p denotes the maximal order of the terms taken into account for the approximation. Note that in this truncated Volterra series representation, $h(m)$ is a placeholder for all the different orders of the kernels.

In general, given a set of input functions $\mathbf{I}$, we are interested in finding a functional $\aleph$, such that $\aleph(\mathbf{I})$ has some desired property. This desired property can be captured by defining a *goodness* functional on the range space of $\aleph$. In cases when the explicit equation relating the input set $\mathbf{I}$ to $\aleph(\mathbf{I})$ is known, various techniques like the harmonic input method, direct expansion etc. ([9]) can be used to compute kernels of the unknown functional. In the absence of such an explicit relation, we propose that the Volterra kernels be learnt from the data using the *goodness* functional. The translation invariance property of the Volterra kernels ensures that if the images are translated by a fixed amount in the domain, the mapped images are also translated by the same amount, and hence the Volterra kernel mapping is stable.

In this framework, the problem of pattern classification can be posed as follows. Given a set of input data $\mathbf{I} = \{g_i\}$

where $i = 1 \dots N$, a set of classes $\mathbf{C} = \{c_k\}$ where $k = 1 \dots K$, and a mapping which associates each g_i to a class c_k, find a functional such that in the range space, the data $\aleph(\mathbf{I})$ is easily classifiable. Here the *goodness* functional could be a measure of the separability of classes in the range space. Once the Volterra kernels have been determined, a new data point can be classified using the learnt functional. $\aleph(\mathbf{I})$ can be approximated to an appropriate accuracy based on computational efficiency and the classification accuracy constraints.

3. Kernel computation as Generalized Eigenvalue problem

For the specific task of image classification, we define the problem as follows. Given a set of input images (2D functions) $\mathbf{I}$, a training set, where each image belongs to a particular class $\mathbf{c_k} \in \mathbf{C}$, compute the Volterra kernels for the unknown functional $\mathcal{N}$ which map the images in such a manner that the *goodness* functional O is minimized in the range space of $\mathcal{N}$. Functional O measures the departure from the complete separability of the data in the range space. In this paper we seek a functional $\mathcal{N}$ that maps all the images from the same class in a manner such that the intraclass L_2 distance is minimized while the interclass L_2 distance is maximized. Once $\mathcal{N}$ has been determined, a new image can be classified using any of the methods like the Nearest Centroid Classifier, Nearest Neighbor Classifiers etc. in the mapped space. With this observation, we define the *goodness* functional O as,

$$O(\mathbf{I}) = \frac{\sum_{\mathbf{c_k} \in \mathbf{C}} \sum_{i,j \in \mathbf{c_k}} \|\mathcal{N}(I_i) - \mathcal{N}(I_j)\|^2}{\sum_{\mathbf{c_k} \in \mathbf{C}} \sum_{m \in \mathbf{c_k}, n \notin \mathbf{c_k}} \|\mathcal{N}(I_m) - \mathcal{N}(I_n)\|^2} \quad (6)$$

where the numerator measures the aggregate intraclass distance for all the classes and the denominator measures the aggregate distance of class $\mathbf{c_k}$ from all other classes in $\mathbf{C}$. Equation (6) can be further expanded as

$$O_k(\mathbf{I}) = \frac{\sum_{\mathbf{c_k} \in \mathbf{C}} \sum_{i,j \in \mathbf{c_k}} \|I_i \otimes_p K - I_j \otimes_p K\|^2}{\sum_{\mathbf{c_k} \in \mathbf{C}} \sum_{m \in \mathbf{c_k}, n \notin \mathbf{c_k}} \|I_n \otimes_p K - I_m \otimes_p K\|^2} \quad (7)$$

where K, like $h(t)$ in eq. (5), is a placeholder for all the different orders of the convolution kernels.

At this juncture we make the linear nature of convolution explicit by converting the convolution operation to multiplication. This conversion to an explicit linear transformational form can be done in many ways, but as the convolution kernel is the unknown in our setup, we wish to keep it as a vector and thus we transform the image I_i into a new representation A_i^p such that

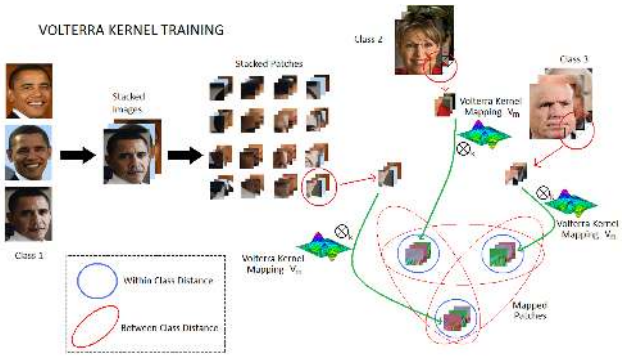

Figure 2. Training images from each class are stacked up and divided into equal sized patches. Corresponding patches from each class are then used to learn Volterra kernels by minimizing intraclass distance over interclass distance. We end up with one Volerra kernel per group of spatially corresponding patches.

$$I_i \otimes_p K = A_i^p \cdot \overline{K} \quad (8)$$

where $\overline{K}$ is the vectorized form of the 2D masks represented by K.

The exact form of A_i^p depends on the order of the convolutions p. In Section 5 we have presented results for up to the second order approximations and thus the structure of A_i^p is explained for only up to second order, but it should be noted that the recognition framework using volterra kernels that we propose is very general and the structure of A_i^p for any order can be analogously derived.

3.1. First Order Approximation

For an image I_i of size $m \times n$ pixels and a first order kernel K_1 of size $b \times b$, the transformed matrix A_i^p has dimensions $mn \times b^2$. It is built by taking neighborhoods of $b \times b$ dimensions at each pixel in I_i, vectorizing and then stacking them one on top of the other. This procedure is illustrated for an image of size 5×5 and kernel of size 3×3 in Figure 1. Border pixels can be ignored or taken into account during convolution by padding the image with zeros without affecting the performance significantly.

Substituting the above defined representation for convolution in eq. (7), we obtain

$$O(\mathbf{I}) = \frac{\sum_{\mathbf{c_k} \in \mathbf{C}} \sum_{i,j \in \mathbf{c_k}} \|A_i^p \cdot \overline{K}_1 - A_j^p \cdot \overline{K}_1\|^2}{\sum_{\mathbf{c_k} \in \mathbf{C}} \sum_{m \in \mathbf{c_k}, n \notin \mathbf{c_k}} \|A_n^p \cdot \overline{K}_1 - A_i^m \cdot \overline{K}_1\|^2}. \quad (9)$$

This can be written as

$$O(\mathbf{I}) = \frac{\overline{K}_1^T \mathbf{S_W} \overline{K}_1}{\overline{K}_1^T \mathbf{S_B} \overline{K}_1} \quad (10)$$

where $\mathbf{S_W} = \sum_{\mathbf{c_k} \in \mathbf{C}} \sum_{i,j \in \mathbf{c_k}} (A_i^p - A_j^p)^T (A_i^p - A_j^p)$ and $\mathbf{S_B} = \sum_{\mathbf{c_k} \in \mathbf{C}} \sum_{m \in \mathbf{c_k}, n \notin \mathbf{c_k}} (A_i^p - A_j^p)^T (A_i^p - A_j^p)$.

Here $\mathbf{S_W}$ and $\mathbf{S_B}$ are symmetric matrices of dimensions $b^2 \times b^2$. Seeking the minimum of eq. (10) leads to solving the generalized eigenvalue problem and thus the minimum of $O(\mathbf{I})$ is given by the minimum eigenvalue of ${\mathbf{S_B}}^{-1}\mathbf{S_W}$ and it is attained when $\overline{K}_1$ equals the corresponding eigenvector.

3.2. Second Order Approximation

The second order approximation of the sought functional contains two terms

$$y(m) = \sum_{q_1=-\infty}^{\infty} h_1(q_1)x(m-q_1) + \sum_{q_1=-\infty}^{\infty}\sum_{q_2=-\infty}^{\infty} h_2(q_1,q_2)x(m-q_1)x(m-q_2) \quad (11)$$

The first term in eq. (11) corresponds to a weighted sum of the first order terms, $x(m-q_1)$, while the second term corresponds to a weighted sum of the second order terms, $x(m-q_1)x(m-q_2)$. For an image I_i of size $m \times n$ pixels and kernels of size $b \times b$, the transformed matrix A_i^2 for the second order approximation in eq. (8) has dimensions $mn \times (b^4+b^2)$ and the kernel vector that multiplies it, $\overline{K}_2$, has dimensions $(b^4+b^2) \times 1$. A_i^2 is built by taking a neighborhood of size $b \times b$ at each pixel in I_i, generating all second degree combinations from the neighborhood, vectorizing them, concatenating the first degree terms and then stacking them one on top of the other. $\overline{K}_2$ is formed by concatenating vectorized second and first order kernels. The structure of A_i^2 for a 5×5 image and 3×3 kernels is illustrated in Figure 1. It must noted that the problem is still linear in the variables being solved for and in fact *by use of this formulation we have ensured that regardless of the order of the approximation, the problem is linear in the coefficients of the Volterra convolution kernels*.

With this definition of A_i^2 we proceed like the first order approximation to obtain analogous equations (9) and (10) with the difference being that the matrices $\mathbf{S_B}$ and $\mathbf{S_W}$ now have dimensions $(b^4+b^2) \times (b^4+b^2)$. Here we must point out an important modification to the structure of A_i^2 which allows us to reduce the size of the matrices. The second order convolution kernels in the Volterra series are required to be symmetrical ([9]) and this symmetry also manifests itself into the structure of A_i^2. By allowing only unique entries in A_i^2 we can reduce the dimensions of A_i^2 to $mn \times \frac{b^4+3b^2}{2}$ and the dimensions of the matrices $\mathbf{S_B}$ and $\mathbf{S_W}$ to $\frac{b^4+3b^2}{2} \times \frac{b^4+3b^2}{2}$. Now as in the first order approximation, the minimum of $O_k(\mathbf{I})$ is given by the minimum eigenvalue of ${\mathbf{S_B}}^{-1}\mathbf{S_W}$, which it is attained when $\overline{K}_2$ equals the corresponding eigenvector.

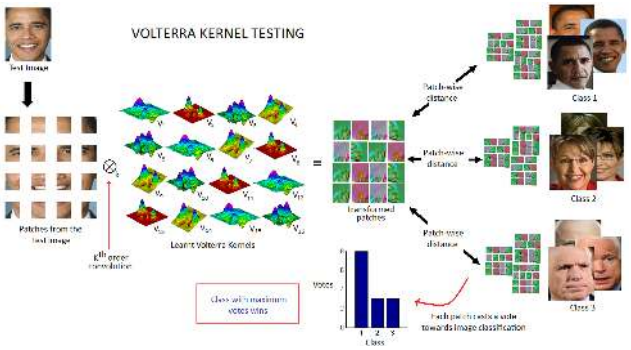

Figure 3. Testing involves dividing the test image according to the scheme used while training. Each patch is then mapped to the range space by the corresponding Volterra kernel where it is classified using a Nearest Neighbor classifier. After being individually classified, each patch from the test image casts a vote towards the parent image classification. The class with the maximum votes wins.

4. Training and Testing Algorithms

The mapping developed in the previous section can be made more expressive if the image is divided into smaller patches and each patch is allowed to be independently mapped to its own discriminative space. This allows us to infer separate kernels for different regions of the face. Further, if each constituent patch casts a vote towards the parent image classification, it introduces robustness to the overall image classification. Thus we adopt a patch based framework for the face recognition task. Note that we do not ignore or select certain patches, as explored in [22].

Training (Fig. 2) in the proposed framework involves learning a volterra kernel from the corresponding patches of the training images. Testing (Fig. 3) in our framework involves two stages. The first stage classifies each patch using the mapping framework described in the previous section and a nearest neighbor classifier. In the second stage, each patch casts a vote towards the parent image classification and the class with maximum votes wins. In the case of a tie, we classify the image using a run-off among the tied classes. As the number of cases with ties are small, a simpler strategy of using a coin toss to break the tie also showed comparable results.

If the kernel size is $k \times k$, the number of patches is p of size $s \times s$, the training set size is t and the number of classes is c, the training complexity is $O(pt^2c^2s^6 + pk^6)$. For testing 1 image, the complexity is $O(pct)$. Since the kernel size is a small constant (3 or 5), the eigenvalue problem can be assumed to take a constant time & thus the training complexity becomes $O(pt^2c^2s^6)$.

Method	Yale A	CMU PIE	Ext. Yale B
2008 CVPR, An *et al*. (KLPPSI) [2]	-	5 10 20 30	5 10 20 30
2008 CVPR, Pham *et al*. (MLASSO) [18]	-	2 3 4	2 3 4
2008 CVPR, Shan *et al*. (UVF) [20]	2 - 8	-	-
2008 TIP, Fu *et al*. (CTA) [10]	-	5 10 15 20	5 10 20 30
2008 TIP, Fu *et al*. (*Lap*,*Eig*,*Fis*) [10]	-	-	5 10 20 30
2007 CVPR, An *et al*. (KRR) [1]	-	5 10 20 30	5 10 20 30
2007 CVPR, Hua *et al*. (ORO) [14]	5	30	20
2007 CVPR, Cai *et al*. (S-LDA) [8]	2 3 4 5	-	-
2007 CVPR, Wang *et al*. (TANMM) [23]	2 3 4	5 10 20	-
2007 ICCV, Cai *et al*. (SR, RDA) [6]	-	30 40	10 20 30 40
2006 TIP, Cai *et al*. (OLAP) [7]	2 3 4 5	5 10 20 30	-
2006 TIP, Cai *et al*. (*Lap*,*Eig*,*Fis*) [7]	2 3 4 5	5 10 20 30	-

Table 1. State-of-the-art methods with which we compare our technique, along with the training set sizes used in their experiments.

4.1. Parameter Selection

Like any other learning algorithm, there are parameters that need to be set before using this method. But unlike most state-of-the-art algorithms where parameters are chosen from a large discrete or continuously varying domain (e.g. initialization and λ in [18], parameter $\alpha \in [0, 1]$ in [8], dimensionality D in [20], reduced dimensionality in TSA [12], 2D-LDA [24] , energy threshold in [19] etc.), Volterra discriminant analysis has a smaller discrete set of parameter choices. We use one of the widely used ([8],[1]) and accepted methods, cross validation, for parameter selection.

Foremost is the selection of the patch size, and for this, starting with the whole face image we define a quad-tree of sub-images. We progressively go down the tree stopping at the level beyond which there is no improvement in the recognition rates as computed using cross validation. Empirically we found that a patch size of 8×8 pixels provides the best results in all cases. Next, we allow patches to be overlapping or non-overlapping. The Volterra kernel size can be of size 3×3 or 5×5 pixels (anything bigger than this severely over-fits a patch of size 8×8). Lastly, the order of the kernel can be quadratic or linear.

In this paper we have presented results for both quadratic and linear kernels. The rest of the parameters were set using a 20-fold leave-one-out cross validation on the training set. It can be noted from the results presented in the next section that the best parameter configuration is fairly consistent not only within a particular database but also across databases.

5. Experiments

In order to evaluate our technique and compare it with existing state-of-the-art methods in learning based face recognition, we have identified 11 recent (Table 1) publications which present the best results on the benchmark databases using very similar (to that in [7]) experimental setups. All of these are embedding methods mentioned in the prologue with the exception of [20] which builds on the concept of Universal Visual Features (UVF). In our study, we present results on Yale A, CMU PIE and Extended Yale B benchmark face databases partly because they are some of the most popular databases which makes a comparative study easy. In addition to the recent methods we also provide comparisons with the traditional baseline methods - Eigenfaces, Fisherfaces and Laplacianfaces. Table. 1 lists these methods along with the number of training images used by them on the above mentioned databases in their experiments. We have presented our results for the whole range of training set sizes so that comparisons with a maximum number of techniques can be made.

For the Yale A [1] face database we used 11 images each of the 15 individuals with a total of 165 images. For the CMU PIE database ([21]) we used all 170 images (except a few corrupted images) from 5 near frontal poses (C05,C07,C09,C27,C29) for each of the 68 subjects. For the Extended Yale B database ([15]) we used 64 (except for a few corrupted images) images each (frontal pose) of 38 individuals present in the database. Note that the methods in Table. 1 used the same subset of images. We obtained the data from the website of the authors of [8] [2]. These images were manually aligned (two eyes were aligned at the same position) and cropped to extract faces, with 256 gray value levels per pixel.

The results (average recognition error rates) on the Yale A, the CMU PIE and the Extended Yale B databases are presented in Table. 1(a), Table. 1(b) and Table. 1(c), respectively. Rows titled Train Set Size indicate the number of training images used and the rows below them list the rates reported by various state-of-the-art and our method (Volterrafaces). Each experiment was repeated 10 times for 10 random choices of the training set. All images other than the training set were used for testing. Specific experimental setup used for Volterrafaces is mentioned below each table. We have reported results with both linear and quadratic masks for the sake of completeness. Best results for a particular training set size are highlighted in bold.

5.1. Discussion

It can be noted that our proposed method (Volterrafaces) consistently outperforms the state-of-the-art and traditional methods on all the three benchmark datasets. On all the three databases, linear masks outperform the quadratic masks but in most cases quadratic masks also provided better performance than the existing methods. For the competing methods we have reported the error rates as mentioned in the original publications (listed next to the method names in the tables). Since we want to present only the best results reported by the competing methods, some of the entries in the tables are left empty because no results for those training set sizes were reported in the original publications.

[1] http://cvc.yale.edu/projects/yalefaces/yalefaces.html

[2] http://www.cs.uiuc.edu/homes/dengcai2/Data/data.html

(a) **Yale A**

Train Set Size	2	3	4	5
S-LDA [8]	42.4	27.7	22.2	18.3
S-LDA [8] (updated)	37.5	25.5	19.3	14.7
UVF [20]	27.11	17.38	11.71	8.16
TANMM [23]	44.69	29.57	18.44	-
OLAP [7]	44.3	29.9	22.7	17.9
Eigenfaces [7]	56.5	51.1	47.8	45.2
Fisherfaces [7]	54.3	35.5	27.3	22.5
Laplacianfaces [7]	43.5	31.5	25.4	21.7
Volterrafaces (Linear)	**15.70**	**12.33**	**9.47**	**6.11**
Volterrafaces (Quad)	**22.15**	**13.36**	15.78	10.19

Train Set Size	6	7	8	9
S-LDA [8] (updated)	12.3	10.3	8.7	-
UVF [20]	6.27	5.07	3.82	-
Volterrafaces (Linear)	**5.78**	**3.96**	**2.61**	**1.43**
Volterrafaces (Quad)	10.04	9.66	9.49	8.74

(b) **CMU PIE**

Train Set Size	5	10	20	30
KLPPSI [2]	27.88	12.32	5.48	3.62
KRR [1]	26.4	13.1	5.97	4.02
ORO [14]	-	-	-	6.4
TANMM [23]	26.98	17.22	5.68	-
SR [6]	-	-	-	6.1
OLAP [7]	21.4	11.4	6.51	4.83
Eigenfaces [7]	69.9	55.7	38.1	27.9
Fisherfaces [7]	31.5	22.4	15.4	7.77
Laplacianfaces [7]	30.8	21.1	14.1	7.13
Volterrafaces (Linear)	**20.26**	**10.24**	**4.94**	**2.85**
Volterrafaces (Quad)	25.29	11.94	**5.45**	4.60

Train Set Size	2	3	4	40
SR [6]	-	-	-	5.2
MLASSO [18]	54.0	43.0	34.0	-
Volterrafaces (Linear)	**43.0**	**36.30**	**23.98**	**2.37**
Volterrafaces (Quad)	**50.48**	**39.66**	**32.67**	**3.04**

(c) **Extended Yale B**

Train Set Size	5	10	20	30
ORO [14]	-	-	-	9.0
SR [6]	-	12.0	4.7	2.0
RDA [6]	-	11.6	4.2	1.8
KLPPSI [2]	24.74	9.93	3.15	1.39
KRR [1]	23.9	11.04	3.67	1.43
CTA [10]	16.99	7.60	4.96	2.94
Eigenfaces [10]	54.73	36.06	31.22	27.71
Fisherfaces [10]	37.56	18.91	16.87	14.94
Laplacianfaces [10]	34.08	18.03	30.26	20.20
Volterrafaces (Linear)	**6.35**	**2.67**	**0.90**	**0.42**
Volterrafaces (Quad)	**13.0**	**3.98**	**1.27**	**0.58**

Train Set Size	2	3	4	40
MLASSO [18]	58.0	54.0	50.0	-
SR [6]	-	-	-	1.0
RDA [6]	-	-	-	0.9
Volterrafaces (Linear)	**26.23**	**18.23**	**9.33**	**0.34**
Volterrafaces (Quad)	**40.81**	**20.47**	**14.42**	**0.43**

Table 2. **Yale A** Training set size: 2-9, Linear kernel size: 5×5, Quadratic kernel size: 3×3, overlapping patches size: 8×8, images size: 64×64. **CMU PIE**: Training set size: 2-9, Linear kernel size: 5×5, Quadratic kernel size: 3×3, overlapping patches size: 8×8, images size: 32×32. **Extended Yale B**, Training set size: 2-5, 5, 10, 20, 30 & 40 Linear kernel size: 3×3, Quadratic kernel size: 3×3, non-overlapping patches size: 8×8, images size: 32×32. For other methods, best results as reported in the respective papers are used.

6. Conclusion

We have introduced the use of Volterra kernel approximations for image recognition functionals in this paper. The kernel learning is driven by the training data, based on a *goodness* functional defined in the range space of the recognition functional. It is shown that for a *goodness* functional that tries to minimize intraclass distances while maximizing interclass distances, the kernel computation reduces to the generalized eigenvalue problem which translates to a very efficient computation of kernels for any order of approximation of the functional. Effectiveness of this technique for face recognition is demonstrated by experiments on three benchmark databases and the results are compared to traditional as well as the state of the art techniques in discriminant analysis for faces. From the results presented in this paper it can be concluded that Volterra kernel approximations show great promise for applications in image recognition tasks.

References

[1] S. An, W. Liu, and S. Venkatesh. Face recognition using kernel ridge regression. In *CVPR*, 2007.

[2] S. An, W. Liu, and S. Venkatesh. Exploiting side information in locality preserving projection. In *CVPR*, 2008.

[3] A. Barmpoutis, R. Kumar, B. C. Vemuri, and A. Banerjee. Beyond the lambertian assumption: A generative model for apparent brdf fields of faces using anti-symmetric tensor splines. *CVPR*, pages 1–6, 24 - 26 June 2008.

[4] P. N. Belhumeur, J. Hespanha, and D. J. Kriegman. Eigenfaces vs. fisherfaces: Recognition using class specific linear projection. *PAMI*, 19(7):711–720, 1997.

[5] V. Blanz and T. Vetter. Face recognition based on fitting a 3d morphable model. *PAMI*, 25(9):1063–1074, 2003.

[6] D. Cai, X. He, and J. Han. Spectral regression for efficient regularized subspace learning. In *ICCV*, 2007.

[7] D. Cai, X. He, J. Han, and H. J.Zhang. Orthogonal laplacianfaces for face recognition. *TIP*, 15(11), 2006.

[8] D. Cai, X. He, Y. Hu, J. Han, and T. Huang. Learning a spatially smooth subspace for face recognition. In *CVPR*, 2007.

[9] J. A. Cherry. Introduction to volterra methods. *Distortion Analysis of Weakly Nonlinear Filters Using Volterra Series*, 1994.

[10] Y. Fu and T. S. Huang. Image classification using correlation tensor analysis. *IEEE TIP*, 17(2), 2008.

[11] X. He, D. Cai, and P. Niyogi. Locality preserving projections. In *NIPS*, 2003.

[12] X. He, D. Cai, and P. Niyogi. Tensor subspace analysis. In *NIPS*, 2005.

[13] X. He, S. Yan, Y. Hu, P. Niyogi, and H. Zhang. Face recognition using laplacianfaces. *IEEE PAMI*, 27(3), 2005.

[14] G. Hua, P. Viola, and S. Drucker. Face recognition using discriminatively trained orthogonal rank one tensor projections. In *CVPR*, 2007.

[15] K. Lee, J. Ho, and D. J. Kriegman. Acquiring linear subspaces for face recognition under variable lighting. *PAMI*, 27(5):684–698, 2005.

[16] X. Li, G. Mori, and H. Zhang. Expression-invariant face recognition with expression classification. *Third Canadian Conference on Computer and Robot Vision*, 2006.

[17] A. Pentland, B. Moghaddam, and T. Starner. View-based and modular eigenfaces for face recognition. *CVPR*, 1994.

[18] D.-S. Pham and S. Venkatesh. Robust learning of discriminative projection for multicategory classification on the stiefel manifold. In *CVPR*, 2008.

[19] S. Rana, W. Liu, M. Lazarescu, and S. Venkatesh. Recognising faces in unseen modes: a tensor based approach. In *CVPR*, 2008.

[20] H. Shan and G. W. Cottrell. Looking around the backyard helps to recognize faces and digits. In *CVPR*, 2008.

[21] T. Sim and T. Kanade. Combining models and exemplars for face recognition: An illuminating example. In *CVPR Workshop on Models versus Exemplars in Comp. Vision 2001*.

[22] Y. Su, S. Shan, X. Chen, and W. Gao. Patch-based gabor fisher classifier for face recognition. In *ICPR*, 2006.

[23] F. Wang and C. Zhang. Feature extraction by maximizing the average neighborhood margin. In *CVPR*, 2007.

[24] J. Ye, R. Janardan, and Q. Li. Two-dimensional linear discriminant analysis. In *NIPS*, 2004.