Trainable Convolution Filters and their Application to Face Recognition

Ritwik Kumar, *Member, IEEE,* Arunava Banerjee, *Member, IEEE,*
Baba C. Vemuri, *Fellow, IEEE,* and Hanspeter Pfister, *Senior Member, IEEE*

Abstract—In this paper we present a novel image classification system that is built around a core of trainable filter ensemble that we call Volterra kernel classifiers. Our system treats images as a collection of possibly overlapping patches and is composed of three components: (a) A scheme for single patch classification that seeks a smooth, possibly non-linear, functional mapping of the patches into a range space, where patches of the same class are close to one another, while patches from different classes are far apart – in the L_2 sense. This mapping is accomplished using trainable convolution filters (or Volterra kernels) where the convolution kernel can be of any shape or order; (b) Given a corpus of Volterra classifiers with various kernel orders and shapes for each patch, a boosting scheme for automatically selecting the best weighted combination of the classifiers to achieve higher per-patch classification rate; (c) A scheme for aggregating the classification information obtained for each patch via voting for the parent image classification. We demonstrate the effectiveness of the proposed technique using face recognition as an application area and provide extensive experiments on Yale, CMU PIE, Extended Yale B, Multi-PIE and MERL Dome benchmark face datasets. We call the Volterra kernel classifiers applied to face recognition Volterrafaces. We show that our technique, which falls into the broad class of embedding-based face image discrimination methods, consistently outperforms various state-of-the-art methods in the same category.

Index Terms—Face Recognition, Convolution, Filtering Classifier, Volterra Kernels, Fisher's Linear Discriminant, Boosting

1 Introduction

We present a novel image classification system that is built around a core of trainable filter ensemble that we call Volterra kernel classifiers. Each Volterra classifier is a filter characterized by a 2D arrangement of real numbers – its convolution kernel. Our image classification system has a three tiered architecture. In the first tier, images are divided into patches and multiple Volterra classifiers with various kernel shapes are learned for each patch location. Given a corpus of Volterra classifiers at each patch location, at the second tier, an aggregate classifier is learned using a boosting scheme. Finally, at the third tier, a label for a given image is picked via plurality voting by all patches. The first two tiers of our system require training while the third tier, which is unsupervised, is only active during testing.

Volterra classifiers use the convolution operation as their foundation. Convolution, perhaps the most important operation used for image analysis, has been applied extensively in the field of computer vision and machine learning. In the domain of image classification, convolution has traditionally been employed at the feature extraction stage, which is typically followed by a discriminator like a Support Vector Machine [12], Fisher's Discriminator [13], Boosting [14] etc. Examples of convolution features include Haar features [38], Gabor features [31], Leung-Malik (LM) features [30] among others. Volterra classifiers depart significantly from these *fixed-feature* traditional approaches in that the convolution filters used are not fixed apriori, rather are learned in a data-driven fashion in conjunction with the discriminator. To elaborate, Volterra classifiers try to find the best filter that groups elements of the same class together and drives elements of different classes apart. Furthermore, the shape of the convolution kernel need not be a 2D matrix in our system, we can enforce any arbitrary 2D shape for the kernel.

Another significant distinguishing characteristics of Volterra classifiers is that, unlike traditional methods, Volterra classifiers are not limited to linear filters like Haar, Gabor, LM etc. Our formulation for Volterra classifiers allows for the use of non-linear higher-order generalizations of convolution provided by the Volterra Theory [39], [11]. This also explains why our classifiers are called Volterra classifiers. We show that the use of non-linear convolution classifiers leads to stronger aggregated classifiers (tier two of our image classification system) than otherwise.

Volterra classifiers also differ from the so-called *kernel based* methods in machine learning. Traditional kernel based classifiers like Kernel SVMs [12], Kernel LDA [13] etc. may possibly allow for non-linear transformation of data, but the form of transformation is explicitly fixed e.g. Gaussian Kernel, Radial Basis Function Kernel etc. Volterra classifiers learn their transformational element, the convolution filters, in a data driven fashion, as part of the discrimination process. Note that although there are kernel methods that use data derived kernel matrices, like Pyramid Matching Kernel (PMK) based SVM, nevertheless the form of the inner product (multiscale histogram intersection in the case of PMK) is fixed explicitly before the discrimination stage.

The second tier of our image classification system, where multiple Volterra classifiers are aggregated to build a stronger classifier, must be contrasted with Multiple Kernel Learning (MKL) methods. In a typical MKL system, a weighted combination of multiple kernels is used in an SVM. The weights of the kernels in MKL are obtained in the data driven fashion,

Digital Object Indentifier 10.1109/TPAMI.2011.225

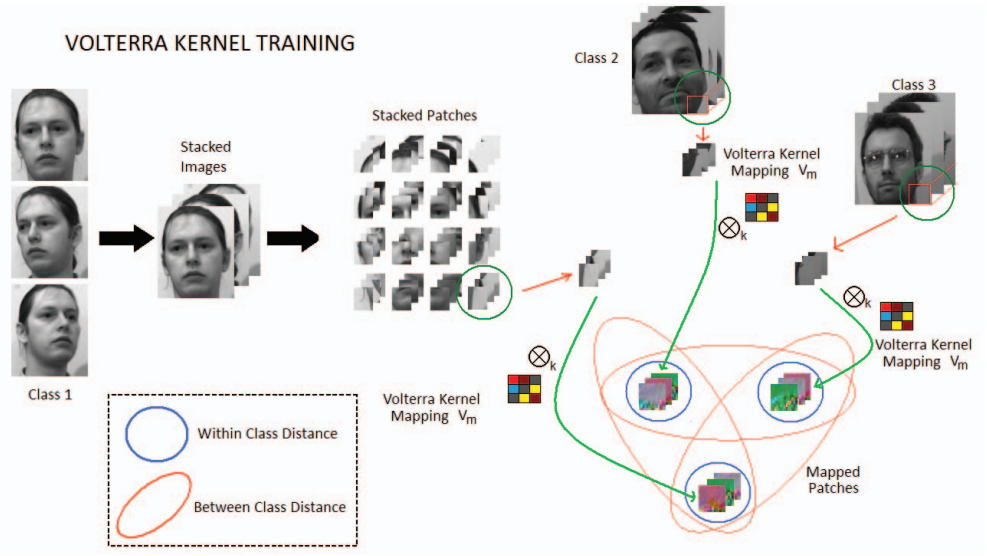

Fig. 1: Training images from each class are stacked up and divided into equal sized patches. Corresponding patches from each class are then used to learn higher order convolutional Volterra kernel by minimizing intraclass distance and maximizing interclass distance. We end up with one Volterra kernel per group of spatially corresponding patches. The size and the order of the kernel is held constant for a given training process. Note that the color images are only used for illustration, so far our implementation works with grayscale images.

as are the weights in our aggregation scheme. The distinction between the two methods lies in the origin of the kernels. In MKL, the kernels themselves are fixed apriori while in Volterra classifiers the convolution kernels are formed such that they are most discriminative for the given data.

The use of voting by patches at the third tier of our system is in part by design and in part required by the computational constraints posed by training convolution kernels. Since in a given image, not all regions are equally informative from the discrimination point of view, voting is a useful way to discount incorrect labels suggested by non-discriminative patches. At the same time, learning classifiers per patch instead for the whole image reduces the dimensionality of the data vectors, making it computationally efficient. This strategy of using voting among patches has been used in the past by various methods ([5], [29]).

Finally, the novel contributions of this paper can be summarized as follows: (1) Though Volterra theory has been around for a long time ($\sim$ 80 years) in the signal processing literature, it is used here for the first time to achieve image classification. (2) Unlike other kernel based methods that seek to obtain a non-linear mapping of the data, our method learns the kernels in a data driven manner and is not predisposed toward a fixed kernel (e.g. Gaussian, Radial Basis Function etc.). (3) In our formulation, the unknown Volterra kernels, irrespective of the order of the kernel, can be obtained by solving a generalized eigenvalue problem. (4) The Volterra kernels can have arbitrary (not necessarily square) shape, which gives them more flexibility in terms of possible transformation they can represent. Finally, (5) our three tier system allows for seamless integration of Volterra classifiers across different orders and shapes of convolution kernels.

1.1 Empirical Study: Face Recognition

We demonstrate the effectiveness of our image classification system using the image based face recognition problem. Following the naming tradition popular in the face recognition literature, we call the Volterra classifiers when applied to faces – Volterrafaces. We conduct a detailed study of our image classification system by breaking it down into three flavors – Linear, Quadratic and Boosted. Boosted Volterrafaces refers to the full three tier Volterra image classification system applied to face recognition. Linear and Quadratic Volterrafaces skip the second tier and use a single square kernel for each patch. An overview of the single kernel training and testing framework is provided in Figures 1 and 2 respectively. The Boosted Volterrafaces method is presented in Figure 3.

For empirical evaluation, we demonstrate the performance of Volterrafaces on five different benchmark face databases – Yale A [1], CMU PIE [35], Extended Yale B [28], MERL Dome [41] and CMU MultiPIE [17]. We compare the obtained results with those obtained from various recent (elaborated below) embedding-based face recognition methods and show that Volterrafaces compares favorably to various state-of-the-art methods.

The face recognition literature is fairly dense and diverse and thus cannot be surveyed in its entirety in the space here. A detailed survey of various types of face recognition methods can be found in [25]. In this paper we focus on the class of face recognition approaches called the embedding-based methods (or subspace methods) that are more closely related to our method. These are also the methods with which we empirically compare Volterrafaces and are summarized in Table 2.

Embedding-based methods, like Volterrafaces, attempt to transform the images such that in the transformed space images are more conducive to classification. A prime instance of such methods is Eigenfaces [36], which attempts to group

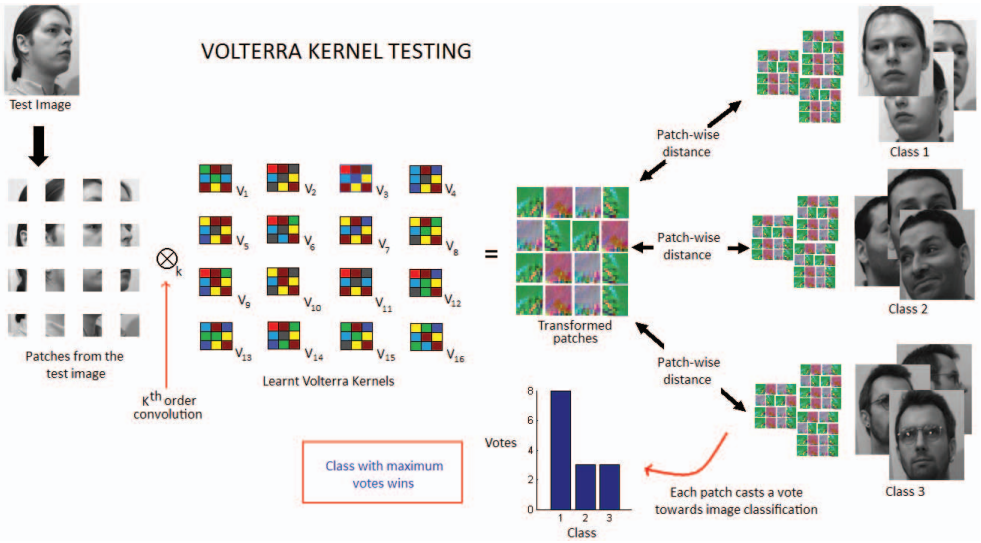

Fig. 2: Testing involves dividing the test image according to the scheme used for training. Then each patch is mapped to the range space by the corresponding Volterra kernel. After the mapping each patch is classified using a Nearest Neighbor classifier in the range space. After all patches have been individually classified, each patch from the test image casts a vote towards the parent image classification. The class with the maximum votes wins.

images by minimizing data variance. Fisherfaces [6] additionally finds a subspace that minimizes the intra-class distances while maximizing the inter-class distances. Tensor extension of both of these methods are also available ([43], [44], [37] and [8]). These methods differ from Volterrafaces in that the only allowed transformations are linear (or multilinear)

A subclass of embedding-based methods is geometrically inspired where the emphasis is on identifying a low dimensional sub-manifold on which the face images lie. The most successful of these methods include those which seek to project images to a lower dimensional subspace such that the local neighborhood structure present in the training set is maintained. These include Laplacianfaces [21], Locality Preserving Projections (LPP) [18], Neighborhood Preserving Embedding [20], Orthogonal Laplacianfaces [7] etc. Over time, improvements and generalization of these methods have appeared in [42], [10], [22] and [19]. Kernel extensions of the above mentioned methods are presented in [9] and [4], but unlike Volterrafaces, they use fixed polynomial or Gaussian kernels. We must mentioned that our current formulation of Volterrafaces does not yet fully exploit the general graph embedding framework, though such an extension should not be very difficult. To be more precise, when put in the graph embedding framework, the distances between graph nodes for Volterrafaces is simply based on the L_2 distances between the transformed images, as in Linear Discriminant Analysis [13]. Additionally, Volterrafaces does not seek reduced dimensionality and the transformed images are of the same size as the input images.

Recently a method that replaces the L_2 distance above with a correlation based distance called Correlation Tensor Analysis (CTA) [16] has been proposed. Other methods like Kernel Ridge Regression [3] take a different geometric approach and try to map images to vertices of a regular simplex such that images with the same labels belong to the same vertex while differently labeled images map to different vertices. Some methods are specifically geared towards the case with very small training set sizes like lasso based classification technique proposed in [32] (LASSO, MLASSO). This method simultaneously tries to find a linear projection and a linear classifier with L_1 constraints on the classifier. It is shown that this method outperforms many graph based embedding methods for small gallery set sizes. Though Volterrafaces does not give any special consideration to training set sizes, we would demonstrate that it outperforms existing methods for both small and large training sets.

Other feature based classification methods seek to extract features from images which can then be used for classification. Notable recent methods in this category were proposed in [40] and [34]. The first of these tries to find features such that the distance between the averages of the homogeneous and heterogeneous neighborhoods of images is maximized. This method works with tensor representations of images and is referred to as Tensor Average Neighborhood Margin Maximization (TANMM). The second method learns so-called Universal Visual Features from a given corpus of natural images (not necessarily face images). Once the features are obtained, they are used in conjunction with ICA to classify the probe set images. Our image classification scheme addresses the disconnect between the feature selection and the discrimination stages, as apparent in these methods.

Volterra Kernel Theory forms the core of Volterrafaces framework and thus we begin by introducing it in Section 2. In Section 3 we present our objective functions and its reduction to the generalized eigenvalue problem. In Section 4 we detail how multiple Volterra kernels can be combined using boosting to obtain a more powerful composite classifier (tier 2 of our image classification system). Having detailed how individual patches are classified thus far, the complete algorithms for training, testing and boosting Volterrafaces are presented in

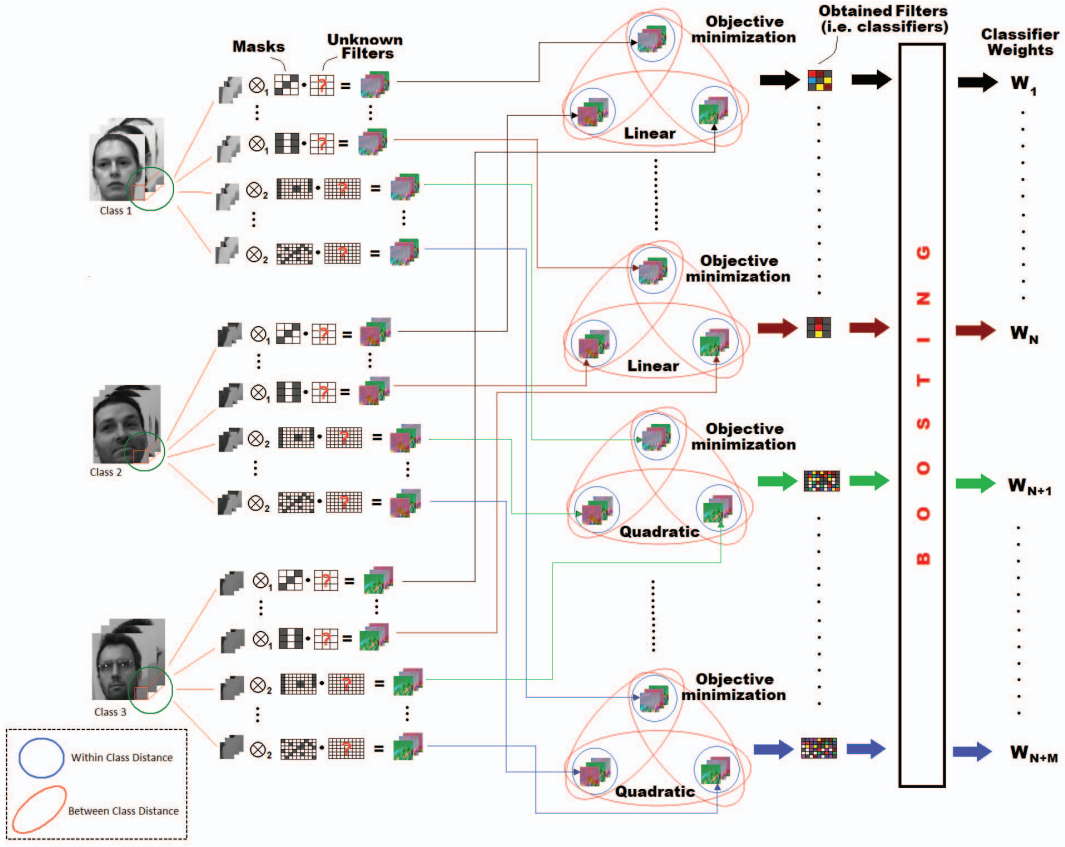

Fig. 3: The Boosted Volterrafaces learns multiple kernels of various shapes and orders for each patch. This makes use of the single kernel training framework presented in Figure. 1. Once these classifiers have been learned, the best weighted combination of these classifiers is picked using a multiclass boosting algorithm. The testing for Boosted Volterrafaces is carried out in the manner outlined in Figure 2 except that the composite classifiers are used to classify each patch.

Section 5. We present detailed experimental setup, results and discussions in Section 6 and finally conclude in Section 7.

2 Volterra Kernel Approximations

Any linear translation invariant (LTI) system can be completely characterized by a linear convolution. If $\Im : \mathbf{H} \rightarrow \mathbf{H}$ is a linear system which maps the input signal $x(t)$ to the output $y(t)$, it is described by a function $h(t)$ such that

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

The symbol $\otimes$ represents the convolution operation.

The Volterra series theory generalizes the above mentioned characterization to non-linear translation invariant systems. If $\aleph : \mathbf{H} \rightarrow \mathbf{H}$ is a non-linear system which maps the input signal $x(t)$ to the output signal $y(t)$, it can be described by an infinite sequence of functions $h_n(t)$ indexed by n as

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

where $\otimes_n$ represents the n^{th} order convolution and $y_n(t) =$

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

Here $h_n(\tau_1, \ldots, \tau_n)$ are called the Volterra kernels of the system $\aleph$. Note that Eq. (1) is a special case of Eq. (3) for $n = 1$. Another intuitive way to think of Volterra series is to see it as the Taylor series expansion of the functional $\aleph$ above. This functional $\aleph$ maps function $x(t)$ to $y(t)$. Further, this Taylor series representation of the functional $\aleph$ is unique.

Switching to a discrete formulation, Eq. (3) can be written as $y_n(m) =$

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

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

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

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

Note that to simplify the notation, we have defined the above equations for one-dimensional signals. All these equations can be seamlessly generalized for two-dimensional signals e.g.

images $I(x, y)$.

2.1 Volterra Theory and Pattern Classification

In general, given a set of input functions $\mathbf{I}$, we are interested in finding a functional $\aleph$, such that $\aleph(\mathbf{I})$ satisfies some desired property. This desired property can be captured by defining an *objective* 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 harmonic input method, direct expansion etc. [11] can be used to compute kernels of the unknown functional. In the absence of such an explicit relation, we propose that Volterra kernels be learned from the data using the *objective* functional.

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 *objective* functional could be a measure of 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 appropriate accuracy based on computational efficiency and classification accuracy constraints.

2.2 Significance of Translation Invariance

Note that the class of systems or functionals covered by Volterra theory is translation invariant. In the simpler one-dimensional case it means that if the input signal is shifted in time by a certain amount, so is the output signal i.e. if $\aleph(x(t)) = y(t)$ then $\aleph(x(t-a)) = y(t-a)$. For the case of images $I(x, y)$, it is more intuitive to think in terms of the two spatial dimensions. The translation invariance ensures that if the input image is shifted in spatial dimensions so would be the output image and by the same amount.

In the context of image classification, an ideal classification system would map all the images of a single subject to the same output function. In absence of such an ideal system, a translation invariant system ensures that images of a single subject, which are not accurately aligned, are not mapped to wildly different output images. Hence the translation invariance property is useful for the task of image classification. In fact, the importance of translation invariance has also been identified in other classification methods that rely on convolution operation [27].

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) $\mathbf{I}$, gallery or 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 *objective* functional O is minimized in the range space of $\mathcal{N}$. Functional O measures the departure from complete separability of data in the range space. In this paper we seek a functional $\mathcal{N}$ that maps all images from the same class in a manner such that the intraclass L_2 distance in minimized while the interclass L_2 distance is maximized. Once $\mathcal{N}$ has been determined, a new image can be classified using any methods like Nearest Centroid Classifier, Nearest Neighbor Classifier etc. in the mapped space. With this observation, we define the *objective* functional O as

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

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

$$\frac{\sum_{\mathbf{c_k} \in \mathbf{C}} \sum_{i,j \in \mathbf{c_k}} \| \sum_{n=1}^{p} I_i \otimes_n K - \sum_{n=1}^{p} I_j \otimes_n K\|^2}{\sum_{\mathbf{c_k} \in \mathbf{C}} \sum_{m \in \mathbf{c_k}, n \notin \mathbf{c_k}} \| \sum_{n=1}^{p} I_n \otimes_n K - \sum_{n=1}^{p} I_m \otimes_n K\|^2}, \tag{7}$$

where K, like $h(t)$ in eq. (5), is a placeholder for all the different order convolution kernels. Note that at first glance the above objective function formulation may seem similar to that of Linear Discriminant Analysis (LDA) [13] but for a closer examination, see Appendix A.

Depending on the order of convolution, the operation can transform the input data linearly or non-linearly. In the following we present a way to transform the input data such that irrespective of the convolution order, the convolution operation can always be represented as a linear operation. The conversion of the convolution operation into 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

$$I_i \otimes_p K = A_i^p \cdot \overline{K}, \tag{8}$$

where $\overline{K}$ is the vectorized form of the 2D kernel represented by K. The exact form of A_i^p depends on the order of the convolutions p as we describe in subsequent sections.

3.1 Volterra Kernel Shape

In an usual convolution operation, the shape of the convolution kernel can be easily controlled by setting desired entries in the kernel to zero. But in the case described above, where the kernel is the unknown, additional constraints are needed. Since a square $(n \times n)$ kernel does not necessarily have to be the one which provides the best mapping in terms of pattern classification rates, we propose to use a binary mask to explicitly set the kernel shape.

This transforms the Eq. (8) as

$$I_i \otimes_p K = A_i^p \cdot M \cdot \overline{K}, \tag{9}$$

where M is a diagonal matrix of size equal to the length of the vector $\overline{K}$. Its i^{th} diagonal entry is set to zero or one depending of whether the i^{th} entry in $\overline{K}$ is desired to be zero or non-zero. As we will describe in subsequent sections, we do not

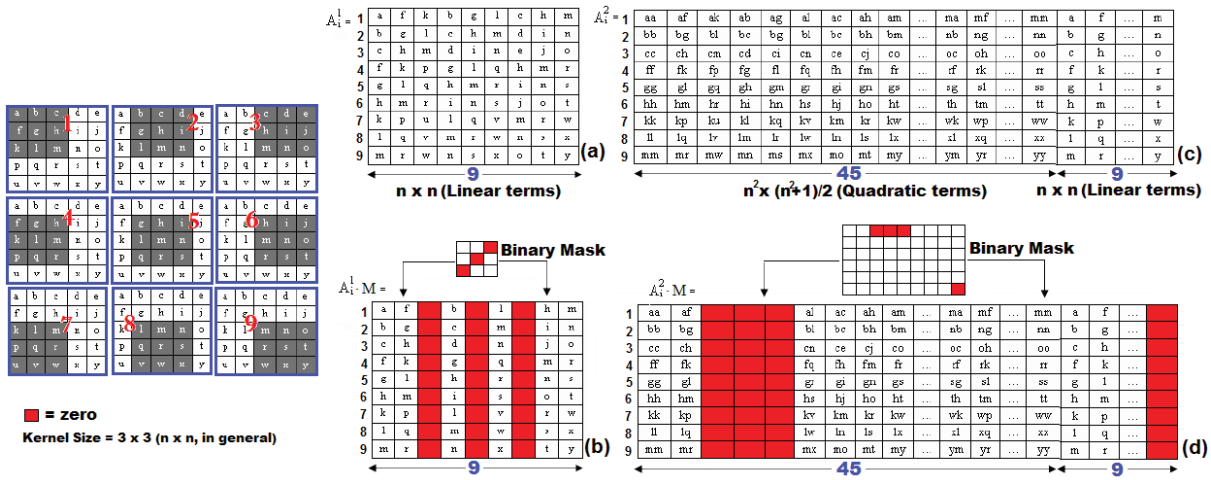

Fig. 4: Structure of A_i^1 and A_i^2 for an image of size 5×5 and kernel of size 3×3. On the left, 9 neighborhoods of the image I_i are highlighted. (a) For first order approximation, each of these neighborhoods become a row in A_i^1. (b) For second order case, we take all the second order combinations of pixel values in each neighborhood and use them as the first 45 $(n^2(n^2+1)/2)$ elements of rows in A_i^2. The rest 9 (n^2) elements are simply the pixel values. Rows are numbered to show the corresponding neighborhood. Parts (c) and (d) show that when binary masks are used to enforce kernel shapes, certain columns in A matrices become zero.

assume that we know the shape of the kernel, rather it is picked automatically.

In the next two subsections we describe the exact transformation of the data and our objective function for the first and second order Volterrafaces classifiers. We have restricted ourselves to second order kernels because of computation ease and satisfactory classification performance. But the procedures described below can be analogously applied to higher order kernels.

3.2 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^1 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 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 4(a). Part (b) of the same figure shows that case when a binary mask is used to enforce a kernel shape. The pixels along the border of the image can be ignored or taken into account by padding the image with zeros. The two cases do not differ significantly in performance.

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

$$O(\mathbf{I}) = \frac{\sum_{\mathbf{c_k} \in \mathbf{C}} \sum_{i,j \in \mathbf{c_k}} \left\| A_i^1 \cdot \overline{K}_1 - A_j^1 \cdot \overline{K}_1 \right\|^2}{\sum_{\mathbf{c_k} \in \mathbf{C}} \sum_{m \in \mathbf{c_k}, n \notin \mathbf{c_k}} \left\| A_m^1 \cdot \overline{K}_1 - A_n^1 \cdot \overline{K}_1 \right\|^2} = \tag{10}$$

$$\frac{\sum_{\mathbf{c_k} \in \mathbf{C}} \sum_{i,j \in \mathbf{c_k}} (A_i^1 \cdot \overline{K}_1 - A_j^1 \cdot \overline{K}_1)^T (A_i^1 \cdot \overline{K}_1 - A_j^1 \cdot \overline{K}_1)}{\sum_{\mathbf{c_k} \in \mathbf{C}} \sum_{m \in \mathbf{c_k}, n \notin \mathbf{c_k}} (A_m^1 \cdot \overline{K}_1 - A_n^1 \cdot \overline{K}_1)^T (A_m^1 \cdot \overline{K}_1 - A_n^1 \cdot \overline{K}_1)} \tag{11}$$

$$= \frac{\sum_{\mathbf{c_k} \in \mathbf{C}} \sum_{i,j \in \mathbf{c_k}} \overline{K}_1{}^T (A_i^1 - A_j^1)^T (A_i^1 - A_j^1) \overline{K}_1}{\sum_{\mathbf{c_k} \in \mathbf{C}} \sum_{m \in \mathbf{c_k}, n \notin \mathbf{c_k}} \overline{K}_1{}^T (A_m^1 - A_n^1)^T (A_m^1 - A_n^1) \overline{K}_1}. \tag{12}$$

This can be written as

$$O(\mathbf{I}) = \frac{\overline{K}_1^T \mathbf{S_W} \overline{K}_1}{\overline{K}_1^T \mathbf{S_B} \overline{K}_1}, \tag{13}$$

where

$$\mathbf{S_W} = \sum_{\mathbf{c_k} \in \mathbf{C}} \sum_{i,j \in \mathbf{c_k}} (A_i^1 - A_j^1)^T (A_i^1 - A_j^1) \tag{14}$$

and

$$\mathbf{S_B} = \sum_{\mathbf{c_k} \in \mathbf{C}} \sum_{m \in \mathbf{c_k}, n \notin \mathbf{c_k}} (A_m^1 - A_n^1)^T (A_m^1 - A_n^1). \tag{15}$$

Here $\mathbf{S_W}$ and $\mathbf{S_B}$ are symmetric matrices of dimensions $b^2 \times b^2$ (much smaller than the number of rows in $\mathbf{S_W}$ or $\mathbf{S_B}$). Seeking the minimum of Eq. (13) 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.

When the binary masks are considered, the matrices $\mathbf{S_W}$ and $\mathbf{S_B}$ in Eq. (13) are transformed to

$$\mathbf{S_W} = \sum_{\mathbf{c_k} \in \mathbf{C}} \sum_{i,j \in \mathbf{c_k}} M(A_i^1 - A_j^1)^T (A_i^1 - A_j^1) M \tag{16}$$

and

$$\mathbf{S_B} = \sum_{\mathbf{c_k} \in \mathbf{C}} \sum_{m \in \mathbf{c_k}, n \notin \mathbf{c_k}} M(A_m^1 - A_n^1)^T (A_m^1 - A_n^1) M. \tag{17}$$

where M is the binary diagonal matrix as described in section 3.1.

Algorithm 1: Training Algorithm

Input: $trainSet$, KernelSize: ks, Convolution Order: ko, $patchSize$
Output: $volterraKernel$

Divide each $I \in trainSet$ into $patchNum$ patches of size $patchSize$
For each patch, X, compute the matrix A_X^{ko} using ks (as described in Section 3)
for $i = 1 : patchNum$ **do**
  $\mathbf{S_W} = \mathbf{S_B} = 0$
  for *Subject* $S \in trainSet$ **do**
    for *Image* $I \in S$ **do**
      $X = i^{th}$ patch $\in I$
      for *Image* $J \in S$ **do**
        $Y = i^{th}$ patch $\in J$
        $D = A_X^{ko} - A_Y^{ko}$
        $\mathbf{S_W} = \mathbf{S_W} + D^T D$
  for *Subject* $S_1 \in trainSet$ **do**
    for *Image* $I \in S_1$ **do**
      $X = i^{th}$ patch $\in I$
      for *Subject* $S_2 \in trainSet \setminus S_1$ **do**
        for *Image* $J \in S_2$ **do**
          $Y = i^{th}$ patch $\in J$
          $D = A_X^{ko} - A_Y^{ko}$
          $\mathbf{S_B} = \mathbf{S_B} + D^T D$
  $volterraKernel(i) =$ min eigvector $\{{\mathbf{S_B}}^{-1}\mathbf{S_W}\}$

3.3 Second Order Approximation

The second order approximation of the sought functional contains two terms

$$y(m) = \sum_{q_1=-\infty}^{\infty} h_1(q_1)x(m-q_1) + \sum_{q_1=-\infty}^{\infty}\sum_{q_2=-\infty}^{\infty} h_2(q_1,q_2)x(m-q_1)x(m-q_2) \quad (18)$$

The first term in Eq. (18) corresponds to a weighted sum of the first order terms, $x(m-q_1)$, while the second order 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 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 Fig. 4(c) while the masked version of the same in shown in Fig. 4(d). Note that we have chosen to have a binary mask which controls individual terms in the second order Volterra approximation rather than applying the binary mask to the linear kernel and then generating second order terms from it.

With this definition of A_i^2 we proceed as for the first order approximation to obtain equations similar to (10) and (13) 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 [11] and this symmetry also manifests itself in 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 matrices $\mathbf{S_B}$ and $\mathbf{S_W}$ to $\frac{b^4+3b^2}{2} \times \frac{b^4+3b^2}{2}$.

For the second order case with binary mask, the objective function is given by

$$O(\mathbf{I}) = \frac{\overline{K}_2^T \mathbf{S_W} \overline{K}_2}{\overline{K}_2^T \mathbf{S_B} \overline{K}_2} \quad (19)$$

where

$$\mathbf{S_W} = \sum_{\mathbf{c_k} \in \mathbf{C}} \sum_{i,j \in \mathbf{c_k}} M(A_i^2 - A_j^2)^T (A_i^2 - A_j^2) M \quad (20)$$

and

$$\mathbf{S_B} = \sum_{\mathbf{c_k} \in \mathbf{C}} \sum_{m \in \mathbf{c_k}, n \notin \mathbf{c_k}} M(A_m^2 - A_n^2)^T (A_m^2 - A_n^2) M. \quad (21)$$

It must be noted that the problem is still no different than the linear case in the variables being solved for and in fact *by use of this formulation we have ensured that regardless of the order of approximation, the problem is linear in the coefficients of the Volterra convolution kernels*. Now as in the first order approximation, the minimum of $O_k(\mathbf{I})$ is given by the minimum eigenvalue of ${\mathbf{S_B}}^{-1}\mathbf{S_W}$ and it is attained when $\overline{K}_2$ equals the corresponding eigenvector.

4 Boosting

In the framework outlined so far, we have described Volterra classifiers which can work with any binary mask to enforce kernel shape and can be of any order. But since we do not know a priori what is the right order or the right binary mask to use in order to achieve the best classifier performance – in our case, the best face recognition results, we present a boosting based algorithm which picks the best combination of the various Volterra classifiers in order to sidestep this problem. In addition, this composite classifier usually achieves higher performance than any of its components.

The number of possible binary masks that can be used to enforce kernel shape grows exponentially in the size of the kernel. For instance, for a 3×3 linear kernel, the number of possible binary masks is 512 while for a quadratic kernel of the same size, this number grows to 1.8×10^{16}. Clearly, exhaustively exploring this space of possible mask is not computationally efficient. In view of this, we propose that we randomly generate a fixed number of linear and quadratic binary masks and use them in our algorithm. For the results

Algorithm 2: Testing Algorithm

Input: $trainSet$, $testImage$, KernelSize: ks, Convolution Order: ko, $patchSize$, $volterraKernel$

1 Divide each $I \in trainSet \cup \{testImage\}$ into $patchNum$ patches of size $patchSize$
2 **for** $i = 1 : patchNum$ **do**
3 **for** *Subject* $S \in trainSet$ **do**
4 **for** *Image* $I \in S$ **do**
5 $X = i^{th}$ patch $\in I$
6 Compute matrix A_X^{ko} using ks
7 $sig(I, S, i) = A_X^{ko} * volterraKernel(i)$
8 $X = i^{th}$ patch $\in testImage$
9 Compute matrix A_X^{ko} using ks
10 $probe = A_X^{ko} * volterraKernel(i)$
11 $class(i) = argmin_{S'}\{||sig(I', S', i) - probe||_2\}$
12 **for** $i = 1 : patchNum$ **do**
13 vote(class(i)) = vote(class(i)) + 1;
14 Classify $testImage$ as $argmax_{S'}\{vote(S')\}$

Database	Subjects	Images per Person	Total
Yale A [1]	15	11	165
CMU PIE [35]	68	170	11560
Extended Yale B [28]	64	38	2432
MERL Dome [41]	242	16	3872
Multi-PIE [17]	249	19	4731

TABLE 1: Databases used in our experiments.

presented in this paper, we have fixed this number to 500 each for the linear and the quadratic kernels. Note that we ensure that the all ones binary mask (square kernel) is always included in the set of binary masks.

Of the various possible boosting strategies for multi-class classification, we have adapted the Stagewise Additive Modeling using a Multi-class Exponential loss function (SAMME) [46]. SAMME is itself an adaptation of the Adaboost algorithm [14] for binary classifiers to the multiclass case. The primary advantage of SAMME over other naive extensions of Adaboost to the multi-class case is that it only requires that the accuracy of individual classifiers be greater than $1/K$, where K is the total number of classes.

In the boosting framework for Volterrafaces, we first train a corpus of linear and quadratic classifiers using the chosen binary masks. Once all the classifiers have been learned, we pick the weighted combination of them (generally 20 or less) to build a composite classifier that we call Boosted Volterrfaces. The details of the training, testing and boosting algorithms are provided in the next section.

5 Algorithms and Parameters

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 develop separate kernels for different face regions. Further, if each constituent patch casts a vote towards the parent image classification, it brings in robustness to the overall image classification. Thus we adopt a patch based framework for the face recognition task. A distinct advantage of the patch based framework is that the aggregated classification results are more robust to changes in regions of the face which are not useful in a discriminative sense.

5.1 Single Kernel Training

The single kernel training algorithm (outlined in Algorithm 1) learns a classifier for each patch separately. The primary computation times in this algorithm is spent in constructing the matrices $\mathbf{S_B}$ (Lines 5 to 11) and $\mathbf{S_W}$ (Lines 12 to 19). For a MATLAB implementation, where matrix multiplications are more efficient than `for` loops, these matrices can be more efficiently computed by constructing a matrix $X = [A_{11}^p M \ A_{12}^p M \ \cdots \ A_{ij}^p M \ \cdots]$ and then computing $\mathbf{S_B}$ and $\mathbf{S_W}$ from the various block matrices embedded in $X^T X$. An overview of the training framework can be found in Figure 1.

5.2 Single Kernel Testing

The single kernel testing algorithm in presented in Algorithm 2. A given test image is first divided into patches (in the same manner as done during training). Each patch is then transformed into the A matrix form (see section 3) and mapped through the learned kernel for that patch. It is then compared against all the patches in the gallery set mapped through the same kernel. The class which is closest to it in L_2 sense is assigned to be the patch's classification.

Once all the patches have been classified, a voting is carried out and the class with the maximum number of votes is chosen to be the input image's classification. In case of a tie, reclassification only among the tied classes can be carried out. But we have found that since such occurrences are rare, randomly assigning the class among the tied classes provides almost the same performance. An overview of the testing framework can be found in Figure 2.

The previous two algorithms described how to learn a Volterra kernel and use it to classify images once the binary mask M and the order of the Volterra kernel p have been fixed. Here we describe the boosted version of Volterrafaces which combines various Volterra classifiers with different kernel shapes and orders to obtain a composite classifier. The objective of this algorithm is that given a corpus of classifiers, obtain a sparse set of weights which can be used to combine the classifiers in a linear fashion.

Our algorithm is outlined in Algorithm 3. For this we divide the gallery set further into a training set and a testing set. Since Volterrafaces requires at least two images per subject for learning kernels, the gallery set – when being used with Boosted Volterrafaces – must have at least three images per person. All the images are divided into patches as before. For each patch, we learn a separate composite classifier. It is also assumed that a set of binary masks (that define the kernel shapes) are provided for both linear and quadratic kernels.

5.3 Boosting with Multiple Kernels

For a given patch location, all the images in the training set are used to learn a Volterra classifier corresponding to each

Algorithm 3: Boosted Volterrafaces Training Algorithm

Input: $trainSet$, KernelSize: ks, $patchSize$, Maximum boosting iterations: M, Number of subjects: K
Output: $volterraKernel$, $kernelWeights$

1 Divide each $I \in trainSet$ into $patchNum$ patches of size $patchSize$
2 Divide the $trainSet$ into $boostingTrainSet$ and $boostingTestSet$. Let $sizeof(boostingTestSet) = L$
3 Generate binary mask for both linear and quadratic kernels
4 **for** $i = 1 : patchNum$ **do**
5 Compute n linear and quadratic kernel using Algorithm 1 for each binary mask using the $boostingTrainSet$
6 Classify $boostingTestSet$ using Algorithm 2 for each linear and quadratic kernel learned above and save classification results in $classiRes \in \{0,1\}^L$
7 Initialize $weights$ to $1/L$
8 **for** $m = 1 : M$ **do**
9 Using $classiRes$ and $KernelWeights$ pick the classifier with least error, call it T^m
10 Compute $err = \sum_{j=1}^{n} KernelWeights(j) * classiRes(j) / \sum_{j=1}^{n} KernelWeights(j)$
11 **if** $(err > (K-1)/K)$ **then**
12 break
13
14 Compute $kernelWeights(m) = log\frac{1-err}{err} + log(K-1)$
15 **for** $j = 1 : L$ **do**
16 $weights(j) = weights(j) * exp(kernelWeights(m) * classiRes(j))$
17 Normalize $weights$ to sum to 1

Method	Citation	Yale A	CMU PIE	Ext. Yale B	MERL Dome	Multi-PIE
Regularized Discriminant Analysis (RDA)	1989 JASA, Friedman [15]	-	-	10 20 30 40	-	-
Eigenfaces	1991 JCN, Turk & Pentland [36]	2 3 4 5	5 10 20 30	5 10 20 30	1 - 10	1 - 10
Fisherfaces	1997 TPAMI, Belhumeur et al. [6]	2 3 4 5	5 10 20 30	5 10 20 30	1 - 10	1 - 10
Laplacianfaces	2005 TPAMI, He et al. [21]	2 3 4 5	5 10 20 30	5 10 20 30	5 - 10	6 - 10
Tensor Subspace Analysis (TSA)	2005 NIPS, He et al. [19]	-	-	-	1 - 10	1 - 10
Orthogonal Laplacianfaces (o-LAP)	2006 TIP, Cai et al. [7]	2 3 4 5	5 10 20 30	-	-	-
Spectral Regression (SR)	2007 ICCV, Cai et al. [9]	-	30 40	10 20 30 40	5 - 10	5 - 10
Smooth Linear Discriminant Analysis (s-LDA)	2007 CVPR, Cai et al. [10]	2 3 4 5	-	-	1 - 10	1 - 10
Smooth Locality Preserving Projections (s-LPP)	2007 CVPR, Cai et al. [10]	-	-	-	1 - 10	1 - 10
Tensor Avg. Nbhood Margin Maximization (TANMM)	2007 CVPR, Wang et al. [40]	2 3 4	5 10 20	-	-	-
Orthogonal Rank One Tensor Projection (ORO)	2007 CVPR, Hua et al. [22]	5	30	20	-	-
Kernel Ridge Regression (KRR)	2007 CVPR, An et al. [3]	-	5 10 20 30	5 10 20 30	-	-
Correlation Tensor Analysis (CTA)	2008 TIP, Fu & Huang [16]	-	5 10 15 20	5 10 20 30	-	-
Universal Visual Features (UVF)	2008 CVPR, Shan et al. [34]	2 - 8	-	-	-	-
Multivariate Lasso Regression (MLASSO)	2008 CVPR, Pham & Venkatesh [32]	2 - 8	-	-	-	-
Kernel LPP with Side Information (KLPPSI)	2008 CVPR, An et al. [4]	-	5 10 20 30	5 10 20 30	-	-
Volterrafaces	**Our Method**	2 - 9	2 3 4 5 10 20 30	2 3 4 5 10 20 30 40	2 - 10	2 - 10

TABLE 2: State-of-the-art methods with which we compare our technique along with the training set sizes used in their experiments.

given kernel shape and order. Once a collection of classifiers have been obtained, we use them along with the testing set to boost the classification performance. This is outlined in the lines 8 to 17 of the Algorithm 3. Note that the step 6 requires us to classify each patch independently and even though we have referred to Algorithm 2, we do not use the voting part of it.

Once the weights of the classifiers have been obtained, a new patch x can be classified by the composite classifier C as

$$C(x) = \arg\max_{m} \sum_{m=1}^{M} kernelWeights(m)\mathbb{I}(T^m(x) = k), \tag{22}$$

where $kernelWeights$ are the M kernel weights as obtained in Algorithm 3 and T is the list of selected Volterra classifiers. $\mathbb{I}$ is the indicator function. An overview of the boosting framework can be found in Figure 3.

5.4 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 [32], parameter $\alpha \in [0, 1]$ in [10], dimensionality D in [34], reduced dimensionality in TSA [19], 2D-LDA [44], energy threshold in [33] etc.), Volterra classifiers have a smaller discrete set of parameter choices. We use one of the widely used [10][3] 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 in 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.

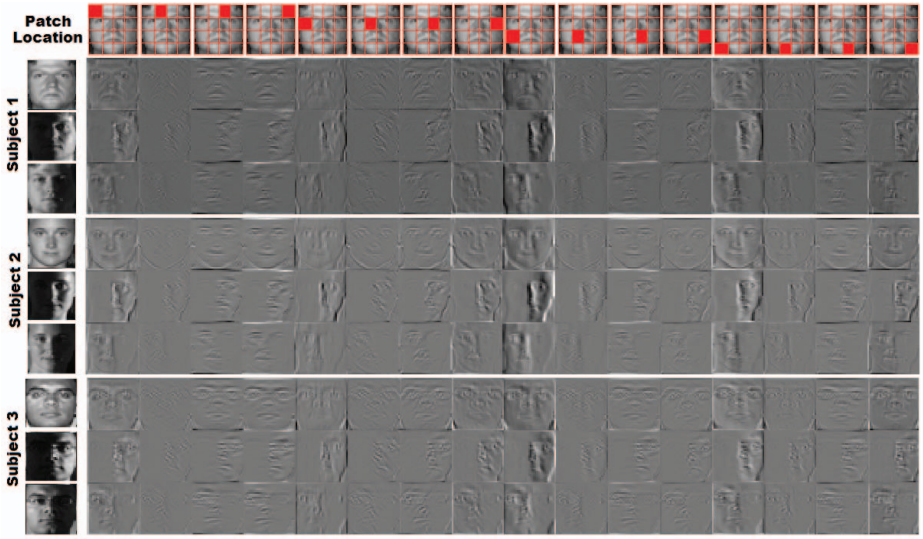

Fig. 5: Volterrafaces: The nature of learnt Volterra filters is shown here. For three different images of three different subjects, we have applied the learned Volterra filter to the whole image. At the top of each column, the red square shows the patch location from which the used linear kernel was learned.

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. Note that the last of these choices is removed in the Boosted Volterrafaces which uses both linear and quadratic kernels to boost performance. Also, when linear and quadratic classifiers are being used without boosting, the shape of the kernels is fixed to be a square.

In this paper we have presented results for both quadratic and linear kernels along with the boosted classifiers. 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 just within a particular database but also across databases. For the Boosted Volterrafaces during boosting three images from the gallery set were used for testing and the rest for training the weak classifiers. In the cases with gallery size of 3 or 4, two images were used for training the weak classifiers and the rest for testing during the boosting stage.

6 Experiments

In order to evaluate our technique we conducted recognition experiments across five different popular benchmark face databases. To provide a context for our results, we also present comparative results for various recent embedding based methods that have been applied to face recognition.

6.1 Databases

We have used Yale A [1], CMU PIE [35], Extended Yale B [28], MERL Dome [41] and CMU MultiPIE [17] datasets for our experiments. As mentioned before, we assume that the faces have been detected in the images and roughly aligned. Note that this is not unusual and other reported methods also make a similar assumption. For the first three databases – Yale A, CMU PIE and Extended Yale B, authors of [10] have provided cropped and manually aligned (two eyes were aligned at the same position) images with 256 gray value levels per pixel. Fortunately, the methods that we compare with have also used the same version of these datasets. For the Yale A face database we used 11 images each of the 15 individuals with a total of 165 images. For CMU PIE database 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 Extended Yale B database we used 64 (except for a few corrupted images) images each (frontal pose) of 38 individuals present in the database. For the more recent MERL Dome (242 subjects) and CMU MultiPIE (Session 1) (249 subjects) datasets we have 16 and 19 images respectively with varying illumination in neutral pose and expression. For these two databases face centers were manually aligned and cropped. Details of the datasets are summarized in Table 1.

Note that among these five datasets various different face images characteristics have been captured. Yale A involves images which have illumination as well as expression variation. Our CMU PIE subset includes images with both illumination and pose variation. The Extended Yale B, the MERL Dome and the Multi-PIE include images with extreme illumination variation. The MERL Dome and the Multi-PIE dataset also represent the recognition scenario where the number of subjects is reasonably large.

6.2 Comparative State-of-the-art

We have attempted to provide comparison with as many of the methods covered in the Section 1.1 as possible. We have included results from any method which has presented results on the same datasets as us and with the same experimental setup. We have also included those methods which have provided the source code for their algorithms. A listing of various traditional and recent state-of-the-art methods included in our comparisons is presented in Table 2.

For the Yale A, CMU PIE and Extended Yale B databases, we have found that since cropped and aligned data was provided by the authors of [10], many papers were able

to report results using very similar setup. Hence for these three databases we have reported results as mentioned in the original publications. For the MERL Dome and the Multi-PIE databases, due to their recency, we have obtained comparative results by running the experiments ourselves. Here we have included those methods for which source code was provided.

It is noteworthy that different methods have chosen to report results using various different gallery and probe set sizes. We have also reported the various configurations used by other methods and included them in our results in Table 2. Since we conducted experiments on MERL Dome and Multi-PIE databases ourselves, we were able to include a much wider range of configurations for other methods. For Volterrafaces, we have included the widest range of gallery-probe set configurations in our experiments as shown in Table 2. Note that a dash in the row corresponding to a method in Table 2 indicates that the cited paper chose not to include results on the database mentioned in the column heading.

6.3 Results and Discussion

Nature of Filters: Before we dive into the recognition performance of our method on various databases, in Fig. 5 we present images to provide a sense of the nature of the learned Volterra filters. The very top row shows the locations of the patches from where the kernels used in the columns below were learned. The subsequent rows show three different images of three different subjects from Multi-PIE database convolved with the learned square linear Volterra kernel. Note that all the images of each subject were normalized together to make them more visible.

Recognition Rates: Average recognition error rates on the Yale A, CMU PIE, Extended Yale B, MERL Dome and Multi-PIE databases are presented in Table 3, Table 4, Table 5, Table 6 and Table 7 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 methods (Volterrafaces and Boosted 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 report results with both linear and quadratic masks for the sake of completeness. Best results for a particular training set size are highlighted in bold. For Boosted Volterrafaces, we have used 500 randomly chosen binary masks for both the linear and the quadratic kernels. This means that the boosting is conducted on a set of 1000 Volterrafaces classifiers. In Table 6 and Table 7 a dash in a cell indicates that the method corresponding to the row failed to converge for one or more of the randomly chosen gallery -probe set configurations of the size corresponding to the column. Note that the citation next to the method name in these tables correspond to the publication from where the reported results or source code were obtained and not necessarily the publication which proposed the method (c.f. Table 2).

Based on these results following conclusions can be drawn:

1) Foremost, we note that Volterrafaces consistently outperforms both traditional and recent methods listed in Table 2 across various databases and training set sizes. The difference in performance is particularly striking for the more difficult cases of small training set sizes. Among the variants of Volterra Kernel classifiers, the Boosted Volterrafaces method, which combines both linear and quadratic kernels, provides the best performance for most of the cases.
2) In all cases, recognition errors go down as the size of the training set increases. It can be noted that the amount of improvement in recognition rates by inclusion of a single additional image in the gallery set is much more pronounced for small gallery set sizes e.g. when the gallery set size increases from 2 to 3 in the MERL Dome and the multi-PIE datasets as compared to when it increases from 7 to 8.
3) The five different databases used here have different characteristics. The difficulty of the face recognition task generally goes up with the number of different subjects in the database. The reported results on the MERL Dome and the Multi-PIE databases provide important test cases for such scenarios with large number of subjects. It also indicates that Volterrafaces maintains its performance even as the problem is scaled.

Relative Performance of Volterra Classifiers: It can be noted from the previous set of experiments that the linear kernels outperform the quadratic kernels. We attribute this phenomenon to the possible over-fitting of data by the quadratic kernels. This is suggested by the fact that for a typical 12×12 pixel patch, a 3×3 linear kernel has 9 unknowns while a quadratic kernel of the same size has 90 unknowns.

This leads us to two important questions: First, why should the Boosted Volterrafaces perform better than both linear and quadratic kernels? And second, do quadratic kernels contribute at all to the superior performance of Boosted Volterrafaces? These can be answered by understanding the workings of Boosting. Boosting performs well as long as the weak classifiers perform better than a random classifier and both linear and quadratic kernels classifiers do so. Thus if we try to boost ensembles of just the linear or the quadratic classifiers, we expect to see improvements in both. We confirm this in Table 8 where we see that both linear and quadratic classifiers improve if they are used in conjunction with Boosting. We have used the Extended Yale B dataset with three different training set sizes to illustrate this. The weak classifier ensemble was built with 50 classifiers with different kernel shapes. The next important point to note is that Boosting performance increases if the ensemble has a wide variety of classifiers (in terms of the samples they make mistakes on). We conjecture that this explains why the inclusion of both linear and quadratic classifiers improves performance beyond what can be achieved by them separately. This is illustrated by the last column in Table 8 where we see that for the exact same settings, a boosted combination of linear and quadratic classifiers outperforms both the boosted linear and the boosted quadratic classifiers. We found that typically Boosting picks a $60\% - 40\%$ mixture of linear and quadratic kernels, respectively.

Computational Efficiency: From the computational efficiency point of view, most of the time is spent during the train-

Train Set Size	2	3	4	5	6	7	8	9
S-LDA [10]	42.4	27.7	22.2	18.3	-	-	-	-
S-LDA [10] (updated)	37.5	25.5	19.3	14.7	12.3	10.3	8.7	-
UVF [34]	27.11	17.38	11.71	8.16	6.27	5.07	3.82	-
TANMM [40]	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**	**5.78**	**3.96**	**2.61**	**1.43**
Volterrafaces (Quadratic)	**22.15**	**13.36**	15.78	10.19	10.04	9.66	9.49	**8.74**
Volterrafaces (Boosted)	-	**11.45**	**7.72**	**8.14**	**5.25**	**5.00**	**3.36**	**1.66**

TABLE 3: **Yale A**: Recognition error rates. Linear kernel size was 5×5 and quadratic kernels size was 3×3. For both cases overlapping patches of size 12×12 were used. Images of size 64×64 were used. For other methods, best results as reported in the respective papers are used.

Train Set Size	2	3	4	5	10	20	30	40
KLPPSI [4]	-	-	-	27.88	12.32	5.48	3.62	-
KRR [3]	-	-	-	26.4	13.1	5.97	4.02	-
ORO [22]	-	-	-	-	-	-	6.4	-
TANMM [40]	-	-	-	26.98	17.22	5.68	-	-
SR [9]	-	-	-	-	-	-	6.1	5.2
OLAP [7]	-	-	-	21.4	11.4	6.51	4.83	-
MLASSO [32]	54.0	43.0	34.0	-	-	-	-	-
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)	**43.0**	**36.30**	**23.98**	**20.26**	**10.24**	**4.94**	**2.85**	**2.37**
Volterrafaces (Quadratic)	**50.48**	**39.66**	**32.67**	25.29	11.94	**5.45**	4.60	**3.04**
Volterrafaces (Boosted)	-	**20.71**	**19.29**	**19.29**	**9.06**	**4.62**	**1.73**	**1.52**

TABLE 4: **CMU PIE**: Recognition error rates. Linear kernel size was 5×5 and quadratic kernels size was 3×3. For both cases overlapping patches of size 12×12 were used. Images of size 32×32 were used. For other methods, best results as reported in the respective papers are used.

Train Set Size	2	3	4	5	10	20	30	40
ORO [22]	-	-	-	-	-	-	9.0	-
SR [9]	-	-	-	-	12.0	4.7	2.0	1.0
RDA [9]	-	-	-	-	11.6	4.2	1.8	0.9
KLPPSI [4]	-	-	-	24.74	9.93	3.15	1.39	-
KRR [3]	-	-	-	23.9	11.04	3.67	1.43	-
CTA [16]	-	-	-	16.99	7.60	4.96	2.94	-
MLASSO [32]	58.0	54.0	50.0	-	-	-	-	-
Eigenfaces [16]	-	-	-	54.73	36.06	31.22	27.71	-
Fisherfaces [16]	-	-	-	37.56	18.91	16.87	14.94	-
Laplacianfaces [16]	-	-	-	34.08	18.03	30.26	20.20	-
Volterrafaces (Linear)	**26.23**	**18.23**	**9.33**	**6.35**	**2.67**	**0.90**	**0.42**	**0.34**
Volterrafaces (Quadratic)	**40.81**	**20.47**	**14.42**	**13.0**	**3.98**	**1.27**	**0.58**	**0.43**
Volterrafaces (Boosted)	-	**8.73**	**6.78**	**4.50**	**0.83**	**0.36**	**0.17**	**0.13**

TABLE 5: **Extended Yale B**: Recognition error rates. All Volterrafaces methods used kernels of size 3×3 and for all cases non-overlapping patches of size 8×8 were used. Images of size 32×32 were used. For other methods, best results as reported in the respective papers are used.

Train Set Size	2	3	4	5	6	7	8	9	10
S-LDA [10]	37.79	16.33	9.91	7.59	1.25	0.44	0.34	0.29	0.16
S-LPP [10]	58.68	37.88	32.44	18.98	14.90	10.21	7.71	6.08	5.09
SR [9]	-	-	-	8.64	1.85	1.04	0.59	0.52	0.40
TSA [19]	48.44	28.56	20.23	16.71	8.02	5.28	3.53	3.28	1.14
Eigenfaces [36]	75.77	68.57	61.03	60.33	54.02	39.86	35.87	35.38	40.52
Fisherfaces [6]	27.50	9.34	6.05	6.07	3.39	2.90	1.18	0.98	0.89
Laplacianfaces [21]	-	-	-	65.83	26.53	16.55	10.73	10.45	9.22
Volterrafaces (Linear)	**5.47**	**3.01**	**0.76**	**0.59**	**0.06**	**0.10**	**0.07**	**0.03**	**0.04**
Volterrafaces (Quadratic)	**12.16**	**5.27**	**3.48**	**2.82**	1.50	0.61	0.75	0.54	0.71
Volterrafaces (Boosted)	-	**2.90**	**0.12**	**0.09**	**0.04**	**0.07**	**0.05**	**0.00**	**0.00**

TABLE 6: **MERL Dome**: Recognition error rates. All Volterrafaces methods used kernels of size 3×3. Linear and Quadratic cases used non-overlapping patches of size 8×8 while the Boosted case used overlapping patches of size 6×6. Images of size 32×32 were used. For other methods, results were obtained using the source code provided by the respective authors.

Train Set Size	2	3	4	5	6	7	8	9	10
S-LDA [10]	37.83	16.98	14.59	9.34	3.96	1.38	0.95	0.67	0.59
S-LPP [10]	58.29	34.41	41.91	31.59	15.65	7.86	5.01	3.26	2.79
SR [9]	-	-	-	10.88	4.37	1.85	1.30	1.00	0.90
TSA [19]	44.22	24.85	21.51	12.13	6.88	2.01	1.48	0.84	0.65
Eigenfaces [36]	65.68	54.44	49.77	42.90	37.00	31.90	26.66	20.36	18.23
Fisherfaces [6]	30.03	13.48	10.93	10.11	3.98	1.39	0.95	0.67	0.59
Laplacianfaces [21]	-	-	-	-	16.16	8.15	5.11	3.30	2.77
Volterrafaces (Linear)	**13.49**	**8.81**	**4.23**	**2.48**	**1.21**	**0.74**	**0.48**	**0.35**	**0.32**
Volterrafaces (Quadratic)	**15.46**	**10.15**	**5.49**	**3.11**	**1.45**	**0.95**	**0.55**	**0.43**	**0.48**
Volterrafaces (Boosted)	-	**3.81**	**2.04**	**0.37**	**0.34**	**0.07**	**0.06**	**0.04**	**0.04**

TABLE 7: **Multi-PIE**: Recognition error rates. All three Volterrafaces methods used kernels of size 3×3 and for all cases non-overlapping patches of size 8×8 were used. Images of size 32×32 were used. For other methods, results were obtained using the source code provided by the respective authors.

Train Set	Linear		Quadratic		Linear + Quadratic
Size	No-Boost	Boosted	No-Boost	Boosted	Boosted
3	14.98	10.98	27.18	17.75	**9.59**
5	7.97	4.99	16.99	9.42	**4.56**
20	0.83	0.32	2.76	0.46	**0.27**

TABLE 8: **Effect of Boosting**: Error rates for three different train set sizes on the Extended Yale B dataset.

ing phase when the Volterra kernels are learned. The training phase involves solving a generalized eigenvalue problem but the size of the involved matrices does not grow with the training set size as detailed in Section 3. The testing phase involves convolution and a Nearest-Neighbor classification step. For the Boosted Volterrafaces, most of the time is spent building the corpus of classifiers. In our experiments, this involved learning 500 different linear and quadratic kernels each. The training phase of the Boosted Volterrafaces algorithm lends itself very well to data shared parallel implementation. The testing phase of the Boosted Volterrafaces involves convolution with 20 or less kernels and a weighted nearest neighbor classification, as given by Eq. (22).

7 Concluding Notes

In this paper we have introduced the Volterra Kernel classifiers, which for the first time use Volterra kernel theory for face image classification. We believe that the basic framework for learning discriminative Volterra kernels introduced in this paper can open a whole new class of image classifiers. A few of these possibilities were explored in this paper, but the Fisher-like objective function that we define in Eq. (7) can be easily modified to account for various different types of neighborhood structures as covered in our discussion about geometrically inspired embedding-based methods (section 1.1).

We believe that the superior performance of the Volterrafaces classifier can be traced to its two important components. The first is its ability to learn linear or non-linear mappings in a data driven fashion. This is in contrast to many of the existing kernel methods (e.g. [3], [4]) which use Gaussian or Polynomial kernels whose parameters are fixed a priori. The second component of Volterrafaces that contributes to its superior performance is the voting mechanism that aggregates the classification information across different parts of an image. In our experiments we have observed that many times the winner class wins with less than 50% of the votes.

Our current boosting framework focused on boosting classification performance of each patch separately. There are various other boosting configurations that can also be explored for instance the patch sizes, patch location as well as the kernel size can be chosen via boosting. Hence boosting still has more scope to provide performance gains. At the same time a drawback of our boosting framework is that it requires a corpus of classifiers, whose numbers can grow exponentially in the kernel size unless they are explicitly limited to some arbitrary number (e.g. 1000 in our case). This is because the number of possible kernel shapes is exponentially related to the kernel size.

It should also be noted that there are various flavors of multi-class boosting algorithms available and we have only adapted one of them to work with Volterrafaces. Other algorithms can possibly also be adapted to work with Volterrafaces. For instance, our current boosting algorithm requires weak classifiers to make hard classifications, which works well with Volterrafaces. But Volterrafaces can be easily modified to provide soft decisions (e.g. probabilities based on nearest neighbor classifier distances) and then boosting algorithms that work with classification probabilities can also be used.

Keeping our algorithms computationally tractable also limits us from exploring kernels of order 7 and higher. Similar to the issue with the kernel shape, the size of the matrix A (in Fig. 4) also grows exponentially with the kernel size. This is also something we would like to address in the future. Voting for the final classification, which forms an integral part of our algorithm, is also conducted in a very straightforward manner. Giving more weights to patches towards the center of the face can possibly be useful. This is yet another issue we would like to explore in the future.

A limitation of Volterrafaces is that it cannot work when a single gallery image is available (note that most of the other methods included in experiments also cannot work with single gallery image). Though we have shown that in conjunction with generative methods which can generate more images from the single gallery image, Volterrafaces can still provide state-of-the-art performance, as shown in [26].

We must explain the use of small sized images (64×64 and 32×32) in our experiments. It is not a limitation of our methods since any large image can be downsampled to a smaller size. In fact in most realistic face recognition setups, high resolution images are not available. Further, it is not clear if high frequency details present in high resolution images are critical for face recognition. It is noteworthy that commercial systems like [23] also work with similar small sized images.

On the broader point of image classification, trainable convolution filters can be seen as a novel class of empirically useful and theoretically interesting features. For a given image classification task, they alone may suffice or may be combined

with existing fixed features like Haar, Gabor etc. Recent neighborhood encoding schemes like Local Binary/Ternary Patterns [2], [45] may also be used in sequence with trainable convolution filters. Finally, trainable convolution filters can be classified as either a local or a global method depending on the patch-size. This syncs well with our more recent work on Kernel Plurality [24] where we have shown that performance of methods that are traditionally considered global (e.g. PCA, LDA) can in fact be enhanced if they are used with a variable patch size followed by voting.

REFERENCES

[1] http://cvc.yale.edu/projects/yalefaces/yalefaces.html.
[2] T. Ahonen, A. Hadid, and M. Pietikainen. Face description with local binary patterns: Application to face recognition. *PAMI*, 2006.
[3] Senjian An, Wanquan Liu, and Svetha Venkatesh. Face recognition using kernel ridge regression. In *CVPR*, 2007.
[4] Senjian An, Wanquan Liu, and Svetha Venkatesh. Exploiting side information in locality preserving projection. In *CVPR*, 2008.
[5] M. Artiklar, P. Watta, and M. Hassoun. Local voting networks for human face recognition. *IJCNN*, 2003.
[6] Peter N. Belhumeur, Joao Hespanha, and David J. Kriegman. Eigenfaces vs. fisherfaces: Recognition using class specific linear projection. *PAMI*, 19(7):711–720, 1997.
[7] D. Cai, X. He, J. Han, and H. J.Zhang. Orthogonal laplacianfaces for face recognition. *TIP*, 15(11), 2006.
[8] Deng Cai, Xiaofei He, and Jiawei Han. Subspace learning based on tensor analysis. Technical Report UIUCDCS-R-2005-2572, UIUC, 2005.
[9] Deng Cai, Xiaofei He, and Jiawei Han. Spectral regression for efficient regularized subspace learning. In *ICCV*, 2007.
[10] Deng Cai, Xiaofei He, Yuxiao Hu, Jiawei Han, and Thomas Huang. Learning a spatially smooth subspace for face recognition. In *CVPR*, 2007.
[11] J. A. Cherry. Introduction to volterra methods. *Distortion Analysis of Weakly Nonlinear Filters Using Volterra Series*, 1994.
[12] Corinna Cortes and V. Vapnik. Support-Vector Networks. *Machine Learning*, 20, 1995.
[13] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. Wiley-Interscience, Hoboken, NJ, 2000.
[14] Y. Freund and R. Schapire. A decision theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139.
[15] J. H. Friedman. Regularized discriminant analysis. *Journal of the American Statistical Association*, 84(405):165–175, 1989.
[16] Yun Fu and Thomas S. Huang. Image classification using correlation tensor analysis. *IEEE TIP*, 17(2), 2008.
[17] R. Gross, I. Matthews, J. Cohn, S. Baker, and T. Kanade. The cmu multi-pose, illumination, and expression (multi-pie) face database. Technical Report Tech Rep TR-07-08, CMU, 2007.
[18] X. He, D. Cai, and P. Niyogi. Locality preserving projections. In *NIPS*, 2003.
[19] X. He, D. Cai, and P. Niyogi. Tensor subspace analysis. In *NIPS*, 2005.
[20] X. He, D. Cai, S. Yan, and H.-J. Zhang. Neighborhood preserving embedding. In *ICCV*, 2005.
[21] X. He, S. Yan, Y. Hu, P. Niyogi, and H. Zhang. Face recognition using laplacianfaces. *IEEE PAMI*, 27(3), 2005.
[22] G. Hua, P. Viola, and S Drucker. Face recognition using discriminatively trained orthogonal rank one tensor projections. In *CVPR*, 2007.
[23] M. Jones and P. Viola. Face recognition using boosted local features. Technical Report TR2003-25, MERL, 2003.
[24] R. Kumar, A. Banerjee, B. C. Vemuri, and H. Pfister. Maximizing all margins: Pushing face recognition with kernel plurality. *ICCV*, 2011.
[25] Ritwik Kumar, Angelos Barmpoutis, Arunava Banerjee, and Baba C. Vemuri. Non-lambertian reflectance modeling and shape recovery of faces using tensor splines. *IEEE TPAMI*, 2009.
[26] Ritwik Kumar, Michael Jones, and Tim K. Marks. Morphable reflectance fields for enhancing face recognition. *CVPR*, 2010.
[27] Steve Lawrence, C. L. Giles, Ah Chung Tsoi, and A. D. Back. Face recognition: A comvolutional neural network approach. *IEEE Transactions on Neural Networks*, 8:98–113, 1997.
[28] K. Lee, J. Ho, and D. J. Kriegman. Acquiring linear subspaces for face recognition under variable lighting. *PAMI*, 27(5):684–698, 2005.
[29] K. C. Lee, J. Ho, M. H. Yang, and D. Kriegman. Video-based face recognition using probabilistic appearance manifolds. *CVPR*, 2003.
[30] T. Leung and J. Malik. Representing and recognizing the visual appearance of materials using three-dimensional textons. *International Journal of Computer Vision*, 2001.
[31] Y. W. Pang, L. Zhang, M. J. Li, Z. K. Liu, and W. Y. Ma. A novel gabor-lda based face recognition method. *Advances In Multimedia Information Processing*, 331:352–358, 2004.
[32] Duc-Son Pham and Svetha Venkatesh. Robust learning of discriminative projection for multicategory classification on the stiefel manifold. In *CVPR*, 2008.
[33] Santu Rana, Wanquan Liu, Mihai Lazarescu, and Svetha Venkatesh. Recognising faces in unseen modes: a tensor based approach. In *CVPR*, 2008.
[34] Honghao Shan and Garrison W. Cottrell. Looking around the backyard helps to recognize faces and digits. In *CVPR*, 2008.
[35] 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*.
[36] M. Turk and A. Pentland. Eigenfaces for recognition. *Journal of Cognitive Neurosciences*, 3:72–86, 1991.
[37] A. Vasilescu and D. Terzopoulos. Multilinear subspace analysis of image ensembles. In *CVPR*, 2003.
[38] P. Viola and M. Jones. Robust real-time face detection. *IJCV*, 57:137–154, 2004.
[39] V. Volterra. Theory of functionals and of integral and integro-differential equations. *Blackie and Sons Ltd.*, 1930.
[40] Fei Wang and Changshui Zhang. Feature extraction by maximizing the average neighborhood margin. In *CVPR*, 2007.
[41] T. Weyrich, W. Matusik, H. Pfister, B. Bickel, C. Donner, C. Tu, J. McAndless, H. W. Jensen J. Lee, A. Ngan, and M. Gross. Analysis of human faces using a measurement-based skin reflectance model. *ACM SIGGRAPH*, pages 1013–1024.
[42] S. Yan, D. Xu, B. Zhang, H.-J. Zhang, Q. Yang, and S. Lin. Graph embedding and extension: A general framework for dimensionality reduction. *PAMI*, (1), 2007.
[43] Jian Yang, David Zhang, Alejandro F. Frangi, and Jing yu Yang. Two-dimensional pca: A new approach to appearance-based face representation and recognition. *IEEE PAMI*, 26(1):131–137, 2004.
[44] J. Ye, R. Janardan, and Q. Li. Two-dimensional linear discriminant analysis. In *NIPS*, 2004.
[45] W. C. Zhang, S. G. Shan, W. Gao, and H. M. Zhang. Local gabor binary pattern histogram sequence (lgbphs): A novel non-statistical model for face representation and recognition. *ICCV*, 2005.
[46] J. Zhu, S. Rossetand H. Zou, and T. Hastie. Multiclass adaboost. Technical report, Stanford University, 2005.

Ritwik Kumar is a Research Staff Member at IBM Research - Almaden. He received his PhD and MS degrees in Computer Engineering at the Dept. of Computer and Information Science and Engineering at the University of Florida, Gainesville, FL, USA in 2009. He completed his post-doctoral training at the School of Engineering and Applied Sciences at the Harvard University, Cambridge, MA, USA. He is a recipient of DAIICT President's Gold Medal (2005) and the University of Florida Alumni Fellowship (2005 - 2009).

Arunava Banerjee is an Associate Professor in the Department of Computer and Information Science and Engineering at the University of Florida, Gainesville. He received his M.S. and Ph.D. degrees in the Department of Computer Science at Rutgers, the State University of New Jersey in 1995 and 2001, respectively. His research interests include algorithms, computer vision, operational research, and computational neuroscience. He received the Best Paper award at the Nineteenth International Conference on Pattern Recognition (ICPR), 2008.

Baba C. Vemuri is currently a Professor and Director of the Center for Vision, Graphics & Medical Imaging at the Department of Computer and Information Sciences at the University of Florida, Gainesville. He was a coprogram chair of ICCV 2007. He has been an AC and a PC member of several IEEE conferences. He was an AE of IEEE TPAMI (from 1992-96) and the IEEE TMI (from 1997-03). He is currently an AE for the Journal of MedIA and CVIU. He is a Fellow of the ACM and the IEEE. His research interests include medical image analysis, computational vision, and information geometry.

Hanspeter Pfister is currently the Gordon McKay Professor of the Practice of Computer Science at the School of Engineering and Applied Sciences at Harvard University, Cambridge, MA. He received his Ph.D. in Computer Science in 1996 from the State University of New York at Stony Brook and his M.S. in Electrical Engineering from the ETH Zurich, Switzerland, in 1991. His research lies at the intersection of visualization, computer graphics, and computer vision. He was a recipient of the IEEE Visualization Technical Achievement Award 2010.