Images as Bags of Pixels

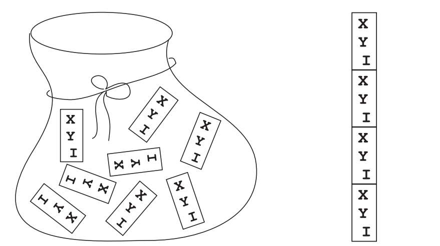
Tony Jebara
Department of Computer Science, Columbia University
New York, NY 10027
jebara@cs.columbia.edu

Abstract

We propose modeling images and related visual objects as bags of pixels or sets of vectors. For instance, gray scale images are modeled as a collection or bag of (X, Y, I) pixel vectors. This representation implies a permutational invariance over the bag of pixels which is naturally handled by endowing each image with a permutation matrix. Each matrix permits the image to span a manifold of multiple configurations, capturing the vector set's invariance to orderings or permutation transformations. Permutation configurations are optimized while jointly modeling many images via maximum likelihood. The solution is a uniquely solvable convex program which computes correspondence simultaneously for all images (as opposed to traditional pairwise correspondence solutions). Maximum likelihood performs a nonlinear dimensionality reduction, choosing permutations that compact the permuted image vectors into a volumetrically minimal subspace. This is highly suitable for principal components analysis which, when applied to the permutationally invariant bag of pixels representation, outperforms PCA on appearance-based vectorization by orders of magnitude. Furthermore, the bag of pixels subspace benefits from automatic correspondence estimation, giving rise to meaningful linear variations such as morphings, translations, and jointly spatio-textural image transformations. Results are shown for several datasets.

1. Introduction

A vital component of any computer vision system is its choice of representation for images and visual data. The way visual information is parameterized, features are extracted, or images are mathematically described remains an active area of research. The success of subsequent vision modules for recognition, segmentation, tracking, and modeling often hinges on the initial representation we chose. For instance, if important variations in our image data generate linear or smooth changes under a given representation, a subsequent recognition module should perform significantly better. In this article we propose a *bag of pixels* or *vector set* representation for images. For example, a gray scale image can be considered as a collection of N pixels each with spatial coordinates (X, Y) and an intensity coordinate (I). Thus, each image in our database is a bag of (X, Y, I) 3-tuples. Similarly, an edge or point-image can be seen as a bag of (X, Y) tuples with no intensity information. Even color video can be described as a vector set of $(X, Y, R, G, B, time)$ 6-tuples. Figure 1(a) depicts this bag of pixels or collection of tuples representation. For comparison purposes, Figure 1(b) shows a traditional appearance-based vectorized representation where the tuples are rigidly concatenated along a fixed ordering.



(a) Bag of Pixels (b) Vectorization

Figure 1: A bag of pixels or vector set versus a direct vectorization representation for a gray scale image.

In a bag of pixels, it is important to maintain that there is no ordering on the pixels and these can be permuted arbitrarily. Concatenating the pixels into a single long vector would obscure this important source of invariance in our representation. Instead of assuming a single ordering or correspondence on the pixels in a bag, we will maintain that each bag of pixels can span a manifold of configurations in a vectorized Euclidean space and treat the ordering as an unknown yet estimable parameter in our modeling process. Figure 2 depicts a manifold representing a single image as a bag or set of vectors of N pixels each of which is a D-dimensional tuple. This manifold is embedded in a $\Re^{N\times D}$-dimensional Euclidean vector space where each point on the manifold is given by concatenating the pixel tuples according to an arbitrary ordering. Admittedly, hard permutations of a vector do not form a continuous manifold. Instead, we *approximate* permutation with soft doubly-stochastic matrices which do in fact create a smooth

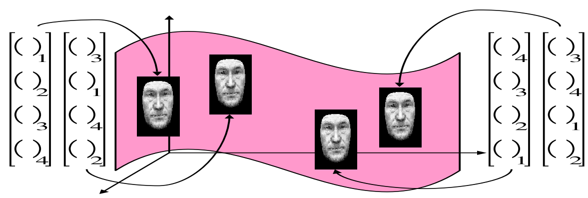


Figure 2: An image as a manifold of vector set configurations in an embedding Euclidean vector space.

and continuous manifold of configurations. In this toy example, we show four configurations of the same image under four different vector orderings. Each vector on the manifold is an arbitrarily ordered concatenation of the 4 pixel tuples. These configurations are invariant since all points on the manifold correspond to the same bag of pixels and render the same identical image. To parametrically consider arbitrary re-orderings, we endow each image or bag of pixels with its own unknown soft block-wise permutation matrix. Movement along each vector set or image's manifold can be represented by varying the unspecified permutation matrix to explore the arbitrary possible re-orderings of the vector set. Instead of proposing a fixed scheme (flow-based, appearance-based, physical deformation , etc.) for establishing the best correspondence or optimal setting of the permutation matrix, we fold this permutation estimation into our overall learning algorithm which jointly computes the permutations while fitting a statistical model to a dataset of many images (e.g. faces, hand-drawn digits, and so forth). Thus, while learning a model of multiple images (for instance a Gaussian or subspace model), we simultaneously estimate the optimal setting for the permutation matrices for each image such that we maximize the likelihood of the model fitting to the dataset. Essentially, the correspondence problem is simultaneously solved for each image in the whole dataset via a global criterion. Potential criteria we will consider include a Gaussian model fit or subspace model fit to a given database of images.

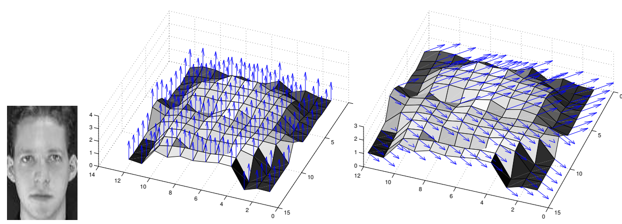
We show that the above estimation problem is solvable as a convex program which yields a unique solution for the joint estimation of the *model and representation*. In other words, we jointly estimate a subspace model and the correspondence or permutation matrices for all images. The permutation matrices are described via a convex hull of constraints and we propose two convex cost functions for estimating them. The first is the maximum likelihood Gaussian mean estimator which gives rise to an estimate of permutation matrices that clusters images towards a common mean. The second is the maximum likelihood Gaussian covariance estimator which gives rise to an estimate of permutation matrices (or correspondences) that aligns images into a minimally low-dimensional subspace, which is an almost ideal pre-processing step for principal components analysis. We show update equations for solving the convex problem for the permutation matrices as an iterative quadratic program. Treating images in this manner provides a general method for solving correspondence and gives rise to more meaningful subspaces of variation for a given dataset. We show interesting experimental results on image datasets including faces and hand-drawn digit images. We note improved modeling, reconstruction, correspondence and representation of images as bags of pixels. For instance, subspace methods such as principal components show improved reconstruction accuracy (by orders of magnitude) when the optimal permutations are estimated instead of being assumed (or heuristically computed). Furthermore, we show various spatio-textural morphing bases emerging automatically after learning from image data. We then conclude with extensions and a summary. Optimized implementation code, additional results and further details are provided online at: www.cs.columbia.edu/∼ jebara.

2. Background

Other efforts in visual representations have explored similar treatments of images as a collection of (X, Y, I) points or (X, Y) points [15, 5]. However, one central issue plaguing this type of representation is the so-called correspondence problem [18, 1, 5]. For instance, it is not clear which tuple in bag A (representing image A) should match with an given tuple in bag B (representing image B). A variety of methods exist for establishing such correspondence, for example physics-based models [15, 18]. Another way to improve appearance models that directly vectorize images is to estimate and apply spatial alignment [14, 13, 4, 12]. Optical flow and variants similarly compute small local correspondences to recover jointly spatial and illumination subspace models [1, 8, 16, 2]. An interesting alternative that is similar to our approach is to consider direct optimization criteria for establishing correspondence such as minimizing the covariance of the data [11] or its description length [3]. However, in most frameworks, correspondence is established a priori through some intermediate criterion, prespecified physical constraints, pairwise matching or alignment optimization instead of emerging automatically from invariances in the overall model learning process. We propose the latter scheme, estimating correspondences to agree with our modeling paradigm without resorting to prior or side constraints.

3. Bags of Pixels vs. Vectors

In this section we further motivate and compare the bag of pixels or vector set representation versus vectorization or appearance-based models. Correspondence issues and the topology of an image are discarded under direct lexigraphic vectorization of the image. Vectorization merely scans the image in a fixed spatial pattern, concatenating each scalar intensity entry into a long vector and ignoring the spatial relationship between adjacent pixels. In this representation, natural variations in the image such as translation appear highly nonlinear. For example, translating a rasterized image vector x_0 by t pixels would involve multiplying it by a shifting matrix M taken to a power of t, i.e. $x_t = M^t x_0$. This is a highly nonlinear operation. In Figure 1(b), direct appearance-based vectorization naively organizes pixels into a fixed vector by *assuming* a fixed ordering and sacrifices our ability to see variation in spatial coordinates, if for example, we were to form an eigenspace over a database of such images. If, we instead represent the image as a *bag* of pixels without specifying the ordering, we can maintain properties of the data such as spatial proximity between adjacent pixels and see variations in X and Y as well as I.



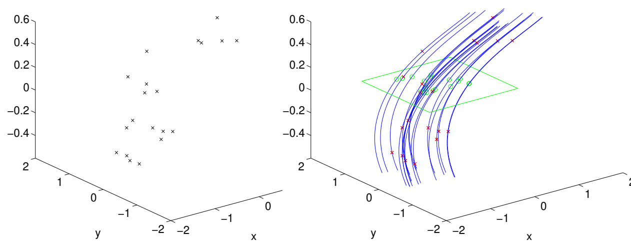
(a) Image (b) Vectorization Basis (c) Bag of Pixels Basis

Figure 3: Morph properties of bag of pixels or vector sets.

One crucial property of the vector set or collection of tuples (i.e. pixels) is that the correspondence between images in the dataset is no longer specified and becomes implicit in the learning process. For example, the aforementioned naive ordering used in vectorization would make the X and Y components of the large vector redundant since these are constant from image to image. The only sources of variation are the I-intensity components. Meanwhile, the collection of pixels representation does not assume an ordering and, if we were to properly estimate correspondence, variations in the X and Y spatial coordinates could emerge. Consider applying a principal component analysis method to vectorized images as in the eigenfaces method [13]. Therein, a basis over the rasterized or vectorized representation would only involve additions and deletions of intensity components in the image. Therefore, a simple change like translation in the image appears highly non-linear. Alternatively, a basis over a collection of tuples where correspondence is optimally estimated allows variations in X and Y coordinates just as easily as variations in intensity. Thus, an image can morph or translate via a linear transformation in this representation. This process is depicted in Figure 3. In (a) we see a regular gray-scale image, which we can consider as a 3D surface with X-coordinates, Y-coordinates and I-intensity values. If vectorized in lexicographic order, the only basis of variation will be a vertical change in intensities as shown, for instance, in (b). In (c), however, we see that a basis of morphings in X and Y coordinates could also emerge if the vectorization is handled more elegantly. In this final figure, the vectors of flow indicate that we can translate the image equally well in X, Y or I merely via linear variations. Thus we note that it is suboptimal to assume an arbitrary ordering. We will instead compute the optimal correspondence or permutation matrix for each image while we perform our model estimation (i.e. estimating a PCA subspace or Gaussian model). One confounding aspect remains in our modeling problem however: each permutation matrix is an unknown transformation parameter which moves the image along a path of invariance.

4. Modeling while Permuting

We can view the unusual vector set representation as a problem of learning a model under permutational invariances of each image (here, each image is a data point in a given database we are attempting to model). A representation often implies invariance properties in an object. In the bag of pixel vectors, ordering of the objects in the bag should be invariant. So, transformations that permute the order of the tuples should not change the representation.



(a) (b)

Figure 4: Invariant manifold learning.

Consider estimating of the optimal permutation or correspondence by using a simple example of a subspace learning problem, such as principal components analysis (PCA). In Figure 4(a) we see a data set in $\Re^3$ which needs to be modeled by a lower dimensional manifold. Clearly, no appropriate manifold is apparent and PCA will not provide a useful result. This is because we have assumed a fixed ordering for each image (each data point) instead of maintaining the images as a manifold of possible vectorized configurations. However, if we generalize PCA and add invariants to the data, this is no longer the case (see Figure 4(b)). For instance, the permutational invariance property of each image provides us with not only a single point for each image but a path or manifold [1] along which each image may move

[1] A path is simply a 1-dimensional manifold

invariantly prior to applying PCA (i.e. a *corresponded principal components analysis*). If points can be moved along their permutation paths invariantly, they can more clearly form a compact two-dimensional subspace. Therefore, we will approach invariant model learning as estimation of permutation transformations on the data (i.e. paths for each datum) while simultaneously forming a model.

5. A Convex Program

For the above estimation problem to have a unique globally optimal solution, we cast it as a jointly convex optimization over permutation matrices and model parameters. In fact, we will implicitly compute the optimal model parameters and fold them into a single cost function exclusively over permutation matrices. This process will be explicated in the next section, but for now, assume we have the following general scenario. We are given an input dataset of T vectors, $X_1, \ldots, X_T$. Each of these T vectors is of size $N \times D$ and results from the concatenation of the tuples in the bag of pixels according to some initial random ordering. We then endow each vector with an affine transformation matrix A_t that interacts linearly with it as follows: $\sum_j A_t^{ij} X_t^j$. These matrices then act to permute the entries in the given vector [7].

For our invariance, these A_t matrices are not just affine matrices but rather block-wise permutation matrices. However, optimizing over hard permutation settings is intractable (max-cut or integer programming methods might help resolve this). Instead, we relax the matrices such that they are *soft* permutation matrices. Also, the matrices can only permute the D-dimensional tuples and not individual scalar entries in the X_t vectors. Therefore, each A_t matrix is a grid of many $D \times D$ identity matrices scaled by an unknown non-negative scalar A_t^{ij}. For example, a matrix permuting two different tuples has the following structure:

$$A_t = \begin{bmatrix} A_t^{11} I & A_t^{12} I \\ A_t^{21} I & A_t^{22} I \end{bmatrix}$$

To relax the hard permutation matrices which only have binary entries, we only constrain each A_t matrix to be doubly-stochastic, in other words:

$$\sum_i A_t^{ij} = 1 \quad \sum_j A_t^{ij} = 1 \quad A_t^{ij} \geq 0$$

Thus, we can re-sort the tuples in each X_t vector as well as take convex combinations of them. In fact, the A_t matrices need not be square, but can be of size $ND \times N_t D$ where N_t is the number of tuples in each X_t image vector. This is useful if we want the correspondence estimation to *combine* or mix two or more pixels in differently sized images so that all images map into a common $\Re^{ND}$ embedding space for PCA or our Gaussian model. We denote these many constrained transformation matrices as $A = A_1, \ldots, A_T$ and note that they satisfy a set of linear equality and inequality constraints. Thus, our solution space over the set of matrices is a *convex hull*. The convex hull is a requirement of any convex programming framework, as shown below:

$$\min_A C(A) \; subject\ to \; \sum_{ij} A_t^{ij} Q_{td}^{ij} + b_{td} \; \geq \; 0 \, \forall t, d \qquad (1)$$

Here, $C(A)$ is a convex cost function to be minimized and the Q_{td} and b_{td} constants specify a hull of constraints (such as our doubly-stochastic constraints). Once we specify $C(A)$, the above formulation is solvable via convex programming techniques, including dual and axis-parallel methods which all yield a global solution. The general optimization picture that emerges is shown in Figure 5.

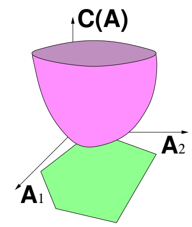


Figure 5: Convex program over doubly-stochastic matrices.

We now propose possible choices for the convex cost function over the $C(A)$ matrices. These emerge automatically from traditional maximum likelihood modeling criteria (such as Gaussian modeling or subspace modeling).

6. Maximum Likelihood Criteria

We now consider two model estimation criteria (although others are possible) and see how they give rise to a convex cost function over the permutation matrices. For jointly performing model estimation while learning invariances, consider the simplest case of estimating a Gaussian mean by maximizing likelihood of the data while we explore different permutation settings. The log-likelihood is $l(A, \mu) = \sum_t \log \mathcal{N}(A_t X_t; \mu, I)$. The optimal mean is $\hat{\mu} = 1/T \sum_t A_t X_t$ which we can plug back into the log-likelihood expression to get:

$$l(A, \hat{\mu}) = -\frac{TD}{2} \log(2\pi) - \frac{1}{2} \sum_t \|A_t X_t - \hat{\mu}\|^2$$

The above likelihood can now be *further* maximized over permutations. By negating, we convert likelihood into a cost function to minimize over A. With further manipulations, the cost function that emerges essentially minimizes the trace of the covariance of the permuted image data:

$$C(A) = tr(Cov(A\,X))$$

The trace of the covariance is a convex quadratic function over the A_t matrices. Combined with linear constraints that make each A_t doubly-stochastic, this cost is minimizable via quadratic programming or iterative methods. This Gaussian mean criterion thus tends to select permutation matrices that cluster data spherically, by moving images along their paths to *center data towards a common mean*.

We generalize to Gaussians of variable covariance as in $\mathcal{N}(AX;\mu,\Sigma)$ and also plug in the maximum likelihood covariance estimate $\hat{\Sigma} = 1/T\sum_t (A_tX_t - \hat{\mu})(A_tX_t - \hat{\mu})^T$ into the likelihood function to obtain:

$$\begin{aligned} l(A,\hat{\mu},\hat{\Sigma}) &= -\frac{TD}{2}\log(2\pi) - \frac{T}{2}\log|\hat{\Sigma}| \\ &\quad -\frac{1}{2}\sum_t (A_tX_t-\hat{\mu})^T\hat{\Sigma}^{-1}(A_tX_t-\hat{\mu}) \end{aligned}$$

After simplifications, the maximum likelihood solution of A is equivalent to minimizing the cost function $|Cov(A\,X)|$ [2]. We will instead minimize the logarithm of the above cost, i.e. $C(A) = \log|Cov(A\,X)|$ since it shares the same optima. We also regularize the cost function by adding a small identity matrix to the covariance and adding a small $tr(Cov(A\,X))$ term (as in the Gaussian mean case), we can avoid rank degeneracies and guarantee the cost stays convex. More specifically, we have:

$$C(A) = \log|Cov(A\,X) + \epsilon_1 I| + \epsilon_2 tr(Cov(A\,X))$$

Both ϵ_1 and ϵ_2 are kept small ($\approx$ 1.0). We prove convexity in the Appendix. Therefore our new $C(A)$ is again convex and Equation 1 results in a convex program. However, it is not a quadratic program. We can instead minimize $C(A)$ by iteratively upper bounding using a quadratic function in A. This permits us to sequentially solve multiple quadratic programs interleaved with variational bounding steps until we converge to the global solution. First consider $\log|S|$ where we have defined $S = Cov(AX)+\epsilon_1 I$. The logarithm of the determinant is concave over covariance matrices [6]. Since $\log|S|$ is concave, we can upper bound it with a tangential linear function in S that is equal and has the same gradient $R = S_0^{-1}$ at the current setting of $S = S_0$ which is computed from our current setting of our permutation matrices, $A = A_0$. The upper bound is then:

$$\log|S| \leq trace(RS) + \log|S_0| - tr(RS_0)$$

Adding our additional regularizer term with ϵ_2 to the above, we obtain the following upper bound on $C(A)$:

$$\begin{aligned} C(A) &\leq trace(R(Cov(A\,X)+\epsilon_1 I)) + \log|S_0| \\ &\quad -tr(RS_0) + \epsilon_2 tr(Cov(A\,X)) \end{aligned}$$

Simplifying the bound by removing terms that are constant over A, we have the following surrogate cost to minimize:

[2]Both the trace and determinant were discussed by [11] yet were not derived via maximum likelihood or convexified into a uniquely solvable program over doubly-stochastic matrices.

$$\begin{aligned} \tilde{C}(A) &= tr(MCov(A\,X)) \\ where\ M &= (Cov(A\,X)+\epsilon_1 I)^{-1} + \epsilon_2 I \end{aligned}$$

We thus update M for the current setting of the A_t matrices (by computing the covariance of the data after each A_t is applied to each X_t), then lock it for a few iterations while we minimize the trace to update the A permutation parameters. Updates of M are interleaved with updates of the A matrices until convergence. The above criterion attempts to cluster data ellipsoidally such that it forms a low-dimensional sub-manifold. It is well known that the determinant of a covariance matrix behaves like a volumetric estimator and approximates the volume of the data. Minimizing volume by varying the permutation matrices (i.e. computing the correspondence) is a valuable preprocessing step for PCA since it concentrates signal energy into a smaller number of eigenvalues, improving the effectiveness and reconstruction accuracy in the PCA subspace. Therefore, this criterion attempts to flatten the data via permutation such that it *forms as flat and low-dimensional a subspace as possible*.

7. Implementation

The cost functions so far both involve minimizing the trace of a matrix in the following general form ($M = I$ for the Gaussian mean case, while in the Gaussian covariance case, M is periodically recomputed from the covariance of the data as it is being transformed):

$$\begin{aligned} tr(MCov(A\,X)) &= \frac{1}{T}\sum_{mpnqi} A_i^{mn}A_i^{pq}X_i^qM^{pm}X_i^n \\ &\quad -\frac{1}{T^2}\sum_{mpnqij} A_i^{mn}A_j^{pq}X_j^qM^{pm}X_i^n \end{aligned}$$

Degeneracies may arise since we approximate permutations using doubly-stochastic matrices. For instance, A_t's entries might all become a constant $c = 1/N$, average out all tuples. To discourage this, we add a quadratic penalty to the cost as $-\lambda\sum_{imn}(A_i^{mn}-c)^2$. This penalizes matrix entries close to the mean and favors entries near 0 or 1. The λ is chosen adaptively to maintain convexity[3].

To minimize the cost with constraints, we use an SMO approach [17], although an SVD-like updated rule for each A_t is also possible. As in SMO, we vary a single A_t matrix for the t'th image at a time and only update 4 of its entries $(A_t^{mn}, A_t^{mq}, A_t^{pn}, A_t^{pq})$ while all others are locked. Iterating randomly, we ultimately update all scalar entries in each A_t matrix. However only 4 scalars are updated at a time in a given iteration. Double-stochasticity gives the equality constraints: $A_t^{mn} + A_t^{mq} = a$, $A_t^{pn} + A_t^{pq} = b$, $A_t^{mn} + A_t^{pn} = c$

[3]Other simplifications are possible. For instance, when minimizing in the determinant, we can force the Gaussian mean to be equal to a single random image in our dataset, i.e. $\mu = X_i$ for the i'th image and also lock its corresponding permutation matrix to identity, i.e. $A_i = I$.

$$A_t^{mn} \leftarrow \frac{NUM}{2DEN}$$

$$NUM = cX_n^T M_{mp} X_n + cX_n^T M_{pp} X_q + 2aX_q^T M_{pm} X_q + cX_n^T M_{pm} X_n + aX_n^T M_{mm} X_q - 2aX_q^T M_{mm} X_q - 2cX_n^T M_{pp} X_n - aX_n^T M_{mp} X_q - cX_n^T M_{pm} X_q - aX_n^T M_{pm} X_q - X_q^T M_{mp} X_q d - X_n^T M_{pp} X_q d + X_n^T M_{pp} X_q a + X_n^T M_{mp} X_q d - X_q^T M_{pm} X_q d + 2X_q^T M_{pp} X_q d - 2X_q^T M_{pp} X_q a + 2aX_q^T M_{mp} X_q + H1 - H3 + H4 - H2 - cX_q^T M_{mp} X_n - aX_q^T M_{mp} X_n - aX_q^T M_{pm} X_n + aX_q^T M_{mm} X_n + cX_q^T M_{pp} X_n + X_q^T M_{pm} X_n d - X_q^T M_{pp} X_n d + X_q^T M_{pp} X_n a + 4a\lambda + 2c\lambda - 2d\lambda$$

$$DEN = X_q^T M_{mp} X_q - X_n^T M_{mm} X_n - X_q^T M_{mm} X_q - X_n^T M_{pp} X_n + X_n^T M_{pm} X_n - X_q^T M_{pp} X_q + X_n^T M_{mm} X_q - X_n^T M_{pm} X_q + X_n^T M_{mp} X_n - X_n^T M_{mp} X_q + X_n^T M_{pp} X_q + X_q^T M_{pm} X_q - X_q^T M_{pm} X_n - X_q^T M_{mp} X_n + X_q^T M_{pp} X_n + X_q^T M_{mm} X_n + 4\lambda$$

$$H1 = (X_n^T M_{um}^T + X_n^T M_{mu})\left(\sum_{u,v \neq \{mn,mq,pn,pq\}} A_{uv} X_v - \frac{1}{T-1} \sum_{u,v,\tau \neq t} A_{\tau,uv} X_{\tau,v}\right)$$

$$H2 = (X_n^T M_{up}^T + X_n^T M_{pu})\left(\sum_{u,v \neq \{mn,mq,pn,pq\}} A_{uv} X_v - \frac{1}{T-1} \sum_{u,v,\tau \neq t} A_{\tau,uv} X_{\tau,v}\right)$$

$$H3 = (X_q^T M_{um}^T + X_q^T M_{mu})\left(\sum_{u,v \neq \{mn,mq,pn,pq\}} A_{uv} X_v - \frac{1}{T-1} \sum_{u,v,\tau \neq t} A_{\tau,uv} X_{\tau,v}\right)$$

$$H4 = (X_q^T M_{up}^T + X_q^T M_{pu})\left(\sum_{u,v \neq \{mn,mq,pn,pq\}} A_{uv} X_v - \frac{1}{T-1} \sum_{u,v,\tau \neq t} A_{\tau,uv} X_{\tau,v}\right)$$

Figure 6: Update rule for iteratively adjusting entries of the permutation matrices.

and $A_t^{mq} + A_t^{pq} = d$. So only one degree of freedom is left to compute *per iteration* and is updated as in Figure 6. The operations involve computing all possible inner products between the X-tuples X_n and X_q weighted by all relevant M_{mm}, M_{mp}, M_{pp} and M_{pm} sub-matrices[4]. We compute the $H1, H2, H3, H4$ and NUM, DEN terms. The ratio of NUM over $2DEN$ gives the optimal update value for the matrix entry A_t^{mn}. We then limit A_t^{mn} to satisfy the inequalities $A_t^{mn} \in [\max(0, a-d, c-1), \min(a, c, 1+a-d)]$. After updating A_t^{mn}, we can update the other 3 entries via their linear dependence on A_t^{mn}. Iterating the update rule randomly over different entries and matrices while intermittently recomputing bounds (via the inverse of M) converges monotonically to the global minimum of $C(A)$.

If a new test image X_{T+1} is observed after training and we wish to compute its A_{T+1} matrix, we could re-optimize the full cost $C(A_1 \ldots A_{T+1})$ again. A more efficient approach is to fix previous estimates of matrices A_t for $t = 1..T$ and just optimize $C(A)$ for the new A_{T+1}. A further simplification is to apply PCA to the covariance matrix since the training data is now flattened into a subspace. We maintain a reasonable number of k eigenvectors and align a new image to this eigenspace by choosing its permutation matrix A_{T+1} to minimize its squared error of the reconstruction in the subspace. If the eigenvectors are $V_1, \ldots, V_k$, we thus minimize the following quadratic cost subject to double-stochasticity constraints on A_{T+1}:

$$A_{T+1} = \arg\min_A \| \sum_k (V_k^T A X_{T+1}) V_k - A X_{T+1} \|^2$$

8. Experimental Results

We evaluated the framework on three datasets: (X, Y) point images of digits, (X, Y, I) intensity images of a single faces and (X, Y, I) intensity images of multiple individuals. In the first dataset, we obtained 28×28 gray scale images of the digits 3 and 9 which were then represented as a collection of 70 (X, Y) pixels by sampling the region of high intensity. This generates clouds of 2D points in the shape of 3's or 9's. A total of 20 such point images was collected with 70 (X, Y) pixels each. We then estimated the A_t permutation matrices by minimizing the covariance's determinant. Figure 7(a) depicts 6 exemplars of the original image data as point clouds. In Figure 7(b), standard PCA with 10 eigenvectors reconstructed the digits point clouds poorly. In Figure 7(c), PCA with 10 eigenvectors was instead applied to the bag of pixels *after* estimation of the permutation matrices. Note that the images are reconstructed more faithfully in the latter case due to the estimation of the permutation or correspondence. Unlike the direct PCA eigenvectors which do not resolve correspondence, the eigenvectors after permutation seem to translate and morph the digits smoothly as in Figure 8(a). Also, the first eigenvectors have more interpretable modes of variation. When applied to the number 9, the first eigenvectors seem to morph the point cloud in interesting ways as in Figure 8(b) and (c).

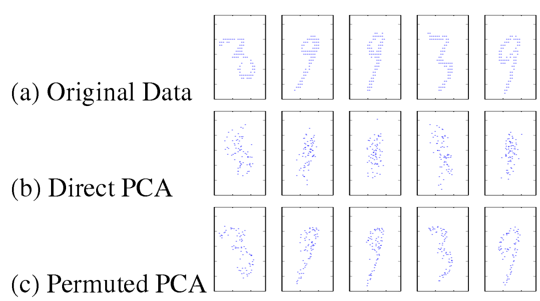


Figure 7: Reconstruction of digit images via PCA with and without permutation estimation.

In a larger experiment, we obtained $T = 300$ gray scale

[4]For clarity, here the matrices and vectors are indexed with subscripts instead of superscripts and all entries where the datum index does not appear refer to the datum at the t'th index.

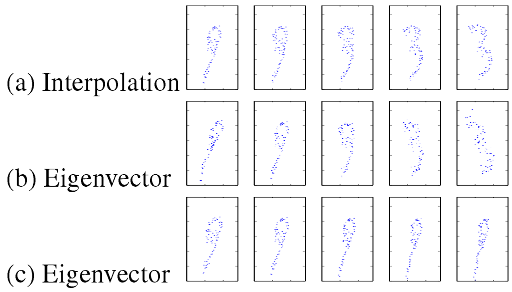
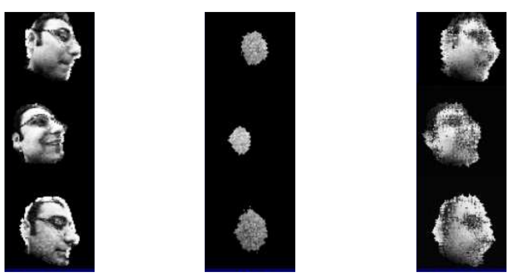


Figure 8: Linear interpolation (morphing) and effect of the first eigenvectors under a bag of pixels representation.

images of faces and sampled the pixels in skin-colored regions to obtain a collection of $N = 2000$ (X, Y, I) pixels for each face. The face images were of a single individual's face as it spans many lighting, 3D pose and expression configurations. Due to the larger size of this dataset, we avoided explicitly storing the A_t matrices which are of size $O(N^2)$. A more efficient storage method for doubly-stochastic matrices can be use where each is saved as $2N$ scalars and estimated with the *Invisible Hand* algorithm [10]. On this dataset, we estimated the permutation transformation matrices and we found an eigenspace of 20 components. We then reconstructed the images from 20 coefficients alone. Figure 9 depicts the accuracy of the reconstructed faces when standard PCA (with 20 eigenvectors) was used as well as when PCA for bags of pixels was used which performs permutation estimation. The images show much higher fidelity when permutation or correspondence is optimized. The permuted (X, Y, I) vector-set eigenvectors act smoothly, rotating and morphing the face in 3D as well as changing its illumination. Traditional appearance-based PCA causes *ghosting* effects where translated images do not move smoothly but, instead, appear to fade in and out. More interestingly, in terms of squared error, PCA had a reconstruction error of 9e5 while permuted PCA had reconstruction error of 5e3. The novel method reduces squared error by approximately 2.5 orders of magnitude.



(a) Original Data (b) Direct PCA (c) Permuted PCA

Figure 9: Reconstruction of (X, Y, I) facial images with vectorized PCA or PCA on permutable bags of pixels.

Finally, a large dataset of 3D synthesized faces of many individuals was used. Permutations were estimated in a semi-supervised way for efficiency and then PCA was performed on the permuted pixels. We show the first few eigenvectors as $\pm$ displacements from the mean face in Figure 10. Each row is one eigenvector (the top row is the top eigenvector) while columns show varying degrees of additions and deletions of the eigenvector. Note how the eigenvectors correspond to natural out-of-plane and in-plane rotations, as well as joint morphings and intensity variations. In current work, we are building a bag of pixels face tracker based on these eigenvectors.

Figure 10: Top 5 Bag of Pixels Eigenvectors applied to the Mean from Multi-Person (X, Y, I) Face Images.

9. Summary and Conclusions

We explored permutation invariance for dealing with images that are organized into collections of tuples, vector sets or bags of pixels. Statistical modeling (Gaussian mean, covariance, appearance subspaces) was shown to clearly benefit from the simultaneous estimation of per-image permutation transformations as well as model parameters. Image data points are effectively permuted and aligned simultaneously during the modeling process. They are also aligned as a whole dataset instead of via pair-wise or ad hoc criteria. We note drastically improved reconstruction accuracy as well as more meaningful linear bases of variation (morphings, spatio-textural change, etc.). Currently, we are exploring a kernel or distance $K(\chi, \chi) - 2K(\chi, \nu) + K(\nu, \nu)$ between bags of pixels which corresponds to finding the closest distance between two permutation manifolds. Surprisingly, this can be done implicitly and efficiently without explicitly computing the optimal permutations[9].

Acknowledgments

Thanks to the reviewers and N. Jojic for valuable feedback. This work was supported in part by NSF ITR 0312690.

Appendix

Theorem *The function* $C(\mathcal{X}) = \log|Cov(\mathcal{X}) + \epsilon_1 I| + \epsilon_2 tr(Cov(\mathcal{X}))$ *of the vector dataset* $\mathcal{X} = \{\vec{x}_1, \ldots, \vec{x}_N\}$ *is convex in the data (and also in variables such as* A_n *that linearly interact with the data) when* $\epsilon_1 \geq 1$ *and* $\epsilon_2 \geq 1$.

Proof *Without loss of generality, assume the vectors in* $\mathcal{X}$ *are zero-mean which yields* $Cov(\mathcal{X}) = \sum_n \vec{x}_n \vec{x}_n^T$ *when we ignore scaling by* $1/N$. *The function then simplifies to:*

$$C(\mathcal{X}) = \log\left|\sum_n \vec{x}_n \vec{x}_n^T + \epsilon_1 I\right| + \epsilon_2 tr\left(\sum_n \vec{x}_n \vec{x}_n^T\right)$$

Compute the Hessian of $C(\mathcal{X})$ *over single large vector argument* $\vec{X}$ *formed by the concatenation of all the* $\vec{x}_1, \ldots, \vec{x}_N$. *The desired* $\frac{1}{2}C''(\vec{X})$ *is then:*

$$|Cov(\mathcal{X}) + \epsilon_1 I|^{-1}\,\mathcal{I} + \epsilon_2 \mathcal{I} - 2\,|Cov(\mathcal{X}) + \epsilon_1 I|^{-2}\,\vec{X}\vec{X}^T$$

Here $\mathcal{I}$ *is a large identity matrix the size of* $\vec{X}\vec{X}^T$. *For convexity, we show the Hessian or a lower bound on it remains positive definite. Note* $-\vec{X}\vec{X}^T$ *is lower bounded by* $-\vec{X}^T\vec{X}\mathcal{I}$ *in the Loewner ordering sense. Also, note* $\vec{X}\vec{X}^T = tr(Cov(\mathcal{X}))$. *Thus, the Hessian is lower bounded by the following scalar times* $\mathcal{I}$:

$$|Cov(\mathcal{X}) + \epsilon_1 I|^{-1} + \epsilon_2 - 2\,|Cov(\mathcal{X}) + \epsilon_1 I|^{-2}\,tr(Cov(\mathcal{X}))$$

Rearranging and writing traces and determinants in terms of the (arbitrary yet non-negative) eigenvalues $\lambda_1, \ldots, \lambda_D$ *of* $Cov(\mathcal{X})$ *we get following non-negativity requirement:*

$$\frac{1}{2}\epsilon_2 \prod_d (\lambda_d + \epsilon_1)^2 + \frac{1}{2}\prod_d (\lambda_d + \epsilon_1) - \sum_d \lambda_d \quad \geq \quad 0$$

In the unidimensional case, there is one eigenvalue λ_1 *and the above is a quadratic which remains positive whenever* $\epsilon_1\epsilon_2 \geq 1/8$. *In the multidimensional case, observe the gradient* $\frac{\partial}{\partial \lambda_i}$ *of the left hand side of the above inequality:*

$$\frac{1}{2}\epsilon_2 \prod_{d \neq i} (\lambda_d + \epsilon_1)^2 (\lambda_i + \epsilon_1) + \frac{1}{2}\prod_{d \neq i} (\lambda_d + \epsilon_1) - 1$$

If $\epsilon_1 \geq 1$ *and* $\epsilon_2 \geq 1$ *the gradient stays positive. Thus, to minimize the left hand side of the bound, eigenvalues can move against the gradient and shrink to 0. However, the inequality is still satisfied at their lowest setting* $\lambda = \vec{0}$ *where the left hand side attains its non-negative minimum ensuring the function* $C(\mathcal{X})$ *is convex for* $\epsilon_1 \geq 1$ *and* $\epsilon_2 \geq 1$.

References

[1] D. Beymer and T. Poggio. Image representation for visual learning. *Science*, 272, 1996.

[2] T. Cootes, G. Edwards, and C. Talyor. Active appearance models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(6):681–685, 2001.

[3] R. Davies, C. Twining, T. Cootes, J. Waterton, and C. Taylor. 3d statistical shape models using direct optimisation of description length. In *European Conference on Computer Vision*, 2002.

[4] B. Frey and N. Jojic. Estimating mixture models of images and inferring spatial transformations using the EM algorithm. In *Computer Vision and Pattern Recognition*, 1999.

[5] S. Gold, C.P. Lu, A. Rangarajan, S. Pappu, and E. Mjolsness. New algorithms for 2D and 3D point matching: Pose estimation and correspondence. In *Neural Information Processing Systems 7*, 1995.

[6] D. Jakobson and I. Rivin. Extremal metrics on graphs. *Forum Math*, 14(1), 2002.

[7] T. Jebara. Convex invariance learning. In *Artificial Intelligence and Statistics 9*, 2003.

[8] M. Jones and T. Poggio. Hierarchical morphable models. In *Computer Vision and Pattern Recognition*, 1998.

[9] R. Kondor and T. Jebara. A kernel between sets of vectors. In *Machine Learning: Tenth International Conference*, 2003.

[10] J. Kosowsky and A. Yuille. The invisible hand algorithm: Solving the assignment problem with statistical physics. *Neural Networks*, 7:477–490, 1994.

[11] A.C.W. Kotcheff and C.J. Taylor. Automatic construction of eigenshape models by direct optimisation. *Medical Image Analysis*, 2(4):303–314, 1998.

[12] E. Miller, N. Matsakis, and P. Viola. Learning from one example through shared densities on transforms. In *Computer Vision and Pattern Recognition*, 2000.

[13] B. Moghaddam and A. Pentland. Probabilistic visual learning for object detection. In *International Conference on Computer Vision*, 1995.

[14] H. Murase and S. Nayar. Learning object models from appearance. In *Proceedings of the AAAI*, 1993.

[15] C. Nastar, B. Moghaddam, and A. Pentland. Generalized image matching: Statistical learning of physically-based deformations. In *European Conference on Computer Vision*, 1996.

[16] A. O'Toole, T. Vetter, and V. Blanz. Three-dimensional shape and two-dimensional surface reflectance contributions to face recognition: An application of three-dimensional morphing. *Vision Research*, 39, 1999.

[17] J. Platt. Using analytic QP and sparseness to speed training of support vector machines. In *Neural Information Processing Systems 11*, 1999.

[18] S. Sclaroff and A. Pentland. Modal matching for correspondence and recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(6), 1993.