A fast and accurate denoising technique for high-resolution image streams using data depth-based robust kernel PCA

Subhabrata Majumdar, Abhirup Mallik
School of Statistics
University of Minnesota - Twin Cities
Minneapolis, MN, USA
Email: majum010@umn.edu; malli066@umn.edu

***Abstract*—Classical kernel principal component analysis (PCA) has been shown to be quite sensitive to the presence of outliers, so many robust versions of kernel PCA exists in the literature. However, they suffer from lack of scalability with increasing amount of data and less accuracy compared to the classical version. Here we propose to use multivariate ranks obtained using data depth calculated in kernel spaces to get a robust version of kernel PCA applicable in big data scenarios, for example, denoising of large images. Given the kernel matrix, the extra calculation required here on top of classical kernel PCA does not grow with number of features, hence being more scalable. We provide theoretical properties related to the method, as well as demonstrate its effectiveness through simulations and a real data application. We also propose a scheme to outline how to use our method to check a continuous stream of images for noise, and denoise them if necessary.**

***Keywords*-Big data; image denoising; data depth; kernel PCA; robust PCA**

I. INTRODUCTION

We propose a fast, robust and efficient method of doing principal component analysis (PCA) in kernel spaces. This method extends upon the idea of doing PCA on multivariate rank vectors based on data depth in place of original data [1]. The time required for this modification does not increase with dimension of the feature space from which data are drawn, thus making it scalable for data with very high number of features. For this reason, the method will be specifically useful in denoising streams of high-resolution images in real time, which is a relevant problem in medical [2] and Geographic Information System (GIS) imaging [3]. It is also possible to adapt our method to existing algorithms of kernel PCA that are fast as well as have less time and memory complexity, like the Kernel Hebbian Algorithm [4] or Incremental Kernel PCA [5], to make them robust.

Image denoising techniques can be divided into two general classes. The first class of methods divide an image into blocks and use local blurring and often parallel computing using CPU or GPU [6]. These methods require increasing amount of computation time for high-resolution images. On the other hand, the second class of methods adapt robust linear PCA methods in kernel settings. These estimators typically have high breakdown points but suffer from poor efficiency compared to classical PCA [7], [8]. The above difficulties make both types of methods difficult to implement in real-time scenarios involving high-resolution images. Compared to them, our simple modification using data depth is capable of accurately reconstructing original images from projections of noisy images on kernel spaces, as well as robust and fast estimation of the underlying kernel subspace from batch-wise processing of incoming data.

Data depth provides a scalar measure for the 'center outward ordering' of a multivariate point with respect to some data cloud [9]. Most depth functions available in the literature generate convex contours irrespective to the shape of the data cloud, and are unsuitable for distributions that have more than one mode or non-convex support [10], which are often encountered in real life. Moreover, calculation of depth functions are often computationally expensive. These two reasons make depth-based methods unsuitable for application in big data scenario. Here we bypass both difficulties by using kernelized spatial depth [11], that respects the local geometry around a data point and given the sample kernel matrix, calculates its depth in time that increases quadratically with sample size, irrespective of the number of features.

This article is organized as follows. Section II reviews the theoretical groundwork required for our analysis by introducing key concepts related to PCA, kernel methods and data depth. Based on these concepts, in section III we obtain theoretical results related to depth-based kernel PCA, explicitly deriving the modified kernel matrix and score vectors corresponding to our method. Sections IV and V demonstrate the performance of our method through simulations and a real data example, respectively. In section VI we introduce a quality index measuring the amount of noise in an incoming image, and based on that provide an algorithm for batch-wise processing and optional denoising of a stream of high-resolution images using our method. We end this paper by providing the summary of our work and motivating further areas of application in section VII. The appendix supplies proofs of the results we state in the main content.

II. Preliminaries

A. Kernel Principal Component Analysis

Given a random variable $\mathbf{X}$ with mean $\mathbf{0}$ taking values in $\mathbb{R}^p$, principal components are defined as columns of the orthogonal matrix $\mathbf{W} = (\mathbf{w}_1, ..., \mathbf{w}_p)^T$ so that

$$\mathbf{w}_1 = \underset{\|\mathbf{w}\|=1}{\arg\min}\, \mathbf{w}^T \mathbf{X}^T \mathbf{X} \mathbf{w} \tag{1}$$

$$\mathbf{w}_k = \underset{\|\mathbf{w}\|=1}{\arg\min}\, \mathbf{w}^T \hat{\mathbf{X}}_k^T \hat{\mathbf{X}}_k \mathbf{w} \tag{2}$$

$$\text{with } \hat{\mathbf{X}}_k = \mathbf{X} - \sum_{s=1}^{k-1} \mathbf{w}_s \mathbf{w}_s^T \mathbf{X} \text{ for } k = 2, ..., p$$

The vectors $\mathbf{w}_k$ turn out to be eigenvectors of the covariance matrix of $\mathbf{X}$ ordered according to the corresponding decreasing eigenvalues. This change of coordinates works well in dimension reduction when the data lies in a linear subspace of dimension lower than feature space, but not so much for data with inherent nonlinearity among variables. In this scenario, we use kernel PCA [12] instead. We map the data to a new (potentially higher dimensional) feature space through a nonlinear transformation $\phi : \mathbb{R}^p \mapsto \mathbb{R}^D$, and perform standard PCA on the transformed variables.

To bypass explicit computation of the function ϕ, we apply the 'kernel trick'. Given n random draws $\mathbf{X}_1, ..., \mathbf{X}_n$, the k^{th} eigen vector ($1 \leq k \leq P$) in the transformed space can be written as a linear combination of the transformed samples: $\mathbf{v}_k = \sum_{i=1}^n a_{ki} \phi(\mathbf{x}_i)$, and given centered transformations, i.e. $\sum_{i=1}^n \phi(\mathbf{x}_i) = \mathbf{0}$, we obtain the vector of coefficients $\mathbf{a}_k = (a_{k1}, ..., a_{kn})^T$ as solution of the following equation:

$$K \mathbf{a}_k = n \lambda_k \mathbf{a}_k \tag{3}$$

using the kernel function induced by the transformation: $k(\mathbf{a}, \mathbf{b}) = \phi(\mathbf{a})^T \phi(\mathbf{b})$, for $\mathbf{a}, \mathbf{b} \in \mathbb{R}^p$, so that $K_{n \times n}$ has $(i, j)^{\text{th}}$ element $k(\mathbf{x}_i, \mathbf{x}_j)$ and λ_k is the k^{th} eigenvalue of the covariance matrix of transformed observations, i.e. $E(\phi(\mathbf{X})\phi(\mathbf{X})^T)$.

Following this, we can obtain the k^{th} principal component score for $\mathbf{x} \in \mathbb{R}$:

$$s_k(\mathbf{x}) = \mathbf{v}_k^T \phi(\mathbf{x}) = \sum_{i=1}^n a_{ki} k(\mathbf{x}, \mathbf{x}_i) \tag{4}$$

Some frquently used kernels in this context are:

- *Gaussian kernel*: $k(\mathbf{a}, \mathbf{b}) = \exp(-\|\mathbf{a} - \mathbf{b}\|^2 / 2\sigma); \sigma > 0$
- *Polynomial kernel*: $k(\mathbf{a}, \mathbf{b}) = (\mathbf{a}^T \mathbf{b})^d; d > 0$
- *Sigmoid kernel*: $k(\mathbf{a}, \mathbf{b}) = \tanh(\eta \mathbf{a}^T \mathbf{b} + \nu)$

Note that setting $d = 1$ in a polynomial kernel results in the usual inner product, or the so-called uniform kernel.

B. Depth functions, and kernel depth

Given a data cloud or a probability distribution, a depth function is any real-valued function that measures the outlyingness of a point in feature space with respect to the data or its underlying distribution. In order to formalize the notion of depth, we consider as data depth any scalar-valued function $D(\mathbf{x}, F_{\mathbf{X}})$ (where $\mathbf{x} \in \mathbb{R}^p$, and the random variable $\mathbf{X}$ has distribution F) that satisfies the following properties [13]:

- *P1. Affine invariance*: $D(A\mathbf{x} + \mathbf{b}, F_{A\mathbf{X}+\mathbf{b}}) = D(\mathbf{x}, F_{\mathbf{X}})$ for any $p \times p$ non-singular matrix A and $p \times 1$ vector $\mathbf{b}$;
- *P2. Maximality at center*: When $F_{\mathbf{X}}$ has center of symmetry $\boldsymbol{\theta}$, $D(\boldsymbol{\theta}, F_{\mathbf{X}}) = \sup_{\mathbf{x} \in \mathbb{R}^p} D(\mathbf{x}, F_{\mathbf{X}})$. Here the symmetry can be central, angular or halfspace symmetry;
- *P3. Monotonicity relative to deepest point*: For any $p \times 1$ vector $\mathbf{x}$ and $\alpha \in [0, 1]$, $D(\mathbf{x}, F_{\mathbf{X}}) \leq D(\boldsymbol{\theta} + a(\mathbf{x} - \boldsymbol{\theta}))$;
- *P4. Vanishing at infinity*: As $\|\mathbf{x}\| \to \infty$, $D(\mathbf{x}, F_{\mathbf{X}}) \to 0$.

Among depth functions, spatial depth is especially attractive because of its computationally frugal nature (time complexity $O(n^2)$). The spatial depth of a point $\mathbf{x} \in \mathbb{R}^p$ with respect to a distribution F is defined as:

$$D(\mathbf{x}, F) = 1 - \left\| \int \mathbf{S}(\mathbf{y} - \mathbf{x}) dF(\mathbf{y}) \right\| \tag{5}$$

where $\mathbf{S}(\mathbf{x}) = \mathbf{x} / \|\mathbf{x}\|$ is the spatial sign functional. For n random draws $\mathcal{X} = \{\mathbf{x}_1, ..., \mathbf{x}_n\}$ we estimate the population spatial depth by its sample version, in which the integral is replaced by average over all sample points. Its kernelized version [11] defined below allows us fast computation of depths *and* depth contours that respect the shape of the data cloud.

Definition 1 (Chen *et al* [11])**.** Consider any kernel function k. Then in the above setup, the *sample kernelized spatial depth* for any $\mathbf{x} \in \mathbb{R}^p$ is defined as:

$$D_k(\mathbf{x}, \mathcal{X}) = 1 - \frac{1}{|\mathcal{X} \cup \mathbf{x}| - 1} \times \left(\sum_{\mathbf{y}, \mathbf{z} \in \mathcal{X}} \frac{k(\mathbf{x}, \mathbf{x}) + k(\mathbf{y}, \mathbf{z}) - k(\mathbf{x}, \mathbf{y}) - k(\mathbf{x}, \mathbf{z})}{\delta_k(\mathbf{x}, \mathbf{y}) \delta_k(\mathbf{x}, \mathbf{z})} \right)^{1/2} \tag{6}$$

where $\delta_k(\mathbf{x}, \mathbf{y}) = \sqrt{k(\mathbf{x}, \mathbf{x}) + k(\mathbf{y}, \mathbf{y}) - 2k(\mathbf{x}, \mathbf{y})}$.

Subsequently, we can easily infer the following:

Proposition 1. *For any transformation ϕ as considered above, and k being the kernel induced by the transformation, consider random samples $\mathcal{X}$ and an arbitrary point $\mathbf{x} \in \mathbb{R}^p$. Then we have*

$$D_k(\mathbf{x}, \mathcal{X}) = D(\phi(\mathbf{x}), \phi(\mathcal{X})) \tag{7}$$

when D is spatial depth and D_k is kernelized spatial depth.

III. Depth-based PCA and its kernelized version

When the underlying data distribution is elliptic, data depth can be used to estimate population eigenvectors in a robust and fairly efficient fashion [1]. For this, suppose the distribution is centered at $\boldsymbol{\mu} \in \mathbb{R}^p$ and any depth function D. Then we define the multivariate rank vector $\tilde{\mathbf{x}} = \tilde{D}(\mathbf{x}, F_{\mathbf{X}})(\mathbf{x} - \boldsymbol{\mu})/\|\mathbf{x} - \boldsymbol{\mu}\|$, where $\tilde{D}(\mathbf{x}, F_{\mathbf{X}}) = \exp(-D(\mathbf{x}, F_{\mathbf{X}}))$ is the outlyingness or *htped function* corresponding to D. After this we simply do PCA on these vectors in place of original data vectors.

Here we assume that the transformed random variable $\phi(\mathbf{X})$ follows an elliptical distribution, centered at $\boldsymbol{\theta} = \phi(\boldsymbol{\mu})$. In this setup multivariate ranks are defined in the new feature space:

$$\begin{aligned} \phi_D(\mathbf{x}) &= \tilde{D}(\phi(\mathbf{x}), \phi(\mathcal{X})).\mathbf{S}(\phi(\mathbf{x}) - \phi(\boldsymbol{\mu})) \\ &= \tilde{D}_k(\mathbf{x}, \mathcal{X}) \frac{\phi(\mathbf{x}) - \phi(\boldsymbol{\mu})}{\sqrt{k(\mathbf{x} - \boldsymbol{\mu}, \mathbf{x} - \boldsymbol{\mu})}} \end{aligned} \quad (8)$$

When the center is unknown, following the case of spherical kernel PCA [14] i.e. kernel PCA on spatial sign vectors in the feature space, we estimate it by the spatial median, i.e. solution $\hat{\boldsymbol{\theta}}$ to the following equation:

$$\sum_{i=1}^n \frac{\phi(\mathbf{x}_i) - \hat{\boldsymbol{\theta}}}{\|\phi(\mathbf{x}_i) - \hat{\boldsymbol{\theta}}\|} = \mathbf{0} \quad (9)$$

This is calculated using a simple iterative algorithm and the kernel function as a linear combination of the samples: $\hat{\boldsymbol{\theta}} = \sum_{i=1}^n \gamma_i \phi(\mathbf{x}_i)$. Finally, we obtain the kernel matrix K_D induced by the transformation ϕ_D:

Theorem 1. *Given that the center of the transformed distribution is estimated by the spatial median as above, the matrix K_D with $k_D(\mathbf{x}_i, \mathbf{x}_j) := \phi_D(\mathbf{x}_i)^T \phi_D(\mathbf{x}_j)$ in its $(i,j)^{th}$ position has the form:*

$$K_D = \Lambda_D K_\Gamma \Lambda_D; \quad K_\Gamma = K + \Gamma^T K \Gamma - \Gamma^T K - K\Gamma \quad (10)$$

where Γ is a $n \times n$ matrix with $\boldsymbol{\gamma} = (\gamma_1, ..., \gamma_n)^T$ as each of its columns, and Λ_D is a diagonal matrix with diagonal elements $\lambda_{D,ii} := \tilde{D}_k(\mathbf{x}_i, \mathcal{X})/\sqrt{K_{\Gamma,ii}}$.

We now perform PCA on the multivariate ranks in the transformed space, i.e. $\{\phi_D(\mathbf{x}_1), ..., \phi_D(\mathbf{x}_n)\}$. Given that the estimate of the k^{th} eigenvector is $\mathbf{v}_k^D = \sum_{i=1}^n a_{ki}^D \phi_D(\mathbf{x}_i)$, the score due to k^{th} principal component is:

$$\begin{aligned} s_k^D(\mathbf{x}) &= \sum_{i=1}^n a_{ki}^D \phi_D(\mathbf{x}_i)^T \left[\phi(\mathbf{x}) - \hat{\boldsymbol{\theta}}\right] \\ &= \sum_{i=1}^n a_{ki}^D \lambda_{D,ii} \times \\ & \left[k(\mathbf{x}, \mathbf{x}_i) + \boldsymbol{\gamma}^T K \boldsymbol{\gamma} - k(\mathbf{x}, \hat{\boldsymbol{\theta}}) - k(\mathbf{x}_i, \hat{\boldsymbol{\theta}})\right] \end{aligned} \quad (11)$$

using the expansion of $\hat{\boldsymbol{\theta}}$.

IV. Simulations

A. Effect of the number of features

We first set up our simulation study following the framework used by Mika *et al* [15]. We generated a data set of eleven Gaussians in $\mathbb{R}^{10}$ with means chosen randomly in $[-1, 1]^{10}$, and variance $\sigma^2 = 1$ in each component. We took training and test sets of size 100 and 33 for each of distributions, and carried out three different kinds of Kernel PCA: classical (kCPCA), spherical (kSPCA) and depth-based (kDPCA). We used the principal components from the training data to recover the points in the test data and calculated the mean squared errors (MSE) in the recovered points and the original points using different number of components. The result is shown in Figure 1. We can see as more components are being used, the MSE in general decreases as expected. However kDPCA performs relatively better than all other methods when we are using fewer components.

B. Effect of number of outliers

We now recreate the simulation setup used by Yang *et al* [16]. The purpose of this experiment is to explore the nature of robustness between several version of robust PCA. The data set used is a two dimensional synthetic dataset $\mathcal{X} : \{x_i, y_i\}_{i=1}^n$ (we used $n = 101$) where x_i were generated from $U[0,1]$. We obtained the responses as $y_i = 0.5\sin(2\pi x_i) + 0.5 + 0.1 r_i$, with $r_i \sim N(0,1)$ being independently generated standard gaussian noise. Outliers were generated from $N(5,1)$ distribution. The ratio of the data to outliers were varied from 20% to 50%. To measure robustness, we compared first eigenvector estimates on the original data using kCPCA and corrupted data using kSPCA and kDPCA using absolute cosine distance, which is defined below:

$$\text{distance}(\boldsymbol{\gamma}_1, \boldsymbol{\gamma}_2) = \cos^{-1} |\boldsymbol{\gamma}_1^T \boldsymbol{\gamma}_2|$$

We compare how the number of outliers affect the principal components based on this distance in figure 2. It seems that as number of outliers increase, the distance between eigenvectors increases at first, then they decrease. Here in this set up after using 18 outliers, we see an decrease. However in all cases, depth-based PCA had lesser distance to the PC from kernel PCA.

V. Application: fast and accurate image denoising

We use the extended Yale Face Database B ([17], [18]) in this example to demonstrate efficacy of our method. We compare its outputs with the same from classical kernel PCA and spherical kernel PCA. The score vectors returned by any kernel PCA method reside in the transformed space, so to apply kernel PCA in image processing, it is imperative to get back pre-images in the original space corresponding to these score vectors. We use the fast and stable method proposed by Kwok and Tsang that approximates pre-images by the multivariate basis spanned by a fixed number (k) of nearest sample points in the feature space [19]. For our purpose we fix $k = 5$. We use gaussian kernel as our chosen kernel function, and obtain the optimal bandwidth $\hat{\sigma}$ as:

$$\frac{1}{2\hat{\sigma}} = \frac{1}{n(n-1)} \sum_{i=1}^{n} \sum_{j=1}^{n} \|\mathbf{x}_i - \mathbf{x}_j\|^2 \tag{12}$$

Here we take 20 grayscale images each for three persons (samples yaleB01, yaleB02 and yaleB28, to be exact) under the same order of lighting conditions. From here on we shall refer to them as person 1, 2 and 3, respectively. We choose the persons so that their facial features are very different. We read in the images in their original resolution (192×168). After this we select 25% pixels at random from 7th images for each person and turn their pixel values to 0 or 1 with probability 0.5. Known as *speckle noise*, this type of noise is common in several imaging techniques, and causes degradation of image quality as well as bad performance of image classification algorithms [20]. After addition of noise, we take each of the 60 images as a vector of length $192 \times 168 = 32256$, extract kernel principal components from the 60×32256 dataset by the three methods under consideration and feed the 3 corrupted images into the models in order to denoise them.

A. Image reconstruction

Panel (a) of figure 3 shows the three original images and their corrupted versions, and panels (b), (c) and (d) show their denoised versions. The five rows correspond to number of kernel principal components used in reconstruction (2, 4, 6, 8 or 10), while the three columns show reconstructions obtained by kCPCA, kSPCA and kDPCA, respectively. In general, kDPCA performs best among the methods for all three images. The method of reconstruction we consider here approximates pre-image of some projection by a basis spanned by only 5 among 60 sample points. An inaccurate estimation of underlying kernel principal components due to one or more corrupted sample point(s) can result in images that belong to different persons being selected among nearest neighbors of the projection of a corrupted image. Here we see that kSPCA suffers from this in all three images. For person 1, reconstructions using 2 and 4 spherical kernel PCs obtain an image belonging to person 2, while all reconstructions of persons 2 and 3 using this method gives back images of person 1 and 2 instead. This reiterates on the fact that non-kernel spherical PCA is considerably less efficient than non-kernel classical PCA when the underlying data distribution is multivariate normal ([1], [21]).

The kCPCA denoisings are comparatively better, except for images 4 and 5 for person 3. Denoised images of persons 1 and 2 turn out to be darker than corresponding original images. kDPCA does not suffer from any of these disadvantages, and accurately denoises all three corrupted images.

B. Signal-to-Noise Ratio comparison

We also calculate the Signal-to-Noise Ratios (SNR) of these images and compare them to those of corresponding noisy images. Here we consider a noisy image as signal, and given the original image, pixel-wise difference of the noisy and original images is taken as noise. SNR is simply defined as ratio of the mean pixel value in signal image and standard deviation of noise. Higher SNR of a denoised image compared to noisy image corresponds to a better denoising.

In table I we list the calculated SNRs. Among the 15 denoisings we consider here, kDPCA has highest SNR among 3 methods for 8 of them. Almost all kDPCA denoisings (except 2 PC denoising for person 2) have higher SNRs than original noisy images. Some more interesting observations are:

- For person 1, kDPCA consistently provides very high SNR.
- For person 2, using 10 PCs and kDPCA gives very good denoising, which is also evident from the corresponding image in figure 3.
- For person 3, using a higher number of PCs deteriorates the performance of kCPCA, but kDPCA still gives higher SNR than noisy image.

Note that a high SNR does not necessarily translate to an accurate reconstructed image in our scenario. Images of person 2 obtained by kCPCA are darker than the original image in general, but still have highest SNR when 4, 6 or 8 PCs are used. For the same person, when using 2 PCs kSPCA gives highest SNR, although in the plot we see that it returns person 1's image instead.

VI. Real-time denoising of high-resolution image sequences

Since kDPCA does not suffer from the dreaded 'curse of dimensionality', its main advantage lies in application on data with very high amount of features, which in our context can be pixel data from high-resolution images. Based on the above outputs, we propose below a scheme of denoising an incoming stream of images using kDPCA. For an image with high amount of noise, we can expect a large difference of pixel values between this image and its denoised version using kDPCA. Keeping this mind, we define the following to quantify the difference:

Definition 2. Given two images, say I_0 and I_1, the *Mean Standardized difference* (MSD) between the images is defined as

$$\text{MSD}(I_0, I_1) = \frac{|\text{mean}(v(I_0) - v(I_1))|}{\text{sd}(v(I_0) - v(I_1))} \tag{13}$$

where for some image I, $v(I)$ is a vector containing all pixel values in I.

To do real-time image denoising, we first use MSD values for a small set of initial training images and use a fixed quantile of these values (for example, 95% or 99%) as threshold. After that we consider a smaller block of incoming images, denoise and calculate their MSD, and compare them with the threshold obtained from the training sample. For images whose MSD values exceed the threshold we store the denoised image instead of the original image. Finally we update the training set by including 'good' test images, i.e. test images that have below-threshold MSD, and discarding equal number of images from the start of the old training set. After this we take in another block of incoming image and repeat the process. A formal sketch of the above scheme is given in algorithm 1.

We choose not to include the denoised images in training samples because their MSD values are extreme compared to those of other, 'good' images in training set. To build a kDPCA model, the only extra calculation compared to an (often inefficient) kSPCA model we need to do is due to calculating kernelized depth values of samples. In an updated train sample one can either calculate depths of all N points, which has time complexity $O(N^2)$; or only calculate depths for new training samples and recycle depths for other points. In this case the extra computation time required will be only $O(n_k^2)$, where n_k is the number of new training samples introduced.

Algorithm 1 Algorithm for fast real-time image denoising using kDPCA

```
 1: procedure KdpcaDenoise(stream of images $\{I_1, I_2, ...\}$)
 2:     Fix integers $N, n \in \mathbb{N}, N > n$; cutoff probability $c$.
 3:     Initialize training set of images $T = \{I_1, ..., I_N\}$.
 4:     Set test set starting position $p = 0$.
 5:     train:
 6:     Use $T$ to build a kDPCA model $\mathcal{M}_T$.
 7:     Denoise images in $T$ using $\mathcal{M}_T$ and get $T' = \{I'_1, ..., I'_N\}$.
 8:     Calculate their MSDs
                $m_i := \text{MSD}(I_i, I'_i)$
        for $i = 1, 2, ..., N$.
 9:     Set $C = c^{\text{th}}$ quantile of $m_1, ..., m_N$.
10:     test:
11:     Read in test set of images $S = \{I_{N+p+1}, ..., I_{N+p+n}\}$.
12:     Use $\mathcal{M}_T$ to denoise images in $S$ and get $S' = \{I'_{N+p+1}, ..., I'_{N+p+n}\}$.
13:     Get their MSDs: $m_{N+p+1}, ..., m_{N+p+n}$.
14:     for $i$ in $1 : n$ do
15:         if $m_{N+p+i} > C$ then
16:             Set $I_{N+p+i} \leftarrow I'_{N+p+i}$
17:     update:
18:     Set $j = 1$.
19:     for $i$ in $1 : n$ do
20:         if $m_{N+p+i} < C$ then
21:             Set $T \leftarrow T - \{I_j\} \cup \{I_{N+p+i}\}$
22:             Set $j \leftarrow j + 1$
23:     Set $p \leftarrow p + n$, goto train
```

VII. Conclusion

In the above sections we outline a robust version of kernel PCA that is fast, provides accurate reconstructions, and most importantly can be scaled to denoise data streams with non-regular shape features that have large amount of features as well as samples. As extension, we think that since the quantity MSD adapts to the constantly updated training data, it can be used as a tool for anomaly or changepoint detection in similar type of data. Also the estimation and reconstruction in kDPCA using gaussian kernels can potentially be improved by using diagonal or positive definite matrices as the tuning covariance matrix instead of a single scalar quantity σ.

Appendix

Proof of proposition 1: For the transformed data in $\mathbb{R}^D$, the sample spatial depth can be written as [11]:

$$D(\phi(\mathbf{X}), \phi(\mathcal{X})) = 1 - \frac{1}{|\mathcal{X} \cup \mathbf{x}| - 1} \left\| \sum_{\mathbf{y} \in \mathcal{X}} \mathbf{S}(\phi(\mathbf{y}) - \phi(\mathbf{x})) \right\|$$

Next we break down the square of the norm above and rewrite using only kernel inner products, making it free of ϕ hence estimable from the given data:

$$\begin{aligned}
&\left\|\sum_{\mathbf{y}\in\mathcal{X}} \mathbf{S}(\phi(\mathbf{y})-\phi(\mathbf{x}))\right\|^2 \\
=&\sum_{\mathbf{y},\mathbf{z}\in\mathcal{X}} (\mathbf{S}(\phi(\mathbf{x})-\phi(\mathbf{y})))^T\mathbf{S}(\phi(\mathbf{x})-\phi(\mathbf{z})) \\
=&\sum_{\mathbf{y},\mathbf{z}\in\mathcal{X}} \frac{(\phi(\mathbf{x})-\phi(\mathbf{y}))^T(\phi(\mathbf{x})-\phi(\mathbf{z}))}{\|\phi(\mathbf{x})-\phi(\mathbf{y})\|\|\phi(\mathbf{x})-\phi(\mathbf{z})\|} \\
=&\sum_{\mathbf{y},\mathbf{z}\in\mathcal{X}} \frac{\phi(\mathbf{x})^T\phi(\mathbf{x})+\phi(\mathbf{y})^T\phi(\mathbf{z})-\phi(\mathbf{x})^T\phi(\mathbf{y})-\phi(\mathbf{x})^T\phi(\mathbf{z})}{\|\phi(\mathbf{x})-\phi(\mathbf{y})\|\|\phi(\mathbf{x})-\phi(\mathbf{z})\|} \\
=&\sum_{\mathbf{y},\mathbf{z}\in\mathcal{X}} \frac{k(\mathbf{x},\mathbf{x})+k(\mathbf{y},\mathbf{z})-k(\mathbf{x},\mathbf{y})-k(\mathbf{x},\mathbf{z})}{\delta_k(\mathbf{x},\mathbf{y})\delta_k(\mathbf{x},\mathbf{z})}
\end{aligned}$$

and the result follows from the definition of $D_k(\mathbf{x},\mathcal{X})$. ■

Proof of Theorem 1: K_Γ is essentially the kernel matrix corresponding to all observations centered at the spatial median. To see that, we look at the $(i,j)^{\text{th}}$ element of the kernel matrix for centered observations:

$$\begin{aligned}
&k(\mathbf{x}_i-\boldsymbol{\mu},\mathbf{x}_j-\boldsymbol{\mu}) \\
=&k(\mathbf{x}_i,\mathbf{x}_j)-k(\mathbf{x}_i,\boldsymbol{\mu})-k(\mathbf{x}_j,\boldsymbol{\mu})+k(\boldsymbol{\mu},\boldsymbol{\mu}) \\
=&k(\mathbf{x}_i,\mathbf{x}_j)+\sum_{k=1}^n\sum_{l=1}^n \gamma_k\gamma_l k(\mathbf{x}_k,\mathbf{x}_l) \\
&-\sum_{k=1}^n \gamma_k k(\mathbf{x}_i,\mathbf{x}_k)-\sum_{k=1}^n \gamma_k k(\mathbf{x}_j,\mathbf{x}_k)
\end{aligned}$$

The first term is $(i,j)^{\text{th}}$ element of K, second term is constant for all (i,j), and the third and fourth terms are cross-products of $\boldsymbol{\gamma}$ with i^{th} and j^{th} rows (or columns) of K, respectively. Accumulating all terms and writing them in matrix notation we get the expansion for K_Γ in 10.

Now we only need to expand $k_D(\mathbf{x}_i,\mathbf{x}_j)$:

$$\begin{aligned}
&\phi_D(\mathbf{x}_i)^T\phi_D(\mathbf{x}_j) \\
=&\tilde{D}_k(\mathbf{x}_i,\mathcal{X})\tilde{D}_k(\mathbf{x}_j,\mathcal{X})\times \\
&\frac{k(\mathbf{x}_i-\boldsymbol{\mu},\mathbf{x}_j-\boldsymbol{\mu})}{\sqrt{k(\mathbf{x}_i-\boldsymbol{\mu},\mathbf{x}_i-\boldsymbol{\mu})}\sqrt{k(\mathbf{x}_j-\boldsymbol{\mu},\mathbf{x}_j-\boldsymbol{\mu})}} \\
=&\frac{\tilde{D}_k(\mathbf{x}_i,\mathcal{X})}{\sqrt{K_{\Gamma,ii}}}K_{\Gamma,ij}\frac{\tilde{D}_k(\mathbf{x}_j,\mathcal{X})}{\sqrt{K_{\Gamma,jj}}}
\end{aligned}$$

to conclude the needed. ■

Acknowledgment

This research is partially supported by the United States National Science Foundation grant IIS–1029711. We thank our doctoral adviser Prof. Snigdhansu Chatterjee for his valuable guidance.

References

[1] S. Majumdar and S. Chatterjee, "Robust estimation of principal components from depth-based multivariate rank covariance matrix," 2015, http://arxiv.org/abs/1502.07042.

[2] A. Ouahabi, "A review of wavelet denoising in medical imaging," in *Signal Processing and their Applications (WoSSPA), 2013 8th International Workshop on Systems*. IEEE, 2013, pp. 19–26.

[3] Z. Ye and H. Mohamadian, "Digital Image Processing for Spatial Object Recognition via Integration of Nonlinear Wavelet-Based Denoising and Clustering-Based Segmentation," in *Advances in Spatial Data Handling and GIS: 14th International Symposium on Spatial Data Handling*, ser. Lecture Notes in Geoinformation and Cartography, W. Shi, A. G. O. Yeh, Y. Leung, and C. Zhou, Eds. Springer, 2012.

[4] K. I. Kim, M. O. Franz, and B. Schölkopf, "Iterative Kernel Principal Component Analysis for Image Modeling," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 9, pp. 1351–1366, 2005.

[5] S. Ozawa, Y. Takeuchi, and S. Abe, "A Fast Incremental Kernel Principal Component Analysis for Online Feature Extraction," in *PRICAI 2010: Trends in Artificial Intelligence*, ser. Lecture Notes in Computer Science. Springer, 2010, vol. 6230, pp. 487–497.

[6] A. Kharlamov and V. Podlozhnyuk, *Image Denoising*, NVIDIA, Jun 2007, v1.0: http://developer.download.nvidia.com/compute/cuda/1.1-Beta/x86_website/projects/imageDenoising/doc/imageDenoising.pdf.

[7] S. Taskinen, I. Koch, and H. Oja, "Robustifying principal component analysis with spatial sign vectors," *Statist. and Prob. Lett.*, vol. 82, pp. 765–774, 2012.

[8] M. Hubert, P. J. Rousseeuw, and K. V. Branden, "ROBPCA: A New Approach to Robust Principal Component Analysis," *Technometrics*, vol. 47-1, pp. 64–79, 2005.

[9] Y. Zuo and R. Serfling, "General notions of statistical depth functions," *Ann. Statist.*, vol. 28-2, pp. 461–482, 2000.

[10] D. Paindaveine and G. Van Bever, "From depth to local depth: A focus on centrality," *J. Amer. Stat. Assoc.*, vol. 108, no. 503, pp. 1105–1119, 2013.

[11] Y. Chen, X. Dang, H. Peng, and H. L. Bart Jr., "Outlier Detection with the Kernelized Spatial Depth Function," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, no. 2, pp. 288–305, 2009.

[12] B. Schölkopf, A. Smola, and K.-R. Müller, *Kernel principal component analysis*. MIT Press, 1999, ch. Advances in Kernel Methods Support Vector Learning, pp. 327–352.

[13] R. Liu, "On a notion of data depth based on random simplices," *Ann. Statist.*, vol. 18, pp. 405–414, 1990.

[14] M. Debruyne, M. Hubert, and J. V. Horebeek, "Detecting Influential Observations in Kernel PCA," *Comput. Stat. Data An.*, vol. 54, no. 12, pp. 3007–3019, 2010.

[15] S. Mika, B. Schölkopf, A. Smola, K. Müller, M. Scholz, and G. Rätsch, “Kernel PCA and De-Noising in Feature Spaces,” *Adv. Neural Inf. Process. Syst.*, pp. 536–542, 1999.

[16] S. Yang, H. Shen, J. Meng, and Z. Shen, “A Fixed-point Iteration Algorithm for Robust Kernel Principal Component Analysis,” *J. Comput. Inf. Syst.*, vol. 10, no. 17, pp. 7489–7497, sep 1999.

[17] A. Georghiades, P. Belhumeur, and D. Kriegman, “From Few to Many: Illumination Cone Models for Face Recognition under Variable Lighting and Pose,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 23, no. 6, pp. 643–660, 2001.

[18] K. Lee, J. Ho, and D. Kriegman, “Acquiring Linear Subspaces for Face Recognition under Variable Lighting,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 5, pp. 684–698, 2005.

[19] J. T. Kwok and I. W. Tsang, “The Pre-Image Problem in Kernel Methods,” *IEEE Trans. Neural Net.*, vol. 15, no. 6, pp. 1517–1525, 2004.

[20] F. Qiu, J. Berglund, J. R. Jensen, P. Thakkar, and D. Ren, “Speckle Noise Reduction in SAR Imagery Using a Local Adaptive Median Filter,” *GISci. Remote Sens.*, vol. 41, no. 3, pp. 244–266, 2004.

[21] A. Magyar and D. Tyler, “The asymptotic inadmissibility of the spatial sign covariance matrix for elliptically symmetric distributions,” *Biometrika*, vol. 101, pp. 673–688, 2014.

Image	No. of kernel PCs	SNR		
		kCPCA	kSPCA	kDPCA
Corrupted Image 1, SNR = 1.94	2	1.26	1.48	**4.99**
	4	1.16	1.56	**5.81**
	6	1.42	3.03	**5.80**
	8	1.55	**6.33**	6.30
	10	1.55	3.30	**8.05**
Corrupted Image 2, SNR = 1.51	2	1.67	**2.13**	1.46
	4	**1.97**	1.01	1.60
	6	**1.91**	1.05	1.56
	8	**1.97**	1.48	1.60
	10	1.65	1.47	**5.72**
Corrupted Image 2, SNR = 1.44	2	1.65	0.90	**1.73**
	4	**2.31**	1.11	2.13
	6	**2.31**	1.04	1.82
	8	0.85	1.46	**2.01**
	10	0.80	0.95	**1.63**

Table I: Signal-to-noise ratios of reconstructions by 3 methods of kernel PCA

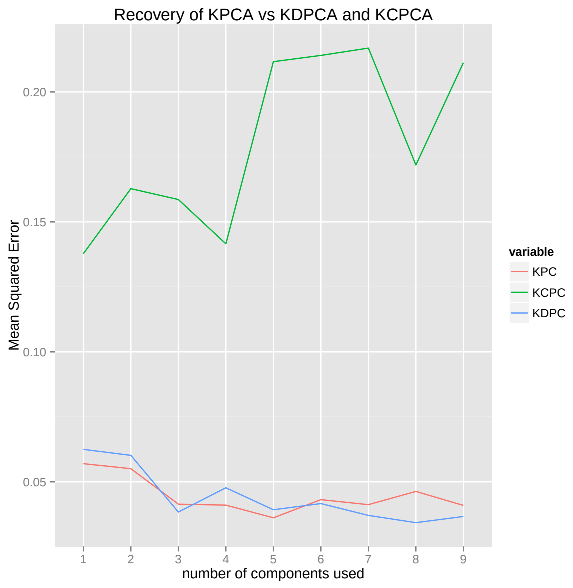


Figure 1: Mean squared error after recovery of kCPCA, kSPCA and kDPCA

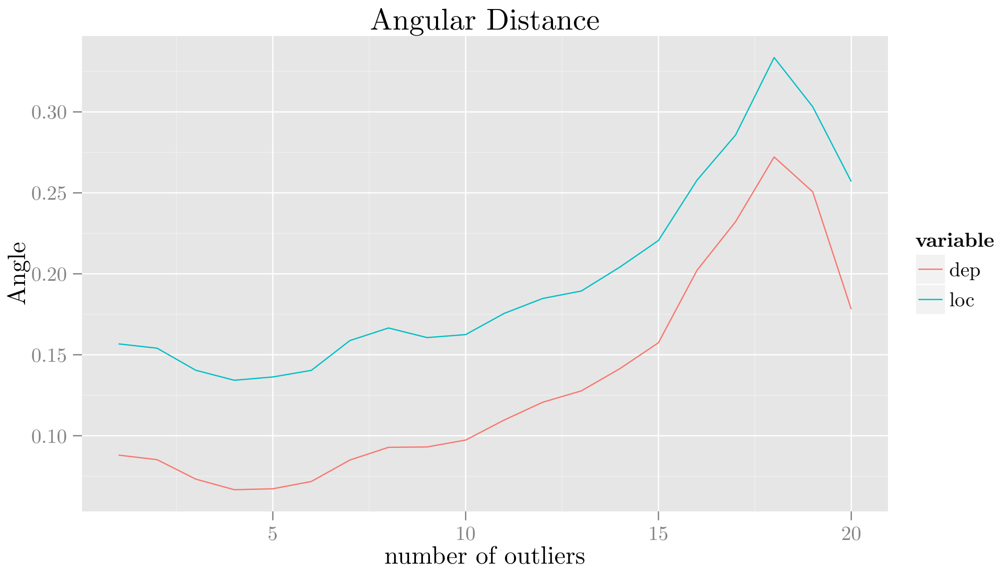
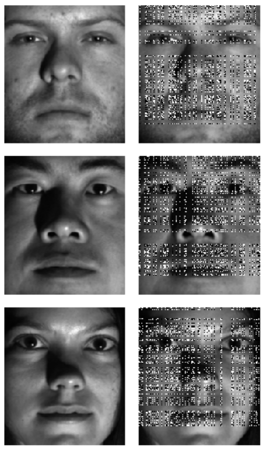
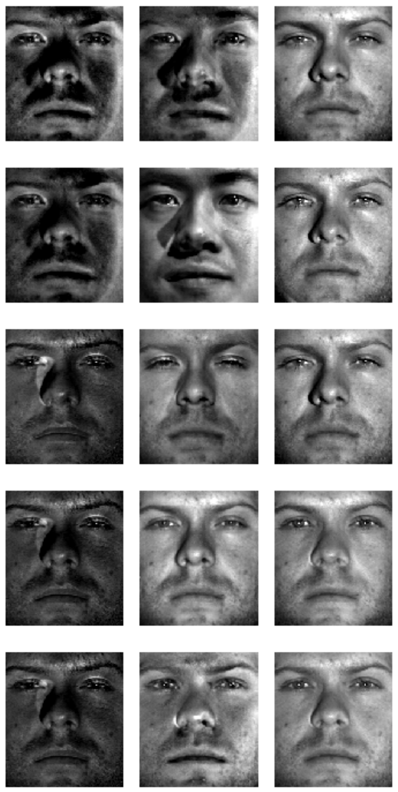


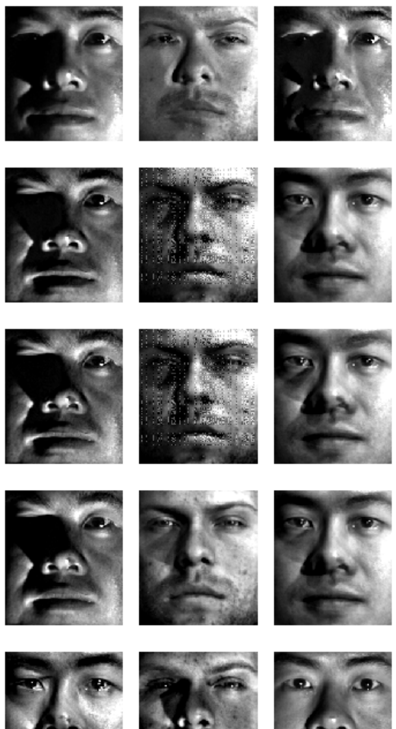
Figure 2: Cosine distance measure of the principle components of kCPCA and kDPCA



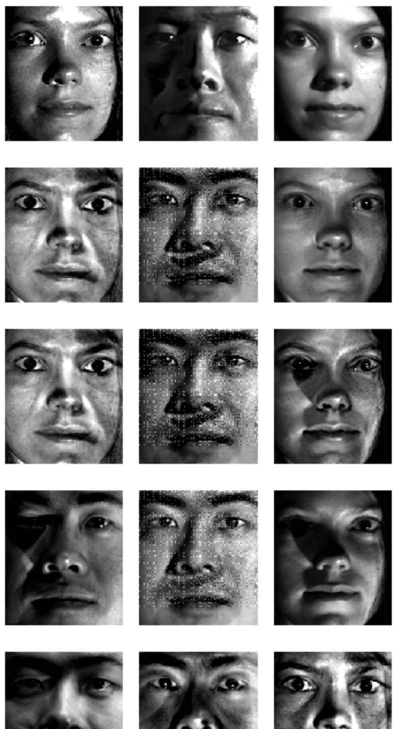
(a)



(b)



(c)



(d)

Figure 3: (a) Original and contaminated images for three persons considered,
(b), (c) and (d) Reconstructions by kCPCA, kSPCA and kDPCA (in columns) for persons 1, 2 and 3 respectively