Image Structure Subspace Learning Using Structural Similarity Index

Benyamin Ghojogh, Fakhri Karray, Mark Crowley

Department of Electrical and Computer Engineering,
University of Waterloo, Waterloo, ON, Canada
{bghojogh, karray, mcrowley}@uwaterloo.ca

Abstract. Literature has shown that Mean Square Error is not a promising measure for image fidelity and similarity assessment, and Structural Similarity Index (SSIM) can properly handle this aspect. The existing subspace learning methods in machine learning are based on Euclidean distance or MSE and thus cannot properly capture the structural features of images. In this paper, we define image structure subspace which captures the intrinsic structural features of an image and discriminates the different types of image distortions. Therefore, this paper provides a bridge between image processing and manifold learning opening future research opportunities. In order to learn this subspace, we propose SSIM kernel as a new kernel which can be used in kernel-based machine learning methods.

Keywords: Image structure subspace, subspace learning, SSIM kernel, SSIM index, structural similarity

1 Introduction

Although Mean Squared Error (MSE) has many nice mathematical properties, such as convexity, it is not a proper measure for image quality, fidelity, or similarity [1]. MSE, or ℓ_2 norm, sees the distortions applied on the image with the same eye; however, the image distortions can be categorized into structural and non-structural distortions [2]. The structural distortions, including blurring, Gaussian noise, and JPEG blocking distortion, are noticeable by Human Visual System (HVS). On the other hand, non-structural distortions, including luminance enhancement and contrast change, are not as annoying to HVS as the structural distortions.

One of the most well-known measures suitable for image fidelity assessment is the structural similarity index (SSIM) [2, 3] which considers luminance and contrast change as non-structural distortions and other distortions as structural ones. Recently, using SSIM in optimization problems has been noticed in the literature [4]. The distance based on SSIM is quasi-convex under some conditions [5] and satisfies the triangular property as a metric [6]. The SSIM has been used in many tasks such as image denoising, image restoration, contrast enhancement, image quantization, compression, etc (e.g., see the references in [5]).

Most of the manifold learning and machine learning algorithms have focused on ℓ_2 norm or MSE [7]. Different subspace learning methods such as Principal Component Analysis (PCA), Kernel PCA, Multi-Dimensional Scaling (MDS) [8], Isomap [9], and Laplacian Eigenmap (LE) [10] have been developed based on MSE. A good question arises here: Why don't we use SSIM in manifold and machine learning? Because SSIM is much more effective than MSE in terms of image structure measurement [1, 3]. In this paper, we introduce and define the new concept of *image structure subspace* which is a subspace capturing the intrinsic features of an image in terms of structural similarity and distortions, and can discriminate the various types of image distortions. The image structure subspace can open a broad new area for future research by connecting the fields of manifold learning and image quality assessment. One of the many possible applications of this subspace is selecting among the denoising methods or tuning their parameters [11] which we consider as a future work. In order to learn the image structure subspace, we propose *SSIM kernel* as a new kernel which can be used in kernel-based learning methods [12].

In the remainder of paper, we briefly review SSIM and then we introduce the SSIM kernel. Using SSIM kernel in several subspace learning methods are explained afterwards. Finally, experiments on comparison of kernels and out-of-sample projection are reported.

2 Structural Similarity Index

The SSIM between two reshaped image patches $\breve{\boldsymbol{x}}_1 = [x_1^{(1)}, \dots, x_1^{(q)}]^\top \in \mathbb{R}^q$ and $\breve{\boldsymbol{x}}_2 = [x_2^{(1)}, \dots, x_2^{(q)}]^\top \in \mathbb{R}^q$, in color intensity range $[0, 255]$, is [2, 3]:

$$\mathbb{R} \ni \text{SSIM}(\breve{\boldsymbol{x}}_1, \breve{\boldsymbol{x}}_2) := \left(\frac{2\mu_{x_1}\mu_{x_2} + c_1}{\mu_{x_1}^2 + \mu_{x_2}^2 + c_1}\right)\left(\frac{2\sigma_{x_1}\sigma_{x_2} + c_2}{\sigma_{x_1}^2 + \sigma_{x_2}^2 + c_2}\right)\left(\frac{\sigma_{x_1,x_2} + c_3}{\sigma_{x_1}\sigma_{x_2} + c_3}\right), \tag{1}$$

where $\mu_{x_1} = (1/q)\sum_{i=1}^{q} x_1^{(i)}$, $\sigma_{x_1} = \Big[\big(1/(q-1)\big)\sum_{i=1}^{q}(x_1^{(i)} - \mu_{x_1})^2\Big]^{0.5}$, $\sigma_{x_1,x_2} = \big(1/(q-1)\big)\sum_{i=1}^{q}(x_1^{(i)} - \mu_{x_1})(x_2^{(i)} - \mu_{x_2})$, $c_1 = (0.01 \times 255)^2$, $c_2 = 2\,c_3 = (0.03 \times 255)^2$, and μ_{x_2} and σ_{x_2} are defined similarly for $\breve{\boldsymbol{x}}_2$. The c_1, c_2, and c_3 are for avoidance of singularity [3] and q is the dimensionality of the reshaped image patch.

Because of $c_2 = 2\,c_3$, the SSIM is simplified to $\text{SSIM}(\breve{\boldsymbol{x}}_1, \breve{\boldsymbol{x}}_2) = s_1(\breve{\boldsymbol{x}}_1, \breve{\boldsymbol{x}}_2) \times s_2(\breve{\boldsymbol{x}}_1, \breve{\boldsymbol{x}}_2)$, where $s_1(\breve{\boldsymbol{x}}_1, \breve{\boldsymbol{x}}_2) := (2\mu_{x_1}\mu_{x_2} + c_1)/(\mu_{x_1}^2 + \mu_{x_2}^2 + c_1)$ and $s_2(\breve{\boldsymbol{x}}_1, \breve{\boldsymbol{x}}_2) := (2\sigma_{x_1,x_2} + c_2)/(\sigma_{x_1}^2 + \sigma_{x_2}^2 + c_2)$. Because of spatial variety of image statistics, the SSIM is usually computed for patches of an image. A sliding window moves pixel by pixel on the two images and calculates the $\text{SSIM}(\breve{\boldsymbol{x}}_1, \breve{\boldsymbol{x}}_2)$ for every patch. We denote the reshaped vectors of the two images by $\boldsymbol{x}_1 \in \mathbb{R}^d$ and $\boldsymbol{x}_2 \in \mathbb{R}^d$, and a reshaped patch in the two images by $\breve{\boldsymbol{x}}_1 \in \mathbb{R}^q$ and $\breve{\boldsymbol{x}}_2 \in \mathbb{R}^q$. Therefore, an SSIM vector denoted by $\boldsymbol{s}(\boldsymbol{x}_1, \boldsymbol{x}_2) \in \mathbb{R}^d$ is obtained whose i-th element is SSIM for the patch around the i-th pixel.

3 SSIM Kernel

We can map the n data points $\{\boldsymbol{x}_i\}_{i=1}^n$, where $\boldsymbol{x}_i \in \mathbb{R}^d$, to a higher-dimensional feature space hoping to have the data fall close to a simpler-to-analyze manifold in the feature space. Suppose $\boldsymbol{\phi} : \boldsymbol{x} \to \mathcal{H}$ is a function which maps the data $\boldsymbol{x}$ to the feature space. In other words, $\boldsymbol{x} \mapsto \boldsymbol{\phi}(\boldsymbol{x})$. Let t denote the dimensionality of the feature space, i.e., $\boldsymbol{\phi}(\boldsymbol{x}) \in \mathbb{R}^t$. We usually have $t \gg d$. If $\boldsymbol{x}$ belongs to the set $\mathcal{X}$, i.e., $\boldsymbol{x} \in \mathcal{X}$, the kernel of two vectors $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ is $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ and is defined as [12]: $k(\boldsymbol{x}_1, \boldsymbol{x}_2) := \boldsymbol{\phi}(\boldsymbol{x}_1)^\top \boldsymbol{\phi}(\boldsymbol{x}_2)$. The kernel matrix for two datasets $\boldsymbol{X}_1 = [\boldsymbol{x}_{1,1}, \dots, \boldsymbol{x}_{1,n_1}] \in \mathbb{R}^{d \times n_1}$ and $\boldsymbol{X}_2 = [\boldsymbol{x}_{2,1}, \dots, \boldsymbol{x}_{2,n_2}] \in \mathbb{R}^{d \times n_2}$ is:

$$\mathbb{R}^{n_1 \times n_2} \ni \boldsymbol{K}(\boldsymbol{X}_1, \boldsymbol{X}_2) := \boldsymbol{\Phi}(\boldsymbol{X}_1)^\top \boldsymbol{\Phi}(\boldsymbol{X}_2), \tag{2}$$

where $\boldsymbol{\Phi}(\boldsymbol{X}_1) := [\boldsymbol{\phi}(\boldsymbol{x}_{1,1}), \dots, \boldsymbol{\phi}(\boldsymbol{x}_{1,n_1})] \in \mathbb{R}^{t \times n_1}$ and $\boldsymbol{\Phi}(\boldsymbol{X}_2)$ is similarly defined. The kernel between a matrix $\boldsymbol{X} \in \mathbb{R}^{d \times n}$ and a vector $\boldsymbol{x} \in \mathbb{R}^d$ is $\mathbb{R}^n \ni \boldsymbol{k}(\boldsymbol{X}, \boldsymbol{x}) := \boldsymbol{\Phi}(\boldsymbol{X})^\top \boldsymbol{\phi}(\boldsymbol{x})$. We denote the kernel over dataset $\boldsymbol{X}$ by $\boldsymbol{K}_x := \boldsymbol{K}(\boldsymbol{X}, \boldsymbol{X}) \in \mathbb{R}^{n \times n}$.

The kernel can be written as [8, 13]:

$$\boldsymbol{K}_x = -(1/2)\, \boldsymbol{H}\boldsymbol{D}\boldsymbol{H}, \tag{3}$$

where $\mathbb{R}^{n \times n} \ni \boldsymbol{H} = \boldsymbol{I} - (1/n)\boldsymbol{1}\boldsymbol{1}^\top$ is the centering matrix, $\boldsymbol{I}$ and $\boldsymbol{1}$ are the identity matrix and $[1, \dots, 1]^\top$, respectively, and $\boldsymbol{D} \in \mathbb{R}^{n \times n}$ is the distance matrix whose (i, j)-th element is a distance measure between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$. For example, if the distance measure for $\boldsymbol{D}$ is $||\boldsymbol{x}_i - \boldsymbol{x}_j||_2^2$, we will have $\boldsymbol{K}_x = \boldsymbol{H}\boldsymbol{X}^\top \boldsymbol{X}\boldsymbol{H}$. The elements of $\boldsymbol{D}$ can be measured by a valid distance metric (see [13] and chapter 2 in [7]). Two examples of distance metrics are Euclidean distance (resulting in metric MDS [8] or PCA) and geodesic distance (resulting in Isomap [9]).

A valid distance metric based on SSIM is [5, 6]:

$$\mathbb{R} \ni d(\breve{\boldsymbol{x}}_1, \breve{\boldsymbol{x}}_2) = \sqrt{2 - s_1(\breve{\boldsymbol{x}}_1, \breve{\boldsymbol{x}}_2) - s_2(\breve{\boldsymbol{x}}_1, \breve{\boldsymbol{x}}_2)}. \tag{4}$$

Calculating this metric for every patch, we will have a distance vector $\boldsymbol{d}(\boldsymbol{x}_1, \boldsymbol{x}_2) \in \mathbb{R}^d$ between the two images $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$. We want to have a scalar distance between two images so we use this theorem: The ℓ_2 norm of a vector of metrics is also a metric (see [5] for proof). Therefore, we define the distance between two images $\boldsymbol{x}_1 \in \mathbb{R}^d$ and $\boldsymbol{x}_2 \in \mathbb{R}^d$ as:

$$\mathbb{R} \ni d(\boldsymbol{x}_1, \boldsymbol{x}_2) := ||\boldsymbol{d}(\boldsymbol{x}_1, \boldsymbol{x}_2)||_2 = \left[\sum_{i=1}^{d} \Big(d_i(\breve{\boldsymbol{x}}_1, \breve{\boldsymbol{x}}_2) \Big)^2 \right]^{(1/2)}, \tag{5}$$

where $d_i(\breve{\boldsymbol{x}}_1, \breve{\boldsymbol{x}}_2)$ is the distance of Eq. (4) for the i-th patch. Note that Eq. (5) is equivalent to the Frobenius norm of the distance map between the two images if we have not reshaped the map to a vector. Calculating Eq. (5) between every two images of the dataset $\boldsymbol{X} = [\boldsymbol{x}_1, \dots, \boldsymbol{x}_n] \in \mathbb{R}^{d \times n}$ gives us the symmetric distance matrix $\boldsymbol{D} \in \mathbb{R}^{n \times n}$. Finally, according to Eq. (3), we define the *SSIM kernel*:

$$\mathbb{R}^{n \times n} \ni \boldsymbol{S}_x = \boldsymbol{S}(\boldsymbol{X}, \boldsymbol{X}) := -(1/2)\, \boldsymbol{H}\boldsymbol{D}\boldsymbol{H}, \tag{6}$$

where $\boldsymbol{D}$ is calculated using Eq. (5). Similarly, as in Eq. (2), we can have the SSIM kernel between two different datasets: $\boldsymbol{S}(\boldsymbol{X}_1, \boldsymbol{X}_2) = -(1/2)\, \boldsymbol{H}_1 \boldsymbol{D} \boldsymbol{H}_2$ where $\boldsymbol{D} \in \mathbb{R}^{n_1 \times n_2}$ is similarly calculated between the images, one from $\boldsymbol{X}_1$ and one from $\boldsymbol{X}_2$. Note that $\boldsymbol{H}_1 \in \mathbb{R}^{n_1 \times n_1}$ and $\boldsymbol{H}_2 \in \mathbb{R}^{n_2 \times n_2}$. The SSIM kernel between a matrix $\boldsymbol{X} \in \mathbb{R}^{d \times n}$ and a vector $\boldsymbol{x} \in \mathbb{R}^d$ is:

$$\mathbb{R}^n \ni \boldsymbol{s}(\boldsymbol{X}, \boldsymbol{x}) := -(1/2)\, \boldsymbol{H} \boldsymbol{d}, \tag{7}$$

where $\boldsymbol{d} \in \mathbb{R}^n$ is the distance vector between $\boldsymbol{X}$ and $\boldsymbol{x}$ where every element is calculated using Eq. (5). The SSIM kernel measures the similarity of data points which are images. Notice that the kernel here should not be confused with the filter kernel in signal processing.

4 Image Structure Subspace Learning

The proposed SSIM kernel can be used with various subspace learning methods in machine learning. Here, we briefly review PCA, kernel PCA, metric MDS, Isomap, and LE, and we also explain how we can use this kernel in these methods in order to obtain the image structure subspace.

4.1 Kernel Principal Component Analysis

Kernel PCA is based on dual PCA which provides the inner products of the vectors, suitable for use with kernels. In dual PCA, Singular Value Decomposition (SVD) is applied on the centered data $\boldsymbol{X}\boldsymbol{H} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top$ where the columns of $\boldsymbol{U} \in \mathbb{R}^{d \times p}$ and $\boldsymbol{V} \in \mathbb{R}^{n \times p}$ are the p leading left and right singular vectors of $\boldsymbol{X}\boldsymbol{H}$, respectively, and $\boldsymbol{\Sigma} \in \mathbb{R}^{p \times p}$ is a diagonal matrix with the p leading singular values (usually $p \ll d$). In dual PCA, the projection of data is $\mathbb{R}^{p \times n} \ni \widetilde{\boldsymbol{X}} = \boldsymbol{U}^\top \boldsymbol{X}\boldsymbol{H} = \boldsymbol{\Sigma}\boldsymbol{V}^\top$ where $\widetilde{\boldsymbol{X}}$ denotes the projected data. On the other hand, from $\boldsymbol{X}\boldsymbol{H} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top$, we have $\boldsymbol{U} = \boldsymbol{X}\boldsymbol{H}\boldsymbol{V}\boldsymbol{\Sigma}^{-1}$. Therefore, projection of out-of-sample data point, denoted by $\boldsymbol{x}_t \in \mathbb{R}^d$, is $\widetilde{\boldsymbol{x}}_t = \boldsymbol{\Sigma}^{-1}\boldsymbol{V}^\top \boldsymbol{H}\boldsymbol{X}^\top(\boldsymbol{x}_t - \boldsymbol{\mu})$ where $\boldsymbol{\mu}$ is the mean of training points $\boldsymbol{X}$.

In kernel PCA, data are mapped to the feature space using a kernel. Applying SVD on the centered mapped data $\boldsymbol{\Phi}(\boldsymbol{X})\boldsymbol{H}$ gives us $\boldsymbol{\Phi}(\boldsymbol{X})\boldsymbol{H} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top$ where $\boldsymbol{U} \in \mathbb{R}^{t \times p}$, $\boldsymbol{V} \in \mathbb{R}^{n \times p}$, and $\boldsymbol{\Sigma} \in \mathbb{R}^{p \times p}$ include the left singular vectors, right singular vectors, and singular values of $\boldsymbol{\Phi}(\boldsymbol{X})\boldsymbol{H}$, respectively. In practice, the $\boldsymbol{V}$ and $\boldsymbol{\Sigma}$ are found by solving the eigenvalue problem for $\boldsymbol{H}\boldsymbol{\Phi}(\boldsymbol{X})^\top\boldsymbol{\Phi}(\boldsymbol{X})\boldsymbol{H} = \boldsymbol{H}\boldsymbol{K}_x\boldsymbol{H}$. Therefore, the kernel should be double-centered. The eigenvalue problem is $\boldsymbol{H}\boldsymbol{K}_x\boldsymbol{H}\boldsymbol{V} = \boldsymbol{V}\boldsymbol{\Sigma}^2$ where $\boldsymbol{V}$ and $\boldsymbol{\Sigma}$ contain the eigenvectors and square root of eigenvalues of $\boldsymbol{H}\boldsymbol{K}_x\boldsymbol{H}$. After picking the p leading eigenvectors to have $\boldsymbol{V} \in \mathbb{R}^{n \times p}$ and $\boldsymbol{\Sigma} \in \mathbb{R}^{p \times p}$, the projection of data in kernel PCA is:

$$\mathbb{R}^{p \times n} \ni \widetilde{\boldsymbol{X}} = \boldsymbol{U}^\top \boldsymbol{\Phi}(\boldsymbol{X})\boldsymbol{H} = \boldsymbol{\Sigma}\boldsymbol{V}^\top. \tag{8}$$

Moreover, from $\boldsymbol{\Phi}(\boldsymbol{X})\boldsymbol{H} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top$ we have $\boldsymbol{U}^\top = \boldsymbol{\Sigma}^{-1}\boldsymbol{V}^\top\boldsymbol{H}\boldsymbol{\Phi}(\boldsymbol{X})^\top$; therefore, projection of out-of-sample point $\boldsymbol{x}_t$ is:

$$\mathbb{R}^p \ni \widetilde{\boldsymbol{x}}_t = \boldsymbol{U}^\top \boldsymbol{\phi}(\boldsymbol{x}_t) = \boldsymbol{\Sigma}^{-1}\boldsymbol{V}^\top\boldsymbol{H}\boldsymbol{k}(\boldsymbol{X}, \boldsymbol{x}_t). \tag{9}$$

In order to learn the image structure subspace, the kernel $\boldsymbol{K}_x$ can be the SSIM kernel $\boldsymbol{S}_x$. Therefore, $\boldsymbol{V}$ and $\boldsymbol{\Sigma}$ are obtained from $\boldsymbol{S}_x \boldsymbol{V} = \boldsymbol{V} \boldsymbol{\Sigma}^2$. Note that the $\boldsymbol{S}_x$ is already double centered according to Eq. (6). Moreover, for out-of-sample projection, $\boldsymbol{s}(\boldsymbol{X}, \boldsymbol{x}_t)$ is used in place of $\boldsymbol{k}(\boldsymbol{X}, \boldsymbol{x}_t)$ in Eq. (9).

4.2 Metric Multi-Dimensional Scaling

In metric MDS, the difference of distances in the original and embedded space, i.e., $||\boldsymbol{D}_x - \boldsymbol{D}_{\widetilde{x}}||_F^2$ is minimized, where $||.||_F$ denotes Frobenius norm. The $\boldsymbol{D}_x$ is the distance matrix whose (i, j)-th element is $||\boldsymbol{x}_i - \boldsymbol{x}_j||_2^2$ and $\boldsymbol{D}_{\widetilde{x}}$ is defined similarly in the embedded space. This optimization problem is equivalent to minimizing the similarities of data points in the original and the embedded space, i.e., $||\boldsymbol{X}^\top \boldsymbol{X} - \widetilde{\boldsymbol{X}}^\top \widetilde{\boldsymbol{X}}||_F^2$. Solving this optimization problem results in [8]:

$$\mathbb{R}^{p \times n} \ni \widetilde{\boldsymbol{X}} = \boldsymbol{\Sigma}^{(1/2)} \boldsymbol{V}^\top, \tag{10}$$

where $\boldsymbol{V} \in \mathbb{R}^{n \times p}$ and $\boldsymbol{\Sigma} \in \mathbb{R}^{p \times p}$ contain the p leading eigenvectors and squre root of eigenvalues of $\boldsymbol{X}^\top \boldsymbol{X}$. Assuming that the kernel over data is linear, i.e., $\boldsymbol{K}_x = \boldsymbol{\Phi}(\boldsymbol{X})^\top \boldsymbol{\Phi}(\boldsymbol{X}) = \boldsymbol{X}^\top \boldsymbol{X}$, the complete $\boldsymbol{V} \in \mathbb{R}^{n \times n}$ and $\boldsymbol{\Sigma} \in \mathbb{R}^{n \times n}$ are obtained by applying SVD on the kernel:

$$\mathbb{R}^{n \times n} \ni \boldsymbol{K}_x = \boldsymbol{\Phi}(\boldsymbol{X})^\top \boldsymbol{\Phi}(\boldsymbol{X}) = \boldsymbol{V} \boldsymbol{\Sigma} \boldsymbol{V}^\top. \tag{11}$$

Note that if the kernel in Eq. (11) is obtained from Eq. (3) where $\boldsymbol{D}$ is the geodesic distance matrix, we will have Isomap [9].

In image structure subspace learning, we use SSIM kernel for MDS. Therefore, Eq. (11) becomes: $\mathbb{R}^{n \times n} \ni \boldsymbol{S}_x = \boldsymbol{V} \boldsymbol{\Sigma} \boldsymbol{V}^\top$. The kernel is required to be double centered [8] and the $\boldsymbol{S}_x$ is already double centered according to Eq. (6).

4.3 Laplacian Eigenmap

The LE [10] addresses the minimization of $\widetilde{\boldsymbol{X}}^\top \boldsymbol{L} \widetilde{\boldsymbol{X}}$ subject to orthogonality of the embedded data points, i.e., $\widetilde{\boldsymbol{X}}^\top \widetilde{\boldsymbol{X}} = \boldsymbol{I}$, where $\boldsymbol{L}$ is the Laplacian matrix, $\boldsymbol{L} = \boldsymbol{M} - \boldsymbol{W}$. The $\boldsymbol{W} \in \mathbb{R}^{n \times n}$ is the similarity matrix whose (i, j)-th element, denoted by $\boldsymbol{W}_{i,j}$, is $\exp(-||\widetilde{\boldsymbol{x}}_i - \widetilde{\boldsymbol{x}}_j||_2^2)$ if the points $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ are connected in the k-nearest neighbor graph, and zero otherwise. Therefore, the LE uses the Radial Basis Function (RBF) kernel for constructing $\boldsymbol{W}$, where the k largest values of every row of the kernel are utilized. The $\boldsymbol{M}$ is a diagonal matrix whose (i, i)-th element is the summation of the i-th row of $\boldsymbol{W}$. Note that $\widetilde{\boldsymbol{X}}^\top \boldsymbol{L} \widetilde{\boldsymbol{X}} = (1/2) \sum_{ij} \boldsymbol{W}_{i,j} ||\widetilde{\boldsymbol{x}}_i - \widetilde{\boldsymbol{x}}_j||_2^2$. Therefore, this optimization problem tries to make $\widetilde{\boldsymbol{x}}_i$ and $\widetilde{\boldsymbol{x}}_j$ close to one another when $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ are similar. Solving this optimization problem results in this eigenvalue problem:

$$\boldsymbol{L} \widetilde{\boldsymbol{X}} = \widetilde{\boldsymbol{X}} \boldsymbol{\Sigma}. \tag{12}$$

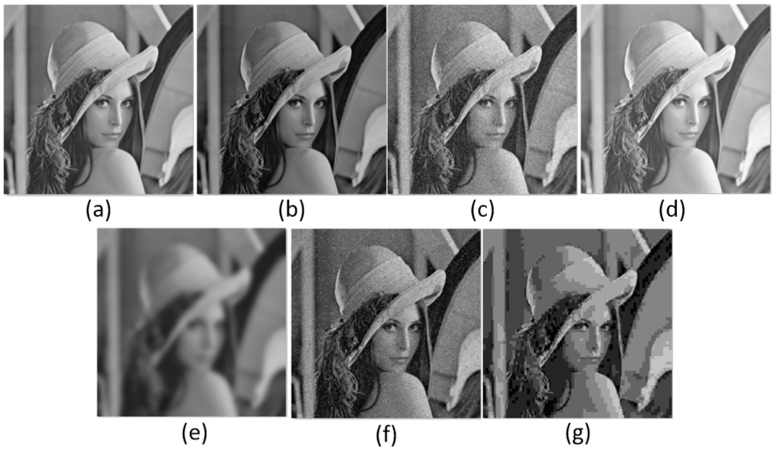

Fig. 1. Examples from the training dataset: (a) original image, (b) contrast stretched, (c) Gaussian noise, (d) luminance enhanced, (e) Gaussian blurring, (f) salt & pepper impulse noise, and (g) JPEG distortion.

Therefore, the embedded data points $\widetilde{\boldsymbol{X}}$ are the p trailing eigenvectors (with smallest eigenvalues) of $\boldsymbol{L}$, except the first smallest eigenvector with eigenvalue zero. Note that finally $\widetilde{\boldsymbol{X}}$ should be transposed to have $\widetilde{\boldsymbol{X}} \in \mathbb{R}^{p \times n}$. In image structure subspace learning, we can use the SSIM kernel for constructing $\boldsymbol{W}$. The k largest values in every row of this kernel are used and the other elements will be zero.

5 Experiments

5.1 Dataset

We made a dataset out of the standard *Lena* image. Six different types of distortions were applied on the original *Lena* image (see Fig. 1), each of which has 20 images in the dataset with different MSE values. Therefore, the size of training set is 121 including the original image. The six used distortions are stretching contrast, Gaussian noise, enhancing luminance, Gaussian blurring, salt & pepper impulse noise, and JPEG distortion. For every type of distortion, 20 different levels of MSE, i.e., from MSE = 45 to MSE = 900 with step 45, were generated in order to have images on the equal-MSE or *iso-error* hypersphere [3].

5.2 Comparison of Kernels

We experimented with the basic subspace learning methods using different kernels including our proposed SSIM kernel. The baseline kernels are Radial Basis Function (RBF) $\exp(-\gamma\, ||\boldsymbol{x}_1 - \boldsymbol{x}_2||_2^2)$, linear kernel $\boldsymbol{x}_1^\top \boldsymbol{x}_2$, polynomial $(\gamma\, \boldsymbol{x}_1^\top \boldsymbol{x}_2 + 1)^3$, sigmoid $\tanh(\gamma\, \boldsymbol{x}_1^\top \boldsymbol{x}_2 + 1)$, cosine $\boldsymbol{x}_1^\top \boldsymbol{x}_2 / (||\boldsymbol{x}_1||_2 ||\boldsymbol{x}_2||_2)$, and geodesic kernel

(Eq. 3 with geodesic distance matrix), where $\gamma := 1/d$. Figures 2 and 3 show the subspaces obtained from experiments using different kernels.

As can be seen in Fig. 2, the image structure subspace propagates the different types of distortions out from the non-distorted original image while the distortions mostly exist on separate trajectories. The more distorted an image is, the further from the original image it is projected or embedded. The subspace obtained using the SSIM kernel discriminates the different distortions much more properly compared to other kernels. In kernel PCA using SSIM, the first dimension puts the luminance enhancement and contrast stretching (non-structural distortions) apart from the structural distortions. Moreover, the third dimension separates the Gaussian blurring as a structural distortion. The fourth dimension shows that the distortions have tilted around this direction in the image distortion subspace. In linear and polynomial kernels, Gaussian noise is not separated from the original image. Also, contrast stretching and impulse noise are treated similarly in a linear kernel. In a polynomial kernel, contrast stretching, impulse noise, and JPEG distortion are not properly separated. The fourth dimension in both linear and polynomial kernels are not promising discriminators.

According to Fig. 2, the results of the SSIM kernel and RBF kernel look very similar, but not identical, especially in the first dimensions. We provide the reason here: Let $d := ||\boldsymbol{x}_1 - \boldsymbol{x}_2||_2$ and $r = d^2$. The Taylor series expansion of RBF kernel is:

$$\exp(-\gamma r) \approx 1 - \gamma r + \frac{\gamma^2}{2} r^2 - \frac{\gamma^3}{6} r^3 + \cdots \tag{13}$$

On the other hand, the SSIM kernel is $\boldsymbol{S}_x \propto -\boldsymbol{D}$ (Eq. 6). Every element of $\boldsymbol{D}$ is based on $d(\boldsymbol{x}_1, \boldsymbol{x}_2)$ in Eq. (5). We have $d(\boldsymbol{x}_1, \boldsymbol{x}_2) \propto d_i(\breve{\boldsymbol{x}}_1, \breve{\boldsymbol{x}}_2)$. Therefore, $\boldsymbol{S}_x \propto -d_i(\breve{\boldsymbol{x}}_1, \breve{\boldsymbol{x}}_2) = -\sqrt{r}$ where $r = (d_i(\breve{\boldsymbol{x}}_1, \breve{\boldsymbol{x}}_2))^2$. By Taylor series expansion, we have:

$$\boldsymbol{S}_x \propto -\sqrt{r} \approx -\frac{5}{16} - \frac{15}{16} r + \frac{5}{16} r^2 - \frac{1}{16} r^3 + \cdots \tag{14}$$

Comparing Eqs. (13) and (14) gives us the hint for why SSIM and RBF kernels had similar results. Note that the r in the RBF kernel is based on Euclidean distance while the r in SSIM kernel is based on SSIM distance.

In Fig. 3, we can see that the sigmoid kernel does not properly discriminate stretching contrast and impulse noise. Moreover, the Gaussian noise is not well separated from the original image. The cosine kernel separates these two but does not preserve the almost uniform distance of different intensities of a distortion, as we had for SSIM kernel. In the first two dimensions of Isomap, on the other hand, the image with Guassian noise and the original image are not separated, and contrast stretching, impulse noise, and JPEG distortion are not discriminated. In the second, third, and fourth dimensions of Isomap, we see that very high amount of JPEG distortion is noticed while low value of JPEG distortion is not respected.

Figure 3 also includes the subspaces of LE using SSIM kernel and RBF kernel [10]. The LE using SSIM kernel strongly outperforms the LE using RBF kernel

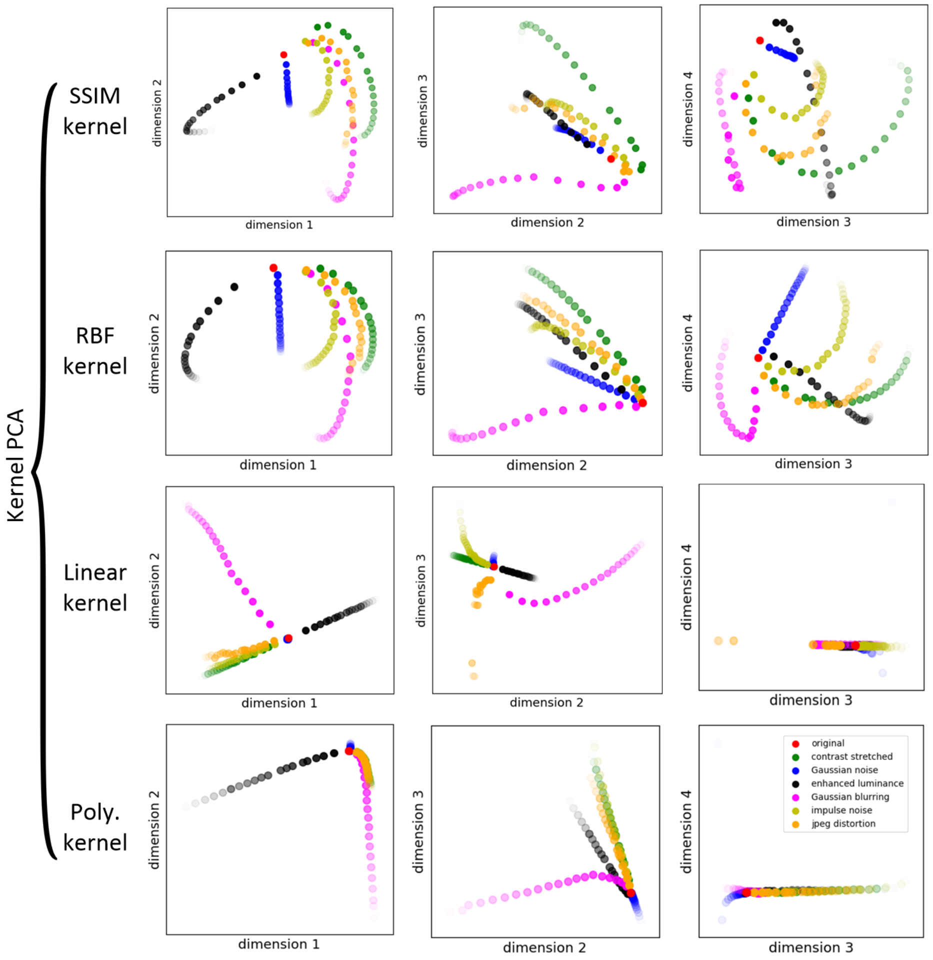

Fig. 2. Image structure subspaces: The first to fourth rows correspond to SSIM kernel, RBF kernel, linear kernel (or dual PCA or MDS [8, 7]), and polynomial kernel in kernel PCA, respectively. More transparent points correspond to more distorted images.

because its second and third dimensions discriminate contrast stretching (non-structural distortion) from Gaussian blurring (structural distortion). The first and second dimensions of LE using SSIM kernel also show much better scatter of data points in the subspace. That is while LE fails to discriminate the distortions properly. Finally, for the sake of better visualization of what the scatter plots in Figs. 2 and 3 mean, please see Fig. 4.

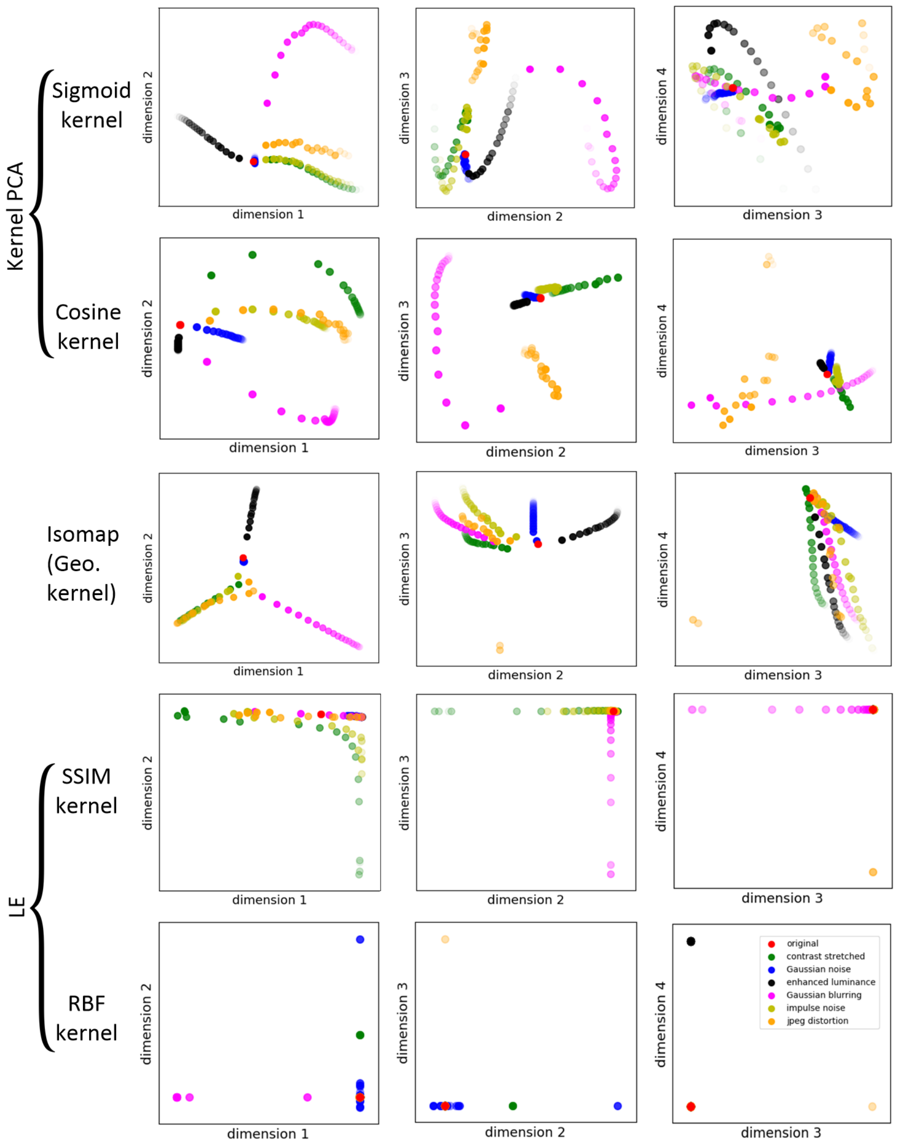

Fig. 3. Image structure subspaces: The first and second rows correspond to sigmoid kernel and cosine kernel in kernel PCA, respectively. The third row is for geodesic kernel (Isomap) [9]. The fourth and fifth rows correspond to LE using SSIM kernel and RBF kernel [10], respectively. More transparent points correspond to more distorted images.

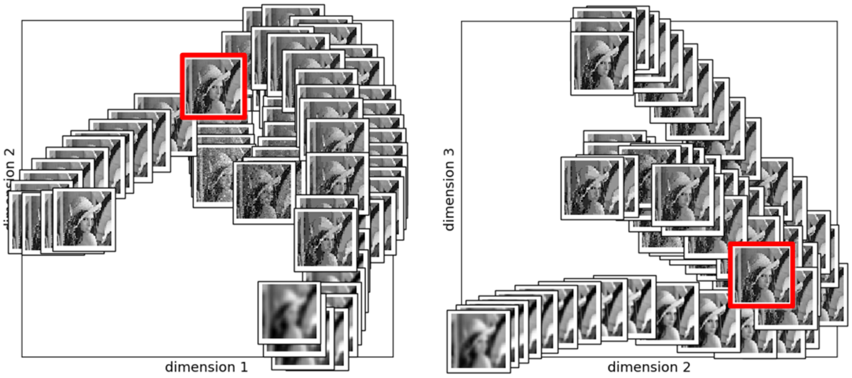

Fig. 4. The projected images onto the subspace of kernel PCA using SSIM kernel. The image with a thick red frame is the original image.

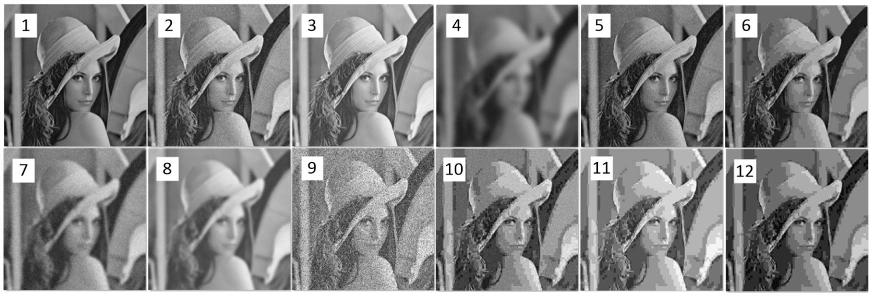

Fig. 5. Out-of-sample images with different types of distortions having MSE = 500: (1) stretching contrast, (2) Gaussian noise, (3) luminance enhancement, (4) Gaussian blurring, (5) impulse noise, (6) JPEG distortion, (7) Gaussian blurring + Gaussian noise, (8) Gaussian blurring + luminance enhancement, (9) impulse noise + luminance enhancement, (10) JPEG distortion + Gaussian noise, (11) JPEG distortion + luminance enhancement, and (12) JPEG distortion + stretching contrast.

5.3 Out-of-sample Projection

For out-of-sample projection using the SSIM kernel, we created 12 test images with MSE = 500 having different distortions. Figure 5 shows the test images where the distortions are reported in the caption of figure. Some of the test images have a combination of different distortions in order to evaluate the image structure subspace with harder out-of-sample images.

The projection of these out-of-sample images onto the kernel PCA subspace obtained using SSIM kernel is shown in Fig. 6. As expected, the images 1 to 6 with have solely one type of distortion fall close enough to the projected training samples of their distortion. Note that in dimensions 3 and 4 of projection, some

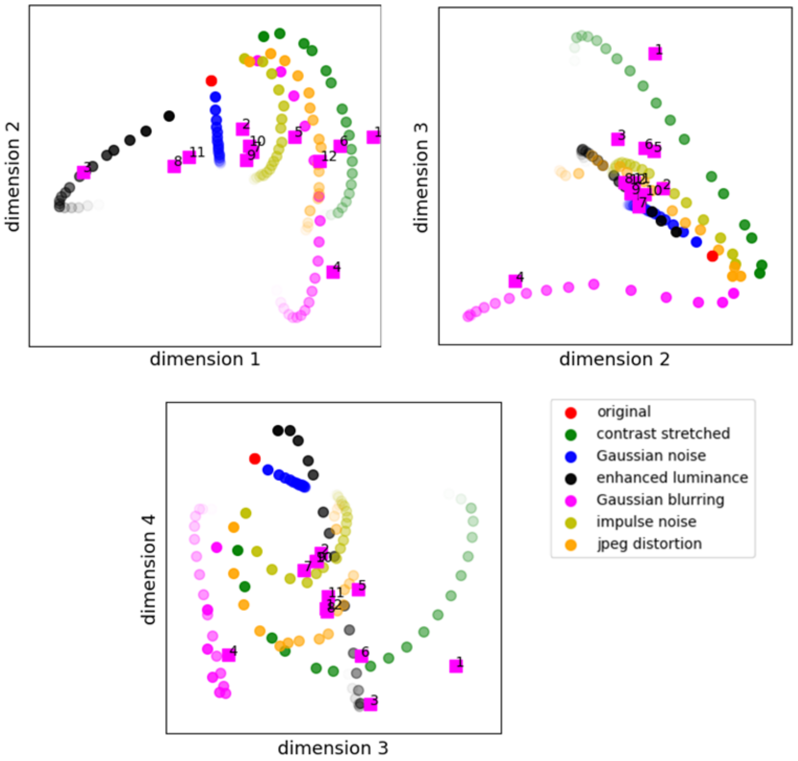

Fig. 6. Projection of out-of-sample images onto the image structure subspace. The pink square points are the out-of-sample images.

points might be thought to fall onto each other but a 3D imagination shows that they are apart in the other dimension. Image 7 falls between the projection of Gaussian blurring and Gaussian noise. Likewise, Image 8 falls between projection of Gaussian blurring and luminance enhancement, image 9 falls between impulse noise and luminance enhancement, image 10 falls between JPEG distortion and Gaussian noise, image 11 falls between JPEG and luminance, and image 12 falls between JPEG and stretching contrast. Note that for example, image 12 falls closer to JPEG distortion than to contrast stretching because JPEG distortion is a structural distortion and carries more amount of image quality distortion. In conclusion, the image structure subspace also supports distortion of out-of-sample images even if they have a combination of different distortions.

6 Conclusion and Future Direction

This paper introduced image structure subspace which captures the intrinsic features of an image in terms of structural similarity and distortions. This paper opens a new research field for a combination of image processing and manifold learning investigations. The SSIM kernel was also proposed which can be used in kernel-based machine learning algorithms. Experiments showed that the image structure subspace can demonstrate interesting discrimination useful to understand the relative effects of different types of distortions. As a future direction, we seek to develop new manifold learning algorithms based on SSIM in order to learn the defined image structure subspace in other ways.

References

1. Wang, Z., Bovik, A.C.: Mean squared error: Love it or leave it? a new look at signal fidelity measures. IEEE signal processing magazine **26**(1) (2009) 98–117
2. Wang, Z., Bovik, A.C., Sheikh, H.R., Simoncelli, E.P.: Image quality assessment: from error visibility to structural similarity. IEEE transactions on image processing **13**(4) (2004) 600–612
3. Wang, Z., Bovik, A.C.: Modern image quality assessment. Synthesis Lectures on Image, Video, and Multimedia Processing **2**(1) (2006) 1–156
4. Brunet, D., Channappayya, S.S., Wang, Z., Vrscay, E.R., Bovik, A.C.: Optimizing image quality. In: Handbook of Convex Optimization Methods in Imaging Science. Springer (2018) 15–41
5. Brunet, D., Vrscay, E.R., Wang, Z.: On the mathematical properties of the structural similarity index. IEEE Transactions on Image Processing **21**(4) (2012) 1488–1499
6. Brunet, D., Vrscay, E.R., Wang, Z.: A class of image metrics based on the structural similarity quality index. In: International Conference Image Analysis and Recognition, Springer (2011) 100–110
7. Strange, H., Zwiggelaar, R.: Open Problems in Spectral Dimensionality Reduction. Springer (2014)
8. Cox, T.F., Cox, M.A.: Multidimensional scaling. Chapman and hall/CRC (2000)
9. Tenenbaum, J.B., De Silva, V., Langford, J.C.: A global geometric framework for nonlinear dimensionality reduction. Science **290**(5500) (2000) 2319–2323
10. Belkin, M., Niyogi, P.: Laplacian eigenmaps for dimensionality reduction and data representation. Neural computation **15**(6) (2003) 1373–1396
11. McCrackin, L., Shirani, S.: Strategic image denoising using a support vector machine with seam energy and saliency features. In: 2014 IEEE International Conference on Image Processing (ICIP), IEEE (2014) 2684–2688
12. Hofmann, T., Schölkopf, B., Smola, A.J.: Kernel methods in machine learning. The annals of statistics (2008) 1171–1220
13. Ham, J.H., Lee, D.D., Mika, S., Schölkopf, B.: A kernel view of the dimensionality reduction of manifolds. In: International Conference on Machine Learning. (2004)