Locally Linear Image Structural Embedding for Image Structure Manifold Learning

Benyamin Ghojogh, Fakhri Karray, Mark Crowley

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

Abstract. Most of existing manifold learning methods rely on Mean Square Error (MSE) or ℓ_2 norm. However, for the problem of image quality assessment, these are not promising measure. In this paper, we introduce the concept of an image structure manifold which captures image structure features and discriminates image distortions. We propose a new manifold learning method, Locally Linear Image Structural Embedding (LLISE), and kernel LLISE for learning this manifold. The LLISE is inspired by Locally Linear Embedding (LLE) but uses SSIM rather than MSE. This paper builds a bridge between manifold learning and image fidelity assessment and it can open a new area for future investigations.

Keywords: Locally linear embedding, locally linear image structural embedding, structural similarity, SSIM, image structure manifold

1 Introduction

Mean Square Error (MSE) is not a good measure for image quality assessment [1]. Two different categories of distortions exist, i.e., structural and non-structural distortions [2]. The structural similarity index (SSIM) [2, 3] is found to be a very promising measure for image fidelity assessment. It encounters luminance and contrast change as non-structural distortions and other distortions as structural ones. Recently, it has been used in optimization problems for different tasks although it is not convex but quasi-convex under certain conditions [4].

The manifold learning methods are designed mostly based on MSE or the ℓ_2 norm. Therefore, they do not perform satisfactorily for image quality discrimination. Locally Linear Embedding (LLE) [5] is an example. In this paper, we introduce the new concept of *image structure manifold* which captures the features of image structure and is useful for discriminating the image distortions. We propose Locally Linear Image Structural Embedding (LLISE), in both original and feature space, which uses SSIM distance rather than ℓ_2 norm. We also propose the out-of-sample extension of LLISE. The derivations of expressions in this paper are detailed more in the supplementary-material paper which will be released in https://arXiv.org.

1.1 Structural Similarity Index

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

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

where $\mu_{x_1} = (1/q)\sum_{i=1}^q x_1^{[i]}$, $\sigma_{x_1} = \Big[\big(1/(q-1)\big)\sum_{i=1}^q (x_1^{[i]} - \mu_{x_1})^2\Big]^{0.5}$, $\sigma_{x_1,x_2} = \big(1/(q-1)\big)\sum_{i=1}^q (x_1^{[i]} - \mu_{x_1})(x_2^{[i]} - \mu_{x_2})$, $c_1 = (0.01 \times l)^2$, $c_2 = 2\,c_3 = (0.03 \times l)^2$, and μ_{x_2} and σ_{x_2} are defined similarly for $\breve{\boldsymbol{x}}_2$. In this work, $l = 1$.

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 $s_2(\breve{\boldsymbol{x}}_1, \breve{\boldsymbol{x}}_2) := (2\sigma_{x_1,x_2} + c_2)/(\sigma_{x_1}^2 + \sigma_{x_2}^2 + c_2)$. If the vectors $\breve{\boldsymbol{x}}_1$ and $\breve{\boldsymbol{x}}_2$ have zero mean, i.e., $\mu_{x_1} = \mu_{x_2} = 0$, the SSIM becomes $\mathbb{R} \ni \text{SSIM}(\breve{\boldsymbol{x}}_1, \breve{\boldsymbol{x}}_2) = (2\breve{\boldsymbol{x}}_1^\top \breve{\boldsymbol{x}}_2 + c)/(||\breve{\boldsymbol{x}}_1||_2^2 + ||\breve{\boldsymbol{x}}_2||_2^2 + c)$, where $c = (q-1)\,c_2$ [6]. 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 block in the two images by $\breve{\boldsymbol{x}}_1 \in \mathbb{R}^q$ and $\breve{\boldsymbol{x}}_2 \in \mathbb{R}^q$. If $\mu_{x_1} = \mu_{x_2} = 0$, the distance based on SSIM, which we denote by $||.||_S$, is [4, 6]:

$$\mathbb{R} \ni ||\breve{\boldsymbol{x}}_1 - \breve{\boldsymbol{x}}_2||_S := 1 - \text{SSIM}(\breve{\boldsymbol{x}}_1, \breve{\boldsymbol{x}}_2) = \frac{||\breve{\boldsymbol{x}}_1 - \breve{\boldsymbol{x}}_2||_2^2}{||\breve{\boldsymbol{x}}_1||_2^2 + ||\breve{\boldsymbol{x}}_2||_2^2 + c}. \quad (2)$$

1.2 Locally Linear Embedding

In LLE [5], first a k-Nearest Neighbor (k-NN) graph is found using pairwise Euclidean distances. Every data point $\boldsymbol{x}_j \in \mathbb{R}^d$ is reconstructed by its k neighbors $\mathbb{R}^{d\times k} \ni \boldsymbol{X}_j := [{}_1\boldsymbol{x}_j, \ldots, {}_k\boldsymbol{x}_j]$ where ${}_r\boldsymbol{x}_j$ denotes the r-th neighbor of $\boldsymbol{x}_j$. If $\mathbb{R}^k \ni \widetilde{\boldsymbol{w}}_j := [{}_1\widetilde{w}_j, \ldots, {}_k\widetilde{w}_j]^\top$ denotes the reconstruction weights for the $\boldsymbol{x}_j$, the reconstruction problem with the weights adding to one is: minimize $\sum_{j=1}^n ||\boldsymbol{x}_j - \sum_{r=1}^k {}_r\widetilde{w}_j\,{}_r\boldsymbol{x}_j||_2^2$, subject to $\sum_{r=1}^k {}_r\widetilde{w}_j = 1, \forall j \in \{1, \ldots, n\}$. Then, the data points are embedded using the obtained weights. Take $\mathbb{R}^n \ni \boldsymbol{w}_j := [{}_1w_j, \ldots, {}_nw_j]^\top$ where ${}_rw_j$ is the weight obtained from linear reconstruction if $\boldsymbol{x}_r$ is a neighbor of $\boldsymbol{x}_j$ and is zero otherwise. If $\boldsymbol{y}_j \in \mathbb{R}^p$ denotes the embedded j-th data point, the embedding problem is with unit covariance is: minimize $\sum_{j=1}^n ||\boldsymbol{y}_j - \sum_{r=1}^n {}_rw_j\,\boldsymbol{y}_r||_2^2$, subject to $(1/n)\sum_{j=1}^n \boldsymbol{y}_j\boldsymbol{y}_j^\top = \boldsymbol{I}$ and $\sum_{j=1}^n \boldsymbol{y}_j = \boldsymbol{0}, \forall j \in \{1, \ldots, n\}$. Kernel LLE [7] finds the k-NN graph and performs linear reconstruction from the neighbors in the feature space.

2 Locally Linear Image Structural Embedding

We partition a d-dimensional image $\boldsymbol{x}$ into $b = \lceil d/q \rceil$ non-overlapping blocks each of which is a reshaped vector $\breve{\boldsymbol{x}} \in \mathbb{R}^q$. In LLISE, we find a p-dimensional image structure manifold for every block. The q is a parameter and is an upper

bound on the desired dimensionality of the manifold of a block ($p \leq q$). This parameter is better not to be a very large number because of spatial variety of image statistics, and not very small to be able to capture the image structure. We denote the i-th block in the j-th image by $\breve{\boldsymbol{x}}_{j,i} \in \mathbb{R}^q$. In LLISE, we first center every image block by removing its mean.

2.1 Embedding The Training Data

***k*-Nearest Neighbors** For every block $\breve{\boldsymbol{x}}_i$ ($i \in \{1, \ldots, b\}$), amongst the n images, a k-NN graph is formed using pairwise Euclidean distances between that i-th block in the n images. Therefore, every block in every image has k neighbors. Let ${}_r\breve{\boldsymbol{x}}_{j,i} \in \mathbb{R}^q$ denote the r-th neighbor of $\breve{\boldsymbol{x}}_{j,i}$ and let the matrix $\mathbb{R}^{q \times k} \ni \breve{\boldsymbol{X}}_{j,i} := [\,{}_1\breve{\boldsymbol{x}}_{j,i}, \ldots,\, {}_k\breve{\boldsymbol{x}}_{j,i}]$ include the neighbors of $\breve{\boldsymbol{x}}_{j,i}$.

Linear Reconstruction by the Neighbors For every block $\breve{\boldsymbol{x}}_i$, we want the j-th image to be linearly reconstructed by its k neighbors. We minimize the reconstruction error while the vector of reconstruction weights for every image block is a unit vector:

$$\begin{aligned} \underset{\widetilde{\boldsymbol{W}}_i}{\text{minimize}} \quad & \sum_{i=1}^{b} \varepsilon(\widetilde{\boldsymbol{W}}_i) := \sum_{i=1}^{b} \sum_{j=1}^{n} \Big|\Big| \breve{\boldsymbol{x}}_{j,i} - \sum_{r=1}^{k} {}_r\widetilde{w}_{j,i}\; {}_r\breve{\boldsymbol{x}}_{j,i} \Big|\Big|_S, \\ \text{subject to} \quad & \sum_{r=1}^{k} {}_r\widetilde{w}_{j,i}^2 = 1, \quad \forall i \in \{1, \ldots, b\}, \;\; \forall j \in \{1, \ldots, n\}, \end{aligned} \tag{3}$$

where $\mathbb{R}^{n \times k} \ni \widetilde{\boldsymbol{W}}_i := [\widetilde{\boldsymbol{w}}_{1,i}, \ldots, \widetilde{\boldsymbol{w}}_{n,i}]^\top$ includes the weights for the i-th block in the images and $\mathbb{R}^k \ni \widetilde{\boldsymbol{w}}_{j,i} := [\,{}_1\widetilde{w}_{j,i}, \ldots,\, {}_k\widetilde{w}_{j,i}]^\top$ includes the weights of linear reconstruction of the i-th block in the j-th image using its k neighbors. The second constraint ensures $\widetilde{\boldsymbol{w}}_{j,i}^\top \widetilde{\boldsymbol{w}}_{j,i} = ||\widetilde{\boldsymbol{w}}_{j,i}||_2^2 = 1$. Note that we can formulate the problem with the constraint $\sum_{r=1}^{k} {}_r\widetilde{w}_{j,i} = 1$ as in LLE; however, with that constraint, the weights start to explode gradually after some optimization iterations. This problem does not happen in LLE because LLE is solved in closed form and not iteratively.

Take $f(\widetilde{\boldsymbol{w}}_{j,i}) := \big|\big|\breve{\boldsymbol{x}}_{j,i} - \sum_{r=1}^{k} {}_r\widetilde{w}_{j,i}\, {}_r\breve{\boldsymbol{x}}_{j,i}\big|\big|_S$ which is restated as $f(\widetilde{\boldsymbol{w}}_{j,i}) = ||\breve{\boldsymbol{x}}_{j,i} - \breve{\boldsymbol{X}}_{j,i}\, \widetilde{\boldsymbol{w}}_{j,i}||_S$. According to Eq. (2), the $f(\widetilde{\boldsymbol{w}}_{j,i})$ is simplified to:

$$\mathbb{R} \ni f(\widetilde{\boldsymbol{w}}_{j,i}) = \frac{\breve{\boldsymbol{x}}_{j,i}^\top \breve{\boldsymbol{x}}_{j,i} + \widetilde{\boldsymbol{w}}_{j,i}^\top \breve{\boldsymbol{X}}_{j,i}^\top \breve{\boldsymbol{X}}_{j,i}\, \widetilde{\boldsymbol{w}}_{j,i} - 2\, \widetilde{\boldsymbol{w}}_{j,i}^\top \breve{\boldsymbol{X}}_{j,i}^\top \breve{\boldsymbol{x}}_{j,i}}{\breve{\boldsymbol{x}}_{j,i}^\top \breve{\boldsymbol{x}}_{j,i} + \widetilde{\boldsymbol{w}}_{j,i}^\top \breve{\boldsymbol{X}}_{j,i}^\top \breve{\boldsymbol{X}}_{j,i}\, \widetilde{\boldsymbol{w}}_{j,i} + c}. \tag{4}$$

The gradient of $f(\widetilde{\boldsymbol{w}}_{j,i})$ with respect to $\widetilde{\boldsymbol{w}}_{j,i}$ is:

$$\mathbb{R}^k \ni \nabla f(\widetilde{\boldsymbol{w}}_{j,i}) = \frac{2\, \breve{\boldsymbol{X}}_{j,i}^\top \Big(\big(1 - f(\widetilde{\boldsymbol{w}}_{j,i})\big) \breve{\boldsymbol{X}}_{j,i} \widetilde{\boldsymbol{w}}_{j,i} - \breve{\boldsymbol{x}}_{j,i} \Big)}{\breve{\boldsymbol{x}}_{j,i}^\top \breve{\boldsymbol{x}}_{j,i} + \widetilde{\boldsymbol{w}}_{j,i}^\top \breve{\boldsymbol{X}}_{j,i}^\top \breve{\boldsymbol{X}}_{j,i}\, \widetilde{\boldsymbol{w}}_{j,i} + c}. \tag{5}$$

The Eq. (3) can be rewritten as:

$$\begin{aligned} \underset{\widetilde{\boldsymbol{w}}_{j,i},\widetilde{\boldsymbol{\xi}}_{j,i}}{\text{minimize}} \quad & \sum_{i=1}^{b}\sum_{j=1}^{n}\big(f(\widetilde{\boldsymbol{w}}_{j,i})+h_1(\widetilde{\boldsymbol{\xi}}_{j,i})\big), \\ \text{subject to} \quad & \widetilde{\boldsymbol{w}}_{j,i}-\widetilde{\boldsymbol{\xi}}_{j,i}=0 \;\; \forall i\in\{1,\dots,b\}, \;\; \forall j\in\{1,\dots,n\}, \end{aligned} \tag{6}$$

where $\mathbb{R}^k \ni \widetilde{\boldsymbol{\xi}}_{j,i} := [{}_1\widetilde{\xi}_{j,i}, \dots, {}_k\widetilde{\xi}_{j,i}]^\top$ and $h_1(\widetilde{\boldsymbol{\xi}}_{j,i}) := \mathbb{I}\big(\widetilde{\boldsymbol{\xi}}_{j,i}^\top \widetilde{\boldsymbol{\xi}}_{j,i} = 1\big)$. The $\mathbb{I}(.)$ denotes the indicator function which is zero if its condition is satisfied and is infinite otherwise. The Eq. (6) can be solved using Alternating Direction Method of Multipliers (ADMM) [8, 9]. The augmented Lagrangian is: $\mathcal{L}_\rho = \sum_{i=1}^{b}\sum_{j=1}^{n}\big(f(\widetilde{\boldsymbol{w}}_{j,i})+h_1(\widetilde{\boldsymbol{\xi}}_{j,i})\big)+(\rho/2)\,||\widetilde{\boldsymbol{w}}_{j,i}-\widetilde{\boldsymbol{\xi}}_{j,i}+\boldsymbol{\jmath}_{j,i}||_2^2-(\rho/2)\,||\boldsymbol{\lambda}_{j,i}||_2^2$, where $\boldsymbol{\lambda}_{j,i}\in\mathbb{R}^k$ is the Lagrange multiplier, $\rho>0$ is a parameter, and $\mathbb{R}^k \ni \boldsymbol{\jmath}_{j,i} := (1/\rho)\boldsymbol{\lambda}_{j,i}$. The term $(\rho/2)\,||\boldsymbol{\lambda}_{j,i}||_2^2$ is a constant with respect to $\widetilde{\boldsymbol{w}}_{j,i}$ and $\widetilde{\boldsymbol{\xi}}_{j,i}$ and can be dropped. The updates of $\widetilde{\boldsymbol{w}}_{j,i}$, $\widetilde{\boldsymbol{\xi}}_{j,i}$, and $\boldsymbol{\imath}_{j,i}$ are performed as [8, 9]:

$$\widetilde{\boldsymbol{w}}_{j,i}^{(\nu+1)} := \arg\min_{\widetilde{\boldsymbol{w}}_{j,i}}\Big(f(\widetilde{\boldsymbol{w}}_{j,i})+(\rho/2)\,||\widetilde{\boldsymbol{w}}_{j,i}-\widetilde{\boldsymbol{\xi}}_{j,i}^{(\nu)}+\boldsymbol{\jmath}_{j,i}^{(\nu)}||_2^2\Big), \tag{7}$$

$$\widetilde{\boldsymbol{\xi}}_{j,i}^{(\nu+1)} := \arg\min_{\widetilde{\boldsymbol{\xi}}_{j,i}}\Big(h_1(\widetilde{\boldsymbol{\xi}}_{j,i})+(\rho/2)\,||\widetilde{\boldsymbol{w}}_{j,i}^{(\nu+1)}-\widetilde{\boldsymbol{\xi}}_{j,i}+\boldsymbol{\jmath}_{j,i}^{(\nu)}||_2^2\Big), \tag{8}$$

$$\boldsymbol{\jmath}_{j,i}^{(\nu+1)} := \boldsymbol{\jmath}_{j,i}^{(\nu)}+\widetilde{\boldsymbol{w}}_{j,i}^{(\nu+1)}-\widetilde{\boldsymbol{\xi}}_{j,i}^{(\nu+1)}, \tag{9}$$

where ν denotes the iteration. The gradient of the objective function in Eq. (7) is $\nabla f(\widetilde{\boldsymbol{w}}_{j,i})+\rho\,(\widetilde{\boldsymbol{w}}_{j,i}-\widetilde{\boldsymbol{\xi}}_{j,i}^{(\nu)}+\boldsymbol{\jmath}_{j,i}^{(\nu)})$. We can use the gradient decent method [10] for solving the Eq. (7). Our experiments showed that even one iteration of gradient decent suffices for Eq. (7) because the ADMM itself is iterative. Hence, we can replace this equation with one iteration of gradient decent.

The proximal operator is defined as [11]:

$$\mathbf{prox}_{\lambda,h}(\boldsymbol{v}) := \arg\min_{\boldsymbol{u}}\,\big(h(\boldsymbol{u})+(\lambda/2)||\boldsymbol{u}-\boldsymbol{v}||_2^2\big), \tag{10}$$

where λ is the proximal parameter and h is the function that the proximal algorithm wants to minimize. According to Eq. (10), the Eq. (8) is equivalent to $\mathbf{prox}_{\rho,h_1}(\widetilde{\boldsymbol{w}}_{j,i}^{(\nu+1)}+\boldsymbol{\jmath}_{j,i}^{(\nu)})$. As $h_1(.)$ is indicator function, its proximal operator is projection [11]. Therefore, Eq. (8) is equivalent to $\Pi(\widetilde{\boldsymbol{w}}_{j,i}^{(\nu+1)}+\boldsymbol{\jmath}_{j,i}^{(\nu)})$ where $\Pi(.)$ denotes projection onto a set. The condition in $h_1(.)$ is $\widetilde{\boldsymbol{\xi}}_{j,i}^\top\widetilde{\boldsymbol{\xi}}_{j,i}=1$; therefore, this projection normalizes the vector by dividing to its ℓ_2 norm.

In summary, the Eqs. (7), (8), and (9) can be restated as:

$$\begin{aligned} \widetilde{\boldsymbol{w}}_{j,i}^{(\nu+1)} &:= \widetilde{\boldsymbol{w}}_{j,i}^{(\nu)}-\eta\,\nabla f(\widetilde{\boldsymbol{w}}_{j,i}^{(\nu)})-\eta\,\rho\,(\widetilde{\boldsymbol{w}}_{j,i}^{(\nu)}-\widetilde{\boldsymbol{\xi}}_{j,i}^{(\nu)}+\boldsymbol{\jmath}_{j,i}^{(\nu)}), \\ \widetilde{\boldsymbol{\xi}}_{j,i}^{(\nu+1)} &:= (\widetilde{\boldsymbol{w}}_{j,i}^{(\nu+1)}+\boldsymbol{\jmath}_{j,i}^{(\nu)})/||\widetilde{\boldsymbol{w}}_{j,i}^{(\nu+1)}+\boldsymbol{\jmath}_{j,i}^{(\nu)}||_2, \\ \boldsymbol{\jmath}_{j,i}^{(\nu+1)} &:= \boldsymbol{\jmath}_{j,i}^{(\nu)}+\widetilde{\boldsymbol{w}}_{j,i}^{(\nu+1)}-\widetilde{\boldsymbol{\xi}}_{j,i}^{(\nu+1)}, \end{aligned} \tag{11}$$

where $\eta > 0$ is the learning rate. Iteratively solving Eq. (11) until convergence gives us the $\widetilde{\boldsymbol{w}}_{j,i}$ for for the i-th block in the j-th image. Note that Eq. (11) can be solved in parallel for the blocks of images.

Linear Embedding In the previous section, we found the weights of linear reconstruction of the i-th block in every image from the i-th block in its k-NN. We can now find the embedding of the i-th block in every image using the obtained weights of reconstruction:

$$\begin{aligned} \underset{\boldsymbol{Y}_i}{\text{minimize}} \quad & \sum_{i=1}^{b} \sum_{j=1}^{n} \big|\big|\boldsymbol{y}_{j,i} - \sum_{r=1}^{n} {}_r w_{j,i}\, \boldsymbol{y}_{r,i}\big|\big|_S, \\ \text{subject to} \quad & \frac{1}{n} \sum_{j=1}^{n} \boldsymbol{y}_{j,i} \boldsymbol{y}_{j,i}^\top = \boldsymbol{I}, \quad \sum_{j=1}^{n} \boldsymbol{y}_{j,i} = \boldsymbol{0}, \quad \forall i \in \{1, \dots, b\}, \end{aligned} \tag{12}$$

where $\boldsymbol{I}$ is the identity matrix, the rows of $\mathbb{R}^{n \times p} \ni \boldsymbol{Y}_i := [\boldsymbol{y}_{1,i}, \dots, \boldsymbol{y}_{n,i}]^\top$ are the embedded i-th block in the images, $\boldsymbol{y}_{r,i} \in \mathbb{R}^p$ is the i-th embedded block in the r-th image, and ${}_r w_{j,i}$ is the weight obtained from the linear reconstruction (last section) if $\boldsymbol{x}_{r,i}$ is a neighbor of $\boldsymbol{x}_{j,i}$ and zero otherwise. The second constraint ensures the zero mean of embedded blocks. The first and second constraints together satisfy having unit covariance for the embedded image blocks.

Suppose $\mathbb{R}^n \ni \boldsymbol{w}_{j,i} := [{}_1 w_{j,i}, \dots, {}_n w_{j,i}]^\top$ and let $\mathbb{R}^n \ni \boldsymbol{1}_j := [0, \dots, 1, \dots, 0]^\top$ be the vector whose j-th element is one and other elements are zero. The Eq. (12) can be restated as:

$$\begin{aligned} \underset{\boldsymbol{Y}_i}{\text{minimize}} \quad & \sum_{i=1}^{b} \sum_{j=1}^{n} ||\boldsymbol{Y}_i^\top \boldsymbol{1}_j - \boldsymbol{Y}_i^\top \boldsymbol{w}_{j,i}||_S, \\ \text{subject to} \quad & \frac{1}{n} \boldsymbol{Y}_i^\top \boldsymbol{Y}_i = \boldsymbol{I}, \quad \boldsymbol{Y}_i^\top \boldsymbol{1} = \boldsymbol{0}, \quad \forall i \in \{1, \dots, b\}. \end{aligned} \tag{13}$$

Let $\theta_j(\boldsymbol{Y}_i) := ||\boldsymbol{Y}_i^\top \boldsymbol{1}_j - \boldsymbol{Y}_i^\top \boldsymbol{w}_{j,i}||_S$. According to Eq. (2), it is simplified to:

$$\mathbb{R} \ni \theta_j(\boldsymbol{Y}_i) = \frac{\textbf{tr}(\boldsymbol{Y}_i^\top \boldsymbol{M}_{j,i}\, \boldsymbol{Y}_i)}{\textbf{tr}(\boldsymbol{Y}_i^\top \boldsymbol{\Psi}_{j,i}\, \boldsymbol{Y}_i) + c}, \tag{14}$$

where $\textbf{tr}(.)$ is the trace of matrix, $\mathbb{R}^{n \times n} \ni \boldsymbol{M}_{j,i} := \boldsymbol{1}_j \boldsymbol{1}_j^\top + \boldsymbol{w}_{j,i} \boldsymbol{w}_{j,i}^\top - 2\, \boldsymbol{1}_j \boldsymbol{w}_{j,i}^\top$, and $\mathbb{R}^{n \times n} \ni \boldsymbol{\Psi}_{j,i} := \boldsymbol{1}_j \boldsymbol{1}_j^\top + \boldsymbol{w}_{j,i} \boldsymbol{w}_{j,i}^\top = \boldsymbol{M}_{j,i} + 2\, \boldsymbol{1}_j \boldsymbol{w}_{j,i}^\top$. The gradient of $\theta_j(\boldsymbol{Y}_i)$ with respect to $\boldsymbol{Y}_i$ is:

$$\mathbb{R}^{n \times p} \ni \nabla \theta_j(\boldsymbol{Y}_i) = \frac{2}{\textbf{tr}(\boldsymbol{Y}_i^\top \boldsymbol{\Psi}_{j,i}\, \boldsymbol{Y}_i) + c} \Big(\boldsymbol{M}_{j,i} - \theta_j(\boldsymbol{Y}_i)\, \boldsymbol{\Psi}_{j,i} \Big)\, \boldsymbol{Y}_i. \tag{15}$$

In Eq. (13), we can embed the constraint as an indicator function in the objective function [8]:

$$\begin{aligned} \underset{\boldsymbol{Y}_i, \boldsymbol{V}_i \in \mathbb{R}^{n \times p}}{\text{minimize}} \quad & \sum_{i=1}^{b} \Big(\sum_{j=1}^{n} \big(\theta_j(\boldsymbol{Y}_i)\big) + h_2(\boldsymbol{V}_i) \Big), \\ \text{subject to} \quad & \boldsymbol{Y} - \boldsymbol{V} = \boldsymbol{0}, \end{aligned} \tag{16}$$

where $h_2(\boldsymbol{V}_i) := \mathbb{I}\big(\boldsymbol{V}_i^\top \mathbf{1} = \mathbf{0} \wedge (1/n)\boldsymbol{V}_i^\top \boldsymbol{V}_i = \boldsymbol{I}\big)$. The $\boldsymbol{U}$ and $\boldsymbol{V}$ are union of partitions, i.e., $\boldsymbol{Y} := \cup_{i=1}^b \boldsymbol{Y}_i$ and $\boldsymbol{V} := \cup_{i=1}^b \boldsymbol{V}_i$ [9].

We can solve the Eq. (16) using Alternating Direction Method of Multipliers (ADMM) [8, 9]. The augmented Lagrangian is: $\mathcal{L}_\rho = \sum_{i=1}^b \Big(\sum_{j=1}^n \big(\theta_j(\boldsymbol{Y}_i)\big) + h(\boldsymbol{V}_i)\Big) + \mathbf{tr}\big(\boldsymbol{\Lambda}^\top(\boldsymbol{Y}-\boldsymbol{V})\big) + (\rho/2)\,||\boldsymbol{Y}-\boldsymbol{V}||_F^2 = \sum_{i=1}^b \Big(\sum_{j=1}^n \big(\theta_j(\boldsymbol{Y}_i)\big) + h(\boldsymbol{V}_i)\Big) + (\rho/2)\,||\boldsymbol{Y}-\boldsymbol{V}+\boldsymbol{J}||_F^2 - (\rho/2)\,||\boldsymbol{\Lambda}||_F^2$, where $\boldsymbol{\Lambda} := \cup_{i=1}^b \boldsymbol{\Lambda}_i$ is the Lagrange multiplier, $\rho > 0$, and $\boldsymbol{J} := (1/\rho)\boldsymbol{\Lambda} = (1/\rho) \cup_{i=1}^b \boldsymbol{\Lambda}_i = \cup_{i=1}^b \boldsymbol{J}_i$. The term $(\rho/2)\,||\boldsymbol{\Lambda}||_F^2$ is a constant with respect to $\boldsymbol{Y}$ and $\boldsymbol{V}$ and can be dropped. The updates of $\boldsymbol{Y}$, $\boldsymbol{V}$, and $\boldsymbol{J}$ are done as [8, 9]:

$$\boldsymbol{Y}_i^{(\nu+1)} := \arg\min_{\boldsymbol{Y}_i} \Big(\sum_{j=1}^n \big(\theta_j(\boldsymbol{Y}_i)\big) + (\rho/2)\,||\boldsymbol{Y}_i - \boldsymbol{V}_i^{(\nu)} + \boldsymbol{J}_i^{(\nu)}||_F^2 \Big), \tag{17}$$

$$\boldsymbol{V}_i^{(\nu+1)} := \arg\min_{\boldsymbol{V}_i} \Big(h_2(\boldsymbol{V}_i) + (\rho/2)\,||\boldsymbol{Y}_i^{(\nu+1)} - \boldsymbol{V}_i + \boldsymbol{J}_i^{(\nu)}||_F^2 \Big), \tag{18}$$

$$\boldsymbol{J}^{(\nu+1)} := \boldsymbol{J}^{(\nu)} + \boldsymbol{Y}^{(\nu+1)} - \boldsymbol{V}^{(\nu+1)}. \tag{19}$$

The gradient of the objective function in Eq. (17) is $\sum_{j=1}^n \big(\nabla\theta_j(\boldsymbol{Y}_i)\big) + \rho\,(\boldsymbol{Y}_i - \boldsymbol{V}_i^{(\nu)} + \boldsymbol{J}_i^{(\nu)})$. Similar to Eq. (7), we replace Eq. (17) with one iteration of gradient descent.

With the same explanation for Eq. (8), Eq. (18) is equivalent to the projection $\Pi(\boldsymbol{Y}_i^{(\nu+1)} + \boldsymbol{J}_i^{(\nu)})$. One of the constraints in Eq. (13) is $\boldsymbol{Y}_i^\top \mathbf{1} = \mathbf{0}$. Therefore, the row mean of the matrix should removed, i.e., $\boldsymbol{Y}_i := \boldsymbol{H}\boldsymbol{Y}_i$, where $\mathbb{R}^{n\times n} \ni \boldsymbol{H} := \boldsymbol{I} - (1/n)\mathbf{1}\mathbf{1}^\top$ is the centering matrix and $\mathbf{1}$ is the vector of ones. The other constraint in Eq. (13) is $(1/n)\boldsymbol{Y}_i^\top \boldsymbol{Y}_i = \boldsymbol{I}$. The variable of proximal operator, which is a projection here, is a matrix and not a vector. According to [11], if F is a convex and orthogonally invariant function and it works on the singular values of a matrix variable $\boldsymbol{A} \in \mathbb{R}^{n\times p}$, i.e., $F = f \circ \sigma$ where the function $\sigma(\boldsymbol{A})$ gives the vector of singular values of $\boldsymbol{A}$, then the proximal operator is $\mathbf{prox}_{\lambda,F}(\boldsymbol{A}) := \boldsymbol{Q}\ \mathbf{diag}\Big(\mathbf{prox}_{\lambda,f}\big(\sigma(\boldsymbol{A})\big)\Big)\ \boldsymbol{\Omega}^\top$. The $\boldsymbol{Q} \in \mathbb{R}^{n\times p}$ and $\boldsymbol{\Omega} \in \mathbb{R}^{p\times p}$ are the matrices of left and right singular vectors of $\boldsymbol{A}$, respectively. In our constraint $(1/n)\boldsymbol{Y}_i^\top \boldsymbol{Y}_i = \boldsymbol{I}$, the function F deals with the singular values of $\boldsymbol{Y}_i$. The reason is that we want: $\boldsymbol{Y}_i \overset{\text{SVD}}{=} \boldsymbol{Q}\boldsymbol{\Sigma}\boldsymbol{\Omega}^\top \implies (1/n)\boldsymbol{Y}_i^\top \boldsymbol{Y}_i = (1/n)\boldsymbol{\Omega}\boldsymbol{\Sigma}\boldsymbol{Q}^\top\boldsymbol{Q}\boldsymbol{\Sigma}\boldsymbol{\Omega}^\top \overset{(a)}{=} (1/n)\boldsymbol{\Omega}\boldsymbol{\Sigma}^2\boldsymbol{\Omega}^\top \overset{\text{set}}{=} \boldsymbol{I} \implies (1/n)\boldsymbol{\Omega}\boldsymbol{\Sigma}^2\boldsymbol{\Omega}^\top\boldsymbol{\Omega} = \boldsymbol{\Omega} \overset{(b)}{\implies} (1/n)\boldsymbol{\Omega}\boldsymbol{\Sigma}^2 = \boldsymbol{\Omega} \implies \boldsymbol{\Sigma} = n\boldsymbol{I}$, where (a) and (b) are because $\boldsymbol{Q}$ and $\boldsymbol{\Omega}$ are orthogonal matrices. Thus, projection onto the second constraint is equivalent to decomposing the matrix with Singular Value Decomposition (SVD) and setting all the singular values to n. To sum up, $\Pi(\boldsymbol{Y}_i^{(\nu+1)} + \boldsymbol{J}_i^{(\nu)})$ first removes the row mean of $(\boldsymbol{Y}_i^{(\nu+1)} + \boldsymbol{J}_i^{(\nu)})$ and then sets the singular values of $(\boldsymbol{Y}_i^{(\nu+1)} + \boldsymbol{J}_i^{(\nu)})$ to n. In summary, the Eqs. (17), (18), and

(19) can be restated as:

$$\begin{aligned}
\boldsymbol{Y}_i^{(\nu+1)} &:= \boldsymbol{Y}_i^{(\nu)} - \eta \sum_{j=1}^{n} \big(\nabla \theta_j(\boldsymbol{Y}_i)\big) - \eta\, \rho\, (\boldsymbol{Y}_i - \boldsymbol{V}_i^{(\nu)} + \boldsymbol{J}_i^{(\nu)}), \\
\boldsymbol{V}_i^{(\nu+1)} &:= \Pi(\boldsymbol{Y}_i^{(\nu+1)} + \boldsymbol{J}_i^{(\nu)}), \\
\boldsymbol{J}^{(\nu+1)} &:= \boldsymbol{J}^{(\nu)} + \boldsymbol{Y}^{(\nu+1)} - \boldsymbol{V}^{(\nu+1)}.
\end{aligned} \tag{20}$$

Iteratively solving Eq. (20) until convergence gives us the $\boldsymbol{Y}_i$ for for the image blocks indexed by i. The rows of $\boldsymbol{Y}_i$ are the p-dimensional embedded image blocks in the *LLISE manifold*. Unlike LLE, the first column of $\boldsymbol{Y}_i$ is not ignored in LLISE because it is not based on ℓ_2 norm and thus eigenvalue problem.

2.2 Embedding The Out-of-sample Data

There exist two methods in the literature for extension of LLE for out-of-sample embedding. The first method is based on the concept of eigenfunctions [12] and the second method uses linear reconstruction of the out-of-sample data [13]. The first method cannot be used for LLISE because it does not result in closed-form eigenvalue problem as in LLE. We use the second approach.

Suppose we have n_t out-of-sample images and $\breve{\boldsymbol{x}}_{j,i}^{(t)}$ denotes the i-th block in the j-th out-of-sample image. For the i-th block in every out-of-sample image, we first find the k-NN among the i-th block in training images. Let ${}_r\breve{\boldsymbol{x}}_{j,i}^{(t)}$ and $\mathbb{R}^{q\times k} \ni \breve{\boldsymbol{X}}_{j,i}^{(t)} := [\,{}_1\breve{\boldsymbol{x}}_{j,i}^{(t)}, \dots,\, {}_k\breve{\boldsymbol{x}}_{j,i}^{(t)}]$ denote the r-th training neighbor of $\breve{\boldsymbol{x}}_{j,i}^{(t)}$ and the matrix including the training neighbors of $\breve{\boldsymbol{x}}_{j,i}^{(t)}$, respectively. We want to reconstruct every out-of-sample image block by its training neighbors:

$$\begin{aligned}
\underset{\widetilde{\boldsymbol{W}}_i^{(t)}}{\text{minimize}} \quad & \sum_{i=1}^{b} \varepsilon(\widetilde{\boldsymbol{W}}_i^{(t)}) := \sum_{i=1}^{b}\sum_{j=1}^{n_t} \Big|\Big| \breve{\boldsymbol{x}}_{j,i}^{(t)} - \sum_{r=1}^{k} {}_r\widetilde{w}_{j,i}^{(t)}\; {}_r\breve{\boldsymbol{x}}_{j,i}^{(t)} \Big|\Big|_S, \\
\text{subject to} \quad & \sum_{r=1}^{k} ({}_r\widetilde{w}_{j,i}^{(t)})^2 = 1, \quad \forall i \in \{1,\dots,b\}, \;\; \forall j \in \{1,\dots,n_t\},
\end{aligned} \tag{21}$$

where $\mathbb{R}^{n_t\times k} \ni \widetilde{\boldsymbol{W}}_i^{(t)} := [\widetilde{\boldsymbol{w}}_{1,i}^{(t)}, \dots, \widetilde{\boldsymbol{w}}_{n_t,i}^{(t)}]^\top$ includes the weights, $\mathbb{R}^k \ni \widetilde{\boldsymbol{w}}_{j,i}^{(t)} := [\,{}_1\widetilde{w}_{j,i}^{(t)}, \dots,\, {}_k\widetilde{w}_{j,i}^{(t)}]^\top$ includes the weights of linear reconstruction of the i-th block in the j-th out-of-sample image using the i-th block in its k training neighbors. Note that Eq. (21) is similar to Eq. (3) and is solved using Eq. (11) where $\widetilde{\boldsymbol{w}}_{j,i}^{(t)}$, $\breve{\boldsymbol{x}}_{j,i}^{(t)}$, and $\breve{\boldsymbol{X}}_{j,i}^{(t)}$ are used in the expressions.

The embedding $\boldsymbol{y}_{j,i}^{(t)}$ of the i-th block in the j-th out-of-sample image, i.e., $\boldsymbol{x}_{j,i}^{(t)}$, is obtained by the linear reconstruction of the embedding of the i-th block in its k training neighbors:

$$\mathbb{R}^p \ni \boldsymbol{y}_{j,i}^{(t)} = \sum_{r=1}^{k} {}_r\widetilde{w}_{j,i}^{(t)}\; {}_r\boldsymbol{y}_{j,i}^{(t)}, \tag{22}$$

where ${}_r\boldsymbol{y}_{j,i}^{(t)} \in \mathbb{R}^p$ is the embedding of ${}_r\breve{\boldsymbol{x}}_{j,i}^{(t)}$ which was found by the linear embedding of the training data, $\boldsymbol{Y}_i$.

3 Kernel Locally Linear Image Structural Embedding

We can map the block $\breve{\boldsymbol{x}}_i \in \mathbb{R}^d$ to higher-dimensional feature space hoping to have the data fall close to a simpler-to-analyze manifold in the feature space. Suppose $\boldsymbol{\phi} : \breve{\boldsymbol{x}} \to \mathcal{H}$ is a function which maps the data $\breve{\boldsymbol{x}}$ to the feature space. In other words, $\breve{\boldsymbol{x}} \mapsto \boldsymbol{\phi}(\breve{\boldsymbol{x}})$. Let t denote the dimensionality of the feature space, i.e., $\boldsymbol{\phi}(\breve{\boldsymbol{x}}_i) \in \mathbb{R}^t$. We usually have $t \gg d$. The kernel of the i-th block in images 1 and 2, which are $\breve{\boldsymbol{x}}_{1,i}$ and $\breve{\boldsymbol{x}}_{2,i}$, is $k(\breve{\boldsymbol{x}}_{1,i}, \breve{\boldsymbol{x}}_{2,i}) := \boldsymbol{\phi}(\breve{\boldsymbol{x}}_{1,i})^\top \boldsymbol{\phi}(\breve{\boldsymbol{x}}_{2,i}) \in \mathbb{R}$.

Let $\boldsymbol{K} = \boldsymbol{\Phi}(\breve{\boldsymbol{X}}_i)^\top \boldsymbol{\Phi}(\breve{\boldsymbol{X}}_i) \in \mathbb{R}^{n \times n}$ be the kernel between the i-th block in the n images. We can normalize it as $\boldsymbol{K}(a,b) := \boldsymbol{K}(a,b)/\sqrt{\boldsymbol{K}(a,a)\boldsymbol{K}(b,b)}$ where $\boldsymbol{K}(a,b)$ denotes the (a,b)-th element of the kernel matrix [14]. Then, the kernel is double-centered as $\boldsymbol{K} := \boldsymbol{H}\boldsymbol{K}\boldsymbol{H}$. The reason for double-centering is that Eq. (2) requires $\boldsymbol{\phi}(\breve{\boldsymbol{x}}_i)$ and thus the $\boldsymbol{\Phi}(\breve{\boldsymbol{X}}_i)$ to be centered. Therefore, in kernel LLISE, we center the kernel rather than centering $\breve{\boldsymbol{x}}_i$. Kernel LLISE maps the data to the feature space and performs the steps of k-NN and linear reconstruction in the feature space.

3.1 Embedding The Training Data

***k*-Nearest Neighbors** The Euclidean distance in the feature space is [15]:

$$||\boldsymbol{\phi}(\breve{\boldsymbol{x}}_{a,i}) - \boldsymbol{\phi}(\breve{\boldsymbol{x}}_{b,i})||_2 = \sqrt{k(\breve{\boldsymbol{x}}_{a,i}, \breve{\boldsymbol{x}}_{a,i}) - 2k(\breve{\boldsymbol{x}}_{a,i}, \breve{\boldsymbol{x}}_{b,i}) + k(\breve{\boldsymbol{x}}_{b,i}, \breve{\boldsymbol{x}}_{b,i})}. \tag{23}$$

For every block i amongst the images, we construct the k-NN graph using the distances of the blocks in the feature space. Therefore, every block has k neighbors in the feature space. Let the matrix $\mathbb{R}^{t \times k} \ni \boldsymbol{\Phi}(\breve{\boldsymbol{X}}_{j,i}) := [\boldsymbol{\phi}({}_1\breve{\boldsymbol{x}}_{j,i}), \dots, \boldsymbol{\phi}({}_k\breve{\boldsymbol{x}}_{j,i})]$ include the neighbors of $\breve{\boldsymbol{x}}_{j,i}$ in the feature space.

Linear Reconstruction by the Neighbors For finding the reconstruction weights $\mathbb{R}^k \ni \widetilde{\boldsymbol{w}}_{j,i} = [{}_1\widetilde{w}_{j,i}, \dots, {}_k\widetilde{w}_{j,i}]^\top$, the Eq. (3) is used in the feature space:

$$\begin{aligned} \underset{\widetilde{\boldsymbol{W}}_i}{\text{minimize}} \quad & \varepsilon(\widetilde{\boldsymbol{W}}_i) := \sum_{i=1}^{b}\sum_{j=1}^{n} \Big|\Big| \boldsymbol{\phi}(\breve{\boldsymbol{x}}_{j,i}) - \sum_{r=1}^{k} {}_r\widetilde{w}_{j,i}\, \boldsymbol{\phi}({}_r\breve{\boldsymbol{x}}_{j,i}) \Big|\Big|_S^2, \\ \text{subject to} \quad & \sum_{r=1}^{k} {}_r\widetilde{w}_{j,i}^2 = 1, \quad \forall i \in \{1,\dots,b\}, \ \ \forall j \in \{1,\dots,n\}. \end{aligned} \tag{24}$$

Let $f^\phi(\widetilde{\boldsymbol{w}}_{j,i}) := \Big|\Big|\boldsymbol{\phi}(\breve{\boldsymbol{x}}_{j,i}) - \sum_{r=1}^{k} {}_r\widetilde{w}_{ij}\, \boldsymbol{\phi}({}_r\breve{\boldsymbol{x}}_{j,i})\Big|\Big|_S^2$. According to Eq. (2), we have:

$$\mathbb{R} \ni f^\phi(\widetilde{\boldsymbol{w}}_{j,i}) = \frac{k_{j,i} + \widetilde{\boldsymbol{w}}_{j,i}^\top \boldsymbol{K}_{j,i}\, \widetilde{\boldsymbol{w}}_{j,i} - 2\, \widetilde{\boldsymbol{w}}_{j,i}^\top \boldsymbol{k}_{j,i}}{k_{j,i} + \widetilde{\boldsymbol{w}}_{j,i}^\top \boldsymbol{K}_{j,i}\, \widetilde{\boldsymbol{w}}_{j,i} + c}, \tag{25}$$

where $\mathbb{R} \ni k_{j,i} := \boldsymbol{\phi}(\breve{\boldsymbol{x}}_{j,i})^\top \boldsymbol{\phi}(\breve{\boldsymbol{x}}_{j,i})$, $\mathbb{R}^k \ni \boldsymbol{k}_{j,i} := \boldsymbol{\Phi}(\breve{\boldsymbol{X}}_{j,i})^\top \boldsymbol{\phi}(\breve{\boldsymbol{x}}_{j,i})$, and $\mathbb{R}^{k\times k} \ni \boldsymbol{K}_{j,i} := \boldsymbol{\Phi}(\breve{\boldsymbol{X}}_{j,i})^\top \boldsymbol{\Phi}(\breve{\boldsymbol{X}}_{j,i})$. The gradient of $f^\phi(\widetilde{\boldsymbol{w}}_{j,i})$ with respect to $\widetilde{\boldsymbol{w}}_{j,i}$ is:

$$\mathbb{R}^k \ni \nabla f^\phi(\widetilde{\boldsymbol{w}}_{j,i}) = \frac{2\Big(\big(1 - f^\phi(\widetilde{\boldsymbol{w}}_{j,i})\big)\boldsymbol{K}_{j,i}\,\widetilde{\boldsymbol{w}}_{j,i} - \boldsymbol{k}_{j,i}\Big)}{k_{j,i} + \widetilde{\boldsymbol{w}}_{j,i}^\top \boldsymbol{K}_{j,i}\,\widetilde{\boldsymbol{w}}_{j,i} + c}. \tag{26}$$

We can use Eq. (11) for solving Eq. (24) where $\nabla f^\phi(\widetilde{\boldsymbol{w}}_{j,i})$ is used in place of $\nabla f(\widetilde{\boldsymbol{w}}_{j,i})$. The linear embedding in kernel LLISE is the same as the linear embedding in LLISE. The rows of obtained $\boldsymbol{Y}_i$ are the i-th embedded block of the images in *kernel LLISE manifold*.

3.2 Embedding The Out-of-sample Data

For embedding every out-of-sample image, we reconstruct it by its training neighbors in the feature space. The Eq. (21) in the feature space is:

$$\begin{aligned} \underset{\widetilde{\boldsymbol{W}}_i^{(t)}}{\text{minimize}} \quad & \sum_{i=1}^{b} \varepsilon(\widetilde{\boldsymbol{W}}_i^{(t)}) := \sum_{i=1}^{b}\sum_{j=1}^{n_t} \Big|\Big| \boldsymbol{\phi}(\breve{\boldsymbol{x}}_{j,i}^{(t)}) - \sum_{r=1}^{k} {}_r\widetilde{w}_{j,i}^{(t)}\, \boldsymbol{\phi}({}_r\breve{\boldsymbol{x}}_{j,i}^{(t)}) \Big|\Big|_S, \\ \text{subject to} \quad & \sum_{r=1}^{k} ({}_r\widetilde{w}_{j,i}^{(t)})^2 = 1, \quad \forall i \in \{1,\dots,b\}, \quad \forall j \in \{1,\dots,n_t\}, \end{aligned} \tag{27}$$

which is similar to Eq. (24) and is solved similarly. Here, the used kernels are $\mathbb{R} \ni k_{j,i} = \boldsymbol{\phi}(\breve{\boldsymbol{x}}_{j,i}^{(t)})^\top \boldsymbol{\phi}(\breve{\boldsymbol{x}}_{j,i}^{(t)})$, $\mathbb{R}^k \ni \boldsymbol{k}_{j,i} = \boldsymbol{\Phi}(\breve{\boldsymbol{X}}_{j,i}^{(t)})^\top \boldsymbol{\phi}(\breve{\boldsymbol{x}}_{j,i}^{(t)})$, and $\mathbb{R}^{k\times k} \ni \boldsymbol{K}_{j,i} = \boldsymbol{\Phi}(\breve{\boldsymbol{X}}_{j,i}^{(t)})^\top \boldsymbol{\Phi}(\breve{\boldsymbol{X}}_{j,i}^{(t)})$. After finding the weights $\widetilde{\boldsymbol{w}}_{j,i}^{(t)} = [{}_1\widetilde{w}_{j,i}^{(t)}, \dots, {}_k\widetilde{w}_{j,i}^{(t)}]$ from Eq. (27), the embedding of the out-of-sample $\boldsymbol{x}_{j,i}^{(t)}$ is found using Eq. (22) where ${}_r\boldsymbol{y}_{j,i}^{(t)} \in \mathbb{R}^p$ is the embedding of ${}_r\breve{\boldsymbol{x}}_{j,i}^{(t)}$ in kernel LLISE.

4 Experiments

Training 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 the training set is 121 including the original image. For every type of distortion, 20 different levels of MSE, i.e., from MSE = 45 to MSE = 900 with step 45, were generated to have images on the equal-MSE or *iso-error* hypersphere [3].

Embedding the Training Images: We embedded the blocks in the training images. In k-NN, we used $k = 10$. For linear reconstruction, we used $\rho = \eta = 0.1$ in LLISE and $10\rho = \eta = 0.1$ in kernel LLISE. For linear embedding, we used $\rho = \eta = 0.01$. We took $q = 64$ (8×8 blocks inspired by [6, 9]), $p = 4$, and $d = 512 \times 512 = 262144$. In order to evaluate the obtained embedded manifold, we used

Fig. 1. Samples 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.

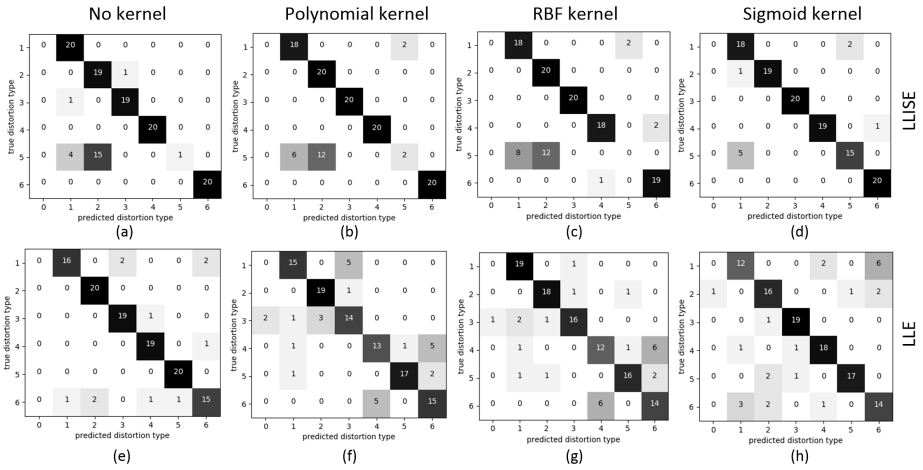

Fig. 2. Confusion matrices for recognition of distortion types with a 1NN classifier used in the embedded space. Matrices (a) and (e) correspond to LLISE and LLE, respectively. Matrices (b) to (d) are for kernel LLISE and (f) and (g) are for kernel LLE with polynomial, RBF, and sigmoid kernels, respectively. The 0 label corresponds to the original image and the labels 1 to 6 are the distortion types with the same order as in Fig. 1.

the 1-Nearest Neighbor (1NN) classifier to recognize the distortion type of every block. The 1NN is useful to show how to evaluate the manifold by closeness of the embedded distortions. The distortion type of an image comes from a majority vote among the blocks. The polynomial $(\gamma\, \breve{\boldsymbol{x}}_1^\top \breve{\boldsymbol{x}}_2 + 1)^3$, Radial Basis function (RBF) $\exp(-\gamma\, ||\breve{\boldsymbol{x}}_1 - \breve{\boldsymbol{x}}_2||_2^2)$, and sigmoid $\tanh(\gamma\, \breve{\boldsymbol{x}}_1^\top \breve{\boldsymbol{x}}_2 + 1)$ kernels were tested for kernel LLISE, where $\gamma := 1/q$. The confusion matrices for distortion recognition are shown in Fig. 2. Also, the LLISE and kernel LLISE are compared with LLE and kernel LLE in this figure. Except for impulse noise, LLISE and kernel LLISE had better performance compared to LLE and kernel LLE. The reason is that impulse noise and Gaussian noise are very similar in terms of structure. In other distortions, especially in JPEG distortion and contrast stretch, the performances of LLE and kernel LLE are not acceptable because LLE uses ℓ_2 norm rather than SSIM distance.

Fig. 3. 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.

Table 1. Recognition of distortions for out-of-sample images. Letters O, C, G, L, B, I, and J correspond to original image, contrast stretch, Gaussian noise, luminance enhanced, blurring, impulse noise, and JPEG distortion, respectively.

image	1	2	3	4	5	6	7	8	9	10	11	12
distortion	C	G	L	B	I	J	B + G	B + L	I + L	J + G	J + L	J + C
LLISE	42.9% C	42.2% G	35.6% L	44.5% B	31.2% G	43.2% J	46.8% G	41.3% B	55.9% G	33.5% G	39.6% J	40.7% J
	22.8% L	29.3% I	29.6% C	15.7% J	28.4% I	16.8% B	34.4% I	14.7% J	39.6% I	26.5% I	16.3% B	16.3% L
kernel LLISE (polynomial)	69.7% C	23.7% L	97.5% L	74.3% B	45.4% C	76.6% J	20.7% G	78.1% L	35.1% G	27.7% L	76.9% L	45.1% J
	14.7% I	21.3% G	0.8% C	14.5% J	19.5% I	13.9% B	17.9% B	5.7% I	23.5% L	16.6% G	7.0% B	28.6% B
kernel LLISE (RBF)	62.1% C	25.6% L	72.6% L	58.1% B	42.3% C	59.6% J	23.7% B	58.1% L	33.3% L	26.5% L	57.8% L	40.0% J
	12.0% I	16.1% B	9.5% C	16.3% J	17.0% I	16.4% B	18.5% J	11.9% B	24.7% G	19.6% J	13.0% B	24.8% B
kernel LLISE (sigmoid)	63.0% C	28.5% L	92.4% L	68.5% B	51.9% C	55.6% J	39.2% B	77.8% L	57.0% L	31.1% L	76.1% L	42.8% J
	14.5% I	24.5% C	3.5% B	15.5% J	14.0% I	19.7% B	19.5% L	10.7% B	12.6% C	24.6% B	10.8% B	29.4% B
LLE	C	L	L	B	C	J	B	C	C	L	L	B
kernel LLE (polynomial)	L	C	L	B	C	J	B	L	L	B	L	J
kernel LLE (RBF)	L	C	C	J	C	J	B	L	L	B	L	J
kernel LLE (sigmoid)	C	L	L	B	C	J	J	C	L	C	C	C

Out-of-sample Embedding: For out-of-sample embedding, we made 12 test images with MSE = 500 having different distortions and some having a combination of different distortions (see Fig. 3). Again, for linear reconstruction, we used $\rho = \eta = 0.1$ in LLISE and $10\rho = \eta = 0.1$ in kernel LLISE. The same 1NN classification was done for the test images. Table 1 reports the top two votes of blocks for every image with the percentage of blocks voting for those distortions. This table also shows the recognition of distortions using LLE and kernel LLE. Note that LLE does not perform block-wise and thus it has only one recognition label for the whole image. As expected for LLISE and kernel LLISE, in most cases, at least one of the two top votes recognized the type of distortion(s) the out-of-sample images had. However, LLE and kernel LLE performed poorly on the out-of-sample images.

5 Conclusion and Future Work

This paper introduced the concept of an image structure manifold which discriminates the types of distortions applied on the images and captures the structure of an image. A new method, named LLISE, was proposed for learning this manifold in both original and feature space. The LLISE is inspired by LLE which uses ℓ_2 norm. As a possible future work, we seek to design other new methods for learning the image structure manifold.

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., Vrscay, E.R., Wang, Z.: On the mathematical properties of the structural similarity index. IEEE Transactions on Image Processing **21**(4) (2012) 1488–1499
5. Roweis, S.T., Saul, L.K.: Nonlinear dimensionality reduction by locally linear embedding. Science **290**(5500) (2000) 2323–2326
6. Otero, D., Vrscay, E.R.: Unconstrained structural similarity-based optimization. In: International Conference Image Analysis & Recognition, Springer (2014) 167–176
7. Zhao, X., Zhang, S.: Facial expression recognition using local binary patterns and discriminant kernel locally linear embedding. EURASIP journal on Advances in signal processing (2012) 1–9
8. Boyd, S., Parikh, N., Chu, E., Peleato, B., Eckstein, J., et al.: Distributed optimization and statistical learning via the alternating direction method of multipliers. Foundations and Trends® in Machine learning **3**(1) (2011) 1–122
9. Otero, D., La Torre, D., Michailovich, O.V., Vrscay, E.R.: Alternate direction method of multipliers for unconstrained structural similarity-based optimization. In: International Conference Image Analysis & Recognition, Springer (2018) 20–29
10. Boyd, S., Vandenberghe, L.: Convex optimization. Cambridge university press (2004)
11. Parikh, N., Boyd, S.: Proximal algorithms. Foundations and Trends® in Optimization **1**(3) (2014) 127–239
12. Bengio, Y., Paiement, J.f., Vincent, P., Delalleau, O., Roux, N.L., Ouimet, M.: Out-of-sample extensions for LLE, Isomap, MDS, Eigenmaps, and spectral clustering. In: Advances in neural information processing systems. (2004) 177–184
13. Saul, L.K., Roweis, S.T.: Think globally, fit locally: unsupervised learning of low dimensional manifolds. Journal of machine learning research **4** (2003) 119–155
14. Ah-Pine, J.: Normalized kernels as similarity indices. In: Pacific-Asia Conference on Knowledge Discovery and Data Mining, Springer (2010) 362–373
15. Schölkopf, B.: The kernel trick for distances. In: Advances in neural information processing systems. (2001) 301–307