Image De-noising with Kernel Principal Component Analysis implementing the Morlet Wavelet Kernel

Kernel Kings
Uddalak Mukherjee
Biswajit Rana
uddalakmukherjee49@gmail.com
ranabiswajit911@gmail.com

May 1, 2024

Abstract

In this project, we present a novel approach for denoising images from the Smartphone Image Denoising Dataset (SIDD) using Kernel Principal Component Analysis (Kernel PCA) with the Morlet kernel. Our method utilizes the Morlet kernel (introduced in [3]) to capture both spatial and frequency information, facilitating effective representation of image features. Employing Kernel PCA, we aim to extract informative features while reducing data dimensionality, thereby enabling efficient denoising. We conduct comprehensive evaluations, comparing our approach with established state-of-the-art methods such as Mean and Median Filters, CBDNet and CGNet, on a subset of the SIDD dataset. While our method may not consistently outperform existing techniques, our results highlight its competitive performance in noise reduction and preservation of crucial image details. These findings underscore the potential of integrating a wavelet-based kernel within the Kernel PCA framework for image denoising applications, offering valuable insights into its efficacy and limitations in real-world scenarios.

1 Introduction

Our project aims to develop an innovative method for denoising images from the Smartphone Image Denoising Dataset (SIDD) by leveraging Kernel Principal Component Analysis (Kernel PCA) with the Morlet Wavelet kernel. Noise reduction in images is crucial for enhancing visual quality and utility, especially in challenging conditions like low-light environments or smartphone photography. **The motivation behind using this Kernel is that it contains both an exponential and a trigonometric component that**

will help us model the noise much better than other kernels using only an exponential component like the Gaussian RBF Kernel.

To address the complexity of high-dimensional images, we adopt a patch-based approach. By denoising smaller patches individually using Kernel PCA with the Morlet kernel and then reconstructing the image, we effectively manage computational complexity while preserving image details.

Through extensive evaluations and comparisons with state-of-the-art methods like Mean and Median Filters, CGNet and CBDNet, we assess the performance of our approach across various scenarios. Our results offer valuable insights into the effectiveness and practicality of our denoising method in real-world image processing applications.

2 Literature review

2.1 Existing Methodologies

Mean Filter: A simple denoising technique that replaces each pixel's value with the average value of its neighborhood. It is effective in smoothing out noise but may blur edges and fine details in the image.

Median Filter: Another simple denoising technique that replaces each pixel's value with the median value of its neighborhood. It effectively removes impulse noise (salt and pepper noise) while preserving edges and fine details better than mean filtering.

CascadedGaze Network (CGNet): CascadedGaze Network (CGNet), an encoder-decoder architecture that employs Global Context Extractor (GCE), a novel and efficient way to capture global information for image restoration. The GCE module leverages small kernels across convolutional layers to learn global dependencies, without requiring self-attention. Extensive experimental results show that our approach outperforms a range of state-of-the-art methods on denoising benchmark datasets including both real image denoising and synthetic image denoising, as well as on image deblurring task, while being more computationally efficient.

Principal Component Analysis (PCA): A dimensionality reduction technique that projects the high-dimensional image data onto a lower-dimensional subspace spanned by the principal components. PCA can effectively separate noise from signal components and is widely used for image denoising.

Convolutional Blind Denoising Network (CBDNet): Training a convolutional blind denoising network (CBDNet) with more realistic noise model and real-world noisy-clean image pairs. On the one hand, both signal-dependent noise and in-camera signal processing pipeline is considered to synthesize realistic noisy images. On the other hand, real-world noisy photographs and their nearly noise-free counterparts are also included to train our CBDNet.

2.2 Kernels

2.2.1 An Introductory Example

Example: Consider the following dataset consisting of three points in two-dimensional space:

$$\mathcal{D} = \{(x_1, y_1), (x_2, y_2), (x_3, y_3)\} = \{(1, 2), (3, 4), (5, 6)\}$$

where x_i are input features and y_i are target labels.

Suppose we want to compute the similarity between each pair of points using a kernel function. One common choice is the Gaussian (RBF) kernel:

$$k(x_i, x_j) = \exp\left(-\frac{\|x_i - x_j\|^2}{2\sigma^2}\right)$$

Let's choose $\sigma = 1$ for simplicity. Then, the kernel matrix K is computed as:

$$K = \begin{pmatrix} k(x_1, x_1) & k(x_1, x_2) & k(x_1, x_3) \\ k(x_2, x_1) & k(x_2, x_2) & k(x_2, x_3) \\ k(x_3, x_1) & k(x_3, x_2) & k(x_3, x_3) \end{pmatrix}$$

Substituting the values, we get:

$$K = \begin{pmatrix} 1 & 0.018 & 1.84 \times 10^{-8} \\ 0.018 & 1 & 0.018 \\ 1.84 \times 10^{-8} & 0.018 & 1 \end{pmatrix}$$

This kernel matrix K represents the pairwise similarities between the data points in the feature space induced by the Gaussian kernel.

2.2.2 Mercer Conditions

Mercer's Conditions:

Mercer's theorem provides necessary and sufficient conditions for a function to be a valid kernel, ensuring that it corresponds to an inner product in some feature space. Let $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ be a symmetric function.

1. **Symmetry:** The kernel function must be symmetric, i.e., $k(x, x') = k(x', x)$ for all $x, x' \in \mathcal{X}$.

2. **Positive Definiteness:** For any finite subset $\{x_1, x_2, \ldots, x_n\} \subset \mathcal{X}$, the Gram matrix $K = [k(x_i, x_j)]_{i,j=1}^n$ must be positive semidefinite. Mathematically, $c^T K c \geq 0$ for any vector $c \in \mathbb{R}^n$, where c^T denotes the transpose of c.

   This conditions can also be expressed in terms of double summations over data points. Let $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ be a symmetric function representing a kernel.

For any finite set of data points $\{x_1, x_2, \ldots, x_n\} \subset \mathcal{X}$, the following condition must hold:

$$\sum_{i=1}^{n} \sum_{j=1}^{n} c_i c_j k(x_i, x_j) \geq 0$$

for all $c_1, c_2, \ldots, c_n \in \mathbb{R}$.

Importance of Mercer's Conditions:

Mercer's conditions are essential for ensuring the validity of kernels in machine learning algorithms such as Support Vector Machines (SVMs) and Kernel Principal Component Analysis (KPCA). Here's why:

- **Inner Product Correspondence:** Mercer's conditions guarantee that the kernel function represents a valid inner product in some feature space. This property is crucial for leveraging kernel methods to implicitly map data into high-dimensional feature spaces.
- **Positive Semidefiniteness:** The positive definiteness requirement ensures that the kernel matrix is positive semidefinite for any finite subset of data points. This property is fundamental for the stability and convergence of algorithms utilizing kernel matrices, such as SVMs and KPCA.
- **Generalization:** Kernels satisfying Mercer's conditions allow algorithms to generalize effectively to unseen data points by capturing complex nonlinear relationships in the data. This enables the application of kernel methods to a wide range of machine learning tasks.

Other Forms of Mercer's Conditions:

In addition to the original Mercer's conditions, there are alternative formulations and extensions, including:

- **Mercer's Theorem (Simplified):** Let $\mathcal{X}$ be a compact subset of $\mathbf{R}^n$. Suppose k is a continuous symmetric function such that the integral operator $T_k : L_2(\mathcal{X}) \to L_2(\mathcal{X})$ defined by

  $$(T_k f)(\cdot) = \int_{\mathcal{X}} k(\cdot, \mathbf{x}) f(\mathbf{x}) d\mathbf{x}$$

  is positive; which here means $\forall f \in L_2(\mathcal{X})$,

  $$\int_{\mathcal{X} \times \mathcal{X}} k(\mathbf{x}, \mathbf{z}) f(\mathbf{x}) f(\mathbf{z}) d\mathbf{x} d\mathbf{z} \geq 0,$$

then we can expand $k(\mathbf{x}, \mathbf{z})$ in a uniformly convergent series in terms of T_k 's eigenfunctions $\psi_j \in L_2(\mathcal{X})$, normalized so that $\|\psi\|_{L_2} = 1$, and positive associated eigenvalues $\lambda_j \geq 0$,

$$k(\mathbf{x}, \mathbf{z}) = \sum_{j=1}^{\infty} \lambda_j \psi_j(\mathbf{x}) \psi_j(\mathbf{z}).$$

The definition of positive semi-definite here is equivalent to the one we gave earlier.

- **Reproducing Kernel Hilbert Space (RKHS) Perspective:** From the perspective of RKHS theory, Mercer's conditions ensure that the kernel function defines a valid inner product space, allowing the construction of feature maps and the characterization of function spaces.

Overall, Mercer's conditions serve as the theoretical foundation for the design and analysis of kernel methods, playing a central role in the development of machine learning algorithms based on kernel functions.

2.3 Kernel Principal Component Analysis

PCA is designed to model linear variabilities in high-dimensional data. However, many high dimensional data sets have a nonlinear nature. In these cases the high-dimensional data lie on or near a nonlinear manifold (not a linear subspace) and therefore PCA can not model the variability of the data correctly. One of the algorithms designed to address the problem of nonlinear dimensionality reduction is Kernel PCA. In Kernel PCA, through the use of kernels, Principal components can be computed efficiently in high-dimensional feature spaces that are related to the input space by some nonlinear mapping. Kernel PCA finds Principal components which are nonlinearly related to the input space by performing PCA in the space produced by the nonlinear mapping, where the low-dimensional latent structure is, hopefully, easier to discover. Consider a feature space $\mathcal{H}$ such that:

$$\begin{array}{ll} \phi : x & \longrightarrow \mathcal{H} \\ \mathrm{x} & \longmapsto \ \phi(x) \end{array}$$

2.3.1 Mathematical Formulation

Suppose $\sum_i^t \Phi(x_i) = 0$ (we will return to this point below, and show how this condition can be satisfied in a Hilbert space). This allows us to formulate the kernel PCA objective as follows:

$$\min \sum_i^t \left\| \Phi(x_i) - U_q U_q^T \Phi(x_i) \right\|$$

By the same argument used for PCA, the solution can be found by SVD:

$$\Phi(X) = U \Sigma V^T$$

where U contains the eigenvectors of $\Phi(X)\Phi(X)^T$. Note that if $\Phi(X)$ is $\mathbf{n} \times t$ and the dimensionality of the feature space $\mathbf{n}$ is large, then U is $\mathbf{n} \times \mathbf{n}$ which will make PCA impractical.

To reduce the dependence on $\mathbf{n}$, first assume that we have a kernel $K(\cdot, \cdot)$ that allows us to compute $K(x, y) = \Phi(x)^\top \Phi(y)$. Given such a function, we can then compute the matrix $\Phi(X)^\top \Phi(X) = K$ efficiently, without computing $\Phi(X)$ explicitly. Crucially, K is $t \times t$ here and does not depend on $\mathbf{n}$. Therefore it can be computed in a run time that depends only on t. Also, note that PCA can be formulated entirely in terms of inner products between data points. Replacing inner products by kernel function K, which is in fact equivalent to the inner product of a Hilbert space yields to the Kernel PCA algorithm.

2.3.2 Centering

In the derivation of the kernel PCA we assumed that $\Phi(X)$ has zero mean. The following normalization of the kernel satisfies this condition.

$$\tilde{K}(x, y) = K(x, y) - E_x[K(x, y)] - E_y[K(x, y)] + E_x\left[E_y[K(x, y)]\right]$$

In order to prove that, define:

$$\tilde{\Phi}(X) = \Phi(X) - E_x[\Phi(X)]$$

Finally, the corresponding kernel is:

$$\tilde{K}(x, y) = \tilde{\Phi}(x)\Phi(y)$$

This expands as follows:

$$\tilde{K}(x, y) = \left(\Phi(x) - E_x[\Phi(x)]\right).\left(\Phi(y) - E_y[\Phi(y)]\right)$$

$$= K(x, y) - E_x[K(x, y)] - E_v[K(x, y)] + E_x\left[E_y[K(x, y)]\right]$$

To perform Kernel PCA, one needs to replace all inner products $x^T y$ by $\tilde{K}(x, y)$ in PCA Algorithm . Note that V is the eigenvectors of $K(X, X)$ corresponding to the top d eigenvalues, and Σ is diagonal matrix of square roots of the top d eigenvalues.

Unfortunately Kernel PCA does not inherit all the strength of PCA. More specifically reconstruction of training and test data points is not a trivial practice in Kernel PCA. In Ordinary PCA data can be reconstructed in feature space easily. However finding the corresponding pattern x in KPCA is difficult and sometimes even impossible.

2.3.3 Algorithm

Step-1 Recover basis: Calculate $K(X, X)$ and let V = eigenvectors of $K(X, X)$ corresponding to the top d eigenvalues. Let Σ = diagonal matrix of square roots of the top d eigenvalues.

Step-2 Encode training data: $Y = U^\top X = \Sigma V^\top$ where Y is a $d \times t$ matrix of encodings of the original data.

Step-3 Reconstruct training data: $\hat{X} = UY = U\Sigma V^\top = XV\Sigma^{-1}\Sigma V^\top = XVV^\top$.

Step-4 Encode test example: $y = U^\top x = \Sigma^{-1}V^\top X^\top x = \Sigma^{-1}V^\top X^\top x$ where y is a d dimensional encoding of x.

Step-5 Reconstruct test example: $\hat{x} = Uy = UU^\top x = XV\Sigma^{-2}V^\top X^\top x = XV\Sigma^{-2}V^\top X^\top x$.

2.4 De-noising Images using KPCA

To utilize Kernel Principal Component Analysis (KPCA) for denoising images, follow these steps:

1. **Data Preprocessing:**
   - Convert the image data into a suitable format, such as a matrix representation where each pixel corresponds to a matrix element.
   - Normalize the pixel values if necessary to ensure numerical stability.
2. **Kernel Selection:**
   - Choose an appropriate kernel function for KPCA. The choice of kernel depends on the characteristics of the data and the denoising task.
   - Common choices include Gaussian (RBF), polynomial, and Morlet kernels. The Morlet kernel, in particular, is known for its effectiveness in capturing complex structures in images.
3. **Kernel Matrix Calculation:**
   - Compute the kernel matrix K using the chosen kernel function applied to the image data. The kernel matrix represents pairwise similarities between data points in the feature space induced by the kernel.
4. **Centering the Kernel Matrix:**
   - Center the kernel matrix K to make the data zero-mean in the feature space. This step involves subtracting the mean of each row/column and adding the overall mean to the diagonal elements.
5. **Eigenvalue Decomposition:**
   - Perform eigenvalue decomposition on the centered kernel matrix K to obtain the eigenvectors and eigenvalues.

- Sort the eigenvectors in descending order of eigenvalues to prioritize the most significant components.

6. **Feature Projection:**
   - Select the top m eigenvectors corresponding to the largest eigenvalues to form the projection matrix W.
   - Project the data onto the subspace spanned by these eigenvectors by multiplying the centered kernel matrix K with W.
7. **Denoising:**
   - Reconstruct the denoised image by computing the dot product of the projected data with the transpose of the projection matrix W^T.
   - Optionally, apply additional post-processing techniques such as thresholding or filtering to enhance the denoising results.

By following these steps, one can effectively denoise images using Kernel Principal Component Analysis (KPCA), leveraging its ability to capture nonlinear structures and preserve important image features. Adjustments to the kernel function and parameter tuning may be necessary to achieve optimal denoising performance for different types of images and noise characteristics.

2.5 Evaluation Metrics

2.5.1 Peak Signal-to-Noise Ratio (PSNR)

The Peak Signal-to-Noise Ratio (PSNR) is a widely used metric to measure the quality of a denoised image compared to the original image. It is defined as:

$$\text{PSNR} = 10 \cdot \log_{10} \left(\frac{\text{MAX}^2}{\text{MSE}} \right)$$

where:

- MAX is the maximum possible pixel value of the image (e.g., 255 for an 8-bit image),
- MSE is the mean squared error between the original and denoised images.

A higher PSNR value indicates better denoising performance, with a maximum value of ∞ when the images are identical.

2.5.2 Structural Similarity Index (SSIM)

The Structural Similarity Index (SSIM) is another metric commonly used to assess image quality. It measures the similarity between the original and denoised images based on luminance, contrast, and structure, taking into account local variations in pixel values. SSIM is defined as:

$$\text{SSIM}(x, y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)}$$

where:

- x and y are the original and denoised images, respectively,
- μ_x and μ_y are the mean intensities of x and y,
- σ_x^2 and σ_y^2 are the variances of x and y,
- σ_{xy} is the covariance between x and y,
- c_1 and c_2 are constants to stabilize the division with weak denominator.

SSIM values range from -1 to 1, with 1 indicating perfect similarity between the images.

3 Proposed methodology

3.1 The Morlet Wavelet Kernel

Definition 1 *[3]The Morlet mother wavelet function is* $\psi(x) = \cos(5x)\exp\left(-x^2/2\right)$. *The Mercer kernel using this Morlet mother wavelet function is*

$$k(\mathbf{x}, \mathbf{y}) = \prod_{i=1}^{d} \cos\left(\frac{5\left(x_i - y_i\right)}{a}\right) \exp\left(-\frac{\left(x_i - y_i\right)^2}{2a^2}\right).$$

The proof that this kernel is a Mercer kernel and the investigation of the performance of this kernel in KPCA are needed for using this type of wavelet kernel.

3.2 Verifying the Mercer Conditions for Morlet Kernel

Proposition 1 *The Morlet Wavelet Kernel satisfies the Mercer Conditions discussed in section 2.2.2.*

Proof.
By definition, K(x,y) = K(y,x) Hence, K(x,y) is symmetric. Also,

$$
\begin{aligned}
&\sum_{i=1}^{n}\sum_{j=1}^{n} K(x_i, x_j) a_i a_j \\
&= \sum_{i=1}^{n}\sum_{j=1}^{n}\prod_{k=1}^{d} \cos\left(\frac{5\left(x_i^k - x_j^k\right)}{a}\right) \exp\left(-\frac{\left(x_i^k - x_j^k\right)^2}{2a^2}\right) a_i a_j \\
&\quad (\text{Let } a = 1) \\
&= \sum_{i=1}^{n}\sum_{j=1}^{n}\prod_{k=1}^{d} \cos\left(5\left(x_i^k - x_j^k\right)\right) \exp\left(-\frac{\left(x_i^k - x_j^k\right)^2}{2}\right) a_i a_j \\
&= \sum_{i=1}^{n}\sum_{j=1}^{n}\prod_{k=1}^{d} \left[\cos\left(5x_i^k\right)\cos\left(5x_j^k\right) + \sin\left(5x_i^k\right)\sin\left(5x_j^k\right)\right] \\
&\quad \times \exp\left(-\frac{\left(x_i^k - x_j^k\right)^2}{2}\right) a_i a_j \\
&= \sum_{i=1}^{n}\sum_{j=1}^{n}\prod_{k=1}^{d} \cos\left(5x_i^k\right)\cos\left(5x_j^k\right) \\
&\quad \times \exp\left(-\frac{{x_i^k}^2}{2}\right) \exp\left(-\frac{{x_j^k}^2}{2}\right) \exp\left(x_i^k x_j^k\right) a_i a_j \\
&\quad + \sum_{i=1}^{n}\sum_{j=1}^{n}\prod_{k=1}^{d} \sin\left(5x_i^k\right)\sin\left(5x_j^k\right) \\
&\quad \times \exp\left(-\frac{{x_i^k}^2}{2}\right) \exp\left(-\frac{{x_j^k}^2}{2}\right) \exp\left(x_i^k x_j^k\right) a_i a_j \\
&= \sum_{i=1}^{n}\sum_{j=1}^{n}\prod_{k=1}^{d} \cos\left(5x_i^k\right)\cos\left(5x_j^k\right) \\
&\quad \times \exp\left(-\frac{{x_i^k}^2}{2}\right) \exp\left(-\frac{{x_j^k}^2}{2}\right) \prod_{k=1}^{d} \exp\left(x_i^k x_j^k\right) a_i a_j \\
&\quad + \sum_{i=1}^{n}\sum_{j=1}^{n}\prod_{k=1}^{d} \sin\left(5x_i^k\right)\sin\left(5x_j^k\right) \\
&\quad \times \exp\left(-\frac{{x_i^k}^2}{2}\right) \exp\left(-\frac{{x_j^k}^2}{2}\right) \prod_{k=1}^{d} \exp\left(x_i^k x_j^k\right) a_i a_j
\end{aligned}
$$

Now, let c_{ij} be such that for all $i, j = 1(1)n$,

$$\prod_{k=1}^{d} \exp\left(x_i^k x_j^k\right) \geq c_{ij} > 0$$

This implies $\prod_{k=1}^{d} \exp\left(x_i^k x_j^k\right) \geq \min_{i,j} c_{ij} = c \geq 0$.

$$\implies \sum_{i=1}^{n}\sum_{j=1}^{n}\prod_{k=1}^{d} \cos\left(5x_i^k\right)\cos\left(5x_j^k\right)\exp\left(-\frac{{x_i^k}^2}{2}\right)\exp\left(-\frac{{x_j^k}^2}{2}\right)\prod_{k=1}^{d}\exp\left(x_i^k x_j^k\right)a_i a_j$$

$$\geq \sum_{i=1}^{n}\sum_{j=1}^{n}\prod_{k=1}^{d} \cos\left(5x_i^k\right)\cos\left(5x_j^k\right)\exp\left(-\frac{{x_i^k}^2}{2}\right)\exp\left(-\frac{{x_j^k}^2}{2}\right)ca_i a_j$$

$$= \left(\sum_{i=1}^{n}\prod_{k=1}^{d} \cos\left(5x_i^k\right)\exp\left(-\frac{{x_i^k}^2}{2}\right)\sqrt{c}a_i\right)^2$$

And similarly,

$$\implies \sum_{i=1}^{n}\sum_{j=1}^{n}\prod_{k=1}^{d} \sin\left(5x_i^k\right)\sin\left(5x_j^k\right)\exp\left(-\frac{{x_i^k}^2}{2}\right)\exp\left(-\frac{{x_j^k}^2}{2}\right)\prod_{k=1}^{d}\exp\left(x_i^k x_j^k\right)a_i a_j$$

$$\geq \left(\sum_{i=1}^{n}\prod_{k=1}^{d} \sin\left(5x_i^k\right)\exp\left(-\frac{{x_i^k}^2}{2}\right)\sqrt{c}a_i\right)^2$$

Therefore,

$$\sum_{i=1}^{n}\sum_{j=1}^{n} K(x_i, x_j)a_i a_j$$

$$\geq \left(\sum_{i=1}^{n}\prod_{k=1}^{d} \cos\left(5x_i^k\right)\exp\left(-\frac{{x_i^k}^2}{2}\right)\sqrt{c}a_i\right)^2$$

$$+ \left(\sum_{i=1}^{n}\prod_{k=1}^{d} \sin\left(5x_i^k\right)\exp\left(-\frac{{x_i^k}^2}{2}\right)\sqrt{c}a_i\right)^2 > 0$$

Therefore, $K(x, y)$ is a positive definite kernel.

3.3 Applying The Morlet Kernel to denoise images through KPCA

The Morlet wavelet kernel, combined with Kernel Principal Component Analysis (KPCA), is applied to denoise SIDD (Spectral Image Denoising Dataset) images. This method effectively separates noise from signal by identifying principal components in a high-dimensional feature space. By retaining only dominant components, the denoised images are reconstructed, preserving structural information while suppressing noise artifacts. The same procedure as in section 2.4 was followed where we implemented our proposed kernel.

4 Experimental result

4.1 Data

4.1.1 Description of Dataset

The Smartphone Image Denoising Dataset (SIDD) [2] consists of a diverse collection of high-quality images captured using smartphone cameras across various scenarios and environments. These scenarios include low-light conditions, indoor settings, outdoor scenes, and different levels of noise. The dataset provides a comprehensive representation of real-world challenges encountered in smartphone photography.

Each image in the SIDD dataset is paired with its corresponding ground truth clean version, allowing for accurate evaluation of denoising algorithms. This ground truth data enables researchers to quantitatively assess the performance of denoising techniques by comparing the output of these algorithms with the pristine images.

The images in the SIDD dataset exhibit a wide range of noise characteristics, including Gaussian noise, Poisson noise, and mixed noise types commonly encountered in smartphone photography. This diversity ensures that denoising algorithms evaluated on the SIDD dataset are robust and capable of handling various noise profiles encountered in real-world scenarios.

Moreover, the SIDD dataset provides annotations and metadata associated with each image, including information about camera settings, exposure parameters, and scene characteristics. These annotations offer valuable insights into the imaging conditions and aid researchers in understanding the relationship between image quality and environmental factors.

Overall, the SIDD dataset serves as a valuable resource for researchers and developers working on image denoising algorithms tailored for smartphone photography. Its comprehensive data representation, including high-quality images, ground truth annotations, and metadata, facilitates the development and evaluation of robust denoising techniques capable of enhancing image quality in real-world smartphone applications.

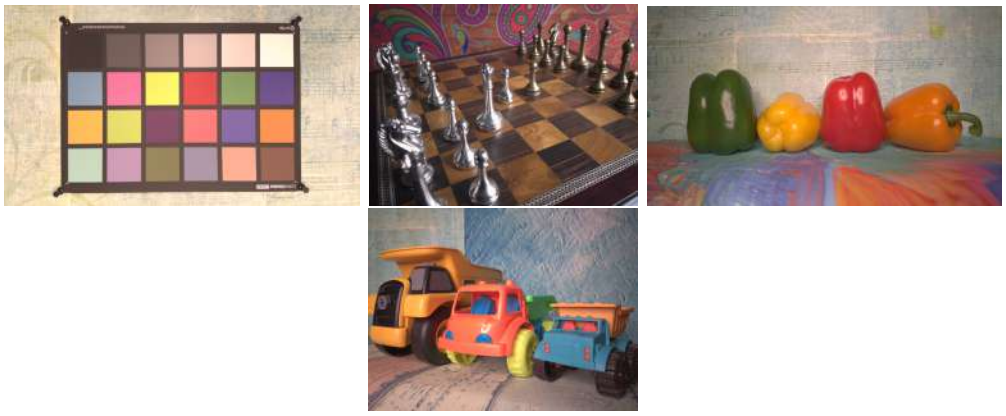

Figure 1: True SIDD Images

Figure 2: Noisy SIDD Images

4.1.2 Sample Test Images

For testing our proposed methodology, 4 images from the SIDD dataset was considered. Figure 1 contain the true images:

The images were named as makeup, chess, capsicum and toys for ease of future referencing.

Figure 2 contain their noisy counterparts:

It is very difficult to identify the noise with bare eyes. We recommend referring the data repository for a better check. Additionally, we also used a standard image called Lena(right) along with its noisy counterpart(left) to test our method.

Figure 3: Lena Image True vs Noisy

4.2 Testing Method on the Standard Image

Method	MSE	PSNR	SSIM
PCA	41.957	31.660	0.709
KPCA + RBF	29.1701	33.2396	0.8419
KPCA + Morlet	29.2611	33.2261	0.8415
Mean Filter	29.926	33.128	0.7481
Median Filter	28.421	33.352	0.7544

Table 1: Results of denoising on the standard Lena image.

As per the table, it is evident that the denoising done using our proposed method is quite on par with the other methods if not better than most of the methods generally used for denoising the Lena image. Figure 4 showcases is the Lena image obtained by denoising the noisy image with the help of the Kernel PCA implementing our proposed Morlet kernel. Results obtained using other denoising techniques is added to the github page linked in the summary.

4.3 Pre-Processing of SIDD Images

- We have tried to denoise the image using two existing dimensionality reduction models, PCA and KPCA-RBF.
- As the resolution of the images are very large (5328,3000), we have used "Patch-Based Image Denoising" process. We have taken multiple same size patches of images then trained the model on those patches and using that we try to regenerate the whole image.

Figure 4: Lena Image denoised using KPCA(Morlet Kernel)

- We have read the image in grey-level then denoised it. After that we recolored the image using the color filter of the noisy image.
- We generated the image one by one.
- First we have applied the PCA model which generates the gray level image as following.
- Next we have applied the Kernel PCA model which generates the image as follows.

4.4 Results and Comparison

For pictorial demonstration in this report we highlight only the makeup box image denoised using some standard techniques

Figure 5: Mean Filter

Figure 6: Median Filter

Figure 7: Kernel PCA with RBF Kernel

With respect to this image the table of comparison with the most commonly used methods is:

Method	**MSE**	**PSNR**	**SSIM**
PCA	50.870	30.89	0.5538
KPCA + RBF	40.699	31.862	0.5596
KPCA + Morlet	36.771	32.307	0.5735
Mean Filter	74.830	29.21	0.6022
Median Filter	77.022	29.0926	0.5854

Table 2: Results of denoising on the makeup box image

Figure 8: Kernel PCA with Morlet Wavelet Kernel

4.4.1 Comparison on a broader scale

For the subset of the data selected, the average comparison table is as follows:

Method	Average PSNR	Average SSIM
Morlet KPCA	31.949	0.7206
RBF KPCA	31.567	0.706
Ordinary PCA	30.535	0.6979
Mean Filter	31.092	0.7374
Median Filter	30.909	0.72895
CGNet*	40.39	0.964
CBDNet*	30.78	0.801

Table 3: Average PSNR and SSIM values across test images for various denoising methods.

* Point to be noted here is that the CGNet and CBDNet averages are over the entire dataset however our results are on a smaller scale. Nevertheless, the images we selected capture quite a good amount of variation to support our comparison.

Our initial implementation of the Morlet KPCA for image denoising has shown promising results, outperforming several primitive techniques such as Mean Filter, Median Filter, PCA, and even KPCA with an RBF Kernel. This underscores the potential effectiveness of leveraging wavelet kernels for image processing tasks. However, it's essential to acknowledge that CGNet and CBDNet represent state-of-the-art methods, indicating there's still room for improvement.

By integrating the Morlet kernel into a more advanced and sophisticated algorithm, we can expect even better denoising performance. Advanced algorithms can exploit the unique properties of the Morlet kernel more effectively, leveraging its ability to capture both time and frequency domain information simultaneously. This integration could lead to enhanced noise reduction, better preservation of image details, and overall superior denoising results compared to our current implementation.

4.5 Time Complexity and Interpretation

Time complexity poses a significant challenge in our approach, particularly due to the multiplicative nature of the Morlet wavelet kernel, which involves multiplying exponential and trigonometric components across all dimensions of the original space. This characteristic puts it at a competitive disadvantage compared to kernels like the Gaussian RBF kernel.

On average, the image denoising process using KPCA with the Gaussian RBF kernel took approximately 3 minutes per image, while using the Morlet wavelet kernel extended this time to 8 minutes per image. In contrast, other techniques, except for CGNet and CBDNet, which took around 5 minutes each as reported by [1], completed the task in less than 2 minutes per image.

Our future objective is to enhance the implementation of this kernel by exploring optimization techniques. By optimizing the computation process, we aim to reduce the time complexity associated with the Morlet wavelet kernel, thereby improving its efficiency for image denoising tasks.

5 Summary

The project focuses on implementing image denoising techniques using kernel principal component analysis (KPCA) with a Morlet wavelet kernel. By leveraging this approach, the team aims to achieve superior denoising results compared to traditional methods like mean and median filters. Despite facing challenges related to time complexity, particularly with the Morlet wavelet kernel's multiplicative form, the project has shown promising outcomes, outperforming other primitive techniques like PCA and KPCA with the radial basis function (RBF) kernel. The team plans to further optimize the implementation of the Morlet wavelet kernel to improve efficiency. For results on more images and access to the project code, visit the GitHub repository: `https://github.com/biswajit-github-2022/RKMVERI_2ND/tree/5ba704868b4a430d2e9a69bfc7317c92826856cd/class/ML/ML_project/work`.

References

[1] Image denoising on sidd. `https://paperswithcode.com/sota/image-denoising-on-sidd`, 2018.

[2] Michael S. Brown Abdelrahman Abdelhamed, Stephen Lin. A high-quality denoising dataset for smartphone cameras. IEEE Explore, 2018.

[3] Sridhar Krishnan Shengkun Xie, Anna T. Lawniczak and Pietro Lio. Wavelet kernel principal component analysis in noisy multiscale data classification. *ISRN Computational Mathematics*, 2012.