Institut für Visualisierung und Interaktive Systeme

Universität Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Bachelorarbeit Nr. 277

Denoising with Patch-based Principal Component Analysis

Fang Bao

Studiengang:	Informatik
Prüfer/in:	Prof. Dr. Andrés Bruhn
Betreuer/in:	Prof. Dr. Andrés Bruhn
Beginn am:	November 20, 2015
Beendet am:	Mai 20, 2016
CR-Nummer:	I.4.4

Abstract

One important task in image processing is noise reduction, which requires to recover image information by removing noise without loss of local structures. In recent decades patch-based denoising techniques proved to have a better performance than pixel-based ones, since a spatial neighbourhood can represent high correlations between nearby pixels and improve the results of similarity measurements. This bachelor thesis deals with denoising strategies with patch-based principal component analysis. The main focus lies on learning a new basis on which the representation of an image has the best denoising effect. The first attempt is to perform principal component analysis on a global scale, which obtains a basis that reflects the major variance of an image. The second attempt is to learn bases respectively over patches in a local window, so that more image details can be preserved. In addition, local pixel grouping is introduced to find similar patches in a local window. Due to the importance of sufficient samples in the principal component analysis transform, the third attempt is to search for more similar patches in the whole image by using a vantage point tree for space partitioning. In the part of implementation, parameter selection and time complexity are discussed. The denoising performance of different approaches is evaluated in terms of both PSNR value and visual quality.

Kurzfassung

Eine der wichtigen Aufgaben in der Bildverarbeitung ist die Entrauschung, die erfordert Bildinformationen ohne Verlust lokaler Strukturen wiederzuherstellen. In den letzten Jahrzehnten hat es sich herausgestellt, dass Patch-basierte Verfahren eine bessere Leistung bei der Bildentrauschung haben als Pixel-basierte Verfahren. Der Grund liegt darin, dass eine räumliche Nachbarschaft die Korrelationen zwischen benachbarten Pixels repräsentiert und die Ergebnisse des Ähnlichkeitsmaß verbessern. In dieser Bachelorarbeit geht es um Entrauschungsstrategien mit der Patch-basierten Hauptkomponentenanalyse. Der Schwerpunkt liegt im Lernen einer neuen Basis, auf welcher die Representation eines Bildes den besten Entrauschungseffekt hat. Der erste Versuch ist, die Hauptkomponentenanalyse global durchzuführen und eine Basis zu erhalten, welche die Hauptvarianz eines Bildes reflektiert. Der zweite Versuch ist, mehrere Basen jeweils über Patches in einem lokalen Fenster zu lernen, um mehr Details zu behalten. Außerdem wird Local Pixel Grouping benutzt um ähnliche Patches in einem lokalen Fenster zu suchen. Die Hauptkomponentenanalyse ist wichtig dass genügend Samples vorhanden sind, daher werden im dritten Versuch weitere ähnliche Patches innerhalb des ganzen Bildes mithilfe von einem Vantage Point Baum gesucht. Im Teil der Implementierung wird über die Auswahl der Parameter und die Zeitkomplexität diskutiert. Die Entrauschungsleistung von unterschiedlichen Verfahren wird nach dem PSNR-Wert und der visuellen Qualität evaluiert.

Contents

1. Introduction 9
   1.1. Motivation . . . 9
   1.2. Solutions . . . 10
   1.3. Outline . . . 10

2. Related Work 11
   2.1. Smoothing Spatial Filtering . . . 11
   2.2. Patch-based Methods . . . 12
   2.3. Denosing with PCA . . . 13
   2.4. Similarity Search for High-dimensional Data . . . 14

3. Methodology 17
   3.1. Patch-based PCA . . . 17
      3.1.1. Construction of Covariance Matrix . . . 18
      3.1.2. Learning Basis via Eigen Decomposition . . . 19
      3.1.3. Selection of Principal Components . . . 21
   3.2. Local Adaption . . . 26
      3.2.1. Local PCA . . . 26
      3.2.2. Local Pixel Grouping . . . 26
   3.3. Non-local Improvement . . . 30
      3.3.1. Similarity Search . . . 31
      3.3.2. Construction of a VP-tree . . . 31
      3.3.3. Searching through a VP-tree . . . 34

4. Implementation 39
   4.1. Framework . . . 39
   4.2. Parameter Selection . . . 40
   4.3. Experimental Results . . . 41
      4.3.1. Denoising Performance . . . 41
      4.3.2. Time Complexity . . . 46

5. Conclusion 47

A. Appendix 49
A.1. Notations 49

Bibliography 51

List of Figures

3.1. Patch Representation 18
3.2. Principal Axes 22
3.3. Advantage of Hard-thresholding 24
3.4. Thresholds for Shrinkage 25
3.5. Denoising Effect of Rare Patterns. 27
3.6. Influence of Sliding Steps 28
3.7. Non-local Similar Patches 30
3.8. The Structure of A VP-tree 32
3.9. The Case of "Pruning" Left Child 35
3.10. The Case of "Pruning" Right Child 36

4.1. Framework of the Implementation 39
4.2. Denoising Effect of *lena* 43
4.3. Denoising Effect of *barbara* 44
4.4. Denoising Effect of *einstein* 45

List of Tables

4.1. Parameter Selection 40
4.2. Performance Comparison 42
4.3. Time Complexity 46

List of Algorithms

3.1. LPG for A Single Patch 29
3.2. LPG-PCA 30
3.3. Construction of A VP-tree 33
3.4. Searching through A VP-tree 37

1. Introduction

Digital images are usually corrupted by noise in the process of acquisition and transmission. Image noise manifests itself as a variation of color and brightness of pixels. It could degrade image quality and disturb further analysis, such as edge detection, image segmentation, pattern recognition, etc. Therefore, denoising is a significant step of image pre-processing.

An image can be regarded as a signal in a two-dimensional space spanned by an orthogonal basis. Frequencies of the signal are represented as the degree of variation in the value of pixels. In general, information of an image often appears in low frequency components since nearby pixels have close values due to high correlation between them. On the contrary, as a random variation of pixel values image noise appears always in high frequency components. However, denoising effects of ordinary low-pass filters are unsatisfactory in most situations, since image edges and details belonging also to high frequency components are removed as well. Thus it can be seen that an ideal denoising effect requires an optimal estimation of noiseless images without causing artifacts or image blurring.

1.1. Motivation

Principal component analysis (PCA) is a common statistical technique for data analysis with reduction of dimensionality. It preserves most of the data variance with the least number of dimensions. By performing PCA transformation on an image, one can learn a new orthogonal basis and select a subset of its basis vectors that represents major variance inside this image. In that way, noise can be separated from the data, as the variance caused by noise usually tends to be small.

Patches are image fragments composed of pixels within a range. They can reflect image features that single pixels are unable to present. For PCA, patch samples are usually overlapping. The redundancy between them is used to remove noise. Therefore, patch-based PCA is a model that is applied to find a basis over patch samples to represent principal components of an image.

This bachelor thesis aims to investigate the factors that influence the denoising effect of the methods with patch-based PCA. They include the scale on which PCA is performed, the degree of similarity between patch samples and the adjustment of various parameters. The patch-based PCA methods are implemented with the consideration of these factors and evaluated with respect to denoising performance and time complexity.

1.2. Solutions

First of all, we perform patch-based PCA in a global scale to learn an orthogonal basis for the whole image. After that, a local adaption is made by performing several times patch-based PCA on local windows respectively. Besides that, we need to search for similar patches to preserve more image details. The search process is first carried out in local regions and then extended to the whole image. In local regions we use Local Pixel Grouping to find similar patches which have the minimal Euclidean distance to a reference patch. For a further non-local search, a Vantage Point Tree is applied for space partitioning. At last, a proper selection of parameters is necessary, which refers to patch size, thresholds for PCA, size of search windows, number of required similar patches etc.

1.3. Outline

This thesis consists of five chapters as follows:

Chapter 1 – Introduction: gives a brief introduction about denoising methods with patch-based PCA and displays the goal and solutions of this thesis.

Chapter 2 – Related Work: provides a literature review of various approaches for image denoising with an emphasis on the combination of patch-based methods and sparsifying transform.

Chapter 3 – Methodology: explains the process of learning a basis by patch-based PCA, proceeds with a local adaption on different image regions and develops a non-local improvement by searching for similar patches in the whole image.

Chapter 4 – Implementation: shows a realization of the denoising methods mentioned above and evaluates the experimental results with respect to denoising performance and computational cost.

Chapter 5 – Conclusion: concludes the contributions of this thesis and discuss extension possibilities for future work.

2. Related Work

This chapter reviews previous research on image denoising techniques related to this thesis. They can be divided into four sections. In the first section, traditional smoothing spatial filters are displayed, which includes mean filter, Gaussian filter, median filter and bilateral filter. In the second section, two representative patch-based methods are reviewed: non-local means and BM3D. In the third part we focus on denoising approaches based on PCA and their different ways for local adaption. At last, we demonstrate clustering and matching techniques for similarity search in a large space.

2.1. Smoothing Spatial Filtering

A spatial filter is an image operation where each pixel value is changed by a function of the intensity value of pixels in a neighbourhood[GW08]. Theoretically, the kernel shape can be arbitrary. But in general, square kernels with odd dimensions are exploited in most cases, as the pixel to be processed can be located at the centre of the kernel. The coefficients in the kernel depends how the the effect of the spatial filter looks like. The filters for image smoothing reduce the amount of intensity variation between a pixel and its neighbours. Hence, all coefficients of low-pass filtering are non-negative.

A linear smoothing filter can be regarded as the convolution product of the signal and a mirrored weighted function[Bru15b]. The simplest linear smoothing filter is mean filter, the idea of which is to replace the value of a pixel with the mean value of its nearby pixels, including itself[GW08]. The smoothing effect can be enhanced by larger kernels. Mean filters distribute same weights to each nearby pixel without consideration of the distance between this pixel and the central pixel. By contrast, the weights of linear Gaussian filters accord with normal distribution. Pixels closer to the centre have a larger contribution to the output value of the pixel to be denoised. The variance of normal distribution determines how wide a Gaussian filter is and how high the degree of smoothing is. Gaussian filters have a better performance of removing noise than mean filters, since the influence of pixels near the kernel centre is considered more significant than the distant ones. However, both mean filters and Gaussian filters result in a loss of sharp edges and image details[GG13].

In Non-Linear spatial filters, median filter replaces each pixel value with the median of its nearby pixels, including itself. It is quite effective for removing salt-and-pepper noise, but not for high level Gaussian noise. Bilateral filter is proposed by Aurich and Weule in their work on non-linear Gaussian filter. It was later used by Smith and Brady in their SUSAN framework. Tomasi and Manduchi gave it the current name. Bilateral filter can smooth images while preserving edges[PKTD07]. For computing weights it considers not only the distance between central pixel and its neighbours, but also their intensity difference. That is, it combines both spatial filtering and range filtering to evaluate the similarity between a nearby pixel and the central pixel. Only nearby and similar pixels achieve a high weight. The drawback of bilateral filter is creation of a staircase effect after denoising, which produces artefact boundaries separating flat regions[BCM06]. Like other smoothing spatial filters mentioned above, it is constrained by pixel-based operations that ignore the spatial context such as texture, shape and pattern.

2.2. Patch-based Methods

The concept of patch-based analysis originates from texture synthesis[EL99] and image inpainting[CPT04]. They use self-similarity between patches to find candidates for target regions. Buades et al introduces this patched-based strategy into image denoising and proposed non-local means(NL means) which can be considered as an extenstion of bilateral filtering to image patches[KB06]. In NL-means, the estimated value for a pixel is computed as a weighted average of all the pixels in the image[BCM05]. The weighted function of a pixel is obtained as a decreasing function of the Euclidean distance between the patch centred at this pixel and the patch centred at the pixel to be denoised. The weight distribution computed by NL-means shows that the pixels with only a similar intensity value but without a similar neighbourhoods are also unable to acquire a high weight. A parameter-free algorithm for NL-means is proposed in [KB06], which jointly chooses optimal values for patch-based weights and variable window sizes with an iterative procedure. Besides that the spatial adaptivity also has a strong robustness.

Dabov et al. proposed a collaborative filtering scheme by patch matching and sparse 3D transform (BM3D)[DFKE07]. They search for similar patches by block matching and group them into 3D data arrays. The reference patch is considered as the centroid of the group. A group is characterized by both intra-patch correlation and inter-patch correlation[DFKE07]. The next three successive steps are 3D transformation of a group, shrinkage of the transform spectrum and inverse 3D transformation[DFKE07]. For basic estimate hard-thresholding is used as shrinkage, and for final estimate Wiener filtering is applied as shrinkage. At last, the final estimate is computed by aggregating all of the obtained local estimates using a weighted average[DFKE07]. BM3D has achieved a

remarkable denoising effect and is widely recognized as the state-of-the-art denoising technique[Bae12]. An extension to colour images is made with a grouping constraint on both chrominance channels, which yields also a consistent improvement of PSNR value.

2.3. Denosing with PCA

In [Tas08], image denoising with PCA is proposed to reduce the dimensionality of patch vectors and accelerate the computations of the NL-means. This method is based on the assumption that patch vectors exist on a lower-dimensional subspace rather than the full space. The result shows that both the accuracy and computational cost of the non-local means image denoising algorithm can be improved by computing similar patches after a PCA projection[Tas08].

Deledalle et al. proposed denoising methods with patch-based PCA in a systematic and comprehensive way. They perform PCA respectively in global, local and hierarchical scale. For global PCA, different shrinkage methods are compared and Hard Thresholding proved to be better for the case of high threshold ratio[DSD+11]. For local PCA, they use a step for the sliding window to highly reduce the computation time with only a slight loss in denoising performance. The step is chosen as half of the sliding window size, which makes the searching zones still overlap. For hierarchical PCA, a geometric partitioning of patch samples is carried out and principal axes for each partition on smaller spaces are re-estimated[DSD+11]. The comparison of the three patch-based PCA displays that local PCA has the best performance and hierarchical PCA saves significantly more time than local PCA.

In terms of learning the best local basis, Muresan and Parks first partitioned the image into patches. Each patch contains a train region and a denoise region which is included in the train region. Then, they utilized an LMMSE estimator to find the principal component basis that minimizes the LMMSE between training vectors and their projection on the basis. To average out the blocking artefacts between different denoised regions, they added an overlap region[MP03]. Another method for learning a locally adaptive basis is Local Pixel Grouping (LPG) which was proposed by Zhang et al.[ZDZS10]. They used an training block centred on the pixel to be denoised. In this block, they selected and grouped the training samples that are similar to the central patch. After the denoising process of the first stage, they updated the noise level and performed the LPG-PCA once more, in order to remove residual noise. They measured the denoising performane with both PSNR value and SSIM value, since SSIM can better reflect the structure similarity between the target image and the reference image[ZDZS10]. The problem of these two

methods lies in overfitting and computational burden. The selection of similar patches in a local region may result in a limited number of samples[DSD+11].

Based on BM3D, Dabov et al. proposed another denoising method which exploits shape-apdaptive PCA (BM3D-SAPCA)[DFKE09]. A pixel obtains its adaptive shape neighbours by using 8-directional LPA-ICI. Similar neighbourhoods are found with block-matching. They also use a threshold to ensure that there are sufficient samples for PCA transform. Then, a PCA-basis is obtained over the similar shape-adaptive neighbourhoods. After that, collaborative filtering is performed like in BM3D. This approach is applied in three iterations for a refinement. For the third iteration, the shrinkage is performed by empirical Wiener filtering rather than hard-thresholding because noise has already been attenuated in the estimate images[DFKE09]. Compared with another data-adaptive transform method[EA06], the transform of BM3D-SAPCA has a more powerful ability to sparsely represent the true-image data[DFKE09].

2.4. Similarity Search for High-dimensional Data

Due to the importance of finding similar patches in a PCA transform, some useful techniques for similarity search in high-dimensional space are reviewed. These techniques can be classified into two types: clustering and matching.

k-means[Mac+67] is a simple clustering method that assigns points to the closest of k centroids[KZN08]. Each centroid is computed as the mean value of the points in this class. The squared Euclidean distance is utilized for measuring the distance between two vectors. Note that a centroid has to be updated after a new point is added into its class. However, the accuracy of k-means can not be guaranteed because of the selection of initial centroids. Arthur and Vassilvitskii proposed k-means++[AV07] to augment k-means with a randomized seeding technique. The idea of this method is to select the first centre uniformly at random and then choose each subsequent centre from the remaining data points with probability proportional to its squared distance from its closest existing centre[AV07]. This ensures that the points which are already very close to a centroid have a small possibility to be chosen.

One problem of clustering is that the clusters are disjoint[DFKE07]. In image denoising, this means each image patch can only belong to one cluster. By contrast, matching makes it possible that an image patch can be similar to multiple other patches. For matching, an efficient search for high-dimensional data depends on the search structure for space partitioning. Bentley's kd-tree[Ben75] is one of the earliest but still the most frequently used methods for space partitioning. One constructs a kd-tree by splitting data set along one dimension into two parts, until a termination criterion is satisfied. The split value is

usually chosen to be the median value along the split dimension[KZN08]. For a kd-tree, node divisions are always axis-aligned. With increase of dimensionality, the performance of kd-tree could quickly decrease. Vantage Point Tree (vp-tree) is another metric tree proposed by Yianilos[Yia93]. A vp-tree splits data set using the absolute distance from a single centroid[KZN08], called vantage point. Each vantage point is the centre of a hypersphere. The data stored in the left child node are inside this hypersphere and the data stored in the right child node are outside the hypersphere. With this structure a search process can quickly be shorter if the requirements for "pruning" are satisfied.[1] An extension to dynamic vp-tree is made in [FCCM00]. The adjustments focus on two aspects: One is the multiple-nearest-neighbour search, the other is the redistribution of nodes in a vp-tree after insertion and deletion.

[1] The details about "pruning" is discussed in Sect. 3.3.

3. Methodology

This chapter attempts to employ various approaches to improve the performance of denoising methods with patch-based PCA. The approaches vary in the scale of performing PCA and degree of sample similarity. In the first section, PCA is performed on the whole image for the purpose of learning a global basis. In the second section, PCA is performed respectively on different local windows such that each image region has their own basis. In addition, similar patches in this window are selected and used as samples for PCA. Due to a limiting number of samples, non-local improvements are made in the third secion. We construct a vp-tree for space partitioning and similarity search.

3.1. Patch-based PCA

Suppose an image $y = f(x)$ is corrupted by a zero-mean additive white Gaussian noise w with constant variance σ^2. The observed noisy image z can be formulated as:

$$z_{i,j} = y_{i,j} + w_{i,j} \quad \text{for} \quad \begin{matrix} i = 1, \ldots, n_1 \\ j = 1, \ldots, n_2 \end{matrix} , \tag{3.1}$$

where n_1 stands for the number of rows and n_2 for the number of columns. In most cases it is hard to separate the noise directly because both y and w are unknown. Thus, more information from z is needed to obtain an estimation of y. Notice that pixel values in an image usually do not appear randomly, as it is only possible to perceive image content if neighbouring pixels are correlated. This correlation can be explored by image patches which are composed of pixels in the neighbourhood. In a noisy image, the correlation is influenced by a random occurrence of noise. We need to restore the image by extracting useful parts from all the components of the correlation. By performing PCA on a noisy image, an orthogonal basis can be learned from patch samples. The basis vectors correspond to the image components that reflect different degrees of data variability. We can pick out principal components to remove image noise.

3.1.1. Construction of Covariance Matrix

Let M be a square patch with $p \times p$ pixels. Thus, there are in total $N = (n_1 - p + 1) \times (n_2 - p + 1)$ patches in the image. Each patch is indexed by its upper left corner. The model of the observed image can be rewritten as follows:

$$z_{u,v} = y_{u,v} + w_{u,v} \quad \text{for} \quad \begin{matrix} u = 1, \ldots, N \\ v = 1, \ldots, M \end{matrix} , \tag{3.2}$$

where $z_{u,v}$ is the value of the v-th pixel in the u-th patch. Fig.3.1 shows how an image is transformed into a $N \times M$ matrix A in which each row vector s_u represents a sample and each column vector f_v represents a feature.

$$s_u = \{z_{(u,\cdot)}\} , \tag{3.3a}$$
$$f_v = \{z_{(\cdot,v)}\} , \tag{3.3b}$$
$$A = (s_1, s_2, \cdots, s_N)^T , \tag{3.3c}$$
$$A = (f_1, f_2, \cdots, f_M). \tag{3.3d}$$

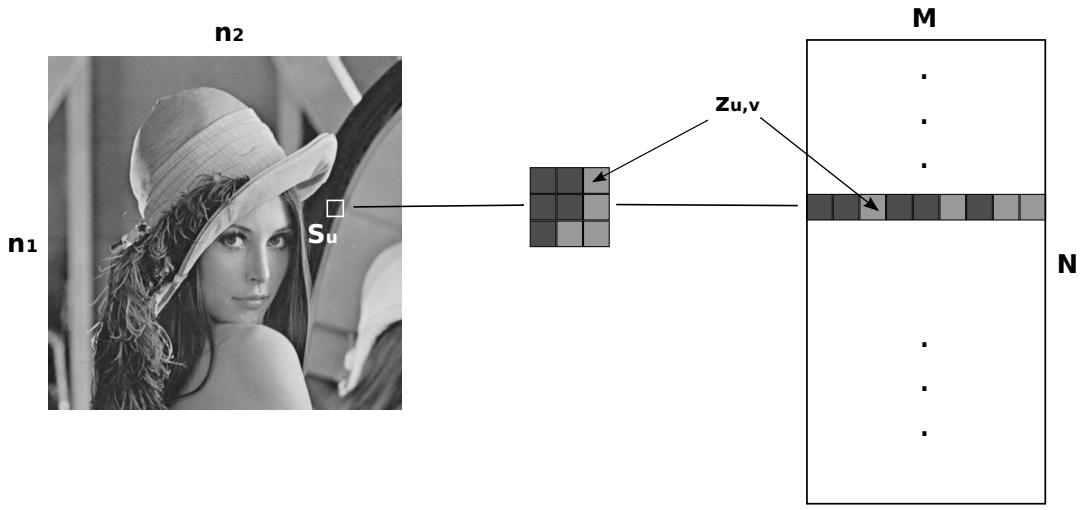


Figure 3.1.: Row-major transform of an image into patch representation.

Before computing the covariance matrix we need to center the data by subtracting the mean vector $E[s_u]$ from each sample s_u, in order to avoid that the first principal component is affected by the mean vector and deviates from the main direction.

$$E[s_u] = \frac{1}{N} \sum_{u=1}^{N} s_u , \tag{3.4a}$$
$$\tilde{s}_u = s_u - E[s_u]. \tag{3.4b}$$

The centred samples form the corresponding matrix $\tilde{A}$, in which the sum of each column is zero.

$$\tilde{A} = (\tilde{s}_1, \tilde{s}_2, \cdots, \tilde{s}_N)^T \tag{3.5a}$$
$$\tilde{A} = (\tilde{f}_1, \tilde{f}_2, \cdots, \tilde{f}_M) \tag{3.5b}$$

The value of pixels in a fixed position to the patch origin can be taken as a feature. The correlation between two arbitrary features can be explored by their covariance. If it is positive, the value of pixels in these positions has the same direction of variation. If it is negative, there is large possibility that their value variates in a reverse direction. If it is zero, then there exists no correlation between these two features. Such correlations in an image is reflected in the following covariance matrix:

$$\begin{aligned}
\Sigma &= \frac{1}{N}\begin{pmatrix} \mathrm{cov}(f_1,f_1) & \mathrm{cov}(f_1,f_2) & \cdots & \mathrm{cov}(f_1,f_M) \\ \mathrm{cov}(f_2,f_1) & \mathrm{cov}(f_2,f_2) & \ddots & \mathrm{cov}(f_2,f_M) \\ \vdots & \ddots & \ddots & \vdots \\ \mathrm{cov}(f_M,f_1) & \mathrm{cov}(f_M,f_2) & \cdots & \mathrm{cov}(f_M,f_M) \end{pmatrix} \\
&= \frac{1}{N}\begin{pmatrix} \tilde{f}_1^2 & \tilde{f}_1\tilde{f}_2 & \cdots & \tilde{f}_1\tilde{f}_M \\ \tilde{f}_1\tilde{f}_2 & \tilde{f}_2^2 & \ddots & \tilde{f}_2\tilde{f}_M \\ \vdots & \ddots & \ddots & \vdots \\ \tilde{f}_1\tilde{f}_M & \tilde{f}_2\tilde{f}_M & \cdots & \tilde{f}_M^2 \end{pmatrix} \\
&= \frac{1}{N}\tilde{A}^T\tilde{A} \ .
\end{aligned} \tag{3.6}$$

As seen here, the covariance matrix is symmetric. The items on the main diagonal are statistical variance of features. They show how large the variety of samples are with respect to a specific feature.

3.1.2. Learning Basis via Eigen Decomposition

So far a mathematical representation of processing an image has been described. The objective of denoising problems is to find a new basis $\{v_1, \ldots, v_M\}$ of the image space. Each vector of the new basis is a linear combination of the original basis vectors. With this new basis an image can be represented with an estimated distinction between information and noise as best as possible. That is to say, the new basis has to satisfy two requirements:

1. Two arbitrary basis vectors are mutually orthogonal, that is, $\forall\, i, j$: $v_i v_j = 0$, if $i \neq j$.

2. If all of the basis vectors are listed from the most significant axis of data variation to the least significant one, each basis vector should reflect the direction of the largest variation of the data among the remaining possible directions.

The first requirement is set to avoid data redundancy. Because in a non-orthogonal basis, the projection of features on one basis vector could be correlated to that on another one, which violates the reduction of dimensionality. The second requirement ensures that with fewer features the information in an image can be preserved as much as possible. This can be expressed mathematically as follows.

We define $d_u = \tilde{s}_u^T v$ to measure how large the projection of the centred data sample s_u on a basis vector v variates. The accumulated variation on v can be computed as the statistical variance of d_u:

$$\begin{aligned}
\sigma_{d_u} &= \sum_{u=1}^{N}(d_u - E[d_u])^2 \\
&= \frac{1}{N}\sum_{u=1}^{N}((\tilde{s}_u^T v)^2 - (E[\tilde{s}_u^T v])^2) \\
&= \frac{1}{N}\sum_{u=1}^{N}((\tilde{s}_u^T v)^2 - (E[\tilde{s}_u^T]v)^2) \\
&= \frac{1}{N}\sum_{u=1}^{N}((\tilde{s}_u^T v)^2 - (0v)^2) \\
&= \frac{1}{N}\sum_{u=1}^{N}(\tilde{s}_u^T v)^T(\tilde{s}_u^T v) \\
&= \frac{1}{N}\sum_{u=1}^{N} v^T \tilde{s}_u \tilde{s}_u^T v \\
&= v^T(\frac{1}{N}\sum_{u=1}^{N}\tilde{s}_u \tilde{s}_u^T)v \\
&= v^T \Sigma v \ .
\end{aligned} \tag{3.7}$$

Now we should seek out the basis vector that has the largest value of σ_{d_u}. Thus, the objective function is defined by:

$$\begin{aligned}
&\arg\max_{v} \quad v^T \Sigma v \\
&s.t. \quad u^T u = 1
\end{aligned} \tag{3.8}$$

The method of Lagrange multipliers is used here to find the maximum of the function value. We define the Lagrange function

$$\mathcal{L}(v, \lambda) = v^T \Sigma v + \lambda(1 - v^T v) \tag{3.9}$$

and solve the equation

$$\bigtriangledown (v^T \Sigma v + \lambda(1 - v^T v)) = 0 \tag{3.10}$$

It yields

$$\Sigma v = \lambda v \ . \tag{3.11}$$

Obviously, v is the eigenvector of the covariance matrix Σ and λ is its corresponding eigenvalue. The maximization of σ_{d_u} can be computed as follows:

$$\max \ \sigma_{d_u} = \max \ u^T \Sigma u = \max \ u^T \lambda u = \max \ \lambda u^T u = \max \ \lambda \tag{3.12}$$

where the first equation is based on (3.7) and the last equation is based on the constraint in (3.8). Therefore, an eigenvalue demonstrates exactly the variance of the data projection on its corresponding eigenvector. Due to the orthogonality between eigenvectors and the maximization of data restoration, the eigenbasis consisting of theses eigenvectors is an optimal basis that we want to find.

According to the finite-dimensional spectral theorem, the eigenvectors of a real symmetric matrix can be obtained by eigen decomposition:

$$\Sigma = V \mathrm{D} V^T \tag{3.13}$$

where V is a matrix whose columns are the eigenvectors of Σ, D is a diagonal matrix constructed from the corresponding eigenvalues. Fig.3.2 shows the first four major axes of the eigenbasis learned from *Lena* with a 7×7 patch and how the projection of the image on these axes looks like. Represented by the eigenbasis, the correlated original features of the image are transformed into linearly uncorrelated components[Bru15a]. As we can see, the first component contains the majority of the image content. The lower a component is ranked, the finer structures in the image it contributes to.

3.1.3. Selection of Principal Components

The slight difference between the true basis B and the observed basis $\tilde{B}$ can be ignored here, as the Gaussian noise lives almost completely in the range of $[-3\sigma, 3\sigma]$[Bru15a]. Under the assumption that $N >> M$ the bias caused by the noise has very little influence on learning the new basis. Now we need to select the components to be retained. There are two common strategies that are discussed here. The first one is to order the eigenvalues from large to small and choose the first largest ones as principal components according to a threshold, such as Kaiser's rule, scree plot and KOK[1]. KOK is the most

[1]The method "Keep or Kill" is performed to preserve a percentage of data variance.[DSD+11]

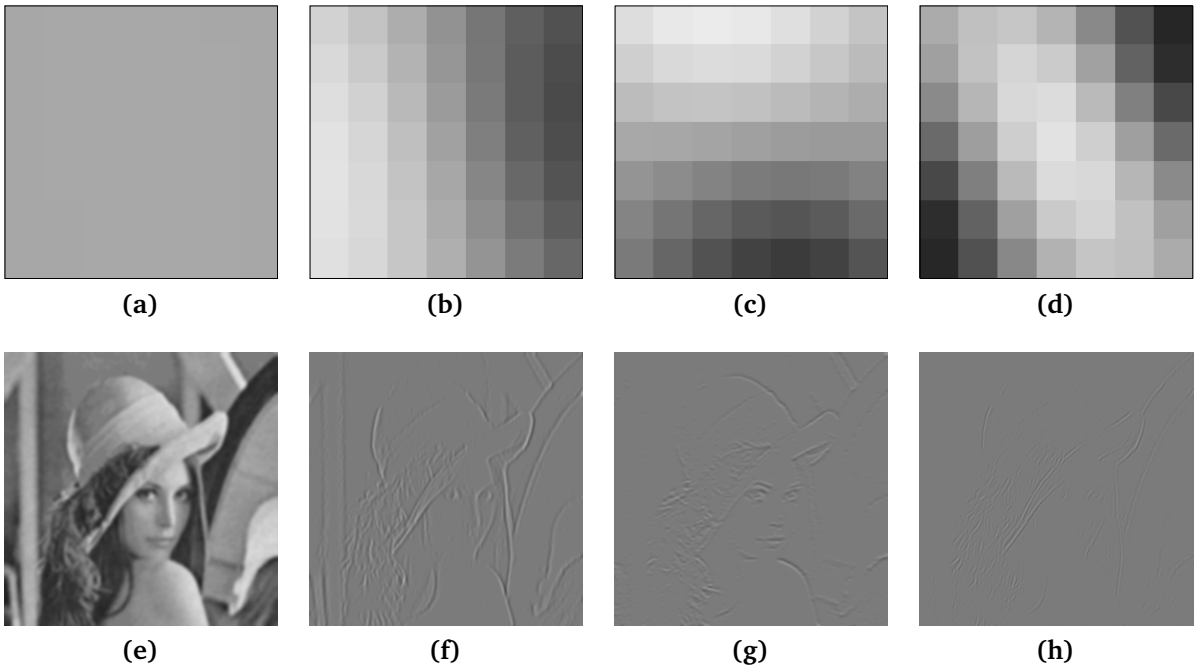


Figure 3.2.: Upper: The first four principal axes of the eigenbasis obtained by performing PCA on the noisy image. **Below:** The projection of the noisy image on these axes.

practical approach. It defines a threshold T_{KOK} to measure how much variation in the image should be kept[DSD+11]:

$$\frac{\sum_{k=1}^{M'} \lambda_k}{\sum_{k=1}^{M} \lambda_k} > T_{KOK} \tag{3.14}$$

where M is the total number of the components[2] and M' is the number of components to be retained. The inequality is solved to determine the M' principal components which form a low dimensional subspace to represent a denoised image.

$$\mathbf{D} = \begin{pmatrix} \mathbf{D}_{M'} & 0 \\ 0 & \lambda \mathbf{I}_{M-M'} \end{pmatrix} \tag{3.15}$$

Each principal component corresponds to an eigenvector. By projecting the centred samples $\tilde{s}_u$ onto these eigenvectors v_k, we attain their coefficients of the eigenba-

[2]Here we suppose there is no repetitive data sample, so the covariance matrix Σ has rank M. As rank(Σ) is equal to the number of non-zero eigenvalues, we have as many components as original features.

sis. The estimate of the samples $\hat{s_u}$ can be obtained after the addition of the mean value[DSD+11]:

$$\hat{s}_u = E[s_u] + \sum_{k=1}^{M'} (\text{proj}_{v_k} \tilde{s}_u) \cdot v_k \tag{3.16}$$

The idea of this strategy is to regard the smaller components as the disturbance of noise because feature values only slightly variate in these axes. Obviously removing these smaller components might make a loss in image details. The improvement for keeping local structures will be discussed in the following sections.

Another idea is to consider all axes as relevant for modelling the patches. But if the coefficient on one axis is less than a defined threshold, this component will be removed from the principal components. A classical example of this idea is Hard Thresholding. It defines a shrinkage function $\boldsymbol{\eta}$ that sets the coefficients to zero if the projection takes little effect[DSD+11]:

$$\hat{s}_u = E[s_u] + \sum_{k=1}^{M} \boldsymbol{\eta} \, (\text{proj}_{v_k} \tilde{s}_u) \cdot v_k \tag{3.17a}$$

$$\boldsymbol{\eta} = x \cdot \mathbb{1}(|x| > \text{T}_{\text{HT}}) \tag{3.17b}$$

where $\mathbb{1}$ stands for indicator function. Hard Thesholding has a good performance in practice. This strategy takes it into account that the uniform principal components obtained on the whole image might not be suitable for all patches, as the relevant axes may vary from one patch to another. In Fig.3.3 a simplified example of two dimensional data is used to explain this phenomenon: A basis $\{v_1, v_2\}$ is learned by PCA from the noisy data, in which v_1 corresponds to the leading direction of variance. Despite of that, there exist still samples that have a larger projection value on v_2 than on v_1. For instance, the data sample s can be more accurately approximated by the projection on v_2 than by the projection on v_1, although v_2 can not represent the major variation of the whole data samples.

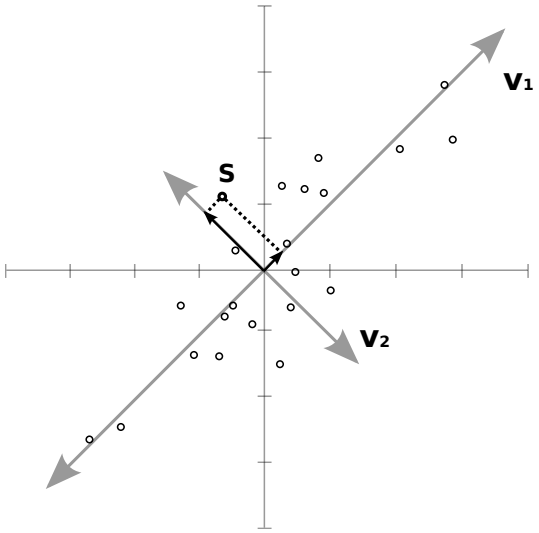


Figure 3.3.: A simlified 2D example that shows the advantage of Hard-thresholding.

Fig.3.4 displays the effect of the two thresholds on denoising performance of *Lena*($\sigma = 20$). T_{KOK} represents the preserved data variation which ranges from 90 to 100 percent. T_{HT} shows the threshold of coefficients by projecting which is proportional to the noise variance σ. The PSNR value is used to measure the denoising performance. Obviously, the PSNR value with Hard Thesholding is much higher than that without Hard Thresholding($T_{HT} = 0$), especially when more data variance is required to be kept. However, the psnr value with setting KOK threshold is higher than that without setting KOK threshold($T_{KOK} = 1$) only when T_{HT} is low.

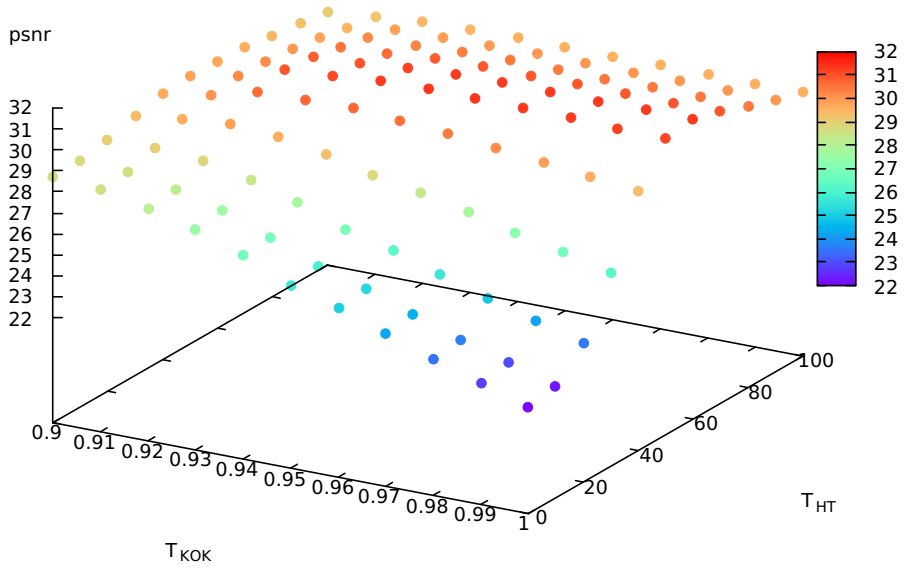


Figure 3.4.: The denoising performance affacted by KOK-threshold and Hard-threshold T_{KOK} and T_{HT}.

Nevertheless, Hard-thresholding is time-consuming compared to KOK. This is due to the fact that it has to traverse all the components to check whether the coefficients are too small or not, even though the majority of information is in fact concentrated only on a few components. Although T_{KOK} plays little role in denoising performance when T_{HT} arrives to an optimal value(here 55), it can accelerate the denoising performance in the case of large dimensionality. Hence, we can first set T_{KOK} to filter out the components that are evidently insignificant for all patches and then use T_{HT} to find the principal components for various patches. The estimate can be revised as follows[DSD+11]:

$$\hat{s}_u = E[s_u] + \sum_{k=1}^{M'} \boldsymbol{\eta} \left(\text{proj}_{v_k} \tilde{s}_u\right) \cdot v_k \tag{3.18}$$

In practice, the denoising performance of this hybrid method is slightly better than that of purely Hard-thresholding. The reason is probably that for Hard-thresholding the samples with small magnitude could lose the coefficients on all axes while the samples with large magnitude remain totally unchanged.

3.2. Local Adaption

As mentioned before, the single eigenbasis learned from the whole image is unable to represent all of the patches due to their large difference in local texture. In order to preserve more image details, we use local PCA to improve the denoising effect. Since patches inside small regions usually have less variability, we can perform PCA on these small regions respectively to obtain locally suitable bases. Furthermore, similar patches inside these small regions can be selected and grouped for learning a more accurate basis. The process of filtering out unsuited patches can be converted into a classification problem which will be solved by block matching.

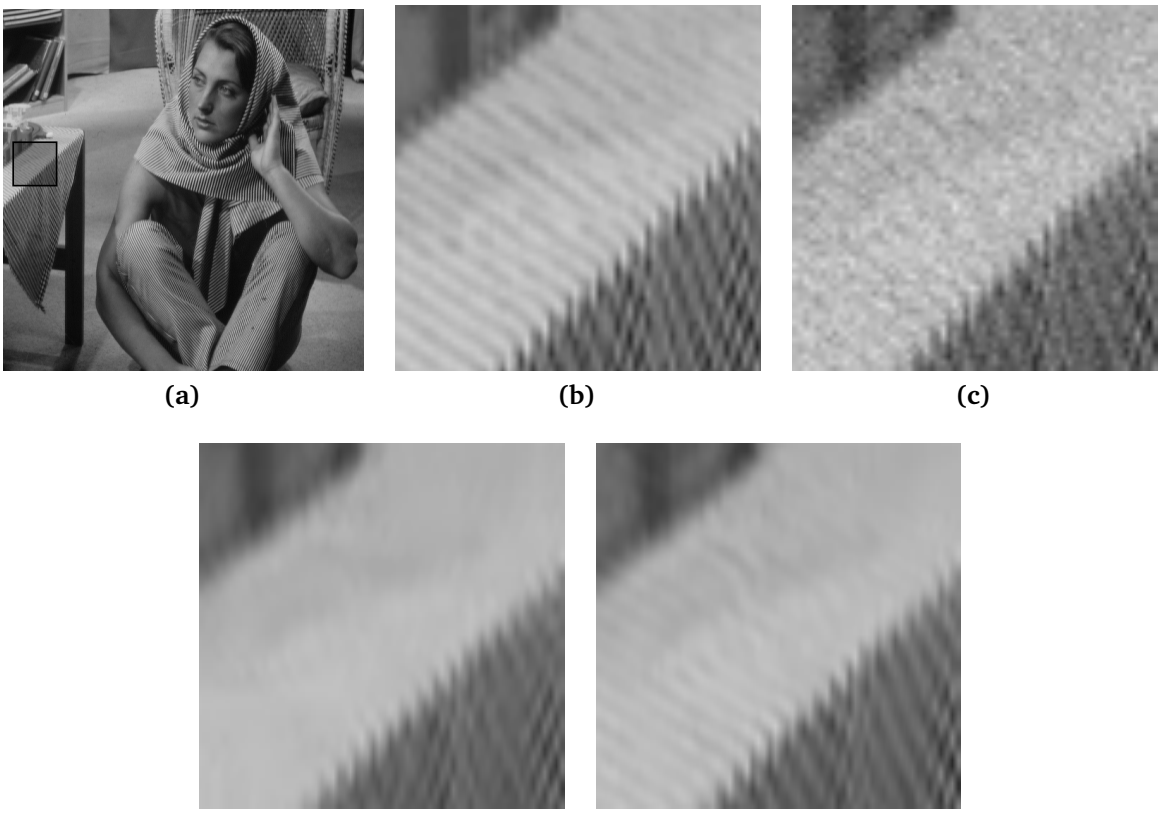
3.2.1. Local PCA

A globally learned basis reflects the main variance of an image. Therefore, frequent patterns can obtain a preferable denoising effect while rare patterns usually suffer severe loss of details. In Fig.3.5 we focus on the image region marked by a transparent square, namely the upper portion of the tablecloth. Due to the fact that this pattern occurs less often than that of the kerchief and the trousers, it can be hardly taken into account in the principal components of a global PCA. So in the result of the global PCA the stripes on the table are totally removed (Fig.3.5(d)). But in a local PCA a fraction of the stripes can still be recognized (Fig.3.5(e)).

In practice, a local PCA can be realized with an $l \times l$ $(l > p)$ search window which contains $(l - p + 1)^2$ patch samples in total. We first consider the case of performing PCA on all of these samples. Each patch in this window obtains an estimate by projecting the noisy data on the locally learned basis. Then, the window is moved with a sliding step δ to the next image region, on which another local basis is learned in the same way as before. Notice that if $\delta < l$, the regions on which PCA is performed can be overlapping, then quite a few pixels acquire probably multiple estimates. Here we take an average of them as the final estimate. Although high overlapping of image regions improves the accuracy of the basis, it is highly time consuming. For a balance between denoising performance and computation time the sliding step is set to half the length of the search window $\delta = \lfloor l/2 \rfloor$. In this case, the computation time is reduced to $1/\delta^2$ of the original time cost but the performance has only a slight loss[DSD+11].

3.2.2. Local Pixel Grouping

When we perform a local PCA, it is hard to determine an optimal size of the search window. A too small window may lead to lack of samples while a too large window



(a) **(b)** **(c)**

(d) **(e)**

Figure 3.5.: Denoising effect of rare patterns. **(a):** Original image; **(b):** an image region; **(c):** noisy image region ($\sigma = 20$); **(d):** denoised image region of a global PCA; **(e):** denoised image region of a local PCA.

inevitably contains patch samples with larger variety. The goal of LPG (Local Pixel Grouping) is to ensure only similar patches in a search window will be selected to form a new basis. To compare the similarity of two matches, we use block matching which is often used in motion estimate[DFKE07].

Suppose there is a $p \times p$ $(= M)$ reference patch that is to be denoised. LPG establishes an $l \times l$ search window in which the reference patch locates exactly at the center. We denote the patch set as P and the central patch as $P[m]$. Each element $P[m]$ is considered as a candidate of similar patches to $P[m]$. The (dis)similarity between $P[m]$ and $P[i]$ is measured with the Euclidean distance for vectors:

$$\mathrm{d}_2(P[m], P[i]) = \sqrt{\sum_{j=1}^{M} (P[m]_j - P[i]_j)^2} \tag{3.19}$$

Alg.3.1 shows the LPG method to search for similar patches. Here we use a threshold ϵ of the distance d_2 to pick out similar patches from the candidates. For the case of insufficient samples, n best matched samples are selected as training samples for basis learning.

Similar to local PCA without selecting similar patches (see Sect.3.2.1), the denoising performance can be evidently increased when LPG is performed on each image patch. However, a sliding step is necessary in practice due to severe computational burden. We need to find a proper sliding step such that the computation time is acceptable and meanwhile the denoising performance has little influence. Fig.3.6 shows how the PSNR value of the image *House* is influenced by an increase of step size. From the blue line we know the denoising performance of a local PCA without LPG is not decreased severely when the step size is within four times of the patch size. The red line shows the acceptable step size for a local PCA with LPG is maximal only two times of the patch size. The reason is that the basis learned by LPG-PCA is specific for the reference patch, while the basis learned by local PCA without LPG is universal for all the patches in a search window.

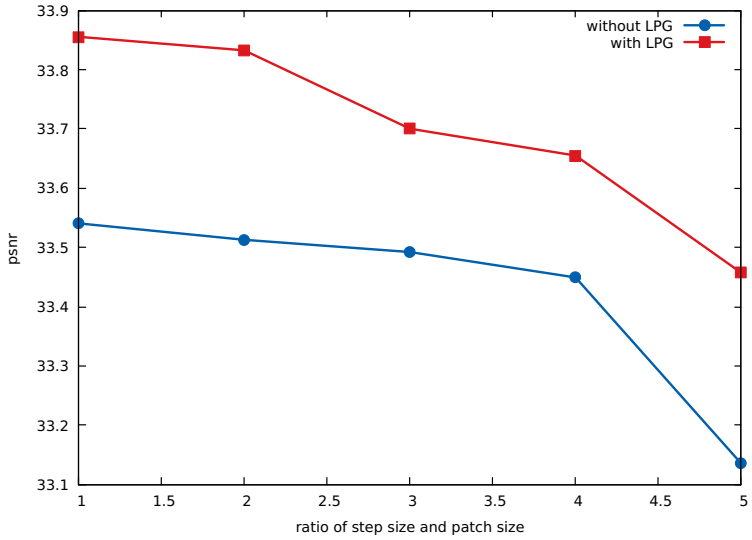


Figure 3.6.: The influence on denoising performance by increasing the sliding step. A comparison between a local PCA with and without LPG is made. Test image: *House*.

Algorithmus 3.1 LPG for A Single Patch

Require:

S // Set of selected training samples
R // Set of residual samples
U // Set of d_2 value of residual samples
sort(A) // Sorts elements in a set from large to small
add(A, a) // Adds an element to a set
getIndex(A, a) // Returns the index of a given element in a set

function LPG(P, ϵ, n, k, l)
  $N \leftarrow (l-p+1)^2$ // Number of samples
  $m \leftarrow \lceil N/2 \rceil$ // Index of the reference patch
  for $i \leftarrow 1$ to N **do**
    if $d_2(P[m], P[i]) < \epsilon$ **then**
      add(S, $P[i]$)
    end if
  end for
  if $|S| < n$ **then**
    $R \leftarrow P \backslash S$
    for $j \leftarrow 1$ to $|R|$ **do**
      add(U, $d_2(P[m], P[j])$)
    end for
    sort(U)
    for $d \leftarrow 1$ to $n - |S|$ **do**
      $x \leftarrow$ getIndex(R, $U[d]$)
      add(S, $R[x]$)
    end for
  end if
  return S
end function

Alg.3.2 displays the algorithm to perform an LPG-PCA on the whole image. With a sliding step δ some patches are skipped during the iterations. Notice that pixels on the boundaries has fewer samples in their search window. The time complexity for denoising an $n_1 \times n_2$ image is $O(\frac{1}{\delta^2} \cdot N \cdot n_1 \cdot n_2)$, where N is the number of samples in a search window.

Algorithmus 3.2 LPG-PCA

Require:
δ // Step size
pca(A) // Returns the basis learned by PCA over the set A
for $i \leftarrow 1$ to n_1 **do**
  for $j \leftarrow 1$ to n_2 **do**
    $t = i * n_2 + j$ // 1D index of a pixel
    $P_t = \{z_{u,v} \mid u \in [\max(i - \frac{l}{2}, 0), \min(i + \frac{l}{2}, n_1]$
      $\cap\, v \in [\max(j - \frac{l}{2}, 0), \min(j + \frac{l}{2}), n_2)\}$
    $S = \text{LPG}(P_t)$
    $B = \text{pca}(S)$
    $\tilde{P}_t = \text{proj}_B P_t$ // Projection onto the new basis
    Extract $P_t[m]$ from $\hat{P}_t$
    $j = j + \delta$
  end for
  $i = i + \delta$
end for

3.3. Non-local Improvement

Local search can increase the similarity between patch samples. However the accuracy of a PCA-basis depends also on the number of samples. In order to obtain sufficient samples, we can further search for non-local similar patches in the whole image. In Fig.3.7, we can see that the patch s_2 is highly similar to the reference patch s_1, although it is not in the local window. For non-local search with high-dimensional data, space partitioning is necessary to make the search process efficient. For this We use a vp-tree as search structure.

Figure 3.7.: Non-local similar patches.

3.3.1. Similarity Search

Similarity search is also called nearest neighbour search (NNS). The goal of NNS is to find a subset of samples $\mathbf{P}_c \subset \mathbf{P}$ such that a criterion is met. There are two types of criteria that are frequently used: ϵ-close neighbours (ϵ-NN) which limits the range from the query q to the candidates p_i[KZN08]:

$$p_i \in \mathbf{P}_c \Leftrightarrow d(p_i, q) \leqslant \epsilon \tag{3.20}$$

and k-nearest neighbours (k-NN) which limits the amount of the best matches to the query[KZN08]:

$$\forall p_i \in \mathbf{P}_c, \forall p_j \notin \mathbf{P}_c : d(p_i, q) \leqslant d(p_j, q) \tag{3.21}$$

where d is a distance metric function that satisfies[KZN08]:

• Symmetry: $d(a, b) = d(b, a)$,
• Non-Negativity: $d(a, a) = 0$ and $d(a, b) > 0, a \neq b$,
• Triangle Inequality: $d(a, b) \leqslant d(a, c) + d(b, c)$.

In this thesis we use the Euclidean distance d_2 as the metric function. Both ϵ-NN and k-NN have been used in Alg.3.1. As can be seen, the running time of exhaustive search is effected by the "curse of dimensionality". Therefore, an efficient data structure for search is necessary, it should quickly exclude dissimilar samples from candidate patches to reduce the number of distance computations.

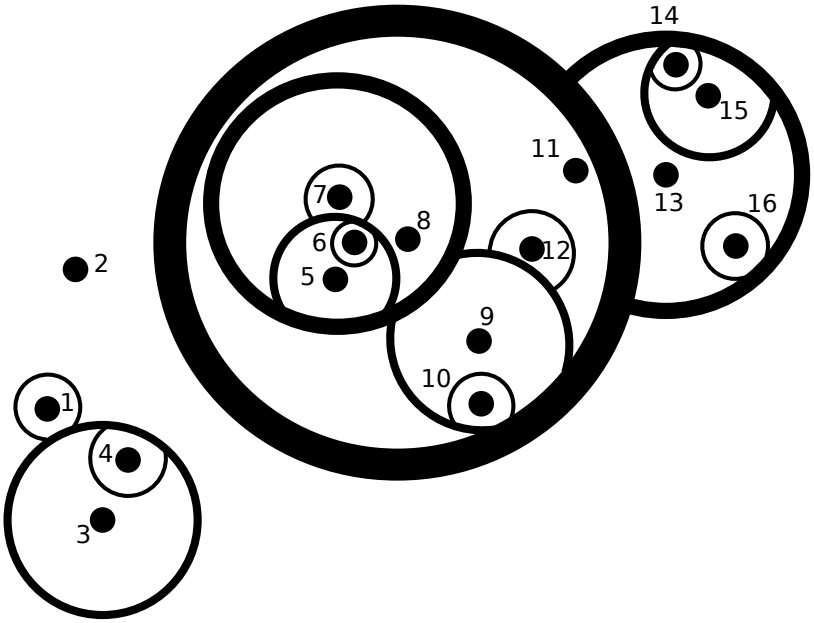
3.3.2. Construction of a VP-tree

We construct a vp-tree to partition sample data. Each node in a vp-tree consists of the following attributes[Yia93]:

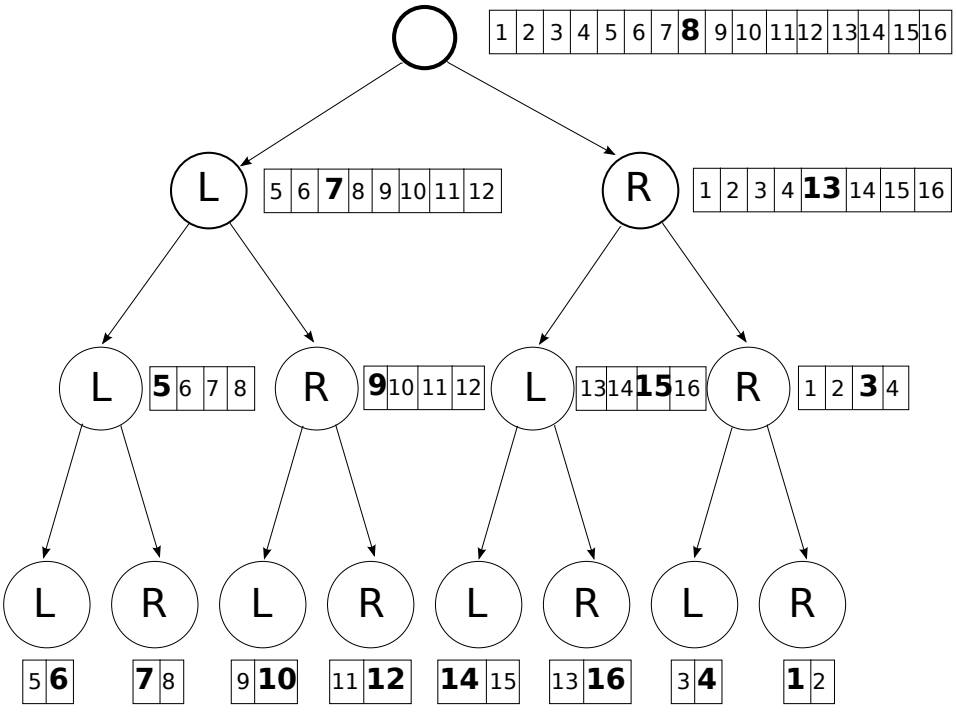
• a set of points (also called elements) assigned to this node (root has the universe of points),
• a vantage point vp which is selected from the set above,
• a distance r which is the radius of a hypersphere centred at vp,
• a left child node, in which all the points are inside the hypersphere,
• a right child nodes, in which all the points are outside the hypersphere.

Fig.3.8(a) displays this data structure with circles, in which line thickness stands for partitioning order. The corresponding representation with tree structure is shown in 3.8(b). It can be seen that the level order corresponds to the partitioning order mentioned above. The blocks near each node record the points assigned to this node.

among them the one in bold is the vantage point of the node, which is illustrated as the centre of circles.



(a)



(b)

Figure 3.8.: A binary vp-tree. **(a):** Circle representation; **(b):** tree structure.

The construction of a vp-tree is a recursive partitioning process similar to that of a kd-tree. The algorithm for constructing a vp-tree is shown in Alg.3.3.

Algorithmus 3.3 Construction of A VP-tree

Require:

```
    D                    // A set to record the distance from vp to other points
    ls                   // Leaf size: Maximal number of points in a leaf node
    leftElements         // A set to save points for left child node
    rightElements        // A set to save points for right child node
    median(A)            // Returns the median of a given set
    chooseVP(A)          // Returns a point as vantage point of the node
 1: struct node:
 2:     elements, vp, leftChild, rightChild, r
 3: function CONSTRUCT_VP_TREE(P)
 4:     node = new Node()
 5:     node.elements = P
 6:     node.vp = chooseVp(P)
 7:     if |P| > ls then
 8:         for i ← 1 to |P| do
 9:             D[i] = d_2(P[i], node.vp)
10:         end for
11:         r = median(D)
12:         for i ← 1 to |P| do
13:             if D[i] < r then
14:                 leftElements.add(P[i])
15:             else
16:                 rightElements.add(P[i])
17:             end if
18:         end for
19:         node.leftChild = construct_vp_tree(leftElements)
20:         node.rightChild = construct_vp_tree(rightElements)
21:     end if
22:     return node
23: end function
```

The parameter P is a data set which contains all the patches in an image. Each patch can be regarded as a high-dimensional point. In addition, a leaf size is defined. This means, if the number of remaining points is smaller than this value, these points should be assigned into one leaf node which is returned to the upper level of the recursion. For the function chooseVP(), we choose a point from our data set as the vantage point of the

root[3]. Then we compute the median distance between the vantage point and all other points in the root. The points within the distance from the vantage point are assigned to the the left child node and the points farther away from the vantage point than the distance are assigned to the right child node. Afterwards we make a recursive process to the left and right child node until there are no more points left. Due to the median distance the elements in each node can be divided in a balanced way if there exists no point that has exactly the same distance to any two arbitrary points.

3.3.3. Searching through a VP-tree

Before discussing the k-NN problem, we first consider the case of single-nearest-neighbour. When we search for the nearest neighbour of a point through a vp-tree, there are two cases in which we can "prune" one of the child nodes of the current node, which significantly accelerates the search process. Before we get to that, some variables need to be defined first:
• τ: A threshold distance from a query point, it ensures that the nearest neighbours are contained within this area. The value of τ could be updated after a node is visited,
• d: The distance from a query point to the current vantage point it visits,
• r: The radius of the hypersphere centred at this vantage point.

In the following graphs, visited points are illustrated with red, a query point with green and a vantage point being visited with blue. A red dashed line stands for the circle area determined by τ. A green dashed line shows the distance between a query point and a vantage point that is being visited. A blue dashed line is the radius of a hypersphere.

[3]For an optimized tree, a vantage point can be chosen by selecting the point with largest spread rather than randomly. Spread is defined as the sum of Euclidean squared distance to all other points. More details see [Yia93].

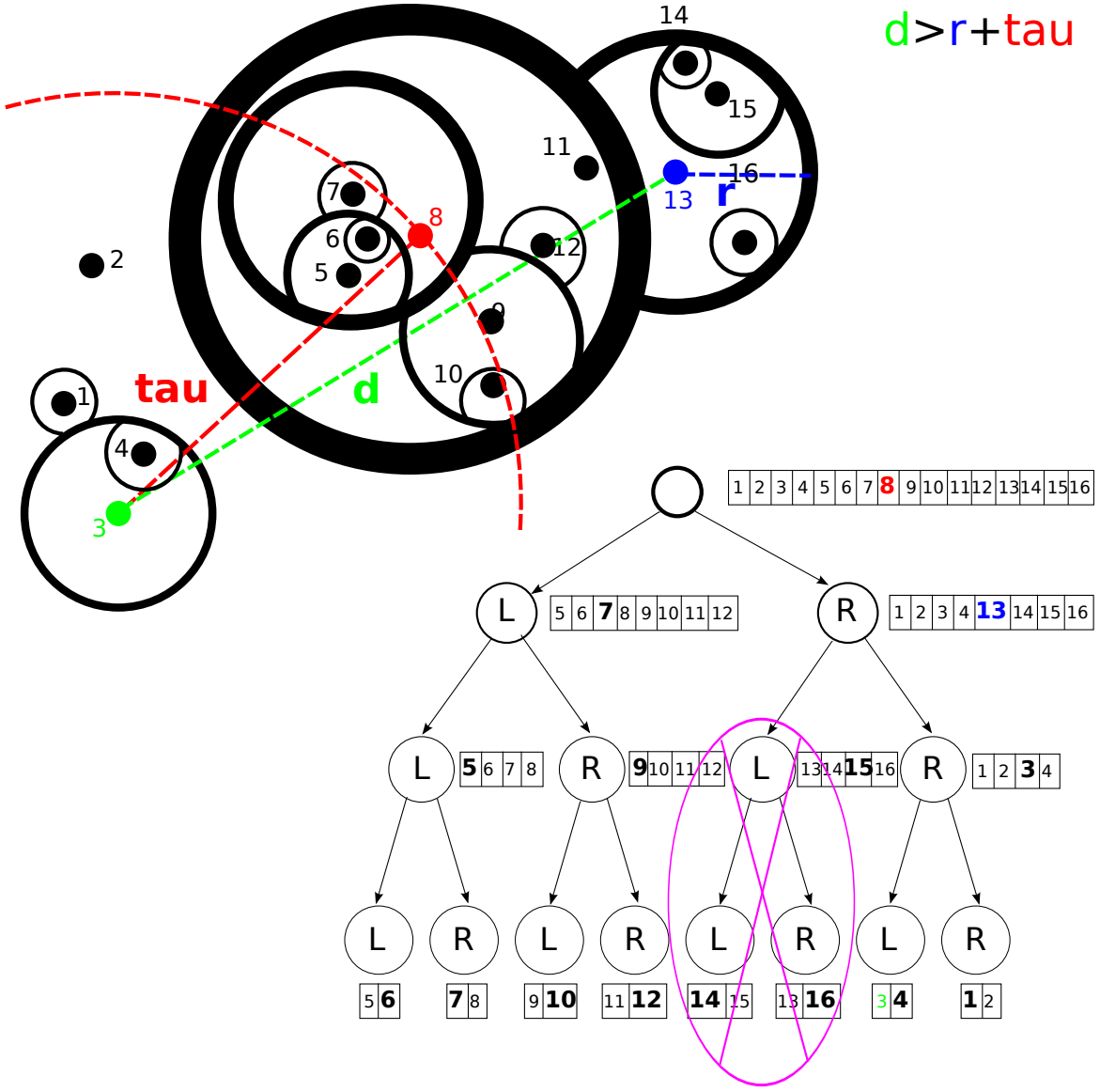


Figure 3.9.: The case of "pruning" left child

We can compare the relation between the three variables above to figure out whether the requirements for the two "prune" cases are satisfied or not. Fig.3.9 displays the first case. The query point (No.3) has obtained the value τ from the visited points (No.8) as the root was added to the nearest neighbours for the moment. When the query point visits the vantage point of the right child (No.13), we find that all the points in the circle centred at point 13 are outside of the circle marked by red dashed line. This means only the points outside the circle centred at point 13 could be even nearer to the query point than the existing near neighbours. Therefore, we can "prune" the left child if $d > r + \tau$ is satisfied.

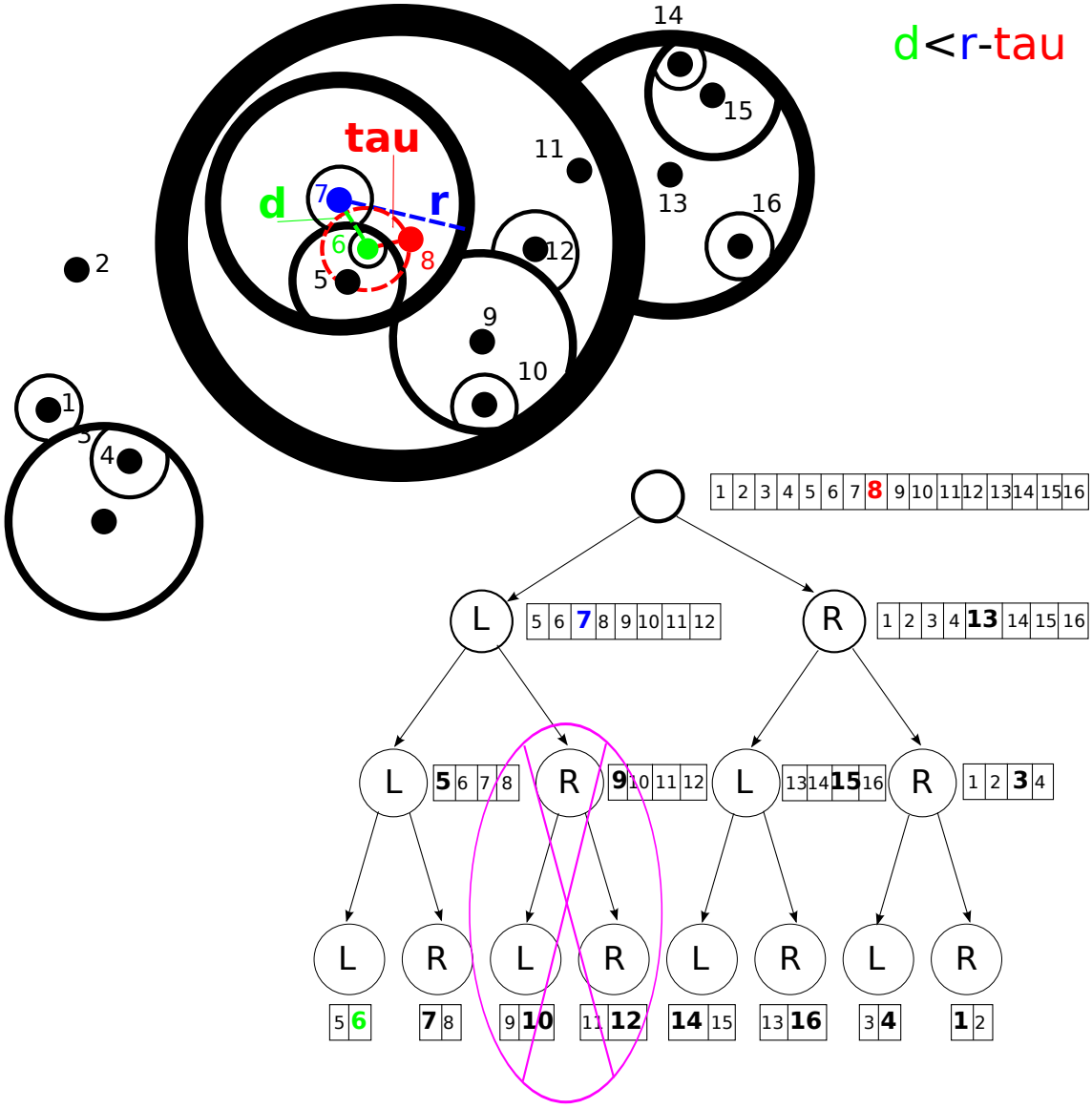


Figure 3.10.: The case of "pruning" right child

The second "prune" case is shown in Fig.3.10. After visiting point 8 which is the vantage point of the root, the query point (No.6) turns to the left child node. It can be seen that the circle marked by red dashed line is totally in the circle centred at point 7. That is to say, we should only consider the points inside the circle centred at 7, as the nearest neighbour of the query point is only possible to appear in the left child. Therefore, the right child can be "pruned" if $d < r - \tau$ is satisfied.

Compared with kd-tree which is regarded as a classic method for space partitioning, VP tree uses the actual distance between samples as partitioning metric rather than their

projection on one of axes. The problem for kd-tree lies in its severe degradation with increase of the data dimensionality, as it is hard to find an optimal axis to partition the scattered data. Therefore, we use VP trees here as search structure.

Algorithmus 3.4 Searching through A VP-tree

Require:
isLeaf(node) // Returns 1 if the node is a leaf
brute_force(q, node) // Returns the nearest point to q in the node
/⋆node: node of a vp-tree, initialized as root⋆/

```
1: function VP_SEARCH(node, q)
2:     if isLeaf(node) then
3:         return brute_force(q, node)
4:     else
5:         d = d_2(q, node.vp)
6:         r=node.r
7:         if d < τ then
8:             τ = d                    // Update τ
9:             if d > r + τ then
10:                vp_search(node.rightChild) // Prune the left child
11:            else if d < r − τ then
12:                vp_search(node.leftChild) // Prune the right child
13:            else
14:                vp_search(node.leftChild)
15:                vp_search(node.rightChild)
16:            end if
17:        end if
18:    end if
19: end function
```

Alg.3.4 shows the process of searching a single-nearest-neighbour through a vp-tree. The exit of the algorithm is reaching a leaf node. We suppose the leaf size is 1 and consider the case of multiple-nearest-neighbours. From Alg.3.4 we know a search for single-nearest-neighbour through a vp-tree over N patches costs $O(\log N)$ time[KZN08]. Then, a search for k-NN through vp-tree costs $O(k \cdot \log N)$ time. By contrast, the computation time for an exhaustive search for k-NN is $O(k \cdot N^2)$[KZN08]. If we add the time for building a vp-tree which is $O(N \log N)$, it's still much faster than the exhaustive search.

4. Implementation

This chapter refers to the implementation of the denoising methods in the previous chapter and a performance test on classic test images. In the first section, we show the framework of the implementation and give an introduction of test conditions. In the second section, we discuss about parameter selection for different methods. In the third section, the experimental results of the implementation are displayed. At last, an evaluation is made w.r.t. the denoising performance.

4.1. Framework

In this thesis four denoising methods with patch-based PCA are realized, which are a global PCA, a local PCA without choosing similar patches, a LPG-PCA and a hybrid PCA with non-local similarity search through a vp-tree. In Fig.4.1 they are depicted with ovals. In the rectangles are the modules we use to realize these methods.

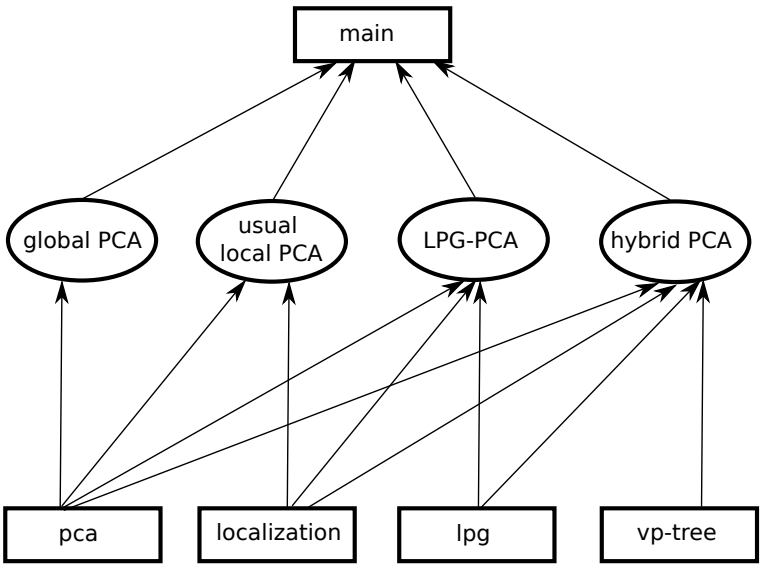


Figure 4.1.: Framework of the implementation.

The program is implemented in C language. The package GNU Scientific Library (GSL) is used for vector and matrix operations. The vp-tree is based on an existing implementation[Fu]. In addition, all test images are in greyscale PGM-format.

4.2. Parameter Selection

Table 4.1.: Parameter Selection

			$\sigma = 5$	$\sigma = 10$	$\sigma = 20$	#samples	#features	similarity of samples	computation time
common		p	7×7	7×7	7×7	-	+	-	+
pca		T_{KOK}	0.99	0.97	0.94				+
		T_{HT}	2.5σ	2.5σ	2.5σ				
loc		δ	4	4	4				-
	lpg	ϵ	$0.9d_{\max}$	$0.9d_{\max}$	$0.9d_{\max}$	+		-	
		n	$0.9N$	$0.9N$	$0.9N$	+		-	+
vp-tree		k	f(n, ϵ)	f(n, ϵ)	f(n, ϵ)	+		-	+
		ls	10	10	10				-

Tab.4.1 shows the selected parameter in this implementation. The left-most column records the modules to which the parameters are categorized. The right-most three columns display whether the listed factors are positive or negative related to the value of a parameter. Patch size p is a common parameter for all the methods. It reflects how far a kernel spreads. A moderate size is used to make a balance between number of samples and number of features.

In the "pca" module there are two shrinkage parameters T_{KOK} and T_{HT}. As mentioned in Sect.3.1, T_{HT} has a more powerful influence on the denoising effect. T_{KOK} stands for the percentage of preserved variation. This optimal value for this parameter decreases with the increase of σ. Because in that case noise accounts for more variation of an image. By contrast, T_{HT} is proportional to σ, as in a highly noisy image, more components of an image are occupied by noise. A higher value of T_{HT} can get rid of more components.

In the "local" module, δ is defined as the ratio of step size and patch size, which determines the frequency of sampling and the computation time. The variation trend of this parameter has been shown in Fig.3.6. Considering the importance of time cost in practice, we use four times of the patch size. Notice that LPG-PCA has a more severe loss in this case than usual local PCA. In LPG-PCA, we use the parameter ϵ to restrict the intensity difference between a reference patch and a candidate patch. The influence of ϵ on number of samples and on similarity of samples is opposite. That is, when we have a

more strict limit on the similarity between a reference and candidates, fewer samples can be obtained, that's the reason why a non-local search is needed. The parameter n decides the number of required samples. But with the increase of samples numbers, the similarity between them is decreased. Here we use the coefficient 0.9 for both ϵ and n, which is a relative high value. That is because the number of samples plays a more important role if there exists a simultaneous influence of these two factors on denoising performance.

In the "vp-tree" module, the parameter k is used for the case of insufficient samples. The value of k is f(n, ϵ) that computes the number of similar patches that have to be searched in the whole image. A too large k my lead to severe computational burden. The parameter ls is the leaf size of a vp-tree. A moderate value of ls could accelerate the the search process since the times of running a overhead is decreased.

4.3. Experimental Results

This section displays the denoising performance of the patch-based PCA methods by performing them on classical test images. These images are cropped and enlarged to represent details and local structures more clearly. The original image of *lena*, *barbara* and *peppers* has a size of 512×512 and *cameraman*, *house* and *einstein* has a size of 256×256.

4.3.1. Denoising Performance

Tab.4.2 shows the PSNR value of the six test images with different degrees of noise. From this table we can see that the local PCA yields in general a better denoising performance than the global PCA. However, the difference between the usual local PCA and the LPG-PCA is comparatively not evident. Although the PSNR value of the LPG-PCA is in most cases slightly higher than that of the usual local PCA, it could still happen that the LPG has a lower PSNR value (i.e. *cameraman* for $\sigma = 5$ and $\sigma = 10$). Therefore, if we only search for similar patches in a local region, the increase of similarity between patches could be cancelled out by the reduction of sample amount. By contrast, hybrid PCA has a robust performance due to sufficient samples.

Besides the internal comparison, we record the corresponding data value of NL-Menas and BM3D from the literature. The methods with patch-based PCA implemented in this thesis have a better performance than NL-Means for the most images. Only *peppers* has a lower PSNR value for $\sigma = 5$. This is probably due to the different size of images. However, our methods are still not competitive to BM3D.

Table 4.2.: Performance Comparison

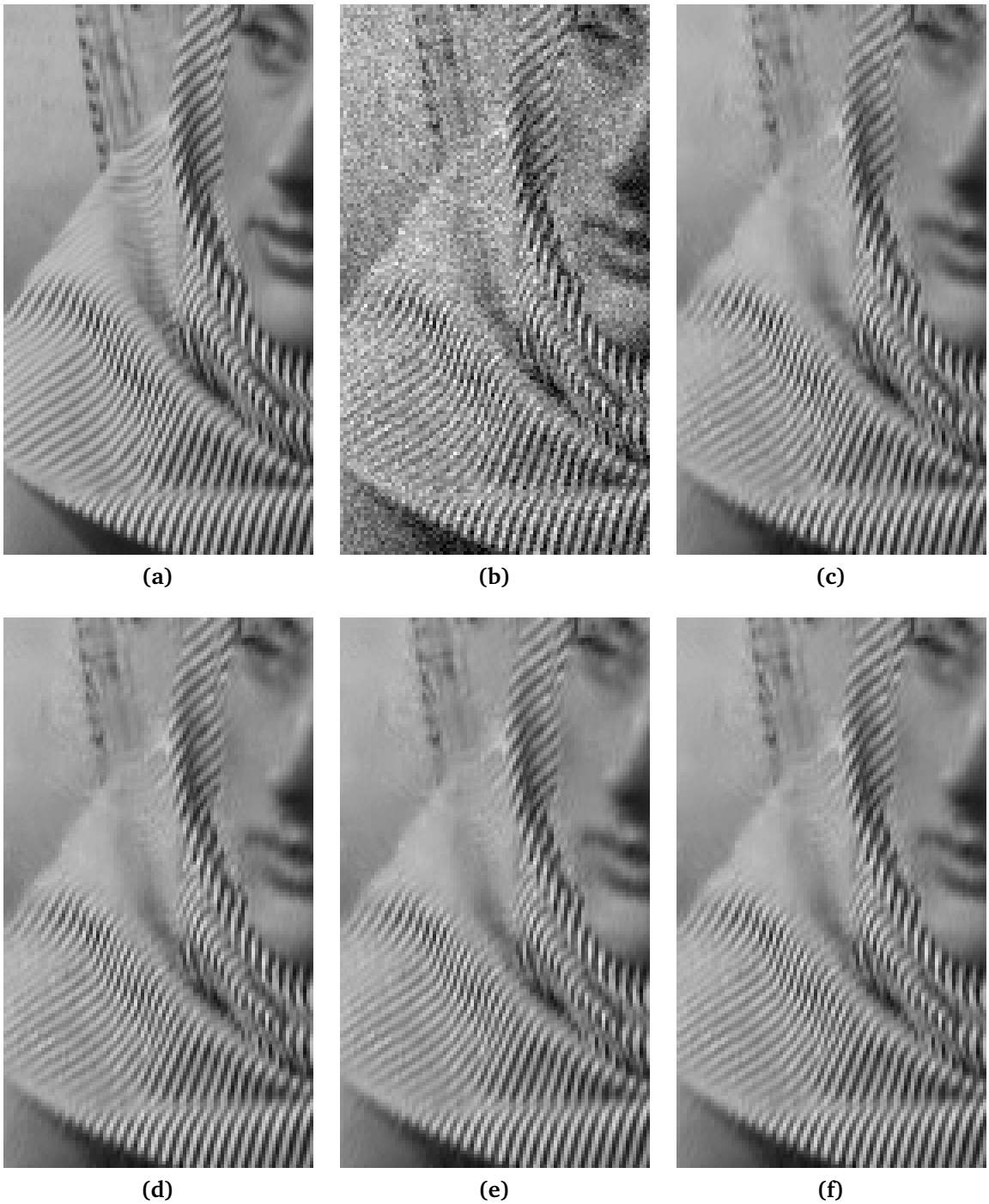
	global PCA	usual local PCA	LPG-PCA	hybrid PCA	NL-Means	BM3D
$\sigma = 5$						
lena	38.50	38.55	38.59	38.63	37.9	38.72
barbara	38.14	38.30	**38.33**	**38.37**	37.0	38.31
cameraman	37.80	37.88	37.84	37.89	37.7	38.29
house	38.50	38.60	38.62	38.70	38.5	39.83
peppers	37.04	37.14	37.15	37.22	37.4	38.12
einstein	37.17	37.24	37.27	37.31	-	-
$\sigma = 10$						
lena	35.32	35.47	35.49	35.55	34.2	35.93
barbara	34.15	34.63	34.70	34.79	33.0	34.98
cameraman	33.33	33.38	33.37	33.44	33.2	34.18
house	35.18	35.33	35.35	35.43	34.8	36.71
peppers	33.52	33.62	33.63	33.71	33.3	34.68
einstein	33.68	33.79	33.80	33.88	-	-
$\sigma = 20$						
lena	32.15	32.29	32.33	32.43	31.0	33.05
barbara	30.17	30.95	30.93	31.02	29.8	31.78
cameraman	29.35	29.45	29.51	29.98	29.4	30.48
house	32.35	32.39	32.36	32.42	31.6	33.77
peppers	29.93	30.03	30.05	30.12	29.8	31.29
einstein	30.50	30.61	30.67	30.71	-	-

A visual contrast of various patch-based PCA methods is shown in Fig.4.2, 4.3 and 4.4. By comparing the visual effect, we know that the advantage of local adaption is mainly represented in clear edges (i.e. Lena's cheek, part of the chair in the background of Barbara and Einstein's hairline). Notice that intensive distributed textures such as stripes on Barbara's kerchief and Einstein's suit are difficult to be restored if the noise is too strong.

(a) **(b)** **(c)**

(d) **(e)** **(f)**

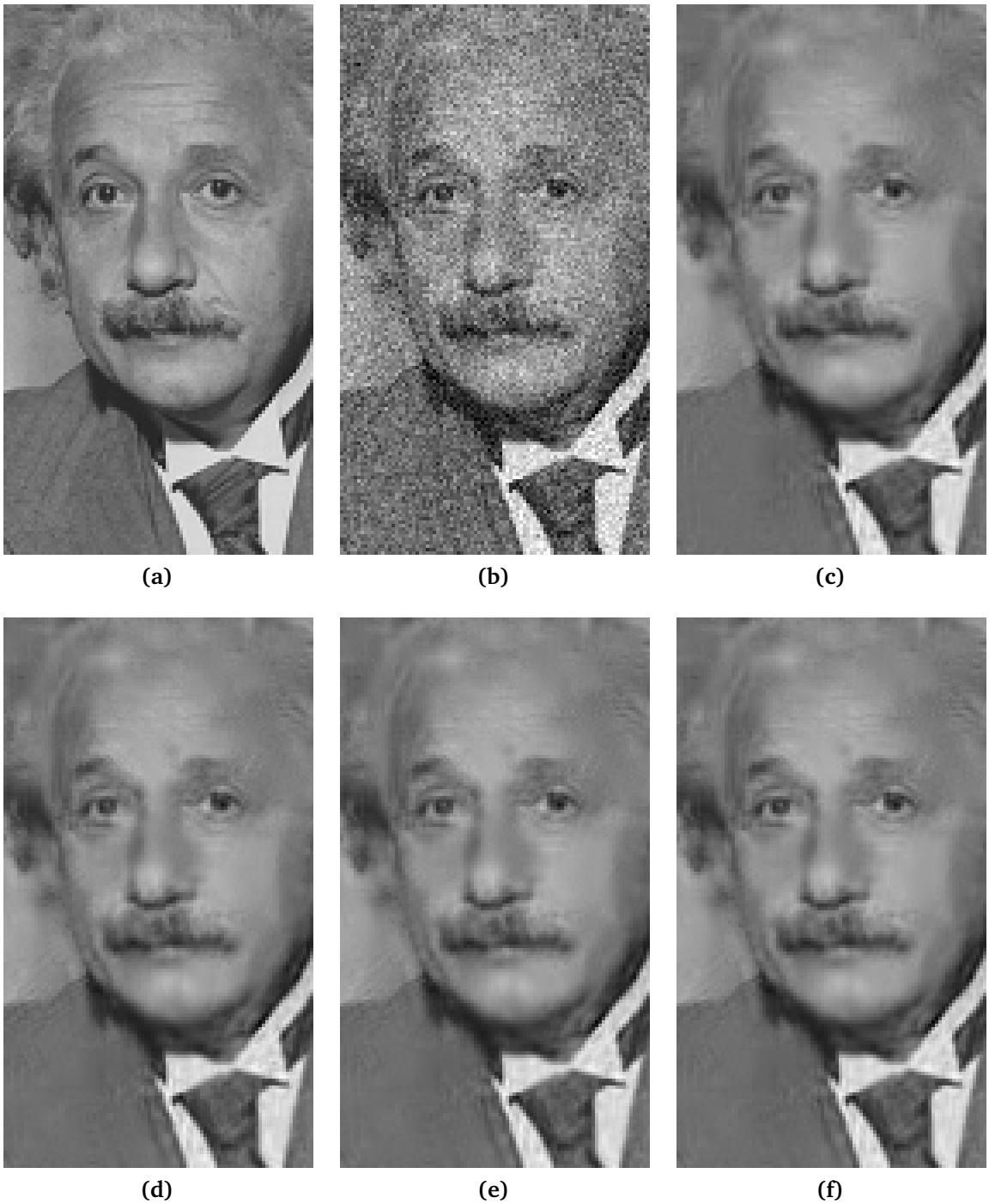
Figure 4.2.: Denoising effect of *Lena*. **(a):** Original image (512×512); **(b):** noisy image ($\sigma = 20$); **(c):** global PCA (PSNR=32.15 dB); **(d):** usual local PCA (PSNR=32.29 dB); **(e):** LPG-PCA (PSNR=32.33 dB); **(f):** hybrid PCA (PSNR=32.43 dB).



(a) **(b)** **(c)**

(d) **(e)** **(f)**

Figure 4.3.: Denoising effect of *Barbara*. **(a):** Original image (512×512); **(b):** noisy image ($\sigma = 20$); **(c):** global PCA (PSNR=30.17 dB); **(d):** usual local PCA (PSNR=30.95 dB); **(e):** LPG-PCA (PSNR=30.97 dB); **(f):** hybrid PCA (PSNR=31.02 dB).



(a) **(b)** **(c)**

(d) **(e)** **(f)**

Figure 4.4.: Denoising effect of *Einstein*. **(a):** Original image (256×256); **(b):** noisy image ($\sigma = 20$); **(c):** global PCA (PSNR=30.50 dB); **(d):** usual local PCA (PSNR=30.61 dB); **(e):** LPG-PCA (PSNR=30.67 dB); **(f):** hybrid PCA (PSNR=30.71 dB).

4.3.2. Time Complexity

Tab.4.3 shows the computation time of the above methods for test images with different size. The sliding step for the usual local PCA is half of the window size and the sliding step for the LPG-PCA is 4, because its denoising effect is senstive to the sampling rate. The CPU of the test computer is an Intel Core i5 2410M.

Table 4.3.: Time Complexity

	global PCA	usual local PCA	LPG-PCA	hybrid PCA
128×128	0.272s	0.680s	0.276s	0.593s
256×256	0.816s	2.652s	4.056s	6.758s
512×512	2.948s	12.740s	53.436s	1m3.622s
1024×1024	10.412s	48.888s	14m9.740s	15m5.439s

As can be seen, the time complexity of the global PCA is sublinear in the image size[1]. With a proper value of the step size, the local PCA runs almost at a linear speed. The LPG has an almost quadratic time complexity even though a step size is given. Therefore, it is hard for the LPG-PCA to find a balance between denoising effect and computational cost when images have a large size. The time complexity of the hybrid PCA is similar to the LPG-PCA, because its local part is exactly the LPG-PCA. That is, the most time cost of the hybrid PCA is occupied by the local part. The non-local part of the hybrid PCA is actually sublinear to the image size.

[1]The time complexity of a global PCA is theoretically linear in the image size. This could be due to the dynamic control of the processor's clock rate.

5. Conclusion

This thesis investigated various denoising methods with patch-based PCA. Based on the strategies of the global PCA, local PCA and LPG-PCA, an extension to a non-local search for similar patches has been made. The hybrid PCA approach yields a robust but slightly better performance than the LPG-PCA. Restricted to the implementation of the local part, the hybrid PCA has a high time cost. However, this method has verified the possibility of non-local similarity search, because the non-local part realized by a vp-tree not only helps to increase the denoising effect, but also has merely a sublinear time complexity for search queries. Besides that, this thesis analysed the factors that have an influence on the denoising performance and computation time. They include the scale on which a PCA is performed, the balance between similarity and number of PCA samples, as well as parameter selection for each stage.

Further improvements for denoising performance can be made in future work. It can be classified into two aspects: First, the combination between the local and non-local part of a hybrid model is needed to be optimized w.r.t. search order and search range. Second, the methods based on PCA assumes that the components with the largest variance also maximize the information contained in an image, which is however not always true. Therefore, more additional information should be obtained such that the transformed data can be better separated.

A. Appendix

A.1. Notations

y	pixel of a noiseless image
z	pixel of a noisy image
n_1	number of rows in an image
n_2	number of columns in an image
w	additive white Gaussian noise
σ	standard deviation of noise distribution
p	patch size
N	number of samples in an image or number of samples in a search window
M	number of features in an image
s	patch sample of a noiseless image
f	patch feature of a noiseless image
A	patch representation of a noiseless image
$E[\ ]$	mean value
$\tilde{s}$	patch sample of a noisy image
$\tilde{f}$	feature sample of a noisy image
$\tilde{A}$	patch representation of a noisy image
cov	covariance of two vectors
Σ	covariance matrix
v	eigenvector
λ	eigenvalue
V	matrix of eigenvectors
D	diagonal matrix of eigenvalues
B	basis
proj_v	projection on a vector
T_{KOK}	threshold for "Keep or Kill" shrinkage
T_{HT}	threshold for Hard-thresholding shrinkage
η	shrinkage function
$\mathbb{1}$	indicate function
l	size of search window
δ	step size
P	patch set
m	index of central patch in a patch set
d_2	Euclidean distance
ϵ	range limit for similarity search
k	number limit for similarity search
vp	vantage point
r	radius of hypersphere
τ	threshold distance from a query point
d	distance between two points
ls	leaf size

Bibliography

[AV07] D. Arthur and S. Vassilvitskii. “k-means++: The advantages of careful seeding.” In: *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*. Society for Industrial and Applied Mathematics. 2007, pp. 1027–1035 (cit. on p. 14).

[Bae12] J. Baek. *Lecture notes in Computational Photography*. 2012 (cit. on p. 13).

[BCM05] A. Buades, B. Coll, and J.-M. Morel. “A non-local algorithm for image denoising.” In: *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*. Vol. 2. IEEE. 2005, pp. 60–65 (cit. on p. 12).

[BCM06] A. Buades, B. Coll, and J.-M. Morel. “The staircasing effect in neighborhood filters and its solution.” In: *IEEE transactions on Image Processing* 15.6 (2006), pp. 1499–1505 (cit. on p. 12).

[Ben75] J. L. Bentley. “Multidimensional binary search trees used for associative searching.” In: *Communications of the ACM* 18.9 (1975), pp. 509–517 (cit. on p. 14).

[Bru15a] A. Bruhn. *Lecture notes in Computer Vision*. 2015 (cit. on p. 21).

[Bru15b] A. Bruhn. *Lecture notes in Imaging Science*. 2015 (cit. on p. 11).

[CPT04] A. Criminisi, P. Pérez, and K. Toyama. “Region filling and object removal by exemplar-based image inpainting.” In: *Image Processing, IEEE Transactions on* 13.9 (2004), pp. 1200–1212 (cit. on p. 12).

[DFKE07] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian. “Image denoising by sparse 3-D transform-domain collaborative filtering.” In: *Image Processing, IEEE Transactions on* 16.8 (2007), pp. 2080–2095 (cit. on pp. 12, 14, 27).

[DFKE09] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian. “BM3D image denoising with shape-adaptive principal component analysis.” In: *SPARS’09-Signal Processing with Adaptive Sparse Structured Representations*. 2009 (cit. on p. 14).

[DSD+11] C.-A. Deledalle, J. Salmon, A. S. Dalalyan, et al. “Image denoising with patch based PCA: local versus global.” In: *BMVC*. 2011, pp. 1–10 (cit. on pp. 13, 14, 21–23, 25, 26).

[EA06] M. Elad and M. Aharon. “Image denoising via sparse and redundant representations over learned dictionaries.” In: *Image Processing, IEEE Transactions on* 15.12 (2006), pp. 3736–3745 (cit. on p. 14).

[EL99] A. A. Efros and T. K. Leung. “Texture synthesis by non-parametric sampling.” In: *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*. Vol. 2. IEEE. 1999, pp. 1033–1038 (cit. on p. 12).

[FCCM00] A. W.-c. Fu, P. M.-s. Chan, Y.-L. Cheung, and Y. S. Moon. “Dynamic vp-tree indexing for n-nearest neighbor search given pair-wise distances.” In: *The VLDB Journal—The International Journal on Very Large Data Bases* 9.2 (2000), pp. 154–173 (cit. on p. 15).

[Fu] P. Fu. *vp-tree implementation*. http://read.pudn.com/downloads150/sourcecode/math/650936/vptree.c__.htm. Accessed: 2016-05-19 (cit. on p. 40).

[GG13] K. Gupta and S. Gupta. “Image Denoising techniques-a review paper.” In: *International Journal of Innovative Technology and Exploring Engineering (IJITEE)* 2 (2013) (cit. on p. 11).

[GW08] R. C. Gonzalez and R. E. Woods. *Digital image processing*. 3. ed. Upper Saddle River, NJ: Pearson Prentice Hall, 2007 [erschienen] 2008, XXII, 954 S. URL: http://www.gbv.de/dms/ilmenau/toc/530915553.PDF (cit. on p. 11).

[KB06] C. Kervrann and J. Boulanger. “Optimal spatial adaptation for patch-based image denoising.” In: *Image Processing, IEEE Transactions on* 15.10 (2006), pp. 2866–2878 (cit. on p. 12).

[KZN08] N. Kumar, L. Zhang, and S. Nayar. “What is a good nearest neighbors algorithm for finding similar patches in images?” In: *Computer Vision–ECCV 2008*. Springer, 2008, pp. 364–378 (cit. on pp. 14, 15, 31, 37).

[Mac+67] J. MacQueen et al. “Some methods for classification and analysis of multivariate observations.” In: *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*. Vol. 1. 14. Oakland, CA, USA. 1967, pp. 281–297 (cit. on p. 14).

[MP03] D. D. Muresan and T. W. Parks. “Adaptive principal components and image denoising.” In: *ICIP (1)*. 2003, pp. 101–104 (cit. on p. 13).

[PKTD07] S. Paris, P. Kornprobst, J. Tumblin, and F. Durand. “A gentle introduction to bilateral filtering and its applications.” In: *ACM SIGGRAPH 2007 courses*. ACM. 2007, p. 1 (cit. on p. 12).

[Tas08] T. Tasdizen. “Principal components for non-local means image denoising.” In: *Image Processing, 2008. ICIP 2008. 15th IEEE International Conference on*. IEEE. 2008, pp. 1728–1731 (cit. on p. 13).

[Yia93] P. N. Yianilos. “Data structures and algorithms for nearest neighbor search in general metric spaces.” In: *SODA*. Vol. 93. 194. 1993, pp. 311–321 (cit. on pp. 15, 31, 34).

[ZDZS10] L. Zhang, W. Dong, D. Zhang, and G. Shi. “Two-stage image denoising by principal component analysis with local pixel grouping.” In: *Pattern Recognition* 43.4 (2010), pp. 1531–1549 (cit. on p. 13).

Erklärung

Ich versichere, diese Arbeit selbstständig verfasst zu haben. Ich habe keine anderen als die angegebenen Quellen benutzt und alle wörtlich oder sinngemäß aus anderen Werken übernommene Aussagen als solche gekennzeichnet. Weder diese Arbeit noch wesentliche Teile daraus waren bisher Gegenstand eines anderen Prüfungsverfahrens. Ich habe diese Arbeit bisher weder teilweise noch vollständig veröffentlicht. Das elektronische Exemplar stimmt mit allen eingereichten Exemplaren überein.

Ort, Datum, Unterschrift