Published in Image Processing On Line on 2021–09–21.
Submitted on 2021–11–05, accepted on 2021–09–08.
ISSN 2105–1232 © 2021 IPOL & the authors CC–BY–NC–SA
This article is available online with supplementary materials,
software, datasets and online demo at
https://doi.org/10.5201/ipol.2021.325

Exploring Patch Similarity in an Image

Jose-Luis Lisani[1], Jean-Michel Morel[2]

[1] Universitat Illes Balears, Spain (joseluis.lisani@uib.es)
[2] Université Paris-Saclay, CMLA (UMR CNRS 8536), France (moreljeanmichel@gmail.com)

Communicated by Gregory Randall *Demo edited by* Jose-Luis Lisani

Abstract

This article describes an experimental procedure to analyze (and verify) the self-similarity concept in natural images and to explore the Gaussianity of groups of similar patches extracted from a single image. The self-similarity assumption means that most image patches of a sufficient size are repeated, of course not identically, but with small variations. The procedure proposed in this paper, and implemented in the accompanying online demo, permits to explore and visualize these clusters of similar patches in a given image. Thanks to it, a user can select a patch in an image, group all patches similar to it up to a translation, or to an isometry, apply PCA to the group, make visual tests about the Gaussianity of the set of patches, and finally apply EM to the set to see if it is a mixture of Gaussians.

Source Code

The reviewed and documented source code and an online demo are available at the web page of this article[1]. Compilation and usage instructions are included in the README.txt file of the archive.

Keywords: patches; sparsity; Gaussianity; Gaussian mixture; PCA; eigenvectors

1 Introduction

Most digital images are obtained as a snapshot of a natural scene or of a human life scene. Because each scene is a snapshot of a local part of the visual surrounding, it generally depicts groups of objects that have the same origin or the same cause. Also, each object is often self-coherent and therefore also shows similar parts. This image self-similarity is generally expressed by the fact that small image square blocks, which we shall call *patches* are repeated in a given image. Images are actually often replete with self-similar structures such as periodic patterns, edges, regular textures and smooth regions. Thus, exploring locally the *patch space* of the image amounts to explore a clustered space where each cluster groups similar patches.

[1] https://doi.org/10.5201/ipol.2021.325

This intuition has been used in most image restoration and analysis algorithms. There are so many articles assuming image self-similarity that it is impossible even to cite a balanced sample of them. As they are now called, "patch based" methods, or "non-local" methods are prominent. We are limited to cite a few, to illustrate the diversity of applications of the supposed self-similarity. It is used for the purpose of denoising [8, 9, 37, 17, 7, 11, 18, 15, 22, 44, 61, 60, 43], sometimes actually analyzing patches by PCA [39, 40, 55, 12, 64, 47, 57]. Repeated patches in an image are used for noise estimation [51, 52], for a random exploration of patches through their similarity [58], for fast image matching [3]. The assumed clustered structure of the set of patches in an image has led to many algorithms building patch dictionaries of images or of groups of images, thus yielding a sparse image representation [41, 42, 46, 1, 27]. Influential texture synthesis or restoration algorithms have been based on the self-similarity assumption [26, 6, 50]. The patch self-similarity is also invoked in image deblurring [38, 36, 45, 65], image segmentation [32, 4], demosaicking [10], stereovision [34], superresolution [25, 54, 19], inpainting [2, 62], and SAR imagery [23, 21]. The representation of the set of patches of an image is often made through a Gaussian mixture computed by the Expectation Maximization algorithm (EM) [63, 67]. Convolutional neural networks are also exploring local receptive fields that can be assimilated to patches, and there is an active search of nonlocal or self-similarity based CNNs [20, 35, 56].

This article describes an experimental procedure that allows exploring the structure of a set of similar patches. It provides several basic data analysis and visualization tools to do so. Indeed, the self-similarity can be explored by taking image patches of any size between 2×2 and 16×16. The self-similarity assumption means that most patches are repeated, of course not identically, but with small variations. Our goal is to have a microscope directed at a group of similar patches. So we shall be looking at the set of patches locally and in a single image. We have no claim at deducing from this exploration a global structure for the set of all patches. The proposed microscope amounts to selecting a patch in an image, then exploring the set of its similar patches, first by a visualization, then by analyzing its structure through PCA [48] or by a Gaussian mixture model [24], as suggested by several papers mentioned above.

Note that assuming that patches are repeated with variations is compatible with several structure assumptions for the set of patches. They might be just clustered, but many papers have claimed that patches belong to low dimensional manifolds [49]. This manifold has even been proposed to have the structure of a Klein bottle [14]. This observation can be considered as an observation artifact, which we shall try to avoid. Indeed, following the argumentation of the authors, most self-similar structures in images are actually edges. Consider the simplest possible non trivial edge image, namely a Heaviside function. Patches in such an image are of two kinds: the patches that do not meet the edge are constant patches and form two perfect clusters. The patches that meet the edge depend on two parameters: the angle $\theta \in [-\frac{\pi}{2}, \frac{\pi}{2}]$ between their most horizontal side and the edge, and the signed distance $\delta \in [-d, d]$ of the center of the patch to the edge. As pointed out by the authors of [14], this 2-parameter set of patches forms a Klein bottle. This structure is therefore inherent to the patch extraction procedure. It may be observed for example if we take all patches of an image containing a black circle on uniform background. Yet, when exploring locally similar patches on an edge, for example, there is no reason to observe this Klein bottle structure. Another view on patch structure is developed in the sparsity framework: the assumption is that patches can be expressed as linear combinations of elementary patches taken in a well chosen dictionary [27]. It was shown in [42] that groups of similar patches indeed are sparse in an adapted dictionary.

This assumption does not contradict the assumption that patches are grouped in Gaussian clusters. Indeed, some of these Gaussian clusters can indeed have a low effective dimension and are therefore sparse (up to some noise residual). One of the goals of the online demo is precisely to verify experimentally that this is the case for patches located on edges. Patches in textures, instead, live in high dimension, as measured by PCA. This will be again easily verified by analyzing the set of

patches similar to a patch selected in a textured part of the image.

The online demo helps analyzing the local variations of patches similar to a seed patch. Such variations may be assumed to be random. So we shall analyse the set of similar patches as a random vector. It is therefore natural to test first the simplest models for random vectors. We limit ourselves to the Gaussian model, which is the simplest one, and to the Gaussian mixture model, which is the next simplest one. Both are classic in image processing as we saw in the bibliographical analysis.

The Gaussian mixture model (GMM), however, is generally used for a larger set of patches, for example all patches of a given image [63], or all patches of a set of images. In that case, the number of Gaussians amounts to several hundreds [67]. In contrast, since we are looking at a single cluster, the online demo applies the expectation minimization algorithm (EM) with the default assumption of only two Gaussians. The goal of this application is to verify visually if the cluster of patches similar to a seed is itself dividable into several clusters. The Gaussian model (which amounts to computing a PCA of the sample patches) is compared to the two Gaussian models of the mixture in several ways: the locations of the patch centers are colored in two different colors, the eigenvectors, eigenvalue spectra, and ten typical samples of both Gaussians are displayed for comparison.

This allows the user to decide first if the choice of N, the number of similar patches, was too large, or too small, and sometimes to relate the Gaussian mixture to some spatial variation. Of course it can also lead to reject one of the models (Gaussian versus GM) in favor of the other, or to reject both. We can anticipate a few empirical conclusions, that users will easily verify in a few clicks on almost any image:

- for most patches, the number of similar patches often goes up to several hundreds;
- the right number can be found by looking at the variations of the "closest to closest" patches and their histogram, in the first section of the online demo;
- clusters of Canny patches on edges are low dimensional and therefore "sparse"; this is checked by looking at the PCA spectrum;
- clusters of Canny patches situated on edges are not Gaussian;
- clusters of textured patches are high dimensional. Furthermore, clusters in natural textures fit well to a Gaussian model. Clusters of patches extracted from a structured manmade texture are intermediate; they are high dimensional, but no longer clearly Gaussian.

The best method to understand the interest of the proposed exploration is to start playing with the online demo. Section 2 describes the visual interaction of the online demo, that helps performing the analysis of the set of patches similar to a selected one. Section 3 analyses in detail three groups of patches extracted from an image. The following sections give all the details on the algorithms to extract the patches, normalize them, group them, and analyze their groups. Section 4 is dedicated to the various procedures to extract similar patches to a given one, up to a choice of patch normalization. It details this normalization, which can make patch comparison invariant to rotations and affine changes of the gray level scale. Section 5 details the algorithms of the main tools to analyze and visualize a group of similar patches: PCA (analyzing the group as samples of a single Gaussian vector) and the application of Expectation Minimization to analyse the group as a Gaussian mixture by the Expectation Maximization algorithm. Normality tests are performed on the components of the PCA.

2 Description of the Online Demo

The positions of the closest patches to a seed patch. Given a patch that has been selected as seed, the algorithm looks at its closest (or most similar) patches for the l^2 distance in a wide enough

image neighborhood. Indeed, our postulate is that the chances of finding similar patches to a given one are much higher nearby, in a spatial neighborhood of the seed. The online demo provides three choices: the user can pick an arbitrary patch for seed (with the "all points" option) and explore all patches similar to it. Or she can be guided to play only among patches with an automatically computed center, either centered on "Canny points" or edges, or on "Harris points" or corners. It may indeed be more sensible to pick patches that have been centered on a relevant image structure, such as edges or corners and to force its similar patches to be also centered in the same way. Forcing similar patches to be centered on an edge excludes the spatial neighbor patches that are still close for the Euclidean distance but are not centered on an edge. Furthermore, picking, for example, Canny edges, leads to a more reliable normalization by orientation, because the gradient orientation is well defined at such points.

Figure 1: Selecting the center of the reference patch by clicking on the image.

By "centered" we mean that the seed patch is centered on an image edge, namely at a Canny point. Thus, it is somewhat centered on a visible image structure, and will be compared with patches of the same kind. By "normalized" we mean that patches will be considered similar up to an arbitrary rotation, as sometimes required in image analysis [66, 28]. This requirement is justified for example if we wish to consider all the patches centered on the boundary of a disk as similar to each other. The distance between patches is made rotation invariant by extracting patches with a vertical direction equal to the direction of the gradient at their center. Such patches have a "normalized orientation".

Our experimental procedure therefore is:

- select or upload an image;
- click on a point in the image, which will be marked in with a red dot (see Figure 1);
- choose the size of the local neighborhood (sub-window size, default: 100) and the type of information that shall be extracted from each pixel (RGB values or only intensity level - computed as average of RGB values-)
- select the patch size (default: 8×8), the extraction method (default: patches centered at Canny points), the normalization options (default: only normalize orientation), and the analysis methods (default: PCA and GM2, a Gaussian mixture with two components);
- decide to normalize or not patch mean or variance;
- fix the number of closest patches to the central patch to be analyzed and displayed (default is 20 but higher values are recommended, up to several hundreds);
- click on the "run" button to start the analysis.

Figure 2 displays the main panel of the online demo with all the available options.

Then the analysis of the patches similar to the seed patches proceeds as follows:

- the patches extracted from the image's sub-window around the central pixel are compared (by l^2 distance) to the central patch. These patches are sorted in increasing order of distance to the central patch, forming the list of closest patches;

Figure 2: Main control panel for patch extraction.

- for each one of the selected closest patches its own list of closest patches is also computed (this is shown as the "closest to closest" list).

The online demo displays as results several illustrations of the closest patches to a given patch, where the number of closest patches displayed is the one specified by the user under *Number of closest patches*. More precisely, the number of displayed closest patches is the minimum of two values:

- the number of closest patches selected by the user;
- the total number of eligible patches in the selected sub-image: if the default option "Canny points" has been selected, it may happen that their number is indeed smaller.

The online demo displays the centers of the closest patches to respectively:

- the central patch;
- the 1st closest patch;
- the 2nd closest patch;
- the 3rd closest patch;
- the third-from-last closest patch;
- the second-to-last closest patch;
- the last closest patch.

In each case the centers of the reference patches are displayed in blue and the centers of their closest patches in red.

Comparing these groups of similar patches allows to verify if the group of selected similar patches is complete or at least self-coherent: if it is, the set of patches closest to the first patch should not differ too much from the set of patches closest to the "last closest" patch. Figure 3 displays such a comparison between the set of 226 patch centers similar to a seed patch 1 (left) and the set of centers of the 226 patches most similar to the least similar patch (with rank 226) to the seed patch 1. Both sets are similar, but not identical, which proves that the user might still extend the cluster without losing its coherence. Figure 4 instead displays such a comparison between the set of 137 patch centers similar to a seed patch 1 (left) and the set of centers of the 137 patches most similar to the least similar patch (with rank 137) to the seed patch 1. Both sets are nearly identical, which proves that we see here a complete cluster.

In addition, this section of the demo displays three 2D histograms:

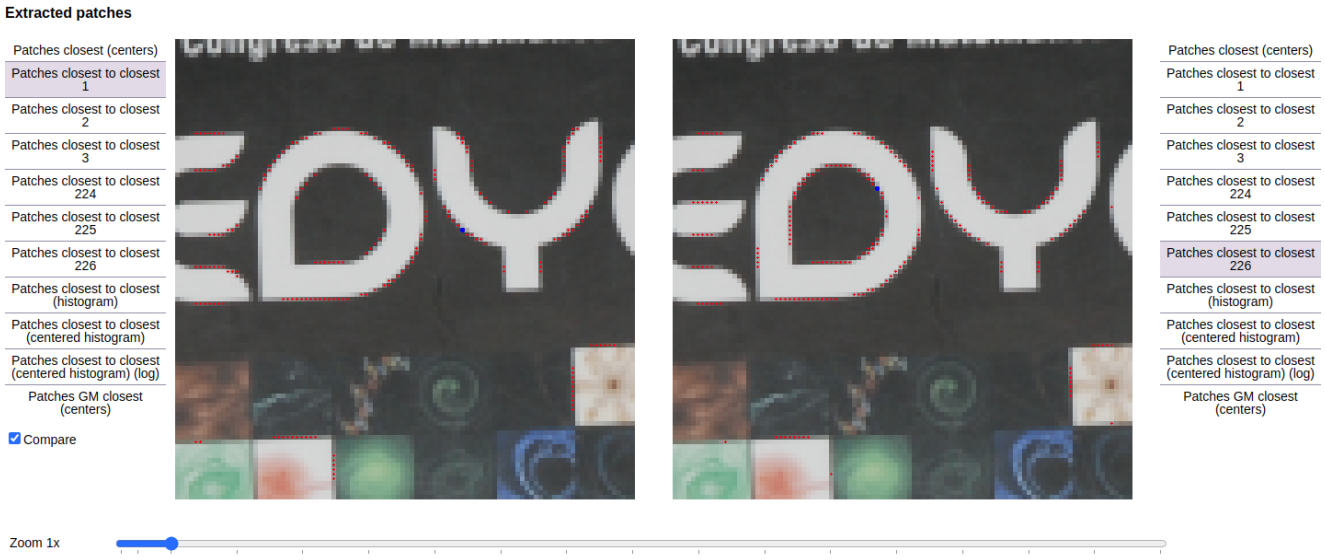

Figure 3: Left: 226 centers (in red) of extracted patches that are similar to the seed patch (with center colored in blue) up to a rotation, selected with the Canny option. Right: the 226 centers of patches that are closest to the 226th closest patch of the seed patch. They include most of the preceding patches, thus illustrating the existence of a stable cluster of similar patches.

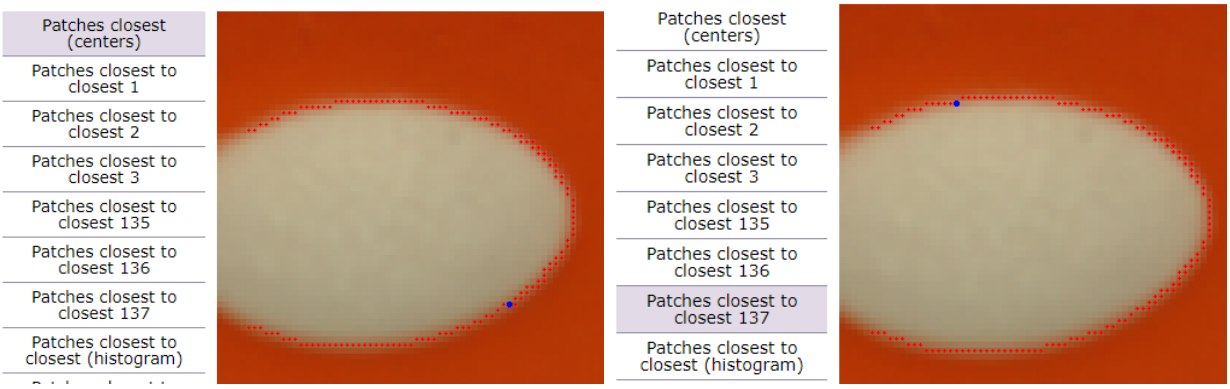

Figure 4: Comparison between the set of 137 patch centers similar to a seed patch 1 (left in blue) and the set of centers of the 137 patches most similar to the least similar patch (right, in blue with rank 137) to the seed patch 1. Both sets are nearly identical, which proves that we see here a coherent cluster.

- the histogram of the positions of the centers of all closest patches to the closest patches. More precisely, let N be the number of closest patches $p_1, \cdots p_N$ to the seed patch p_0. Then the N closest patches p_{ij} to p_i are found. The histogram of the patch centers p_{ij} therefore has $N(N+1)$ samples.

- the histogram of positions of the centers of closest to closest patches p_{ij}, after normalization of their coordinates. This normalization consists, for each of the N patches p_i in turn, to place it in a fixed reference system centered at the patch's center and whose vertical axis is oriented in the direction of the gradient at center of the patch p_i. Then the centers of all of its N similar patches p_{ij} are placed in this common reference system.

- the previous histogram displayed in logarithmic scale. All views are designed to explore the extent of the cluster, its completeness and its internal structure. Indeed the second histogram presents a synopsis of all planar transforms sending patches to their similar patches.

Exploring the model of the patches similar to a seed patch. The analysis of the similar patches to our seed patch is pursued with an analysis of their structure, with the assumption that it might be represented by a single Gaussian (hence applying PCA to the set), or by a Gaussian mixture. The online demo therefore performs a PCA and a Gaussian mixture analysis (by the EM algorithm, with the number of Gaussians selected by the user - from 2 up to 10 -) of the closest patches and shows:

- an image (*Patches GM closest (centers)*), where the centers of the closest patches are represented in different colors depending on the Gaussian component they belong to;
- the seed patch, its top ten closest patches and, for each Gaussian in the mixture, the ten patches with highest probability in that Gaussian. (see Figure 5);

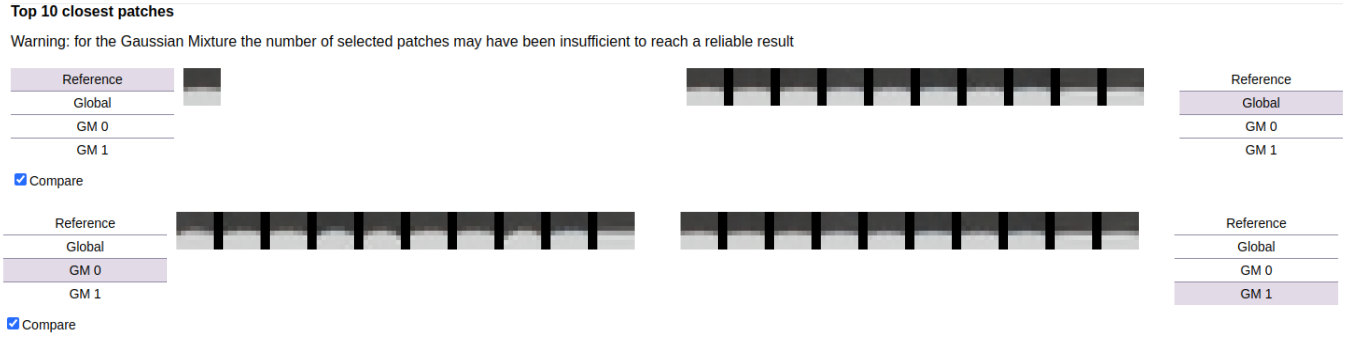

Figure 5: Above: seed patch and its top ten closest patches. Below, the 10 patches with highest probability of each component of the Gaussian mixture.

- the result of the PCA analysis: basis of eigenvectors of the Gaussian distribution fitted to the set of closest patches, its spectrum in decreasing order (namely its eigenvalues in decreasing order), and simulated random patches drawn from that distribution;
- the result of the Gaussian mixture analysis: for each Gaussian, its basis of eigenvectors and its spectrum;
- 3D views of different projections of the set of patches onto the space generated by three vectors of the PCA basis. These vectors are chosen by the user in the main control panel of the demo. Different colors distinguish the patches belonging to different Gaussians in the Gaussian mixture;
- the histograms of the coefficients of the patches on the basis vectors selected by the user, and their comparison with the Gaussian having same mean and variance.

We now give more detail and explain the visualization. Our local representation of the patch space aims at exploring the assumption that patches are clustered in groups of similar patches. It culminates with the synthetic representation of the maximum likelihood estimate of a Gaussian model for a given set of similar patches. As we mentioned, the Gaussian model is obtained by PCA or by EM, as a component of a Gaussian mixture. This representation is illustrated in Figure 6.

In this figure, the seed patch of a set of similar patches is a 8×8 color patch centered at a Canny point on a straight edge. Its dimension is $192 = 3 \times 64$. The displayed orthonormal basis of eigenvectors of PCA applied to the set of similar patches has therefore 192 elements. These patches are displayed from left to right and from to bottom in decreasing order of magnitude of their corresponding eigenvalues. On the right, these magnitudes are displayed in the same order, hence demonstrating that this particular set of patches is sparse, namely has only a few non-negligible coefficients. Eight simulated patches of the Gaussian vector associated with the PCA are also displayed. These samples

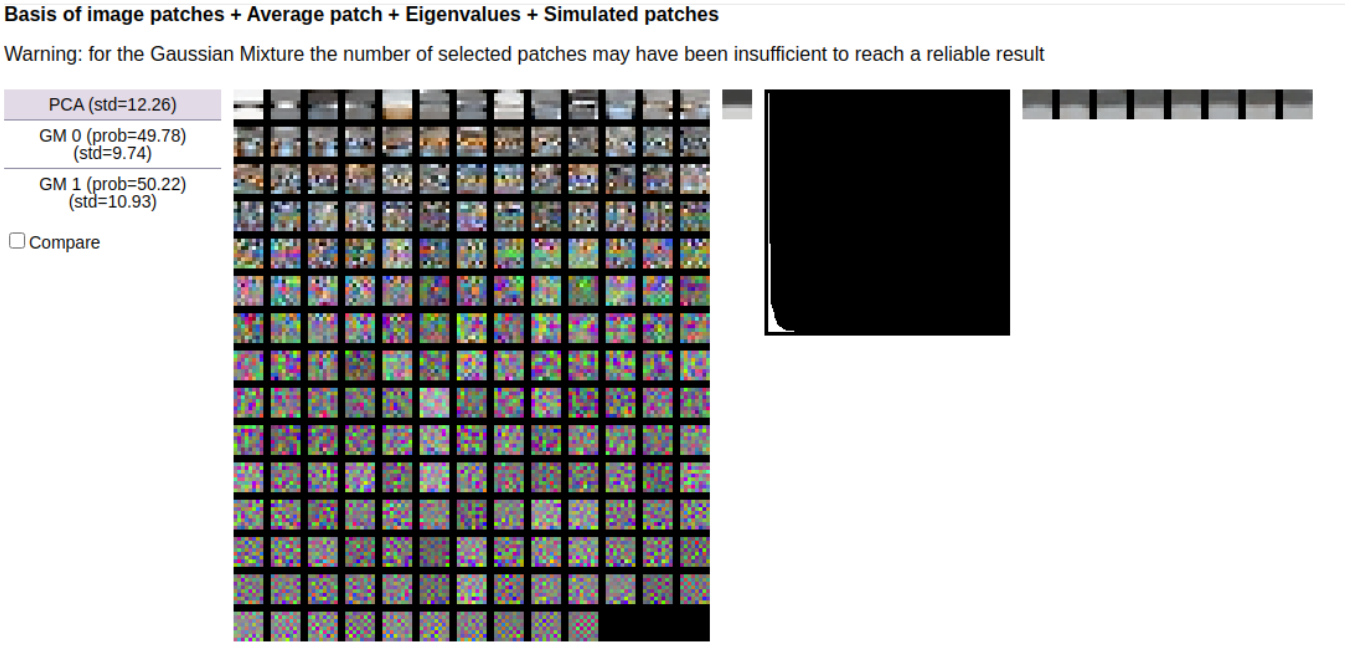

Figure 6: Representing a Gaussian modeling a group of similar patches. The seed patch was a 8×8 color patch centered at a Canny point on a straight edge, and reoriented so that the direction of the edge is horizontal. Its dimension is $192 = 3 \times 64$. The displayed orthonormal basis of eigenvectors of PCA applied to the set of similar patches has therefore 192 elements. These patches are displayed from left to right and from top to bottom in decreasing order of magnitude of their corresponding eigenvalues. On the right, these magnitudes are displayed in the same order, hence demonstrating that this particular set of patches is sparse, namely has only a few non-negligible coefficients. Eight simulated patches of the Gaussian vector associated with the PCA are also displayed. These samples are built by simulating their independent Gaussian random coordinates on the PCA basis, each with a variance equal to the corresponding eigenvalue of the PCA. They represent generic patches of this Gaussian, and should resemble the seed patch. The mean vector of the patch cluster, which is a denoised version of the seed patch, is also displayed on the right of the basis.

have independent Gaussian random coordinates on the PCA basis, each with a variance equal to the corresponding eigenvalue of the PCA. They represent generic patches of this Gaussian, and of course resemble the seed patch. The mean vector of the patch cluster, which is a denoised version of the seed patch, is also displayed on the right of the basis. Only the first ten eigenvalues matter in this decomposition, so the effective basis of this cluster has only about ten dimensions. An examination of the first row of the basis confirms that the basis indeed is adapted to such an edge patch and its similar patches.

However, the dimension of such a cluster of patches, even if low, remains considerable and hardly accessible to human perception. At best we can visualize three dimensions by simulating a 3D volume. This is done by selecting three dimensions and by projecting the cluster on these three dimensions, so that the coordinates of patches can be visualized in a cube. An example of this 3D visualization is given in Figure 7. Each patch is represented with a different color, depending on the Gaussian of the mixture it belongs to.

The histograms of projections onto the chosen coordinate axes are shown, together with a Gaussian distribution with the same mean and variance included for comparison. See Figure 8.

A χ-squared test is performed to evaluate the normality of each PCA projection. Experience shows that such a test is very often negative, thus negating the normality. How strongly this normality is negated is given by the p-value.

In addition, our visualization of the components of the PCA and its comparison with a Gaussian with the same parameters gives hints about the normality or about the cause of non-normality. There are at least two observation biases that may explain this non-normality. First, one generally observes

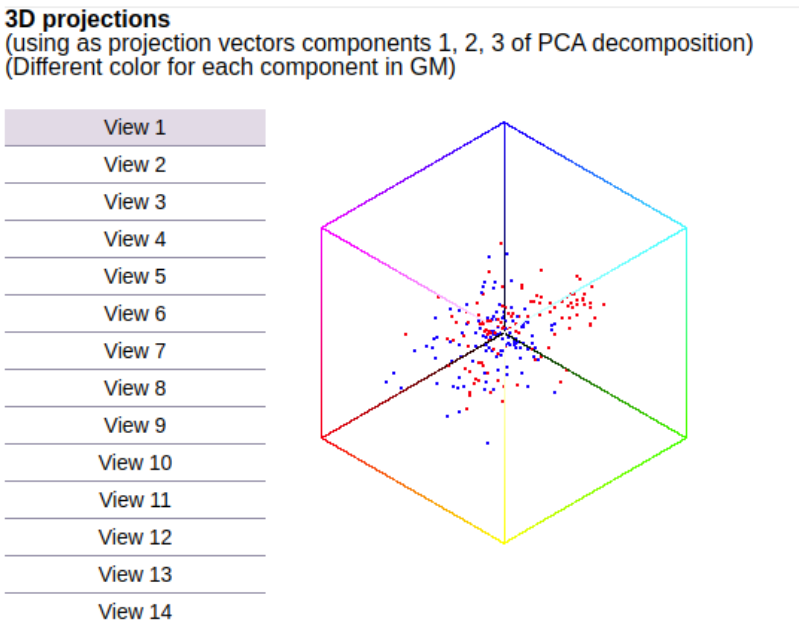

Figure 7: The online demo allows to select three dimensions on which to project the patch cluster. Each dot corresponds to the 3D coordinates of a patch. Blue and red distinguish the patches belonging to the first or the second component of a Gaussian mixture. By moving a pointer on the column of tags on the left, one can animate the cube and see various views of the cluster. This is a clear example where the Gaussian mixture is not dividing the cluster into sub-clusters. So a single Gaussian seems sufficient to describe this cluster.

the presence of several subgroups of similar patches due to the grid sampling, which quantizes the set of similar patches. For example if a patch is on a vertical edge, a first group of similar patches will be perfectly aligned, but a second group will be centered in the column on the left, and a third group on the right. Of course, there are other reasons for the non-normality. A simple one is the fact that the set of the k most similar patches to a given patch is not Gaussian if k is too small, even if these patches indeed obey the same Gaussian law!

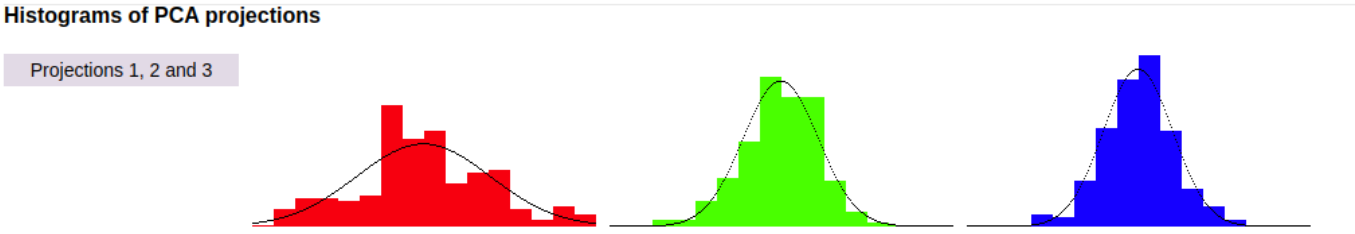

Figure 8: Histogram of PCA projections on the first three components and their fit to Gaussians. See text for details.

Normalization of color patches. When the normalization options are chosen in the control panel of the demo, mean and variance normalization of color patches is performed based on their intensity (the average of the three color channels). More precisely:

- the intensity value I (i.e. average of R, G and B) is computed for each pixel in the patch;
- a new intensity I' is computed as $I' = aI + b$, with a and b such that $\text{Mean}(I') = 0$ and $\text{Var}(I') = 1$;

- the normalized values R', G', B' are computed as $R' = R\frac{I'}{I}$, $G' = G\frac{I'}{I}$, $B' = B\frac{I'}{I}$. This procedure guarantees that the intensity of the normalized color patch has mean 0 and variance 1, while keeping the original $R/G/B$ ratios.

Visualization of patches. For patch visualization, the following conventions are adopted:

- original patches are shown when there is no normalization involved (neither mean or variance normalization),
- patches normalized by mean are re-normalized for display, so that the mean intensity value of the displayed patch is 128,
- mean and variance normalized patches are re-normalized for display, so that the mean intensity value of the displayed patch is 128 and its standard deviation 40.

3 A Study of Three Groups of Patches

In this section we illustrate the kind of observations that can be drawn from the application of the algorithms to a set of similar patches. Three groups of patches are extracted from the three zones of the Traffic image shown in Figure 9. One of the zones contains strong geometric edges on the city bus. The second zone is a manmade texture, the bellows of the city bus, which makes a somewhat periodic structure with variations due to its lighting and flexibility. The third zone is the foliage of a tree, a clear example of natural texture.

Figure 9: Three seed patches will be picked from each of the three zones of the Traffic image surrounded by a white square. One of the zones contains strong geometric edges on the city bus. The second zone is a manmade texture, the bellows of the city bus, which makes a somewhat periodic structure with variations due to its lighting and flexibility. The third zone is the foliage of a tree, a clear example of natural texture.

Figure 10 shows the first zone. The seed is a patch centered on a classic horizontal edge situated in the exact center of the displayed image. We show in red the centers of its 199 closest patches, that border the upper and lower part of the bus' windows. To verify that the group of patches is a sufficient sample of all really similar patches in the image, we also show the centers of the 199 patches that are closest to the last closest patch of the seed. Both sets are in good agreement, they share many patches. Finally we display the histogram of all groups of 199 patches closest to any of the initial 200. So this is the histogram of the positions of the centers of 39800 patches in the image.

Figure 10: The seed is on a classic edge situated in the center of the image. Left: the 199 closest patches. Middle: the 199 patches that are closest to the last closest. Both sets are in good agreement. Right: the histogram of all groups of 199 patches closest to any of the initial 200: it is coherent, which indicates that we do have a self-coherent group of 200 similar patches.

This histograms aligns with the upper and lower boundaries of the bus windows, which indicates that the group of 200 similar patches is a complete enough representative of these edges.

We show in Figure 11 the 199 closest patches to this edge seed. This long list of patches starts with the seed itself, followed by its closest one, and so on. One can observe that, though very similar, after the third line some of the patches have their main edge displaced upward or downward by approximately one pixel. This is due to the interference of the sampling grid, that is almost horizontal and therefore samples the similar patches with shifted grid origins. In short, sampling issues impair the self-similarity of observed patches.

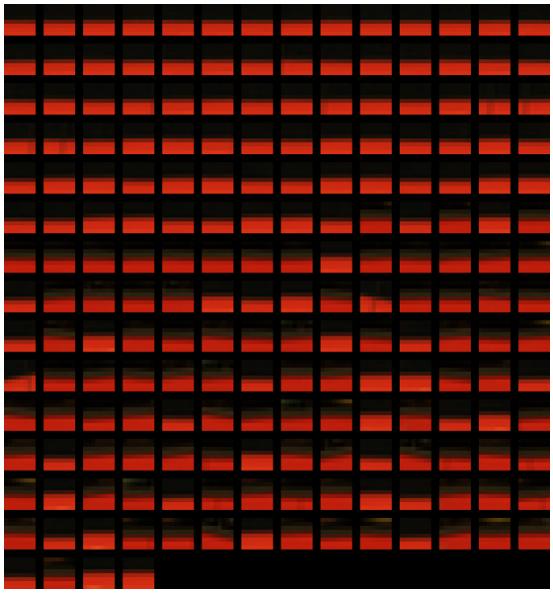

Figure 11: From left to right and from top to bottom the 199 closest patches to a seed extracted from the Traffic image. The seed patch is the first one. Sampling issues impair the self-similarity of edges: some of the edges are displaced upward or downward by one pixel.

We shall examine if this group is better represented by a single PCA or by the two Gaussians obtained by a Gaussian mixture model obtained by expectation maximization. In Figure 12 we show in blue and red the centers of the patches selected by a Gaussian mixture model with two Gaussians. One of the groups is on the edge and the other one stays farther from the edge. So the constitution

of these two groups is due to image sampling effects. We compare the mean patches of the PCA and of the Gaussians 0 and 1 of the Gaussian mixture. It appears that the mean of the second Gaussian contains an edge shifted downwards by approximately one pixel. So the GMM uncovered here two groups of patches that are similar, but different due to a sampling grid effect. On the bottom right of the figure we show the extremely sparse spectra of the three Gaussians, which are hardly more than three-dimensional. The normalized standard deviations of the global PCA and of the two Gaussians of the Gaussian mixture are respectively 10.49, 7.92 and 9.32. These numbers indicate that the two Gaussians of the Gaussian mixture are not redundant: they have different means and variances, and correspond to patches that are located on two different positions with respect to the sampling grid. It made sense to use two Gaussians instead of one, since this has led to a coherent division of the patches in two classes. These classes are rather an artifact caused by image sampling, though.

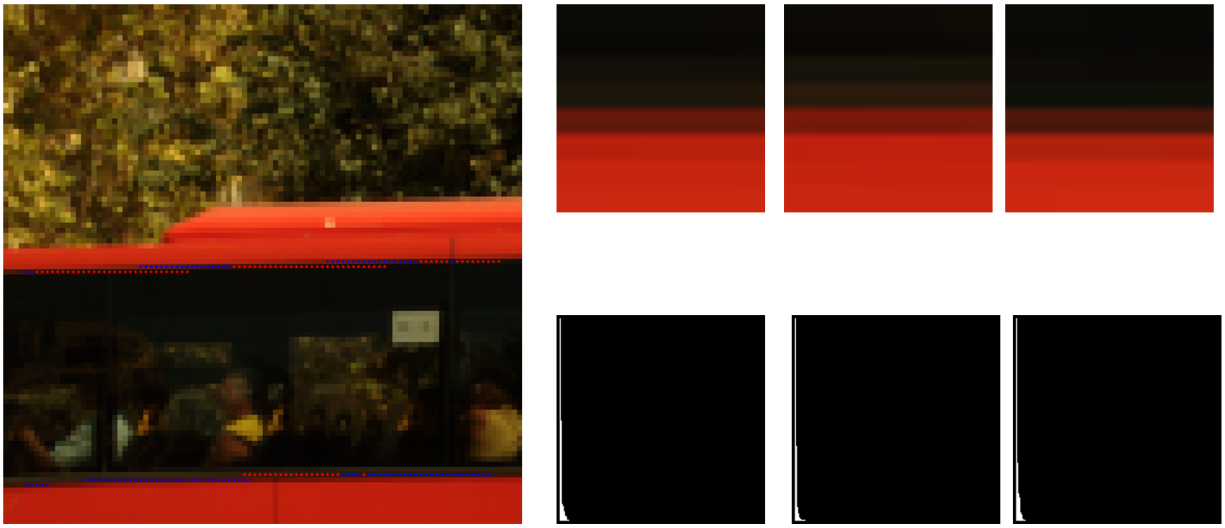

Figure 12: Left : In blue and red the centers of the patches selected by a Gaussian mixture model with two Gaussians. Top right: The mean patches of the PCA and of the Gaussians 0 and 1 of the Gaussian mixture. Bottom right: the extremely sparse spectra of the Gaussians, which are hardly more than three-dimensional.

Figure 13 shows the 192 eigenvectors of the PCA for the 199 patches most similar to the seed patch. According to the spectrum of the PCA, only the first six eigenvalues are non negligible! So the only relevant eigenvectors are the first six eigenvectors, which indeed show some geometric structure, namely a vertical oscillation. The concentration of this set of similar patches is confirmed by the ten patches sampled from this Gaussian shown on the bottom left of the figure. This is a clear example where the "sparsity" of patches is verified. Similar observations can be made on all groups of patches similar to a patch located on an image edge. PCA is often associated with the Gaussian model, because it yields the maximum likelihood estimate of the Gaussian's parameters. Yet, the fact that PCA efficiently uncovers the low dimension of the data set does not imply normality. As Figure 14 shows, this set of patches is not at all Gaussian. This figure shows on its left column, from top to bottom, the projection of the patches on dimensions 1, 2, 3, then dimensions 4, 5, 6. It is not necessary to display more dimensions. As we just saw, the PCA spectrum is almost zero for the higher dimensions. The figure shows from left to right and from top to bottom the six histograms of the 1D projections of the patches on the first six PCA dimensions. They clearly aren't Gaussian. The Anderson-Darling normality test confirms it with p-values $< 10^{-4}$ in all the cases. Not only these PCA projections are clearly non Gaussian but the first component is bi-modal, which can be attributed as we saw to a sampling grid bias. The behavior of this (sparse and non-Gaussian) set of edge patches is observed on most edge patches.

We now pass to examine a more complex set of patches, with its seed situated in the center of

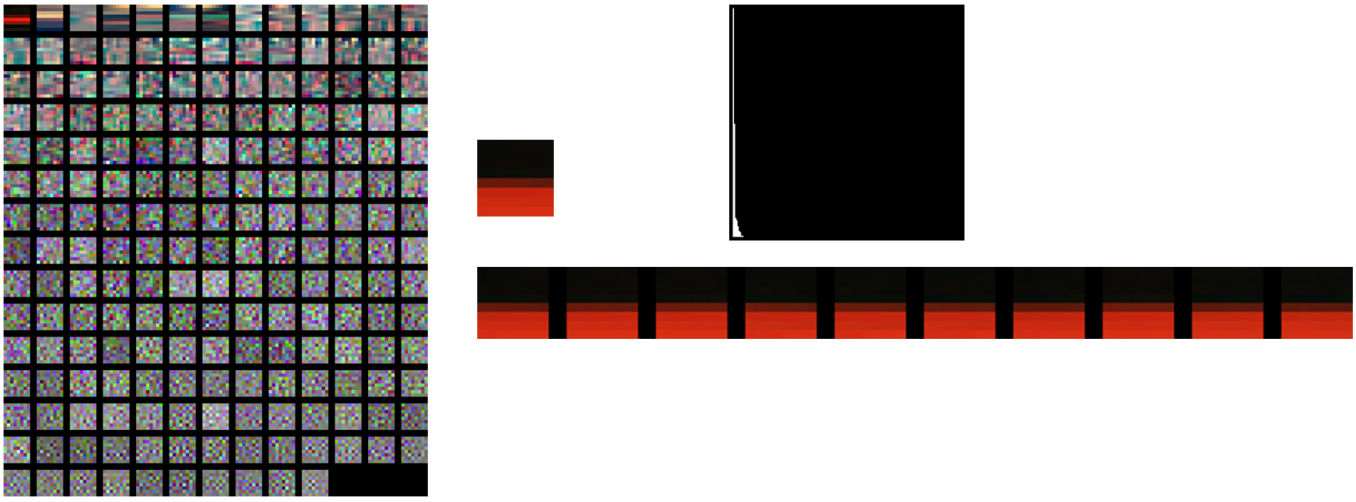

Figure 13: Left : the 192 eigenvectors of the PCA for the 199 patches most similar to the seed (top, middle). Top, right: the spectrum of the PCA with only six non-negligible eigenvalue. Bottom left: ten patches sampled from this Gaussian, very similar to the seed. This is a clear example where the "sparsity" of patches is verified.

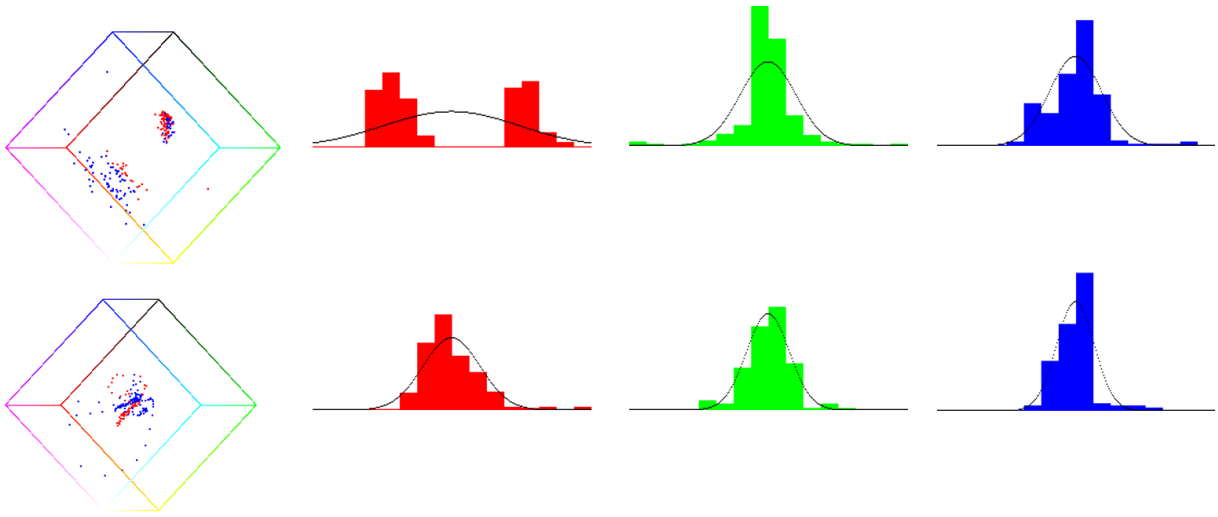

Figure 14: Left column from top to bottom: the patch projections on dimensions 1, 2, 3, then dimensions 4, 5, 6. Right: from left to right and top to bottom the histograms of the 1D projections of the patches on the six PCA dimensions.

the regular zone occupied by the bellows of a city bus. Figure 15 shows the set of closest patches to the central seed. It is large, containing not less than 399 patches. The choice of such a large number will be justified by several visual checks about the coherence of the patch set. In the middle, the figure displays the centers of the 399 patches that are closest to the last closest. Both sets are in good agreement. This is confirmed by the histogram displayed on the right of the figure. It is the histogram of all groups of 399 patches closest to any of the initial 400. It contains 159600 samples, most of which overlap, and anyway occupy a consistent region of the bellows.

We can now examine this whole set of patches and its coherence (see Figure 16). Starting from the seed patch, these 399 closest patches are displayed in the order of their distance to the seed patch, from left to right and from top to bottom. On the right of the figure, the positions of the centers of these patches are displayed on the image itself. To see if there is any rationale in the division operated by the Gaussian mixture model, we display in blue the patch centers of the first Gaussian of the GM and in red the patch centers of the second. The positions of these centers are interlaced and it is difficult to find any interpretation to this division.

Figure 15: The seed is the central patch situated in a regular man made texture. Left: the centers of the 399 closest patches. Middle: the 399 patches that are closest to the last closest. Both sets are in good agreement. Right: the histogram of all groups of 399 patches closest to any of the initial 400. Almost all are contained in the same texture.

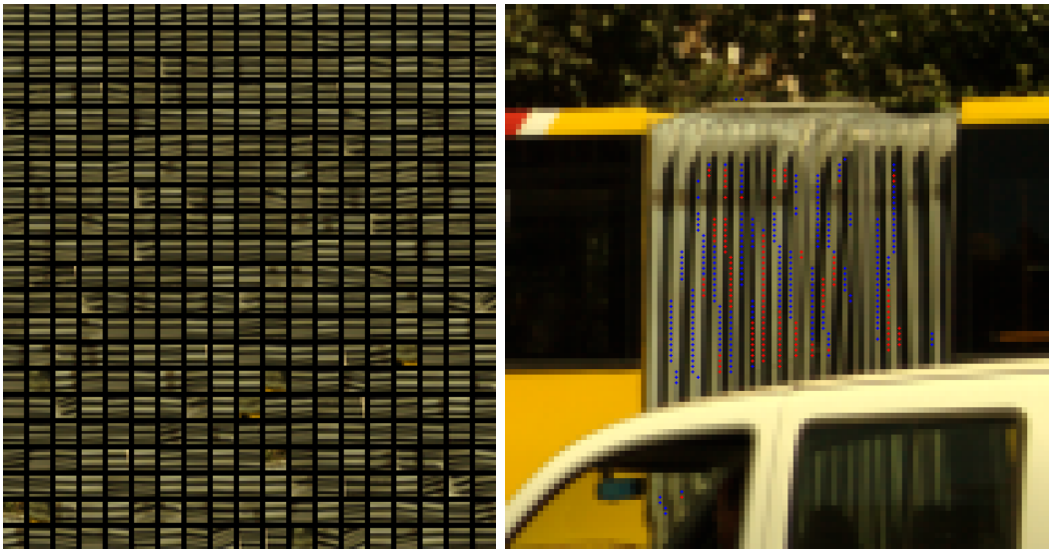

Figure 16: Left: from left to right and from top to bottom the 399 closest patches to the seed extracted from the Traffic image in the zone of the bellows of the city bus, starting from the seed. Right: the positions of these patches in the image. In blue the patch centers of first Gaussian of the GM and in red the patch centers of the second Gaussian.

This is confirmed in Figure 17 where we show all details of the PCA and GM analysis and compare the spectra and simulated patches for the three Gaussians. The figure shows the seed patch, then the spectrum of the PCA, then ten patches sampled from the PCA of the 399 patches most similar to the seed, ten patches sampled from the first Gaussian in the GM, ten patches sampled from the second Gaussian in the GM, finally the respective spectra of the PCA and of both Gaussians of the GM. The spectra of the PCA and of both Gaussians are almost identical and their respective standard deviations are 21.47, 20.41, 20.51. This leads to conclude that a single PCA is sufficient to represent the set of patches. The high dimensionality, with about 25 dimensions relevant to give an account of this group of patches, leads to surmise that the texture is not sparse at all. This is a clear example where the "sparsity" of a set of patches is not verified. Similar observations can be made on many groups of similar patches situated in structured textures.

But is it licit to talk about Gaussians? We already saw that the group of similar patches to a simple geometric edge patch is not Gaussian at all. In Figure 18 we apply the normality tests again, this time on the first nine dimensions of the PCA. There is no particular reason to examine each GM separately, as we have just seen that they are largely redundant. With the same presentation

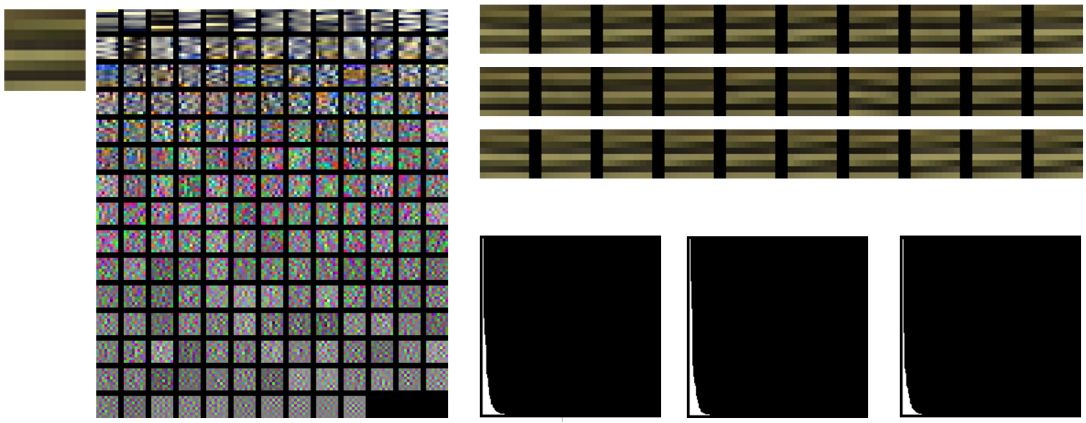

Figure 17: From left to right and top to bottom: the seed patch, the spectrum of the PCA, ten patches sampled from the PCA of the 399 patches most similar to the seed, ten patches sampled from the first Gaussian in the GM, ten patches sampled from the second Gaussian in the GM, finally the respective spectra of the PCA and of both Gaussians of the GM.

as in Figure 14, we present in the first column from top to bottom: the projection of the patches on dimensions 1, 2, 3, then dimensions 4, 5, 6, finally dimensions 7, 8, 9. Then, from left to right and top to bottom we display the nine histograms of the 1D projections of the patches on the first nine PCA dimensions. Their Anderson-Darling normality test (p-values) are respectively $< 10^{-4}$, 0.0069, $< 10^{-4}$, $< 10^{-4}$, 0.3353, 0.0002, $< 10^{-4}$, 0.0034, 0.0064. (If p-value > 0.05 the hypothesis that data follow a Gaussian distribution cannot be rejected.) This leads to reject normality for this group of similar patches in a structured texture. Similar observations can be made on most manmade textures.

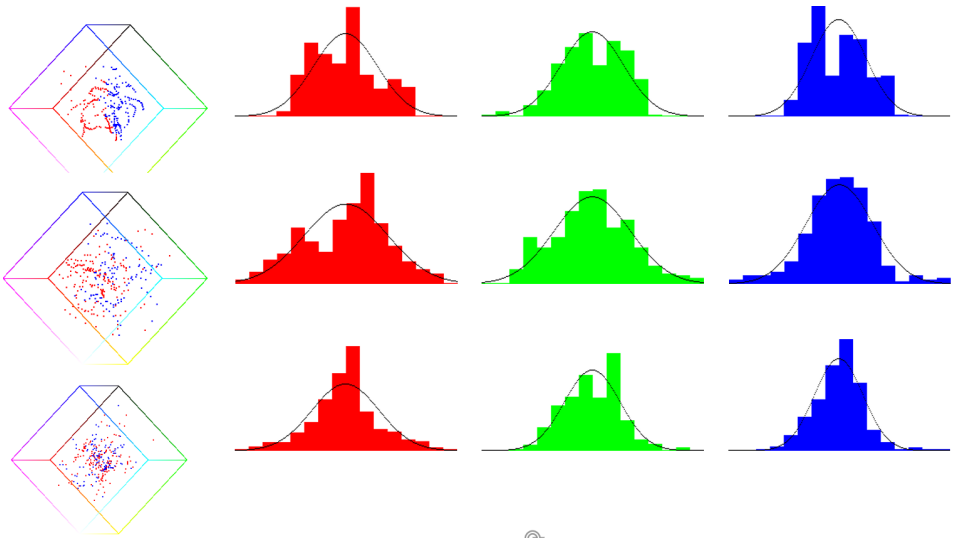

Figure 18: Left column from top to bottom: the projection of the patches on dimensions 1, 2, 3, then dimensions 4, 5, 6, finally dimensions 7, 8, 9. Right: from left to right and top to bottom the nine histograms of the 1D projections of the patches on the first nine PCA dimensions. Their Anderson-Darling normality test (p-values) are respectively $< 10^{-4}$, 0.0069, $< 10^{-4}$, $< 10^{-4}$, 0.3353, 0.0002, $< 10^{-4}$, 0.0034, 0.0064.

We now pass to the third patch type announced in Figure 9, extracted from an image region containing the foliage of a tree. Fig. 19 shows this textured region centered at the seed patch. On the left the centers of the 399 closest patches are displayed as red dots. On the middle, we display in the same way the centers of the 399 patches that are closest to the last closest. Both sets are in good agreement, which is reassuring, as it means that we have a coherent group of similar patches. This is confirmed on the right image showing the histogram of all groups of 399 patches closest to

any of the initial 400. The shape of this histogram with $399 * 400 = 159600$ samples agrees with the first two groups, which indicates that we do have a self-coherent group of 400 similar patches.

Figure 19: Analysis of the coherence of a group of similar patches. The seed is a textured patch situated in the center of the image. Left: the 399 closest patches. Middle: the 399 patches that are closest to the last closest. Right: the histogram of all groups of 399 patches closest to any of the initial 400.

Figure 20 displays all of the 399 closest patches to the seed extracted from the foliage of a tree in the Traffic image, starting from the first seed. One can observe a high variability in the similar patches. In a natural texture, the probability that two 8×8 patches strongly resemble is very low. The right image of the same figure shows the positions of these patches in the image. To explore if a classification of these patches makes sense we applied EM to find a Gaussian mixture with two components. In blue we show the patch centers of the first Gaussian and in red the patch centers of the second Gaussian. They interlace uniformly, so their division is arguably irrelevant.

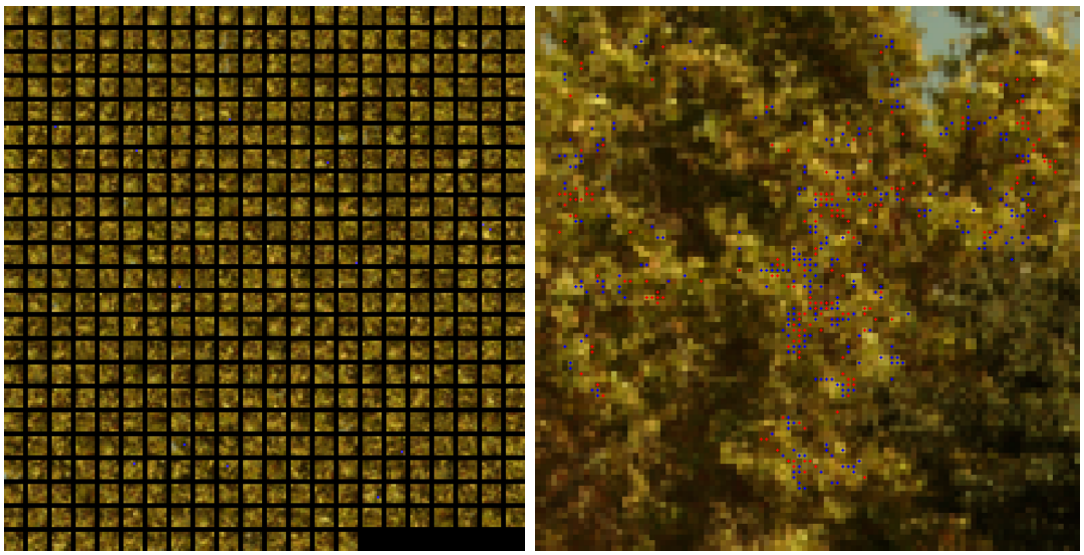

Figure 20: Left: left to right and top to bottom, the 399 closest patches to the seed extracted from the foliage of a tree in the Traffic image. The seed is the first one. Right: the positions of these patches in the image. In blue the patch centers of the first Gaussian of the GM and in red the patch centers of the second GM.

This is actually confirmed by the experiment displayed in Figure 21. On the left, one can examine the 192 eigenvectors of the PCA for the 399 8×8 patches most similar to the seed (top, middle). According to the spectrum of the PCA (bottom, middle left), more than 64 dimensions are necessary to give a precise account of this group of textured patches. It is not sparse at all. Thus, this is a

clear example where the "sparsity" of patches is not verified. Similar observations can be made on many groups of similar patches situated in unstructured textures. The other two spectra displayed on the bottom right are the the spectra of the first and second Gaussian of the GM. They are almost identical to the spectrum of the global PCA. The respective standard deviations of the PCA and of the first and second Gaussians are respectively 22.40, 21.63 and 22.11. This means that the variability of the patches is not reduced by dividing them into two different Gaussians. So the Gaussians of the GM are redundant and we can conclude that a single Gaussian is enough to model the set. We observed that the variability of appearance of the patches is high in this group. This is also reflected by the fact that their mean (top right) is blurry, compared to the seed.

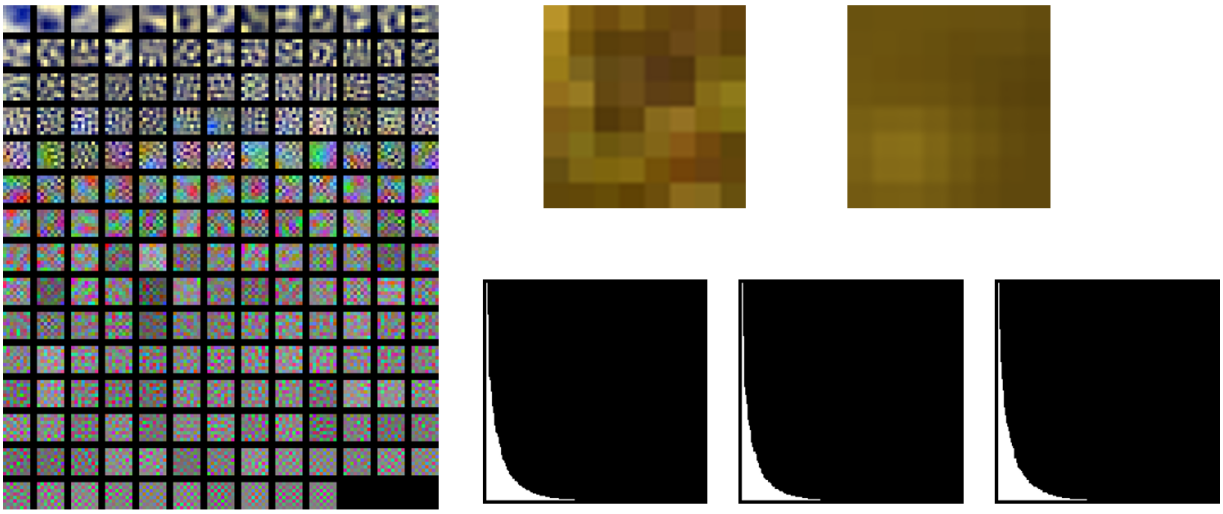

Figure 21: Left : the 192 eigenvectors of the PCA for the 399 8×8 patches most similar to the seed (top, middle). Top middle and right : the seed patch, and the mean of the PCA. Bottom right: The spectrum of the PCA and the spectra of the first and second Gaussians of the GM.

In Figure 22 we show on the left column, from top to bottom, the projection of the patches on dimensions 1, 2, 3, then dimensions 4, 5, 6, finally dimensions 7, 8, 9. Then from left to right and top to bottom we display the nine histograms of the 1D projections of the patches on the first nine PCA dimensions. Their Anderson-Darling normality test gives respectively p= 0.260, 0.015, 0.035, 0.101, 0.078, 0.052, 0.074, 0.032, 0.8808. The p-value is the probability that the hypothesis that the observed data follow a Gaussian distribution cannot be rejected. The p-values observed for this set are therefore close to Gaussian, as can also be observed visually. This leads to conclude that, in contrast with what happens with edges and structured textures, similar groups in unstructured natural textures are approximately Gaussian. Similar observations can be made on most natural textures.

4 Extraction of a Group of Similar Patches

This section describes all tools to select a patch, normalize it up to a rotation, or an affine map of the gray scale. Patches can be selected in a subclass, like the ones centered at Canny points [13] (edges) or at Harris corners [33]. So we also describe how to compute the gradient and then the Canny or Harris points which will be the center of a patch. We also describe the computation of the structure tensor [30], which yields a robust way to associate an intrinsic orientation around each point. Once the orientation at a point has been fixed, the patch centered at this point with sides parallel or orthogonal to the local orientation is sampled by interpolation so we also describe this interpolation and sampling operation. The main algorithm for patches extraction is summarized in

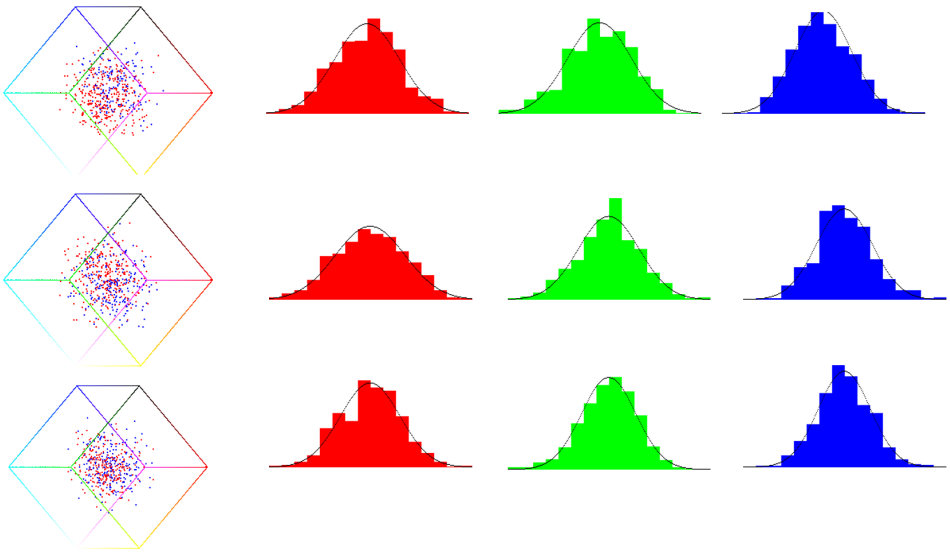

Figure 22: Left column from top to bottom: the projection of the patches on dimensions 1, 2, 3, then dimensions 4, 5, 6, finally dimensions 7, 8, 9. Right: from left to right and top to bottom the nine histograms of the 1D projections of the patches on first nine PCA dimensions. Anderson-Darling normality p-values: p= 0.260, 0.015, 0.035, 0.101, 0.078, 0.052, 0.074, 0.032, 0.8808.

Algorithm 1. This algorithm lists first all the options of the demo. Then it specifies how the set of patches, which will be the object of the demo, is being computed, depending on the chosen options. The output of Algorithm 1 is the set of patches extracted from the image that will thereafter be searched to find the N closest patches to a seed patch selected by the user. In short, Algorithm 1 is a synopsis of the preparation of the set of patches which will be explored. The multiple options of the algorithm are further described in more detail in the following sections.

4.1 Gradient Computation

Let $u(i,j)$, $i = 0, \dots w-1$, $j = 0, \dots, h-1$, be a digital image with values in the range $[0, 255]$. The *raw* gradient vectors $\boldsymbol{G} = (G_x, G_y)$ are computed using one of the following formulas:

- *Sobel gradient*[2]. The gradient at point (i,j) is computed by the following convolutions

$$G_x(i,j) = \frac{1}{4}\begin{pmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{pmatrix} * u(i,j) \qquad G_y(i,j) = \frac{1}{4}\begin{pmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & +2 & +1 \end{pmatrix} * u(i,j). \tag{1}$$

- *Scharr gradient*[2]. The gradient at point (i,j) is computed by the following convolutions

$$G_x(i,j) = \frac{1}{16}\begin{pmatrix} -3 & 0 & +3 \\ -10 & 0 & +10 \\ -3 & 0 & +3 \end{pmatrix} * u(i,j) \qquad G_y(i,j) = \frac{1}{16}\begin{pmatrix} -3 & -10 & -3 \\ 0 & 0 & 0 \\ +3 & +10 & +3 \end{pmatrix} * u(i,j). \tag{2}$$

- *Bilinear gradient* (gradient of the bilinear interpolate of u). The gradient at point $(i{+}0.5, j{+}0.5)$

[2]Sobel operator, Wikipedia,http://en.wikipedia.org/wiki/Sobel_operator (as of Oct. 18, 2012, 13:00 GMT)

is

$$G_x(i+0.5, j+0.5) = \frac{(u(i+1,j) + u(i+1,j+1)) - (u(i,j) + u(i,j+1))}{2},$$
$$G_y(i+0.5, j+0.5) = \frac{(u(i,j+1) + u(i+1,j+1)) - (u(i,j) + u(i+1,j))}{2}. \tag{3}$$

Algorithm 1: Main algorithm for patches extraction

Input : u gray scale or color image

1 **Parameters** (defaults in **black**):

- *color* flag: if set (and the image has 3 channels), then process 3 channel patches, else process patches with the intensity component
- patch size (κ, consider square patches) ($\boldsymbol{\kappa = 8}$)
- normalization options:
  - location normalization
    (method for selection of patch centers): *all*, ***Canny*** or *Harris*
  - orientation normalization: *nOrientation* flag (**active**)
  - average value normalization:*nMean* flag (**active**)
  - variance normalization: *nVariance* flag (**active**)
- gradient computation options
  (only needed if *Canny* or *Harris* are selected or if *nOrientation* flag is active):
  - gradient computation method: *Sobel*, ***Scharr***, *bilinear*
  - gradient refinement: *none*, ***smooth***
- interpolation options (for patches sampling): *nearest*, *bilinear* [31], ***cubic B-spline*** [59].

Output : patches data (set of vectors of dimension $d \times \kappa \times \kappa$, where $d = 1$ if input image is gray or *color* flag is set to *false*, else $d = 3$)

2 **Basic algorithm:**

- select the centers of the patches (use all pixels or apply Algorithms 3 or 4)
- sample an image patch of size $d \times \kappa \times \kappa$ around each center (Algorithm 5)

The values of $|G_x|$ and $|G_y|$ are in the range $[0, 255]$. Note that $\boldsymbol{G}$ is a discrete gradient field and the set of discrete points on which it is defined is either of the form $\{(i,j) : (i,j) \in [0,w) \times [0,h) \subset \mathbb{Z}^2\}$ (Sobel/Scharr formulas) or $\{(i+0.5, j+0.5) : (i,j) \in [0,w) \times [0,h) \subset \mathbb{Z}^2\}$ (bilinear formula). For simplicity, in the following, by $\boldsymbol{G}(i,j)$ we will denote either the gradient at (i,j) or $(i+0.5, j+0.5)$, the exact meaning depending on the method used for its computation.

It is possible to refine the value of the gradient at a given point by using information of the gradients at neighbor points. The goal is to obtain a smoother gradient field.

Gradient information around a point is summarized by the *structure tensor*, a 2×2 matrix computed as described in Algorithm 2.

This algorithm can be understood as the computation of the image's dominant direction around (i,j). Indeed, $M(i,j)$ is the linear projector on the line directed by the gradient at (i,j), and its first eigenvector is precisely this gradient, up to its sign. But $M(i,j)$ is positive, so an average of such

Algorithm 2: Computation of structure tensors

Input : $\boldsymbol{G}$ (discrete gradient field), r (radius of neighborhood)
Output : M_S structure tensor at each point

foreach *discrete point* (i,j) **do**

define

$$M(i,j) = \begin{pmatrix} G_x^2(i,j) & G_x(i,j)G_y(i,j) \\ G_x(i,j)G_y(i,j) & G_y^2(i,j) \end{pmatrix}$$

compute

$$M_S(i,j) = \sum_{(i',j')\in D_r} M(i',j')w_G(i',j')$$

// $D_r = \{(i',j') : d((i,j),(i',j')) \leq r\}$ is the set of neighbors of (i,j)
// $d((i,j),(i',j'))$ is the Euclidean distance between (i,j) and (i',j')
// w_G are weighting factors, either constant or regular samples of a Gaussian function
// In any case, they are normalized so that $\sum_{(i',j')\in D_r} w_G(i',j') = 1$

end

matrices makes sense. So $M_S(i,j)$ can be interpreted as a mean of projectors on gradients. Being again positive, one can take its first eigenvector and consider it as the mean gradient direction. The only inconvenience of this mean gradient is that it has lost its orientation. Once the structure tensor at each point is computed, the **gradient smoothing algorithm** is applied. Its goal is to give a definite orientation to this robust mean gradient. Since the first eigenvector of the structure tensor is interpreted as an average gradient, but defined up to its sign, this sign must be specified. To that aim we start from u, the gray scale image, M_S the structure tensor at each discrete point (i,j) and the neighborhood D_r, both defined in Algorithm 2. We diagonalize M_S and denote by λ_1 its largest eigenvalue and by t_1 its associated eigenvector. Then the "smooth gradient vector" at (i,j) is defined by

$$G_s(i,j) = \kappa\sqrt{\lambda_1}\frac{t_1}{|t_1|}, \tag{4}$$

where

$$\kappa = \text{sign}\Big(\sum_{(i',j')\in D_r} \text{sign}((i'-i,j'-j)\cdot\frac{t_1}{|t_1|})u(i',j')\Big).$$

In that way, the smooth gradient points in the direction of increasing u.

4.2 Selection of Patch Centers

The centers of the patches can be selected in three different ways:

- Select all: all pixels positions (i,j) are used as centers of the patches (except the ones near the border of the image, so that the patches are inside the image domain). If bilinear interpolation has been used for computing the gradients, then an offset factor $(0.5, 0.5)$ is added to the selected positions (since gradients are known at points of the form $(i+0.5, j+0.5)$).
- Centers at Canny points (i.e. *edge* points). These points are defined as extrema of the modulus of the gradient in the direction of the gradient, and are computed as described in Algorithm 3.

Algorithm 3: Extraction of Canny points

Input : $\boldsymbol{G}$ (discrete gradient field, raw or smooth, computed with Equations (1), (2) or (3) or (4)), Th (threshold on the gradient magnitude)
Output : set of Canny points

foreach *discrete point* (i, j) **do**
  if $||\boldsymbol{G}(i,j)|| > Th$ **then**
    compute
      – unitary vector in the direction of the gradient $\boldsymbol{G_u}(i,j) = \frac{\boldsymbol{G}(i,j)}{||\boldsymbol{G}(i,j)||}$
      – (i_n, j_n): nearest discrete point to $(i,j) + \boldsymbol{G_u}$
      – (i_p, j_p): nearest discrete point to $(i,j) - \boldsymbol{G_u}$
    if $||\boldsymbol{G}(i,j)|| > ||\boldsymbol{G}(i_n,j_n)||$ *and* $||\boldsymbol{G}(i,j)|| > ||\boldsymbol{G}(i_p,j_p)||$ **then**
      label (i,j) as a *Canny point*
    end
  end
end

- Centers at Harris points (i.e. *corner* points [33]). These points are computed as described in Algorithm 4.

Algorithm 4: Extraction of Harris points

Input : $\boldsymbol{G}$ (discrete gradient field, raw, computed with Equations (1), (2) or (3))
Output : set of Harris points

foreach *discrete point* (i, j) **do**
  compute the structure tensor $M_S(i,j)$ (Algorithm 2), using a neighborhood of radius 1.

  define the *cornerness* of a point as $C(i,j) = (ab - c^2) - k(a+b)$ `// k is a fixed value` $(k = 0.05)$ `and` a, b, c `are the elements of` $M_S(i,j) = \begin{pmatrix} a & c \\ c & b \end{pmatrix}$.

  `//` T `is a fixed threshold` $(T = 0.01)$ `and` $C_{\max} = \max_{i,j} C(i,j)$
  if $C(i,j) < T \cdot C_{\max}$ **then**
    set $C(i,j) = 0$
  end
end
`// find local maxima of the cornerness function:`
foreach *pixel* (i, j) **do**
  if $C(i,j) > C(i',j')$ *for all* (i',j') *connected to* (i,j) **then**
    `// use 4 connectivity:` (i,j) `is connected with` $(i \pm 1, j)$ `and` $(i, j \pm 1)$
    label (i,j) as a *Harris point*
  end
end

4.3 Sampling of Patches

This section consists of two algorithms. Algorithm 5 takes a list of patch centers and the chosen patch normalization options. It delivers a list of resampled patches in vector form. Algorithm 6 describes

the simple normalization of mean and variance of a given patch by applying an affine map to its value, so the patch mean is zero and its variance 1. Note that one can go far beyond normalizing patches by translation, rotation, and affine maps on the intensity as we do here. In [29] for example, one finds the implementation of a method that extracts image patches normalized with respect to a geometric affine transformation. As such, it needs to define a sampling grid and interpolate the image to sample the patches, much like Algorithm 5.

5 Analysis of the Space of Patches

5.1 Principal Components Analysis (PCA)

The principal components [48] of a set of data vectors with equal dimension are defined as the eigenvectors of the empirical covariance of this set of vectors.

The PCA algorithm described in this section (see Algorithm 7) takes as input a set of data vectors of dimension d and gives a set of d directions, the eigenvectors of the empirical covariance matrix, and the corresponding variances, which are the eigenvalues of this same matrix. In the presentation we adopt, the eigenvectors are sorted in decreasing value of variance. The set of principal components form an orthonormal basis adapted to the set of data vectors. In many natural data sets, the decay of the eigenvalues is steep and can lead to a *dimension reduction* of the data set, by simply projecting the vectors on the subset generated by the eigenvectors with nonnegligible associated eigenvalue.

Note that the variance of the data along each principal component is its corresponding eigenvalue. This last property assumes that the original data vectors have been shifted so that their mean vector is $\mathbf{0}$.

5.2 Gaussian Mixture Estimation

A popular method of analysis of a set of data is to assume that they follow a Gaussian mixture model, that is, each data vector has been drawn from one of several possible multidimensional Gaussians. This has been for example applied to image restoration in [67, 63].

Assuming known the number of Gaussians in the model (K), the goal is to infer their parameters (mean and covariance) and to estimate the probability for each data vector to be drawn from each Gaussian (or, alternatively, the probability of each Gaussian in the mixture). This can be achieved with the EM (expectation-maximization) algorithm [24, 53], which minimizes the overall likelihood $\mathcal{L}$ of the model (see Algorithm 8). We use a C version of a classic Matlab implementation[3].

In order to understand the computations performed in the expectation step we have to take into account the following considerations:

The **likelihood** of the model is $\mathcal{L}' = \prod_{i=1}^{N} p(\boldsymbol{x_i})$.

The **log-likelihood** of the model is $\mathcal{L} = \log(\mathcal{L}') = \sum_{i=1}^{N} \log(p(\boldsymbol{x_i}))$.

$p(\boldsymbol{x_i})$ can be computed with the Total Probability formula: $p(\boldsymbol{x_i}) = \sum_{j=1}^{K} p(\boldsymbol{x_i}|j)p(j)$, where $p(\boldsymbol{x_i}|j)$ is the probability of vector $\boldsymbol{x_i}$ given a Gaussian distribution j

$$p(\boldsymbol{x_i}|j) = \frac{1}{(2\pi)^{m/2} det(\Sigma_j)^{1/2}} e^{-\frac{1}{2}(\boldsymbol{x_i}-\boldsymbol{\mu_j})\Sigma_j^{-1}(\boldsymbol{x_i}-\boldsymbol{\mu_j})},$$

Since the product of small quantities may lead to overflows, it is preferable to use logarithms when working with probabilities. If we define $\alpha_{ij} = \log(p(\boldsymbol{x_i}|j)) + \log(p(j))$, then

[3] Mo Chen, EM algorithm for Gaussian mixture model, Matlab Central (as of Oct. 25, 2012, 15:00 GMT). http://www.mathworks.com/matlabcentral/fileexchange/26184-em-algorithm-for-gaussian-mixture-model

Algorithm 5: Sampling of patches

Input : $\boldsymbol{u}$ (gray scale or color image, of size $w \times h$), C (set of centers of patches)
Input : (Optional) $\boldsymbol{G}$ (discrete gradient field, raw or smooth, computed with Eq. (1), (2), (3) or (4))

Parameters:

- patch size (κ, consider square patches)
- *color* flag: if set (and the image has 3 channels), then compute patches for each one of the channels, else compute patches only for the intensity component
- normalization options:
  - orientation normalization: *nOrientation* flag
  - average value normalization:*nMean* flag
  - variance normalization: *nVariance* flag
- interpolation options: *nearest*, *bilinear*, ***cubic B-spline***

Output : a set of vectors of dimension $d \times \kappa \times \kappa$ containing the sampled values of each patch ($d = 1$ if input image is gray or *color* flag is set to *false*, else $d = 3$)

The square patches are converted to vector form by scanning them row by row, from left to right and from top to bottom.

begin
  if $\boldsymbol{u}$ *is gray (or color flag is false)* **then**
    $\boldsymbol{u} = \{u\}$ (or $\boldsymbol{u} = \{\text{average}(R, G, B)\}$) (1 channel image)
  else
    $\boldsymbol{u} = \{R, G, B\}$ (3 channels image)
  end
  foreach *discrete point* (i, j) *in* C **do**
    if $\boldsymbol{G}(i, j)$ *is known* **then** define $\boldsymbol{v} = (v_x, v_y) = \frac{\boldsymbol{G}(i,j)}{||\boldsymbol{G}(i,j)||}$ `// unitary vector`
    else consider $\boldsymbol{v} = (1, 0)$
    define $\boldsymbol{w} = (w_x, w_y) = (-v_y, v_x)$ `// orthonormal vector to` $\boldsymbol{v}$
    consider the set of sampling points

$$S = \{(x, y) : (x, y) = (x_0, y_0) + k\boldsymbol{v} + l\boldsymbol{w}, 0 \leq k \leq \kappa - 1, 0 \leq l \leq \kappa - 1\}$$

    where

$$(x_0, y_0) = (i, j) - \frac{\kappa - 1}{2}\boldsymbol{v} - \frac{\kappa - 1}{2}\boldsymbol{w}$$

    is the top left point of the sampling grid

    if *all the points* (x, y) *in* S *belong to the image domain (*$0 \leq x \leq w - 1$, $0 \leq y \leq h - 1$*)* **then**
      foreach *channel* u *of image* $\boldsymbol{u}$ **do**
        compute sampled values $u_s(x, y)$ and store them in a vector structure
        `// Three interpolation methods can be used for computing` $u_s(x, y)$`:`
        `// Nearest neighbor:`
        `//` $u_s(x, y) = u(i', j')$ `such that` i' `and` j' `are the closest integers to` x `and` y`, respectively`
        `// Bilinear:`
        `//` $u_s(x, y) = u_B(x, y)$`, where` u_B `is the bilinear interpolate of` u `[31]`
        `// Cubic B-spline:`
        `//` $u_s(x, y) = u_C(x, y)$`, where` u_C `is the cubic B-spline interpolate of` u `[59, 5]`
        if *any of the normalization flags nMean or nVariance is active* **then**
          normalize mean and variance of the stored values `// see Algorithm 6`
        end
      end
    end
  end
end

Algorithm 6: Normalization of mean and variance of patches

Input : set of N patches: $\boldsymbol{x_i}$, $(1 \leq i \leq N)$, *nMean* flag, *nVariance* flag
Output : set of N normalized values: $\boldsymbol{y_i}$, $(1 \leq i \leq N)$

begin
  for *i=1* **to** N **do**
    μ=mean of $\boldsymbol{x_i}$
    σ=standard deviation of $\boldsymbol{x_i}$
    `// If flags are active, then normalize patches to zero mean and unit variance`
    if *nMean flag active* **then**
      $\boldsymbol{y_i} = \boldsymbol{x_i} - \mu$
    end
    if *nVariance flag active and* $\sigma \neq 0$ **then**
      $\boldsymbol{y_i} = (\boldsymbol{x_i} - \mu)/\sigma$
    end
  end
end

Algorithm 7: PCA

Input : a set of N vectors $\boldsymbol{x_i}$ of dimension d: $\boldsymbol{x_i} = (x_i(1), \cdots, x_i(d))$
Output : mean vector $\boldsymbol{m}$ (dimension d), set of variances λ_i $(1 \leq i \leq d)$ associated to each principal component (sorted in decreasing order), set of principal components $\boldsymbol{v_i}$ $(1 \leq i \leq d)$ (dimension d, sorted in decreasing order of variance)

begin
  compute mean (average) vector:

$$\boldsymbol{m} = \frac{1}{N} \sum_{i=1}^{N} \boldsymbol{x_i} \tag{5}$$

  compute the empirical covariance matrix $C = (c_{kl})$, $1 \leq k, l \leq d$:

$$c_{kl} = \frac{1}{N} \sum_{i=1}^{N} x_i(k) x_i(l) - m(k) m(l)$$

  diagonalize C: get eigenvalues λ_i and associated eigenvectors $\boldsymbol{v_i}$ $(1 \leq i \leq d)$

  order eigenvalues and associated eigenvectors (decreasing value of eigenvalue)

  normalize eigenvectors
end

Algorithm 8: EM algorithm for Gaussian Mixture estimation

Input : a set of N vectors $\boldsymbol{X} = \{\boldsymbol{x_i}\}$ of dimension d: $\boldsymbol{x_i} = (x_i(1), \cdots, x_i(d))$, $1 \leq i \leq N$; number of d-dimensional Gaussians in the mixture K

Parameters

- maximum number of iterations $it_{\max}$ (default: 500)
- tolerance factor T (default: 10^{-6})

Output : mean $\boldsymbol{\mu_j}$, covariance matrix Σ_j and probability p_j of each Gaussian in the mixture ($1 \leq j \leq K$); probability $p(j|\boldsymbol{x_i})$ for each input vector $\boldsymbol{x_i}$ of being drawn from Gaussian j; label for each input vector $l(\boldsymbol{x_i})$, indicating the index of the Gaussian from which it has been drawn with the highest probability: $l(\boldsymbol{x_i}) = \underset{j}{\operatorname{argmax}}\, p(j|\boldsymbol{x_i})$; final number of Gaussians in the mixture K

begin
  `// Initialize model:`
  begin
    `// Initialize mean vectors:`
    for *j=1* **to** K **do** $\boldsymbol{\mu_j}$=random vector from $\boldsymbol{X}$ `// check that all the` $\boldsymbol{\mu_j}$ `are different`
    `// Initialize labels and probabilities`
    for *i=1* **to** N **do**
      $l(\boldsymbol{x_i}) = \underset{j}{\operatorname{argmin}}\, ||\boldsymbol{x_i} - \mu_j||$ `// Gaussian with closest mean vector`
      for *j=1* **to** K **do** $p(j|\boldsymbol{x_i}) = 0$
      $p(l(\boldsymbol{x_i})|\boldsymbol{x_i}) = 1$
    end
  end
  `// Iterate until convergence:`
  $it = 1$ `// iterations counter`
  stop=**false**
  repeat
    `// Maximization step: reestimate mean and covariances`
    Maximization() `// see Algorithm 9`
    `// Expectation step: reestimate probabilities and labels`
    `// and compute log likelihood of the model`
    $\mathcal{L}$=Expectation() `// see Algorithm 10`
    `// Reevaluate number of Gaussians`
    set $K_0 = K$
    for *j=1* **to** K_0 **do**
      if $j \neq l(\boldsymbol{x_i})$, $\forall i \in \{1, \cdots, N\}$ **then**
        remove Gaussian component j
        set $K = K - 1$
      end
    end
    `// Stop criteria`
    stop = ($it > 2$ **and** $\frac{\mathcal{L} - \mathcal{L}_{prev}}{|\mathcal{L}|} < T$) **or** $it > it_{\max}$
    $it = it + 1$
    $\mathcal{L}_{prev} = \mathcal{L}$
  until *stop* = ***true***
end

Algorithm 9: Maximization step of the EM algorithm

Maximization()
Parameter: $\varepsilon = 10^{-6}$ (used for numerical stability)
begin
    `// update probability of each Gaussian:`
    for *j=1* **to** K **do** $p_j = \frac{1}{N}\sum_{i=1}^{N} p(j|\boldsymbol{x_i})$
    `// update mean of each Gaussian:`
    for *j=1* **to** K **do** $\boldsymbol{\mu}_j = \frac{\sum_{i=1}^{N} p(j|\boldsymbol{x_i})\boldsymbol{x_i}}{\sum_{i=1}^{N} p(j|\boldsymbol{x_i})}$
    `// update covariance matrix of each Gaussian:`
    for *j=1* **to** K **do**
        $\Sigma_j = (c_{jkl}), 1 \leq k, l \leq d$
        $c_{jkl} = \frac{\sum_{i=1}^{N} p(j|\boldsymbol{x_i})x_i(k)x_i(l)}{\sum_{i=1}^{N} p(j|\boldsymbol{x_i})} - \mu_j(k)\mu_j(l)$
        `// add a small regularizing term for numerical stability`
        for *k=1* **to** d **do** $c_{jkk} = c_{jkk} + \varepsilon$
    end
end

$$\mathcal{L} = \sum_{i=1}^{N} \log(\sum_{j=1}^{K} e^{\alpha_{ij}}),$$

The term $\log(\sum_{j=1}^{K} e^{\alpha_{ij}})$ is computed with the log-sum-exp formula to prevent underflows

$$\log(\sum_{j=1}^{K} e^{\alpha_{ij}}) = \alpha_i^{\max} + \log(\sum_{j=1}^{K} e^{(\alpha_{ij} - \alpha_i^{\max})}),$$

where $\alpha_i^{\max} = \max_j \alpha_{ij}$.
On the other hand

$$\log(p(\boldsymbol{x_i}|j)) = -\frac{1}{2}(\boldsymbol{x_i} - \boldsymbol{\mu_j})\Sigma_j^{-1}(\boldsymbol{x_i} - \boldsymbol{\mu_j}) - \frac{d}{2}\log(2\pi) - \frac{1}{2}\log(det(\Sigma_j)).$$

The terms $(\boldsymbol{x_i} - \boldsymbol{\mu_j})\Sigma_j^{-1}(\boldsymbol{x_i} - \boldsymbol{\mu_j})$ and $\log(det(\Sigma_j))$ can be efficiently computed using a Cholesky decomposition [16] of Σ_j, which is a symmetric and positive-definite matrix:

- $\Sigma_j = L \cdot L^T$, where L is an lower-triangular matrix.
  $(\boldsymbol{x_i} - \boldsymbol{\mu_j})\Sigma_j^{-1}(\boldsymbol{x_i} - \boldsymbol{\mu_j}) = (\boldsymbol{x_i} - \boldsymbol{\mu_j})(L \cdot L^T)^{-1}(\boldsymbol{x_i} - \boldsymbol{\mu_j}) = ||L^{-1}(\boldsymbol{x_i} - \boldsymbol{\mu_j})||^2 = ||\boldsymbol{a}||^2$,
  where $\boldsymbol{a}$ is the solution of the linear system $L\boldsymbol{a} = (\boldsymbol{x_i} - \boldsymbol{\mu_j})$.
- $\log(det(\Sigma_j)) = \log(det(L \cdot L^T)) = \log(det(L)^2) = 2\log(\prod_{k=1}^{d} L_{kk}) = 2\sum_{k=1}^{d}\log(L_{kk})$

It must be clarified that, in the EM algorithm described here, the Expectation step is only a fast approximation, as each patch is assigned to "only" one Gaussian during this step, whereas the model specifies that all patches are drawn from a mixture of Gaussian. This is a standard procedure, as the exact E-step would be too costly. See for example the detailed explanation in [67].

Algorithm 10: Expectation step of the EM algorithm

Expectation()
Output : Log Likelihood of the model

begin
  // compute $\log(p(\boldsymbol{x_i}|j))$
  for *j=1* **to** K **do**
    // Cholesky decomposition of Σ_j
    L=lower triangular matrix such that $\Sigma_j = L \cdot L^T$

    compute $\boldsymbol{a}$, the solution of $L\boldsymbol{a} = (\boldsymbol{x_i} - \boldsymbol{\mu_j})$

    $\log(p(\boldsymbol{x_i}|j)) = -\frac{1}{2}||\boldsymbol{a}||^2 - \frac{d}{2}\log(2\pi) - \sum_{k=1}^{d}\log(L_{kk})$
  end
  // compute $\log(p(\boldsymbol{x_i}))$, use log-sum-exp formula
  for *i=1* **to** N **do**
    for *j=1* **to** K **do** $\alpha_{ij} = \log(p(\boldsymbol{x_i}|j)) + \log(p(j))$

    define $\alpha_i^{\max} = \max_j \alpha_{ij}$
    $\log(p(\boldsymbol{x_i})) = \alpha_i^{\max} + \log(\sum_{j=1}^{K} e^{(\alpha_{ij} - \alpha_i^{\max})})$
  end
  // compute log likelihood
  $\mathcal{L} = \sum_{i=1}^{N}\log(p(\boldsymbol{x_i}))$

  // update $p(j|\boldsymbol{x_i})$, use Bayes formula in logarithmic form
  for *i=1* **to** N **do**
    for *j=1* **to** K **do**
      $\log p(j|\boldsymbol{x_i}) = \log p(\boldsymbol{x_i}|j) + \log p(j) - \log p(\boldsymbol{x_i})$
      $p(j|\boldsymbol{x_i}) = e^{\log p(j|\boldsymbol{x_i})}$
    end
  end
  // update labels
  for *i=1* **to** N **do** $l(\boldsymbol{x_i}) = \operatorname{argmax}_j p(j|\boldsymbol{x_i})$ // Gaussian with highest probability

  return $\mathcal{L}$
end

Acknowledgements

The first author has been partially sponsored by MINECO/AEI/FEDER, UE projects TIN2017-85572-P, DPI2017-86372-C3-3-R, and by the Comunitat Autonoma de les Illes Balears through the Direcció General de Política Universitària i Recerca with funds from the Tourist Stay Tax Law ITS 2017-006 (PRD2018/26). The second author has been partly financed by Office of Naval research grant N00014-17-1-2552 and N00014-20-S-B001, DGA Astrid project "filmer la Terre" n ANR-17-ASTR-0013-01.

Image Credits

 Miguel Colom, CC-BY

References

[1] M. Aharon, M. Elad, and A. Bruckstein, *K-SVD: Design of dictionaries for sparse representation*, in Proceedings of SPARS, vol. 5, 01 2005.

[2] P. Arias, V. Caselles, and G. Sapiro, *A variational framework for non-local image inpainting*, International Workshop on Energy Minimization Methods in Computer Vision and Pattern Recognition (EMMCVPR), (2009). https://doi.org/10.1007/978-3-642-03641-5_26.

[3] C. Barnes, E. Shechtman, A. Finkelstein, and D.B. Goldman, *Patchmatch: A randomized correspondence algorithm for structural image editing*, ACM Transactions on Graphics, 28 (2009), p. 24. https://doi.org/10.1145/1531326.1531330.

[4] X. Bresson and T.F. Chan, *Non-local unsupervised variational image segmentation models*, tech. report, 2008. UCLA CAM Report.

[5] T. Briand and P. Monasse, *Theory and Practice of Image B-Spline Interpolation*, Image Processing On Line, 8 (2018), pp. 99–141. https://doi.org/10.5201/ipol.2018.221.

[6] T. Brox and D. Cremers, *Iterated nonlocal means for texture restoration*, Lecture Notes in Computer Science, 4485 (2007), p. 13. https://doi.org/10.1007/978-3-540-72823-8_2.

[7] A. Buades, B. Coll, and J.M. Morel, *Nonlocal image and movie denoising*, International Journal of Computer Vision, 76 (2008), pp. 123–139. https://doi.org/10.1007/s11263-007-0052-1.

[8] A. Buades, B. Coll, and J.-M. Morel, *A non-local algorithm for image denoising*, in IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), vol. 2, 2005, pp. 60–65. https://doi.org/10.1109/CVPR.2005.38.

[9] A. Buades, B. Coll, and J-M. Morel, *A review of image denoising algorithms, with a new one*, Multiscale Modeling and Simulation, 4 (2006), pp. 490–530. https://doi.org/10.1137/040616024.

[10] A. Buades, B. Coll, J-M. Morel, and C. Sbert, *Self-Similarity Driven Color Demosaicking*, IEEE Transactions on Image Processing, 18 (2009), pp. 1192–1202. https://doi.org/10.1109/TIP.2009.2017171.

[11] A. Buades, Y. Lou, JM Morel, and Z. Tang, *A note on multi-image denoising*, in International Workshop on Local and Non-Local Approximation in Image Processing, IEEE, 2009, pp. 1–15. https://doi.org/10.1109/LNLA.2009.5278408.

[12] J. Salmon C-A. Deledalle and A. Dalalyan, *Image denoising with patch based PCA: local versus global*, in British Machine Vision Conference, BMVA Press, 2011, pp. 25.1–25.10. http://dx.doi.org/10.5244/C.25.25.

[13] J. Canny, *A computational approach to edge detection*, IEEE Transactions on Pattern Analysis and Machine Intelligence, (1986), pp. 679–698. https://doi.org/10.1109/TPAMI.1986.4767851.

[14] G. Carlsson, T. Ishkhanov, V. De Silva, and A. Zomorodian, *On the local behavior of spaces of natural images*, International Journal of Computer Vision, 76 (2008), pp. 1–12. https://doi.org/10.1007/s11263-007-0056-x.

[15] P. Chatterjee and P. Milanfar, *Patch-based near-optimal image denoising*, IEEE Transactions on Image Processing, 21 (2012), pp. 1635–1649. https://doi.org/10.1109/TIP.2011.2172799.

[16] A-L. Cholesky, *Sur la résolution numérique des systèmes d'équations linéaires*, Bulletin de la Sabix. Société des amis de la Bibliothèque et de l'Histoire de l'École polytechnique, (2005), pp. 81–95.

[17] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian, *Image denoising by sparse 3D transform-domain collaborative filtering*, IEEE Transactions on Image Processing, 16 (2007), pp. 2080–2095. https://doi.org/10.1109/TIP.2007.901238.

[18] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian, *BM3D image denoising with shape-adaptive principal component analysis*, Workshop on Signal Processing with Adaptive Sparse Structured Representations, (2009).

[19] A. Danielyan, A. Foi, V. Katkovnik, and K. Egiazarian, *Image And Video Super-Resolution Via Spatially Adaptive Block-Matching Filtering*, in International Workshop on Local and Non-Local Approximation in Image Processing, 2008.

[20] A. Davy, T. Ehret, J-M. Morel, P. Arias, and G. Facciolo, *A non-local CNN for video denoising*, in IEEE International Conference on Image Processing (ICIP), IEEE, 2019, pp. 2409–2413. https://doi.org/10.1109/ICIP.2019.8803314.

[21] C.A. Deledalle, L. Denis, and F. Tupin, *NL-InSAR: Nonlocal interferogram estimation*, IEEE Transactions on Geoscience and Remote Sensing, 49 (2011), pp. 1441–1452. https://doi.org/10.1109/TGRS.2010.2076376.

[22] C.A. Deledalle, F. Tupin, and L. Denis, *Poisson NL means: Unsupervised non local means for Poisson noise*, in International Conference on Image Processing (ICIP), IEEE, 2010, pp. 801–804. https://doi.org/10.1109/ICIP.2010.5653394.

[23] ——, *Polarimetric SAR estimation based on non-local means*, in IEEE International Geoscience and Remote Sensing Symposium (IGARSS), IEEE, 2010, pp. 2515–2518. https://doi.org/10.1109/IGARSS.2010.5653936.

[24] A.P. Dempster, N.M. Laird, and D.B. Rubin, *Maximum likelihood from incomplete data via the EM algorithm*, Journal of the Royal Statistical Society: Series B (Methodological), 39 (1977), pp. 1–22.

[25] M. Ebrahimi and E.R. Vrscay, *Solving the Inverse Problem of Image Zooming Using "Self-Examples"*, Lecture Notes in Computer Science, 4633 (2007), p. 117. https://doi.org/10.1007/978-3-540-74260-9_11.

[26] A.A. Efros and T.K. Leung, *Texture synthesis by non-parametric sampling*, in International Conference on Computer Vision (ICCV), vol. 2, 1999, pp. 1033–1038. https://doi.org/10.1109/ICCV.1999.790383.

[27] M. Elad and M. Aharon, *Image denoising via sparse and redundant representations over learned dictionaries*, IEEE Transactions on Image Processing, 15 (2006), pp. 3736–3745. https://doi.org/10.1109/TIP.2006.881969.

[28] V. Fedorov and C. Ballester, *Affine non-local means image denoising*, IEEE Transactions on Image Processing, 26 (2017), pp. 2137–2148. https://doi.org/10.1109/TIP.2017.2681421.

[29] V. Fedorov and C. Ballester, *An affine invariant patch similarity*, Image Processing On Line, 8 (2018), pp. 490–513. https://doi.org/10.5201/ipol.2018.202.

[30] W. Förstner, *A Feature Based Corresponding Algorithm for Image Matching*, International Archives of Photogrammetry, 26 (1986), pp. 150–166.

[31] P. Getreuer, *Linear Methods for Image Interpolation*, Image Processing On Line, 1 (2011), pp. 238–259. https://doi.org/10.5201/ipol.2011.g_lmii.

[32] G. Gilboa and S. Osher, *Nonlocal linear image regularization and supervised segmentation*, Multiscale Modeling and Simulation, 6 (2008), pp. 595–630. https://doi.org/10.1137/060669358.

[33] C.G. Harris and M. Stephens, *A combined corner and edge detector*, in Alvey Vision Conference, vol. 15, Citeseer, 1988, pp. 10–5244. https://doi.org/10.5244/C.2.23.

[34] Y.S. Heo, K.M. Lee, and S.U. Lee, *Simultaneous depth reconstruction and restoration of noisy stereo images using non-local pixel distribution*, in IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2007, pp. 1–8. https://doi.org/10.1109/CVPR.2007.382999.

[35] F. Jia, X-C. Tai, and J. Liu, *Nonlocal regularized CNN for image segmentation*, Inverse Problems & Imaging, 14 (2020), p. 891. http://dx.doi.org/10.3934/ipi.2020041.

[36] M. Jung and L.A. Vese, *Nonlocal variational image deblurring models in the presence of Gaussian or impulse noise*, Lecture Notes in Computer Science, 5567 (2009). https://doi.org/10.1007/978-3-642-02256-2_34.

[37] C. Kervrann, J. Boulanger, and P. Coupe, *Bayesian non-local means filter, image redundancy and adaptive dictionaries for noise removal*, Lecture Notes In Computer Science, 4485 (2007). https://doi.org/10.1007/978-3-540-72823-8_45.

[38] S. Kindermann, S. Osher, and P.W. Jones, *Deblurring and denoising of images by nonlocal functionals*, Multiscale Modeling and Simulation, 4 (2006), pp. 1091–1115. https://doi.org/10.1137/050622249.

[39] M. Lebrun, A. Buades, and J-M. Morel, *A nonlocal Bayesian image denoising algorithm*, SIAM Journal on Imaging Sciences, 6 (2013), pp. 1665–1688. https://doi.org/10.1137/120874989.

[40] M. Lebrun, M. Colom, A. Buades, and J-M. Morel, *Secrets of image denoising cuisine*, Acta Numerica, 21 (2012), pp. 475–576. https://doi.org/10.1017/S0962492912000062.

[41] J. Mairal, F. Bach, J. Ponce, and G. Sapiro, *Online learning for matrix factorization and sparse coding*, The Journal of Machine Learning Research, 11 (2010), pp. 19–60. https://dl.acm.org/doi/10.5555/1756006.1756008.

[42] J. Mairal, F. Bach, J. Ponce, G. Sapiro, and A. Zisserman, *Non-local sparse models for image restoration*, in IEEE International Conference on Computer Vision (ICCV), IEEE, 2009, pp. 2272–2279. https://doi.org/10.1109/ICCV.2009.5459452.

[43] J.V. Manjón, J. Carbonell-Caballero, J.J. Lull, G. García-Martí, L. Martí-Bonmatí, and M. Robles, *MRI denoising using Non-Local Means*, Medical Image Analysis, 12 (2008), pp. 514–523. https://doi.org/10.1016/j.media.2008.02.004.

[44] J.V. Manjón, M. Robles, and N.A. Thacker, *Multispectral MRI Denoising Using Non-Local Means*, in Annual Conference on Medical Image Understanding and Analysis (MIUA), vol. 7, 2007, pp. 41–45.

[45] M. Mignotte, *A non-local regularization strategy for image deconvolution*, Pattern Recognition Letters, 29 (2008), pp. 2206–2212. https://doi.org/10.1016/j.patrec.2008.08.004.

[46] B.A. Olshausen and D.J. Field, *Sparse coding with an overcomplete basis set: A strategy employed by VI?*, Vision research, 37 (1997), pp. 3311–3326. https://doi.org/10.1016/S0042-6989(97)00169-7.

[47] J. Orchard, M. Ebrahimi, and A. Wong, *Efficient Non-Local-Means Denoising using the SVD*, in IEEE International Conference on Image Processing (ICIP), 2008. https://doi.org/10.1109/ICIP.2008.4712109.

[48] K Pearson, *On lines and planes of closest fit to systems of points in space*, The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science, 2 (1901), pp. 559–572. https://doi.org/10.1080/14786440109462720.

[49] G. Peyré, *Manifold models for signals and images*, Computer Vision and Image Understanding, 113 (2009), pp. 249–260. https://doi.org/10.1016/j.cviu.2008.09.003.

[50] ——, *Sparse modeling of textures*, Journal of Mathematical Imaging and Vision, 34 (2009), pp. 17–31. https://doi.org/10.1007/s10851-008-0120-3.

[51] N.N. Ponomarenko, V.V. Lukin, S.K. Abramov, K.O. Egiazarian, and J.T. Astola, *Blind evaluation of additive noise variance in textured images by nonlinear processing of block DCT coefficients*, in Proceedings of SPIE, vol. 5014, 2003, p. 178. https://doi.org/10.1117/12.477717.

[52] N. N. Ponomarenko, V. V. Lukin, M. S. Zriakhov, A. Kaarna, and J. T. Astola, *An automatic approach to lossy compression of AVIRIS images*, IEEE International Geoscience and Remote Sensing Symposium, (2007). https://doi.org/10.1109/IGARSS.2007.4422833.

[53] William H Press, *Numerical recipes 3rd edition: The art of scientific computing*, Cambridge university press, 2007. ISBN: 9780521880688.

[54] M. Protter, M. Elad, H. Takeda, and P. Milanfar, *Generalizing the non-local-means to super-resolution reconstruction*, IEEE Transactions on Image Processing, (2008). https://doi.org/10.1109/TIP.2008.2008067.

[55] M. Raphan and E.P. Simoncelli, *An Empirical Bayesian interpretation and generalization of NL-means*, tech. report, Computer Science Technical Report TR2010-934 , Courant Institute of Mathematical Sciences, New York University, 2010.

[56] C. Ren, X. He, and Y. Pu, *Nonlocal similarity modeling and deep CNN gradient prior for super resolution*, IEEE Signal Processing Letters, 25 (2018), pp. 916–920. https://doi.org/10.1109/LSP.2018.2829766.

[57] T. Tasdizen, *Principal components for non-local means image denoising*, in IEEE International Conference on Image Processing (ICIP), 2008, p. 1728. https://doi.org/10.1109/ICIP.2008.4712108.

[58] K.M. Taylor and F.G. Meyer, *A random walk on image patches*, SIAM Journal on Imaging Sciences, 5 (2012), pp. 688–725. http://dx.doi.org/10.1137/110839370.

[59] P. Thévenaz, T. Blu, and M. Unser, *Interpolation revisited [medical images application]*, IEEE Transactions on Medical Imaging, 19 (2000), pp. 739–758. https://doi.org/10.1109/42.875199.

[60] N. Wiest-Daesslé, S. Prima, P. Coupé, S.P. Morrissey, and C. Barillot, *Non-local means variants for denoising of diffusion-weighted and diffusion tensor MRI*, Lecture Notes in Computer Science, 4792 (2007), p. 344. https://doi.org/10.1007/978-3-540-75759-7_42.

[61] N. Wiest-Daessle, S. Prima, P. Coupé, S.P. Morrissey, and C. Barillot, *Rician noise removal by non-local means filtering for low signal-to-noise ratio MRI: Applications to DT-MRI*, Lecture Notes in Computer Science, 5242 (2008), pp. 171–179. https://doi.org/10.1007/978-3-540-85990-1_21.

[62] A. Wong and J. Orchard, *A nonlocal-means approach to exemplar-based inpainting*, in IEEE International Conference on Image Processing (ICIP), 2008, pp. 2600–2603. https://doi.org/10.1109/ICIP.2008.4712326.

[63] G. Yu, G. Sapiro, and S. Mallat, *Solving Inverse Problems with Piecewise Linear Estimators: From Gaussian Mixture Models to Structured Sparsity.* Arxiv preprint arXiv:1006.3056, 2010. https://arxiv.org/abs/1006.3056.

[64] L. Zhang, W. Dong, D. Zhang, and G. Shi, *Two-stage image denoising by principal component analysis with local pixel grouping*, Pattern Recognition, 43 (2010), pp. 1531–1549. https://doi.org/10.1016/j.patcog.2009.09.023.

[65] X. Zhang, M. Burger, X. Bresson, and S. Osher, *Bregmanized Nonlocal Regularization for Deconvolution and Sparse Reconstruction*, SIAM Journal on Imaging Sciences, 3 (2010), pp. 253–276. https://doi.org/10.1137/090746379.

[66] S. Zimmer, S. Didas, and J. Weickert, *A rotationally invariant block matching strategy improving image denoising with non-local means*, in International Workshop on Local and Non-Local Approximation in Image Processing, 2008.

[67] D. Zoran and Y. Weiss, *From learning models of natural image patches to whole image restoration*, in International Conference on Computer Vision (ICCV), IEEE, 2011, pp. 479–486. https://doi.org/10.1109/ICCV.2011.6126278.