Covariance Trees for 2D and 3D Processing

Thierry Guillemot, Andrés Almansa, Tamy Boubekeur

▶ **To cite this version:**

Thierry Guillemot, Andrés Almansa, Tamy Boubekeur. Covariance Trees for 2D and 3D Processing. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Jun 2014, Colombus, United States. pp.556-563, 10.1109/CVPR.2014.78 . hal-01117099v2

HAL Id: hal-01117099
https://imt.hal.science/hal-01117099v2

Submitted on 24 Jan 2017

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Covariance Trees for 2D and 3D Processing

Thierry Guillemot Andrés Almansa Tamy Boubekeur
Télécom ParisTech - CNRS – LTCI – Institut Mines Télécom - Paris, France
{guillemot,almansa,boubekeur}@telecom-paristech.fr

Abstract

Gaussian Mixture Models have become one of the major tools in modern statistical image processing, and allowed performance breakthroughs in patch-based image denoising and restoration problems. Nevertheless, their adoption level was kept relatively low because of the computational cost associated to learning such models on large image databases. This work provides a flexible and generic tool for dealing with such models without the computational penalty or parameter tuning difficulties associated to a naïve implementation of GMM-based image restoration tasks. It does so by organising the data manifold in a hirerachical multiscale structure (the Covariance Tree) that can be queried at various scale levels around any point in feature-space. We start by explaining how to construct a Covariance Tree from a subset of the input data, how to enrich its statistics from a larger set in a streaming process, and how to query it efficiently, at any scale. We then demonstrate its usefulness on several applications, including non-local image filtering, data-driven denoising, reconstruction from random samples and surface modeling from unorganized 3D points sets.

1. Introduction

Statistical Priors for Image Restoration witnessed two quantum leaps in the last decade: The first one, Non Local Means (NLM) [5, 4, 13], allowed to go beyond local regularization priors like TV, by observing that non-local patch-based priors enable to much better capture the self-similar structure of natural image textures. The second one, including Non Local Bayes (NLB) [9], and Piecewise Linear Estimator (PLE) [18], perfected this idea by means of a more detailed description of the manifold containing natural image patches, which turns out to be piecewise regular and low-dimensional with respect to the high-dimensional embedding patch-space.

Actual implementations of these ideas require algorithmic accelerations and model simplifications. Thus, in NLB [9] the manifold is assumed to be locally linear, and approximated by an anisotropic Gaussian model, based on the kNNs in feature space. Ideally the manifold should be estimated over the clean feature but in the absence of better information, it is approximated iteratively from the noisy features. This choice is potentially inaccurate and requires a lot of computation since a local Gaussian Model has to be learnt around each single point in the image.

Another approach has been made popular in image processing by PLE [18] which is closely related to Structured Sparsity as introduced in [12]. In this approach the problem is simplified by modelling the patch manifold as a Gaussian Mixture Model (GMM) that is fitted to the unknown restored patches via their noisy measurements and an iterative Expectation Maximisation (EM) algorithm. In this case the number of Gaussians in the model is fixed in advance to no more than two dozens, in order to keep computational complexity under control, and to ensure that initialization heuristics are sufficient to guide the EM procedure to a good local minimum of the non-convex objective function.

The purpose of this work is to provide a generic data structure which can be used to estimate the patch manifold from a big database of clean patches. Our approach can be seen as a hybrid between PLE and NLB, in the sense that it is based on a GMM (like PLE) that is locally defined in the feature space (like NLB). Contrarily to NLB there is no need to re-learn the model around each patch. Hence the number of gaussians is no longer limited to a few dozens thanks to a hierachical data-structure that allows to : *(a)* quickly insert a new patch to enrich the learnt model and *(b)* quickly query the learnt model parameters that are pertinent around a given patch.

Experimental evidence shows that our scheme closely matches the restoration quality of top-notch state-of-the-art image restoration methods like PLE or NLB. We demonstrate that various 2D and 3D problems –usually formulated in terms of Bayesian *a posteriori* expectation (EAP)– can be reformulated as a posterior likelihood maximization (MAP) which can be solved by our scheme. Then we provide a few examples showing how our CovTree has an advantage when applied to several problems such as image denoising, image reconstruction, point set surfaces reconstruction.

2. Background

Notation used in the paper Let us consider a spatial domain $\mathcal{S} \subset \mathbb{R}^{d_\mathcal{S}}$ and let $\{\mathbf{p}_1, \ldots, \mathbf{p}_N\}$ be a point set of N samples. To fix ideas let us consider a mapping $f : \mathcal{S} \to \mathcal{R}$ associating each sample $\mathbf{p}_i \in \mathcal{S}$ to a value $\mathbf{f}_i$ of a range domain $\mathcal{R} \subset \mathbb{R}^{d_\mathcal{R}}$ [1]. For example, this representation includes RGB images associating to each pixel $\mathbf{p}_i = (x_i, y_i)^T$ a color value $\mathbf{f}_i = (r_i, g_i, b_i)^T$, 3D point sets defined by their spatial positions and normals $\mathbf{p}_i = \mathbf{f}_i = (x_i, y_i, z_i, nx_i, ny_i, nz_i)^T$, a more complex representation used to compute bilateral filters concatenating spatial and range coordinates by $\mathbf{p}_i = (x_i, y_i, r_i, g_i, b_i)^T$ of an image to a color value $\mathbf{f}_i = (r_i, g_i, b_i)^T$ or a non local representation by replacing $\mathbf{p}_i$ and $\mathbf{f}_i$ by image patches.

High-Dimensional Filtering Linear filters such as the bilateral filter [3, 15, 17] or the non-local means filter [5] can be computed as a weighted average of values in the range domain $\mathcal{R}$, where the weights measure the dissimilarity between points in a spatial domain $\mathcal{S}$, by means of *e.g.* a Gaussian kernel ϕ_Σ with diagonal covariance matrix Σ. Thus, for any $\mathbf{q} \in \mathcal{S}$ the filtered signal $\hat{f}$ is defined by:

$$\hat{f}(\mathbf{q}) = \sum_{\mathbf{p}_i \in \mathcal{S}} \phi_\Sigma(\mathbf{p}_i - \mathbf{q})\mathbf{f}_i / \sum_{\mathbf{p}_i \in \mathcal{S}} \phi_\Sigma(\mathbf{p}_i - \mathbf{q}) \qquad (1)$$

For most applications, a naïve implementation of Eq. 1 requieres a quadratic complexity and needs to be accelerated. The main idea of most acceleration techniques is to perform filtering by a linear interpolation of values computed by downsampling $\mathcal{S}$. For bilateral filtering, Paris and Durand [14] introduce a tesselation of $\mathcal{S}$ into hypercubes using a regular grid defined into the spatial $5d$ domain. Nevertheless, such a grid defines a lot of unnecesserary cells yielding the use of this approach difficult for higher-dimensionnal applications.

Adams et al. [2] use a *kd-tree* dividing $\mathcal{S}$ into hyperrectangles depending on signal variations thus avoiding all the empty cells defined by the regular grid. Then they [1] tesselate $\mathcal{S}$ using uniform simplicies. The filter's response is computed by performing multi-linear or barycentric interpolation. They consider that the signal is a linear manifold and as the dimension $d_\mathcal{S}$ of $\mathcal{S}$ increases, the number of necessary cells to enclose the signal explodes.

Gastal and Oliveira [7] defines a correct downsampling of $\mathcal{S}$ by using non-linear manifolds. They iteratively separate samples from different populations into different clusters using recursive low-pass filters to define adaptative manifolds. Thus, the computation is performed only where needed and the filter response is computed in linear time.

Collaborative Filter When $\mathbf{p}_i = \mathbf{f}_i$ is a set of patches corrupted by a Gaussian noise of variance σ_n^2, Eq 1 can be reformulated – in the case of NLM – in terms of Bayesian *a posteriori* expectation (EAP). Lebrun et al. [9] propose the Non Local Bayes (NLB) filter by replacing the EAP by a posterior likelihood maximization (MAP). Thus, in addition to a non-local mean μ, each patch is associated to a covariance matrix describing the variability of the patch group. For any noisy patch $\tilde{\mathbf{p}}$, an optimum $\hat{\mathbf{p}}$ is computed by :

$$\hat{\mathbf{p}} = \mu_{\tilde{\mathbf{p}}} + \Sigma_{\tilde{\mathbf{p}}}[\Sigma_{\tilde{\mathbf{p}}} + \sigma_n^2 Id]^{-1}(\tilde{\mathbf{p}} - \mu_{\tilde{\mathbf{p}}}) \qquad (2)$$

Ideally the covariance matrix $\Sigma_{\tilde{\mathbf{p}}}$ corresponds to clean patches variability but, in pratice, it is computed from a noisy covariance $\tilde{\Sigma}_{\tilde{\mathbf{p}}}$ describing variability of noisy patches. Such a filter is called "collaborative" because we process all the patches group at the same time by the filtering operation $\Sigma_{\tilde{\mathbf{p}}} = \tilde{\Sigma}_{\tilde{\mathbf{p}}} - \sigma^2 Id$. Nevertheless, this approach requires a lot of computation since a local Gaussian model has to be learnt twice around each patch in the images to ensure the correctness of $\Sigma_{\tilde{\mathbf{p}}}$. In practice, the estimation step is approximated by a kNN search in the feature space.

Related to the work about Structure Sparsity [12], Yu et al. [18] introduce the Piecewise Linear Estimator (PLE) that defines a general framework for solving inverse problems in imaging such as inpainting, zooming, or deblurring. The manifold of patches is approximated by a Gaussian Mixture Model (GMM) which is fitted to the unknown restored patches via MAP using an iterative Expectation Maximisation (EM) algorithm. This approach can be seen as a faster implementation of NLB which strongly discretizes the manifold of patches. In this context finding a good initialization and a correct number of classes for the GMM is a quite tricky problem.

The aim of our work is to use the latest progress concerning high dimensional filtering to generalize ideas introduced by collaborative filtering. We propose the *Covariance Tree*, a data structure able to learn points distributions from data and to provide, for any query location and scale, an anisotropic Gaussian corresponding to the local learned distribution. In particular, the Covariance Tree provides several **key benefits** such as:

1. *on-the-fly learning* which allows, given an initial structure, to progressively refine the precision of the learned model by streaming additional data points, for a constant memory budget, accounting for a potentially high amount of data points while controlling the size of the tree. This is a **a key aspect of our work** to recover texture details;

[1] Remark : It helps to think of f as a function even though our structure does not require the mapping f to be well and uniquely defined for any $\mathbf{p} \in \mathcal{S}$ nor does it require this mapping to be known explicitly. Rather the association between $\mathbf{p}$ and $\mathbf{f}$ is learnt by the algorithm from the given database of pairs $(\mathbf{p}_i, \mathbf{f}_i)$.

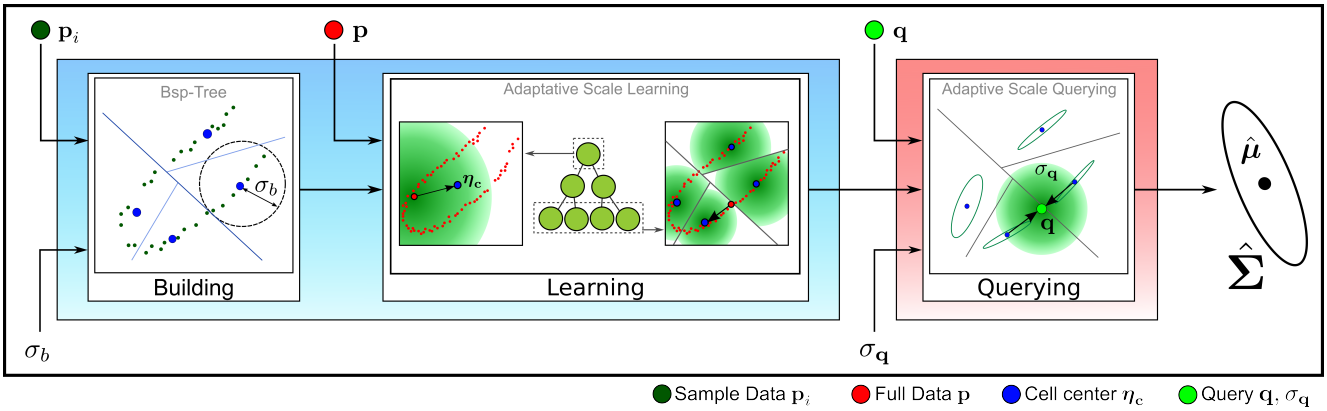


Figure 1. *Pipeline:* To fix ideas, we illustrate the different steps of our algorithm when the two sets $\{\mathbf{p}_i\} = \{\mathbf{f}_i\}$ Our CovTree is based on tree main steps : (a) from a sampled data set, we build a binary tree structure from a space partionning of $\mathcal{S}$ based on $\{\mathbf{p}_i\}$ up to a cell size σ_b, (b) each node learns the local statistical distribution modeled by an anisotropic kernel by streaming data points through the tree by considering the $\mathbf{p}_i$ and summing a weighted contribution of $\mathbf{f}_i$ to all traversed' nodes kernel, (c) for any query point $\mathbf{q} \in \mathcal{S}$ and scale $\sigma_\mathbf{q} \in \mathbb{R}$, our CovTree models the local distribution of learned data at $\mathbf{q}$ at the scale $\sigma_\mathbf{q}$ by providing a multivariate Gaussian distribution defined from a mean $\hat{\boldsymbol{\mu}}$ and a covariance matrix $\hat{\boldsymbol{\Sigma}}$.

2. *fast local distribution estimate*, at different scales, without recomputing the data structure;

3. *genericity*, allowing to solve a number of 2D and 3D processing problems by instantiating our structure with specific spatial and range domains, which includes Non-Local Bayes filtering, data-driven image denoising, image holes completion and 3D Non-Local Point Set Surface reconstruction.

3. The Covariance Tree

Assume that we want to restore a data point $\mathbf{f} \in \mathcal{R}$ that is either noisy or incomplete in some way. We also have access to $\mathbf{q} \in \mathcal{S}$ which is related to $\mathbf{f}$ in the following manner: the prior distribution of $\mathbf{f}$ given $\mathbf{q}$ can be modelled as a multivariate Gaussian $N(\mu_\mathbf{q}, \Sigma_\mathbf{q})$ with parameters varying smoothly as a function of $\mathbf{q}$. [2]

If we know the mapping $\mathbf{q} \mapsto (\mu_\mathbf{q}, \Sigma_\mathbf{q})$, and the degradation model (given as the conditional probability of the degraded or incomplete $\tilde{\mathbf{f}}$ given the clean $\mathbf{f}$) then we can use standard Bayesian techniques (such as MAP or EAP) to estimate $\mathbf{f}$ from the degraded pair $(\mathbf{q}, \tilde{\mathbf{f}})$. In general such a mapping is unknown, but we can estimate it from a large database of examples $(\mathbf{p}_i, \mathbf{f}_i) \in \hat{\mathcal{R}} \times \hat{\mathcal{S}} \subset \mathcal{R} \times \mathcal{S}$, by averaging a sufficiently large number of neighbours :

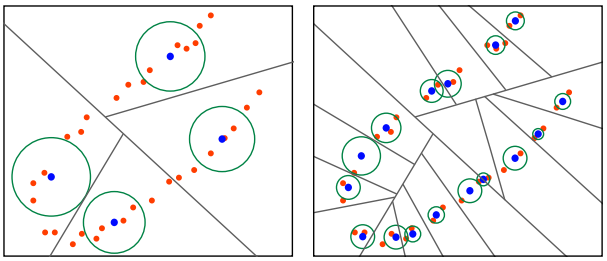
$$\begin{aligned} \mu_{\mathbf{q},\sigma_\mathbf{q}} &= \sum_{\mathbf{p}_i \in \hat{\mathcal{R}}} \phi_{\sigma_\mathbf{q}}(\|\mathbf{p}_i - \mathbf{q}\|)\mathbf{f}_i \\ \Sigma_{\mathbf{q},\sigma_\mathbf{q}} &= \sum_{\mathbf{p}_i \in \hat{\mathcal{R}}} \phi_{\sigma_\mathbf{q}}(\|\mathbf{p}_i - \mathbf{q}\|)\bar{\mathbf{f}}_i \bar{\mathbf{f}}_i^T \end{aligned} \tag{3}$$

where $\bar{\mathbf{f}}_i = (\mathbf{f}_i - \mu_{\mathbf{q},\sigma_\mathbf{q}})$. A small scale parameter $\sigma_\mathbf{q}$ provides a better localized prior, but larger values are often required to provide a statistically significant estimation of the local Gaussian distribution from the given examples, especially when the dimensions $d_\mathcal{S}$ and $d_\mathcal{R}$ are large, as is the case in non-local patch-based filtering or restoration for instance.

Unfortunately, high-dimensional restoration problems require large example databases and a brute-force approach to the computation of Eq. (3) at each query point $\mathbf{q}$ is untractable. We tackle this problem by introducing an hierarchical data structure which summarizes and indexes the database at multiple scales, with a limited amount of memory. This structure is equipped with a *fast* query mecanism providing an approximation to (3), for any query point $\mathbf{q}$ and for a large range of scales $\sigma_\mathbf{q}$. Our basic idea is to model the database as an hierarchical set of anisotropic multivariate Gaussians approximating smoothly and progressively the distribution of $\{\mathbf{f}_i\}$ in the database. At each level of the structure, the kernels are formed by the mean and the covariance of the local distribution. Consequently, we name our structure *Covariance Tree* (or *cov-tree*).

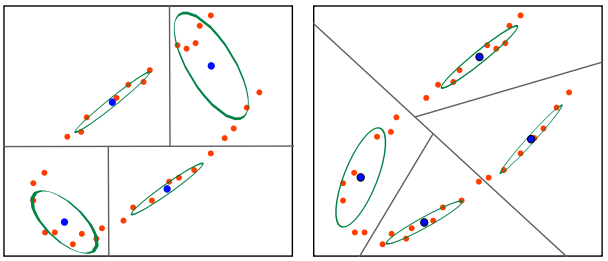
More precisely, a cov-tree is a Binary Space Partition Tree [6] (*bsp-tree*) carrying anisotropic Gaussians learned from data on its nodes. It defines a rotation-invariant tessellation of $\mathcal{S}$ into space cells, as well as an hierarchical

[2] Note that if this assumption holds, then the data $\mathbf{f}_i$ will be more compactly represented by multivariate Gaussians than by simpler isotropic ones (see Figure 2). Indeed, by using isotropic Gaussian we consider that f is propagating the same way in every direction of the space defining smaller and more numerous cells and decreasing the quality of the learning.



(a) Large cells (b) Small cells

Figure 2. *Distribution models*: Local isotropic Gaussians are too poor a model for manifolds. At a given scale, the balls are too coarse to describe local variations *(a)*. The only solution is to refine the partition *(b)* increasing the number of representative cells.



(a) Kd-tree (b) Bsp-tree

Figure 3. *Tree structure.* A *Kd-tree* (a) defines partitions of $\mathcal{S}$ independantly of the anisotropy of the data. A *bsp-tree* (b) allows to be rotation-invariant and better model distributions, diminishing the error of the estimated multivariate Gaussian.

partition allowing to approximate the database at different scales.

Our approach (as summarized in Figure 1) is essentially based on three operators:

building we perform a top-down hierarchical space partitioning of $\mathcal{S}$ based on $\{\mathbf{p}_i\}$ up to a cell size σ_b, resulting in a binary tree structure for which each node will later carry an anisotropic kernel modeling the local statistical distribution of $\{\mathbf{f}_i\}$ in its related space cell (Sec. 3.1);

learning we learn this distribution by streaming (training) data points through the tree, classifying them using $\mathbf{p}_i$ and summing a weighted contribution of $\mathbf{f}_i$ to all traversed nodes' kernels (Sec. 3.2);

querying for any query point $\mathbf{q} \in \mathcal{S}$ and scale $\sigma_\mathbf{q} \in \mathcal{R}$, our cov-tree provides a multivariate Gaussian distribution, in the form a mean $\hat{\boldsymbol{\mu}}$ and a covariance matrix $\hat{\boldsymbol{\Sigma}}$ interpolated at $\mathbf{q}$ and modelling the distribution of learned data at scale $\sigma_\mathbf{q}$ (Sec. 3.3).

The two parameters σ_b and $\sigma_\mathbf{q}$ (that determine the locality of learning resp. query) are set to ensure a reasonable approximation of equation (3). Usually, σ_b is defined smaller than the noise level σ_n of the training data and $\sigma_\mathbf{q} \simeq \sigma_n$. Indeed, given the noise in the data, values $\sigma_\mathbf{q}$ finer than the noise σ_n provide too poor statistical estimate of $(\hat{\boldsymbol{\mu}}, \hat{\boldsymbol{\Sigma}})$. The result of the successive steps of building, learning and querying approximates the estimation of covariance matrices with a Gaussian kernel of size $\sqrt{2}\sigma_\mathbf{q}$.

3.1. Building the tree

Instead of using a *kd-tree* [2], which fails at capturing anisotropy accurately (see Fig. 3(a)), we give to our cov-tree a *bsp* structure based on the direction of maximum variation of the input points. Let us consider a tree node η and its associated set of points $\mathbf{p}_j$. We compute and store in η a splitting plane $\{\boldsymbol{\eta_d}, \boldsymbol{\eta_c}\}$, with its normal $\boldsymbol{\eta_d}$ defined as the normalized eigenvector associated to the largest eigenvalue of the covariance of $\{\mathbf{p}_j\}$ and its center $\boldsymbol{\eta_c}$ defined as the average of the $\mathbf{p}_j$. The cell radius is given by $\eta_r = \max_{\mathbf{p}_j} \|\mathbf{p}_j - \boldsymbol{\eta_c}\|$. We then subdivide $\{\mathbf{p}_j\}$ into two distinct sets based on their signed distance to the plane $(\mathbf{p}_j - \boldsymbol{\eta_c})^t \boldsymbol{\eta_d}$. Finally, we construct the two children of η based on these two sets. Starting from the root and the entire input $\{\mathbf{p}_i\}$, we perform this recursive construction while $\eta_r > \sigma_b$. Consequently, our approach is *output-sensitive* and well adapted to large data sets, the memory footprint of the cov-tree depending only on the desired precision σ_b.

3.2. Learning local distributions

Once the tree structure is initialized, we can compute the statistics of its cells by streaming pairs of training data point $\{\mathbf{p}_i, \mathbf{f}_i\}$ through it. Starting from the root node, a pair is classified top-down using $\mathbf{p}_i$ and enriches each traversed node η by summing $\mathbf{f}_i$ to its local distribution using a weight w_i defined from a Gaussian ϕ_{η_r} centered at $\boldsymbol{\eta_c}$:

$$\begin{aligned} \boldsymbol{\mu}_\eta &:= \boldsymbol{\mu}_\eta + w_i \mathbf{f}_i \\ \boldsymbol{\Sigma}_\eta &:= \boldsymbol{\Sigma}_\eta + w_i \mathbf{f}_i^t \mathbf{f}_i \end{aligned} \tag{4}$$

Each point we stream during the learning step increases the precision of our cov-tree without adding additional nodes (i.e., constant memory cost). For a large database, the tree nodes'kernels are typically learned over the full data while the tree's structure is built from a subset only. Interestingly enough, the dataset used for the building step can be different from the learning one.

3.3. Querying local distributions

Once fully built, the tree can be queried using any $\mathbf{q} \in \mathcal{S}$, providing an anisotropic Gaussian describing the distribution of the learned values around $\mathbf{q}$ at scale $\sigma_\mathbf{q}$. To do so, we first collect a set of tree nodes in the vicinity of $\mathbf{q}$, at different scales, by traversing the tree top-down, gathering every node η intersecting the $[\mathbf{q}, \sigma_\mathbf{q}]$ ball and being either a leaf or verifying $\eta_r > \sigma_\mathbf{q}$. Traversing the tree to a finer precision level $\eta_r << \sigma_\mathbf{q}$ would increases the precision, but increases significantly the computational cost (see Fig 4). Second,

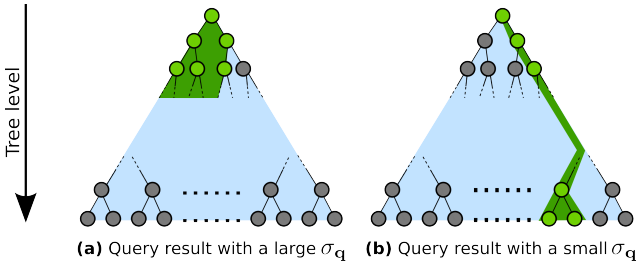


Figure 4. Our fast query limits the number of cov-tree nodes gathered (in green) to reconstruct a local anisotropic distribution. *(a)* When the requested scale $\sigma_{\mathbf{q}}$ is large, only the top nodes are retained. *(b)* As $\sigma_{\mathbf{q}}$ gets smaller, the gathering shaft gets thinner, collecting deeper nodes mostly.

we estimate the distribution at $\mathbf{q}$ as a weighted combination of the gathered nodes' distributions, using weights w_i defined from a Gaussian $\phi_{\sigma_{\mathbf{q}}}$ centered on $\mathbf{q}$:

$$\begin{aligned} \hat{\boldsymbol{\mu}} &= \frac{1}{\hat{w}} \sum_i w_i \boldsymbol{\mu}_{\eta_i} \\ \hat{\boldsymbol{\Sigma}} &= \frac{1}{\hat{w}_2} \sum_i w_i \boldsymbol{\Sigma}_{\eta_i} - \hat{w} \hat{\boldsymbol{\mu}}^t \hat{\boldsymbol{\mu}} \end{aligned} \tag{5}$$

Where $\hat{w}$ and $\hat{w}_2$ are two normalization values.

3.4. Complexity Analysis

We recall that $\mathcal{S}$ is the $d_{\mathcal{S}}$-dimensional spatial domain and $\mathcal{R}$ is the $d_{\mathcal{R}}$-dimensional range domain. Each of the N_b points used during the building step appears only once in the K_b cov-tree clusters, resulting in a *building* cost of $O(d_{\mathcal{S}} N_b log(K_b))$. Using N_l points to learn the nodes'statistics, classifying them has a cost of $O(N_l d_{\mathcal{S}} log(K_b))$ and learning variations in all nodes has a cost of $O(N_l log(K_b) d_{\mathcal{R}}^2)$, resulting in a *learning* cost of $O(N_l log(K_b)(d_{\mathcal{S}} + d_{\mathcal{R}}^2))$. When querying N_q points, gathering the lists of K_q contributing nodes has a cost of $O(N_q d_{\mathcal{S}} K_q)$; the additional anisotropic Gaussian estimation ($O(N_q K_q d_{\mathcal{R}}^2)$) leads to a total *querying* cost of $O(N_q K_q (d_{\mathcal{S}} + d_{\mathcal{R}}^2))$. Last, the memory cost of our cov-tree is $O(N_b K_b d_{\mathcal{R}}^2)$ and remains constant during the online steps (learning and querying), allowing to re-learn and/or reuse numerous times a cov-tree precomputed once.

4. Applications

We provide a few example applications of our cov-tree to solve inverse problems in 2D imaging and 3D rendering. Whenever possible the results are compared with state of the art techniques for the same problems.

The performance numbers reported in this paper were measured on a 2.4 GHz Intel Xeon processor with 12 GB of memory and 8 cores, but running a non-optimized C++

Algorithm 1 Cov-tree main functions

function BUILD(C_k, σ_b)
  Allocate new node η
  $\boldsymbol{\eta_c} \leftarrow$ means of points in C_k
  $\boldsymbol{\eta_d} \leftarrow$ largest eigenvector of $cov(C_k)$
  $\eta_r \leftarrow max_{\mathbf{p}_i \in C_k} ||\mathbf{p}_i - \boldsymbol{\eta_c}||_2$
  if $\eta_r \geq \sigma_b$ **then**
    Mark η as leaf node
  else
    $C_l \leftarrow \{\mathbf{p}_i \in C_k, (\mathbf{p}_i - \boldsymbol{\eta_c})^t \boldsymbol{\eta_d} \leq 0\}$
    $\eta_{left} \leftarrow$ BUILD(C_l)
    $C_r \leftarrow \{\mathbf{p}_i \in C_k, (\mathbf{p}_i - \boldsymbol{\eta_c})^t \boldsymbol{\eta_d} \geq 0\}$
    $\eta_{right} \leftarrow$ BUILD(C_r)
  end if
  return η
end function

function LEARN($\mathbf{p}, \mathbf{f}, \eta$)
  // Compute weight by using the node radius
  $w_i \leftarrow \phi_{\eta_r}(\mathbf{p} - \boldsymbol{\eta_c})$
  Update node statistics using Eq.(4)
  if η is not a leaf node **then**
    if $(\mathbf{p} - \boldsymbol{\eta_c})^t \boldsymbol{\eta_d} \leq 0$ **then**
      LEARN($\mathbf{p}, \mathbf{f}, \eta_{left}$)
    else
      LEARN($\mathbf{p}, \mathbf{f}, \eta_{right}$)
    end if
  end if
end function

function COMPUTE STAT($\mathbf{q}, \sigma_{\mathbf{q}}, \eta$)
  if $\eta_r < \sigma_{\mathbf{q}}$ or η is not a leaf node **then**
    // Compute weight by using $\sigma_{\mathbf{q}}$
    $w_i \leftarrow \phi_{\sigma_{\mathbf{q}}}(\mathbf{q} - \boldsymbol{\eta_c})$
    $\boldsymbol{\mu} := \boldsymbol{\mu} + w_i \boldsymbol{\mu}_{\eta}$
    $\boldsymbol{\Sigma} := \boldsymbol{\Sigma} + w_i \boldsymbol{\Sigma}_{\eta}$
  else
    if $(\mathbf{q} - \boldsymbol{\eta_c})^t \boldsymbol{\eta_d} \leq \sigma_{\mathbf{q}}$ **then**
      $\{\boldsymbol{\mu}, \boldsymbol{\Sigma}\} \leftarrow$ COMPUTE STAT($\mathbf{q}, \sigma_{\mathbf{q}}, \eta_{left}$)
    end if
    if $(\mathbf{q} - \boldsymbol{\eta_c})^t \boldsymbol{\eta_d} \geq -\sigma_{\mathbf{q}}$ **then**
      $\{\boldsymbol{\mu}, \boldsymbol{\Sigma}\} \leftarrow$ COMPUTE STAT($\mathbf{q}, \sigma_{\mathbf{q}}, \eta_{right}$)
    end if
  end if
  return $\{\mu, \boldsymbol{\Sigma}\}$
end function

function QUERY($\mathbf{q}, \sigma_{\mathbf{q}}$)
  $\{\boldsymbol{\mu}, \boldsymbol{\Sigma}\} \leftarrow$ COMPUTE STAT($\mathbf{q}, \sigma_{\mathbf{q}}, \eta_{root}$)
  Compute $\hat{\boldsymbol{\mu}}$ and $\hat{\boldsymbol{\Sigma}}$ from $\{\boldsymbol{\mu}, \boldsymbol{\Sigma}\}$ using Eq.(5)
  return $\{\hat{\boldsymbol{\mu}}, \hat{\boldsymbol{\Sigma}}\}$
end function

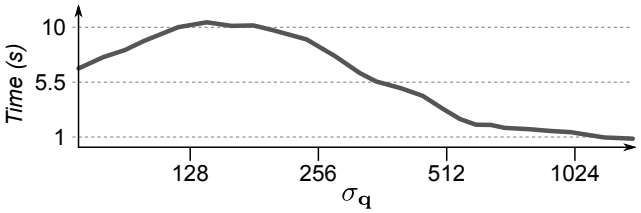


Figure 5. Computation time (in seconds) of $5 \cdot 10^5$ queries (excluding the building step which is run only once) of our cov-tree at different scales $\sigma_{\mathbf{q}}$. Time is measured by specifically constraining the use of a single core.

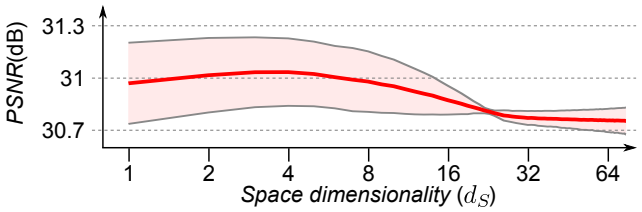


Figure 6. PSNR (with respect to ground-truth) of NLB denoising using our cov-tree with PCA dimensionality reduction in the spatial domain. We measured the $\mathrm{PSNR}_i(d_S)$ on 20 test images i for different values of the spatial dimension d_S. The mean curve $\overline{\mathrm{PSNR}}(d_S)$ is represented in red and in gray the standard deviations of the centered curves $\mathrm{PSNR}_i(d_S) - \overline{\mathrm{PSNR}}_i$. Although the absolute levels of PSNR present a large variation (between 29 and 34 dB) among all images, the shape of the $\mathrm{PSNR}_i(d_S)$ curve as a function of d_S is the same for all images and shows a peak around $d_S = 4$ in agreement with the findings of [16] for NLM.

code[3] which most often takes advantage of only one core.

4.1. Non Local Bayes Filtering

Non-local Bayes denoising [9][4] is a bayesian MAP estimation from noisy image patches $\tilde{\mathbf{p}}$. [5] The prior multivariate Gaussian model for the clean patch is estimated from a neighborhood $N_{\tilde{\mathbf{p}}}$ of noisy patches, so the estimated covariance matrix $\mathbf{\Sigma}_{\tilde{\mathbf{p}}}$ is also corrupted by noise. When we combine the denoising of this covariance matrix with the MAP estimation in equation (2), we obtain the estimated (denoised) patch as

$$\hat{\mathbf{p}} = \mu_{\tilde{\mathbf{p}}} + [\mathbf{\Sigma}_{\tilde{\mathbf{p}}} - \sigma_n^2 Id]\mathbf{\Sigma}_{\tilde{\mathbf{p}}}^{-1}(\tilde{\mathbf{p}} - \mu_{\tilde{\mathbf{p}}}) \qquad (6)$$

The neighborhood $N_{\tilde{\mathbf{p}}}$ is not only restricted to patches $\tilde{\mathbf{p}}_i$ that are close to $\tilde{\mathbf{p}}$ in the feature space ($R = \mathbb{R}^{3n^2}$ for $n \times n$ patches of color images), but also to those associated to nearby pixels (x_i, y_i). This restriction makes the search for similar patches tractable in very large images, since the search region does not grow with image size. Using the cov-tree, however, we can afford such large search regions without any significant performance penalty.

We propose three variants of NLB denoising which only differ in the choice of the neighborhood $N(\tilde{\mathbf{p}})$:

Global Search. The spatial domain $S = R = \mathbb{R}^{3n^2}$ is the same as the feature space, so the neighborhood $N_{\tilde{\mathbf{p}}}$ only depends on $\tilde{\mathbf{p}}$.

Local Search. In order to reproduce the neighborhood of the original NLB we augment the spatial domain with the pixel coordinates (x_i, y_i), so $S = \mathbb{R}^{d_S}, d_S = 3n^2 + 2$.

Compressed Local Search. As an adaptation of [16] to NLB, we can reduce the spatial domain S to the first 6 principal components in a global PCA analysis of all noisy patches in the image.

Figure 7 presents the denoising results of the original NLB compared to our local variant based on the cov-tree. Our local approach (e) produces better results than the original NLB (d). We can explain it by our use of the approximate Gaussian kernel ϕ as a weighting function. In the original NLB, means and covariance matrices are estimated by averaging the k nearest neighbors with a constant weight. Consequently, covariance matrices can be more strongly affected by outliers.

As shown in Figure 6, PCA dimensionality reduction over the S domain not only accelerates the search (as expected), but also produces better performances. The latter can be attributed to the denoising effect of dimensionality reduction, which improves the relevance of the computed neighborhoods $N(\tilde{\mathbf{p}})$.

Figure 5 shows that the time required to solve a query in the cov-tree is only mildly affected by the neighborhood size $\sigma_{\mathbf{q}}$ of the query. Indeed, as shown in Figure 4, coarse-sized queries explore a lot of nodes in breadth but stop the search at the top of the tree while a fine-sized query explores fewer nodes in breadth but explores the tree in depth. Consequently, the number of nodes (and hence the computational complexity) for each radius size is equivalent, except for medium sized queries which involve the largest number of nodes.

4.2. Data Driven Image Denoising

In [11] the use of huge image databases is advocated as a way to learn the prior underlying natural image patches. Their procedure (called shotgun NLM in [10]) was shown to serve as a way to estimate the fundamental limits of non-local image denoising methods, but no attempt was done to make the computation time actually tractable for real appli-

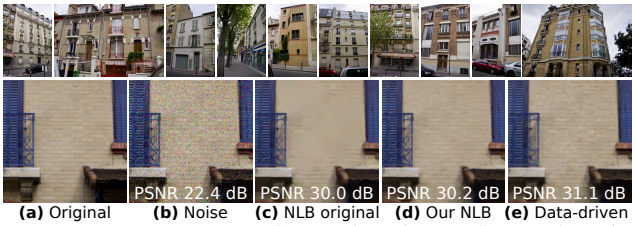
[3] The querying process can be parallelized by using one thread for each query. A similar solution can be used for the learning process but care must be taken to avoid concurrent writing to the same cell. Parallelizing the building step is more involved but has less impact since this is the least time-consuming part of the pipeline.

[4] for simplicity we only consider NLB's first stage here

[5] where noise is zero-mean Gaussian with variance σ^2

(a) Original **(b)** Noise std. dev. 0.2 **(c)** Accelerated NLM (6+2-D) **(d)** NLB original (147+2-D) **(e)** Our filter (6+2-D)

Figure 7. *Non-local filtering*: We use our CovTree to learn 7×7 RGB patches extracted from the noisy image *(b)*. The original patch dimensionality (147-D) is reduced to 6-D using PCA as suggested for NLM by [16]. When compared to an accelerated implementation of NLM [7] *(c)* or the original NLB *(d)*, our filter *(e)* better recovers features thanks to the use of an approximate Gaussian kernel ϕ.



(a) Original **(b)** Noise **(c)** NLB original (147+2-D) **(d)** Our NLB (147+2-D) **(e)** Data-driven NLB (147-D)

Figure 8. *Data driven image denoising*: We use our CovTree to learn from about 10^8 clean 7×7 RGB patches extracted from an image database (*first line*) to denoise an image corrupted by a noise of standard deviation of 0.1 *(b)* by exploiting the prior underlying natural images. Compared to the original image *(a)*, our data driven filter *(e)* better preserves features than using the original NLB filter *(c)* or our single-image CovTree-NLB filter *(d)*.

cations. A similar idea was proposed in [19] for a PLE-like algorithm with 200 Gaussians, thus requiring many days to learn the prior on a largedatabase. In this section we apply the same idea to implement a "shotgun NLB" denoising algorithm, but using the cov-tree to make both learning and restoration computationally tractable with large learning databases.

Indeed, we propose to use an external noiseless image database instead of noisy patches to denoise a noisy image. This idea has two main benefits: we can increase the number of learned patches (without a significant penalty in computation time or memory use) and data is not degraded by noise, thus increasing the precision of the estimated anisotropic Gaussian.

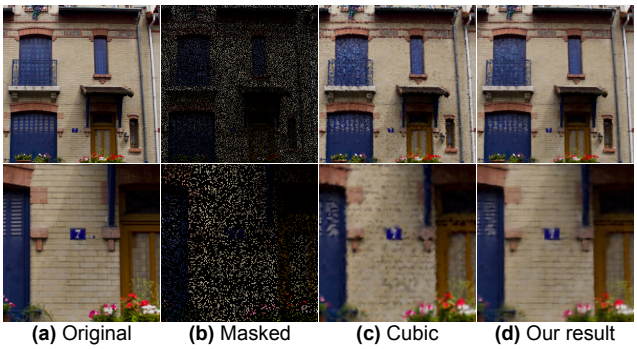
In practice, we build the tree at a scale $\sigma_b = \sigma_n$ by considering the colored noised patches (without the pixel coordinate) to fix the hierarchy of spatial domain cells. Then, we learn the corresponding range-domain Gaussian models from the database of noiseless colored patches. Finally, the covariance matrix $\mathbf{\Sigma}_{\tilde{\mathbf{p}}}$ and the mean vector $\mu_{\tilde{\mathbf{p}}}$ are estimated from a noisy patch $\tilde{\mathbf{p}}$ with $\sigma_q = \sigma_n$. In contrast to the previous section, the estimated anisotropic gaussians $\mathcal{N}(\mu_{\tilde{\mathbf{p}}}, \mathbf{\Sigma}_{\tilde{\mathbf{p}}})$ are noiseless, consequently, we applied equation (2) directly, without need for denoising the covariance matrix.

Figure 8 shows our result of denoising an image of a façade using a database of noiseless images of similar façades in the same city (but not the same façade). As expected, the database-driven denoising performs better. More extensive experimentation is needed to check if this reconstruction is actually close to the fundamental limits announced in [11] .

Our database contains about 10^8 patches and the learning phase takes about 5 hours and 8GB of RAM to hold the data-structure. This is several orders of magnitude faster than the times reported in [11, 19], while our database is also larger. Reconstruction takes about 5 minutes.

4.3. Reconstruction from random samples

One of the most visually striking applications of Bayesian MAP estimation with a Gaussian mixture prior model for image patches, is the reconstruction of an image, from a small random subset of its pixels (20% in our case), as showcased in [18] among others. Let's consider a patch $\mathbf{q}$ sampled by a known random sampling operator $\tilde{\mathbf{q}} = S\mathbf{q}$. As before, we consider that image patches are locally modeled as an anisotropic Gaussian distribution, with mean $\mu_{\tilde{\mathbf{q}}}$ and covariance $\mathbf{\Sigma}_{\tilde{\mathbf{q}}}$ estimated from the local dictionary. We assume that $\hat{\mathbf{q}}_0$ is an initial estimate of the complete patch (obtained by a cubic interpolation over the Delaunay triangulation). This patch serves as a query to extract a local statistical prior for $\mathbf{q}$ from the cov-tree. Then by Bayesian



(a) Original **(b)** Masked **(c)** Cubic **(d)** Our result

Figure 9. *Data driven reconstruction from sparse samples*: We obtain from an original image *(a)* a masked image by extracting randomly 20% of the original pixels *(b)*. By learning about 10^8 patches with our CovTree, we reconstruct an image *(d)* by applying Eq. (7) with a coarse-to-fine query scales, and starting from a cubic interpolation *(c)*.

MAP estimation, we write the recovered patch $\hat{\mathbf{q}}_i$ as :

$$\hat{\mathbf{q}}_{i+1} = (\mathbf{\Sigma}_{\hat{\mathbf{q}}_i} S^H S + \frac{\sigma_i^2}{2} Id)^{-1} (\mathbf{\Sigma}_{\hat{\mathbf{q}}_i} S^H \tilde{\mathbf{q}} + \frac{\sigma_i^2}{2} \mu_{\hat{\mathbf{q}}_i}) \quad (7)$$

Combining all reconstructed patches by aggregation, we obtain a first reconstruction I_1 from the interpolated image I_0. The patches $\hat{\mathbf{q}}_i$ in this first reconstruction $(i = 1)$ can in turn be used as an initialisation/query to obtain more accurate prior and reconstruction $\hat{\mathbf{q}}_{i+1}$. We iterate the process with until $\hat{\mathbf{q}}_i$ equals $\hat{\mathbf{q}}_{i+1}$ by using a coarse-to-fine query scale $\sigma_i = \alpha^i \sigma_0$ where $\alpha \in [0, 1]$ is a scale factor. Figure 9 presents the result of our approach over an image were only 20% random samples were retained. In our experiments $\alpha = 0.8$ and $\sigma_0 = 0.65$

5. Conclusion and perspectives

We proposed a novel data structure and associated algorithms that allow to deal with a continuous family of local multivariate Gaussian models: both efficient learning over a large database and fast queriyng are supported. The extracted model varies continuously over a range domain R when the query point varies over a (possibly different) spatial domain S, and different scale levels can be specified by the query resulting in various degrees of spatial locality of the extracted statistical model.

The relevance of such a data structure is motivated by image restoration problems via Bayesian MAP with local Gaussian priors on the image patches. Such models became incresingly sucessful in image processing during the last 5 years, because they are very close to an accurate statistical model of natural images. However, progress in this area has been slow because we lack the required tools for efficiently handling the massive amounts of data that need to be fed to learn these models in order to approach optimal results. The cov-tree is an attempt to fill this gap.

Given the genericity of the cov-tree, its applicability reaches potentially far beyond image restoration. In the supplementary material we include an application that improves an NLM-like algorithm for denoising and interpolation of 3D point clouds [8]. In this case a data structure is definitely required to index and query a set of patches even for point cloud of 10^6 points. The reason is that common acceleration and search techniques that are used by non-local methods in 2D image processing do not apply to irregular 3D point sets over a surface that has no implicit parameterization.

Acklowedgements This work has been partially funded by the EC under contracts FP7-287723 REVERIE, FP7-323567 HARVEST4D, by the French government under ANR iSpace&Time, FUI CEDCA and CNES R&T 128435 projects.

References

[1] A. Adams, J. Baek, and M. A. Davis. Fast high-dimensional filtering using the permutohedral lattice. In *CGF*, volume 29, pages 753–762, 2010.

[2] A. Adams, N. Gelfand, J. Dolson, and M. Levoy. Gaussian kd-trees for fast high-dimensional filtering. *TOG*, 28(3):21, 2009.

[3] V. Aurich and J. Weule. Non-linear gaussian filters performing edge preserving diffusion. In *Mustererkennung 1995*, pages 538–545. 1995.

[4] S. Awate and R. Whitaker. Higher-order image statistics for unsupervised, information-theoretic, adaptive, image filtering. In *CVPR*, volume 2, pages 44–51, 2005.

[5] A. Buades, B. Coll, and J.-M. Morel. A non-local algorithm for image denoising. In *CVPR*, volume 2, pages 60–65, 2005.

[6] H. Fuchs, Z. M. Kedem, and B. F. Naylor. On visible surface generation by a priori tree structures. In *SIGGRAPH*, volume 14, pages 124–133, 1980.

[7] E. S. Gastal and M. M. Oliveira. Adaptive manifolds for real-time high-dimensional filtering. *TOG*, 31(4):33, 2012.

[8] T. Guillemot, A. Almansa, and T. Boubekeur. Non local point set surfaces. In *3DIMPVT*, pages 324–331, 2012.

[9] M. Lebrun, A. Buades, and J.-M. Morel. Implementation of the "Non-Local Bayes" Image Denoising Algorithm. *IPOL*, 2013:1–42, 2013.

[10] M. Lebrun, M. Colom, A. Buades, and J. Morel. Secrets of image denoising cuisine. *Acta Numerica*, 21:475–576, 2012.

[11] A. Levin and B. Nadler. Natural image denoising: Optimality and inherent bounds. In *CVPR*, pages 2833–2840, 2011.

[12] J. Mairal, F. Bach, J. Ponce, G. Sapiro, and A. Zisserman. Non-local sparse models for image restoration. In *ICCV*, pages 2272–2279, 2009.

[13] G. Motta, E. Ordentlich, I. Ramirez, G. Seroussi, and M. J. Weinberger. The dude framework for continuous tone image denoising. In *ICIP*, volume 3, pages III–345, 2005.

[14] S. Paris and F. Durand. A fast approximation of the bilateral filter using a signal processing approach. *IJCV*, 81(1):24–52, 2009.

[15] S. M. Smith and J. M. Brady. Susan—a new approach to low level image processing. *IJCV*, 23(1):45–78, 1997.

[16] T. Tasdizen. Principal components for non-local means image denoising. In *ICIP*, pages 1728–1731, 2008.

[17] C. Tomasi and R. Manduchi. Bilateral filtering for gray and color images. In *ICCV*, pages 839–846, 1998.

[18] G. Yu, G. Sapiro, and S. Mallat. Solving inverse problems with piecewise linear estimators: from gaussian mixture models to structured sparsity. *TIP*, 21(5):2481–2499, 2012.

[19] D. Zoran and Y. Weiss. From learning models of natural image patches to whole image restoration. In *ICCV*, pages 479–486, 2011.