Sparse PDF Maps for Non-Linear Multi-Resolution Image Operations

Markus Hadwiger
KAUST

Ronell Sicat
KAUST

Johanna Beyer
KAUST

Jens Krüger
IVDA, DFKI, Intel VCI

Torsten Möller
Simon Fraser University

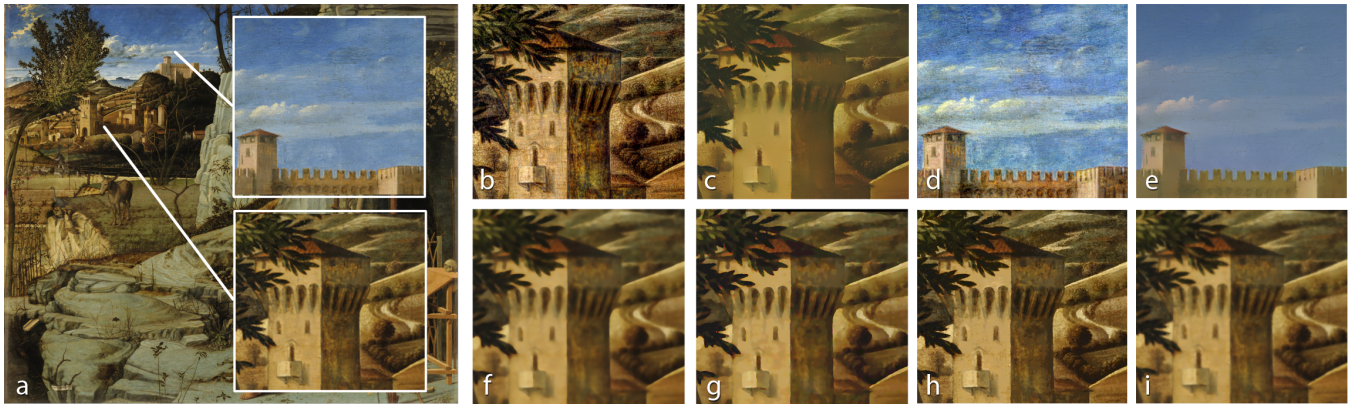

Figure 1: ***sPDF-maps*** *are a compact multi-resolution image pyramid data structure that sparsely encodes pre-computed pixel neighborhood probability density functions (pdfs) for all pixels in the pyramid. They enable the accurate, anti-aliased evaluation of non-linear image operators directly at any output resolution. A variety of operators can be computed at run time from the same pre-computed data structure in a way that scales to gigapixel images, such as local Laplacian filters for (b,d) detail enhancement or (c,e) smoothing, (f) median filters, (g) dominant mode filters, (h) maximum mode filters, (i) bilateral filters. The original image (a) has resolution* $16,898 \times 14,824$ *(250 Mpixels).*

Abstract

We introduce a new type of multi-resolution image pyramid for high-resolution images called *sparse pdf maps* (sPDF-maps). Each pyramid level consists of a sparse encoding of continuous probability density functions (pdfs) of pixel neighborhoods in the original image. The encoded pdfs enable the accurate computation of non-linear image operations directly in any pyramid level with proper pre-filtering for anti-aliasing, without accessing higher or lower resolutions. The sparsity of sPDF-maps makes them feasible for gigapixel images, while enabling direct evaluation of a variety of non-linear operators from the same representation. We illustrate this versatility for anti-aliased color mapping, $\mathcal{O}(n)$ local Laplacian filters, smoothed local histogram filters (e.g., median or mode filters), and bilateral filters.

CR Categories: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Color, shading, shadowing, and texture;

Keywords: image pyramids, mipmapping, anti-aliasing, multi-resolution filtering, display-aware filtering, smoothed local histogram filtering, bilateral filtering, local Laplacian filtering

Links: DL PDF

1 Introduction

The recent increase in the resolution of acquired and computed image data has resulted in a need for novel multi-resolution techniques, e.g., representing, processing, and rendering gigapixel images [Kopf et al. 2007; Summa et al. 2010]. One of the most prevalent types of multi-resolution image hierarchies are pyramid representations, such as mipmaps [Williams 1983], or Gaussian pyramids [Burt and Adelson 1983]. Image pyramids store pre-filtered and downsampled versions of the original image, where the pre-filtering is crucial for avoiding aliasing. Because standard pre-filters are linear operators, further linear operators (e.g., further smoothing) can be applied accurately in any pre-computed pyramid level, as they commute with the pre-filter. However, this does not apply to non-linear operators, such as color mapping or edge-preserving filters. In this case, one has to process the original image and re-compute the pyramid. This is impractical for gigapixel images, especially when the goal is the interactive display of processed images at a lower output resolution, which is becoming more important as the gap between the acquired image and display resolutions widens. Therefore, in such a scenario all operators are usually applied directly at the output resolution. Nevertheless, downsampling the image first using a standard pre-filter, e.g., bicubic, followed by a non-linear operation, can introduce false colors. These artifacts due to resampling are a form of aliasing, though not of the same kind as the well-known moiré patterns. A naive solution would be to remove the pre-filter and resort to nearest-neighbor downsampling, but then moiré patterns would appear.

This paper introduces *sparse pdf maps* (sPDF-maps), which are probabilistic image pyramids that sparsely encode probability density functions (pdfs) of local pixel neighborhoods. Our main goals are (1) the accurate evaluation of non-linear operators at any output resolution, and (2) scalability to gigapixel images. The sPDF-map representation is computed in a pre-computation stage. However, at run time different operators with arbitrary parameter settings can be evaluated interactively. sPDF-maps enable the accurate evaluation of non-linear operators directly at any resolution without accessing

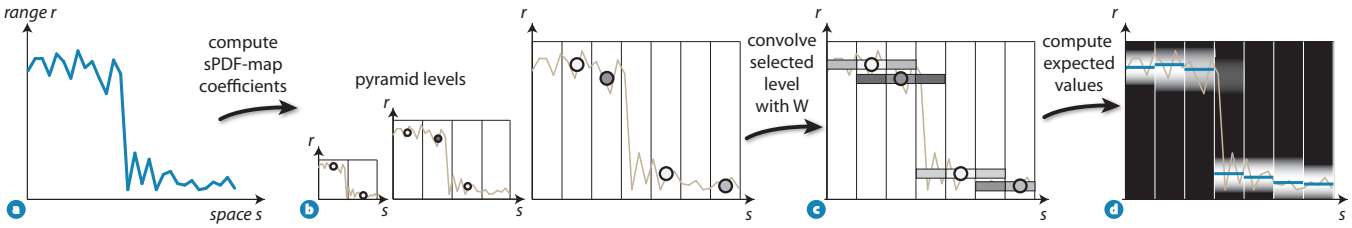

Figure 2: ***Non-linear operator evaluation from pdfs encoded as sPDF-map coefficients.*** *The original signal (a) is represented in coarser pyramid levels by coefficients in the (space × range) domain (b). A non-linear operator, such as bilateral filtering, is evaluated via (c) spatial convolution of the coefficients, and (d) computing the expectation of each pixel (blue) as the sum of look-ups in pre-computed 1D functions.*

higher or lower resolutions. This property makes our approach fully scalable to gigapixel images. Efficient out-of-core processing and rendering are greatly facilitated by only having to load the visible image tiles of the desired output resolution. In this context, a drawback of other multi-scale representations such as wavelets, e.g., [Fattal 2009], is that they always require accessing multiple scales.

Each pyramid level of an sPDF-map comprises a set of coefficients that sparsely represent the 2D image domain combined with its 1D range (i.e., intensities). Such a combined 3D domain was previously exploited for fast bilateral filtering [Paris and Durand 2006; Chen et al. 2007]. Non-linear operators in the 2D image domain can then be evaluated from linear convolutions in the 3D sPDF-map domain. We compute the sPDF-map coefficients via Matching Pursuit [Mallat and Zhang 1993], which is common in sparse signal representation. In fact, our representation is so sparse that we usually require no more coefficients than image pixels. Despite this compactness, our representation implicitly represents the entire continuous 1D pdf associated with each image pixel. See Fig. 2 for an overview.

We present a general method that uses the sPDF-map coefficients for the efficient and accurate evaluation of a variety of non-linear operators in the combined (space × range) domain, with the range being *continuous* (Fig. 2). We introduce the sPDF-map data structure, which compactly stores the coefficients in a manner similar to standard image pyramids. This facilitates efficient parallel implementation on GPUs. We demonstrate the versatility of sPDF-maps for color mapping, edge-aware processing with local Laplacian filters [Paris et al. 2011], smoothed local histogram filters [Kass and Solomon 2010], and bilateral filters [Chen et al. 2007]. Our method unifies all of these operations, enabling their computation with arbitrary parameters from the same pre-computed data structure.

Main contribution. Our work is the first to consistently and compactly store a pre-computed representation of the continuous 1D probability distribution describing the neighborhood of each image pixel in each level of an image pyramid, while providing efficient and easy means for directly using this pdf representation for non-linear operator evaluation anywhere in this multi-resolution hierarchy.

2 Related Work

Mipmaps [Williams 1983] are pre-computed texture pyramids that enable efficient pre-filtering for anti-aliasing in texture mapping [Heckbert 1989]. In contrast to using multi-scale decompositions for image processing, mipmaps focus on using any desired resolution directly. This is also an important property of sPDF-maps, but the encoding of probability distributions enables us to faithfully apply non-linear operators after the pre-computed pre-filtering.

Image pyramids such as Gaussian and Laplacian pyramids [Burt and Adelson 1983] target multi-scale analysis and filtering. Multi-scale image decompositions can also be computed via edge-preserving filters, e.g., for shape and detail enhancement [Fattal et al. 2007]. A very powerful family of multi-scale representations are wavelets [Mallat 2009], which can be adapted to better observe edges, e.g., edge-avoiding wavelets [Fattal 2009], or *à-trous* wavelets [Hanika et al. 2011]. Our goal, however, is the direct evaluation of non-linear operators at any desired output resolution, which enables their scalable evaluation for gigapixel images [Kopf et al. 2007], whose display, filtering, and processing is an active area of current research [Kazhdan and Hoppe 2008; Summa et al. 2010].

Anti-aliasing. Our main goal is preserving the non-linearity of image operations in a pyramid of images that have been pre-filtered for anti-aliasing [Crow 1977]. Our approach is conceptually equivalent to knowing subpixel coverage and weighting subpixel contributions accordingly [Carpenter 1983]. For shadow mapping, this is equivalent to percentage-closer filtering [Reeves et al. 1987]. We represent the entire range of pixel values, although we "forget" the exact pixel locations like locally orderless images [Koenderink and Van Doorn 1999]. In contrast, many methods represent only Gaussian distributions faithfully [Donnelly and Lauritzen 2006; Younesy et al. 2006].

Non-linearity of shading. An important non-linear operation in graphics is shading. Computing pre-filtered normal maps is a hard problem. Approaches in this area that are similar to ours are fitting the underlying distribution of normals [Han et al. 2007], or the distribution of reflectance (BRDFs) [Tan et al. 2008]. Both of these approaches use expectation maximization (EM) for fitting, whereas we obtain a sparse representation via Matching Pursuit [Mallat and Zhang 1993]. A major difference of sPDF-maps is that they exploit coherence between neighboring pixels in the 3D (space × range) domain, which leads to very few coefficients, whereas all previous approaches are fitting the distribution of each pixel individually.

Local Laplacian filters enable powerful edge-preserving filtering via simple non-linear operators [Paris et al. 2011]. sPDF-maps allow these filters to be evaluated for gigapixel images at near-interactive rates, by computing each Laplacian pyramid coefficient directly. This results in $\mathcal{O}(n)$ complexity, instead of the original $\mathcal{O}(n \log(n))$. Recent work by Aubry et al. [2011] also presents an $\mathcal{O}(n)$ approach, however at the cost of storing multiple pre-computed pyramids.

Bilateral filtering [Tomasi and Manduchi 1998] is a non-linear weighting of the neighborhood $\mathcal{N}(\boldsymbol{p})$[1] of a pixel $\boldsymbol{p}$, with intensity $I_{\boldsymbol{p}}$, with Gaussians G_{σ_s} (spatial weight) and G_{σ_r} (range weight):

$$I_{\boldsymbol{p}}^{\text{bilat.}} = \frac{1}{w_{\boldsymbol{p}}} \sum_{\boldsymbol{q} \in \mathcal{N}(\boldsymbol{p})} G_{\sigma_s}\left(\|\boldsymbol{p}-\boldsymbol{q}\|\right) G_{\sigma_r}\left(|I_{\boldsymbol{p}} - I_{\boldsymbol{q}}|\right) I_{\boldsymbol{q}}, \qquad (1)$$

normalizing with the sum of all weights $w_{\boldsymbol{p}}$. This equation is non-linear, because the intensity $I_{\boldsymbol{q}}$ appears in the argument of $G_{\sigma_r}(\cdot)$. It can be computed from linear convolutions by going from the

[1] Technically, the sum is over the whole image, but we use a neighborhood $\mathcal{N}(\boldsymbol{p})$ in 2D ($\mathcal{N}_3(\boldsymbol{p}, I_{\boldsymbol{p}})$ in 3D) to be closer to the actual implementation.

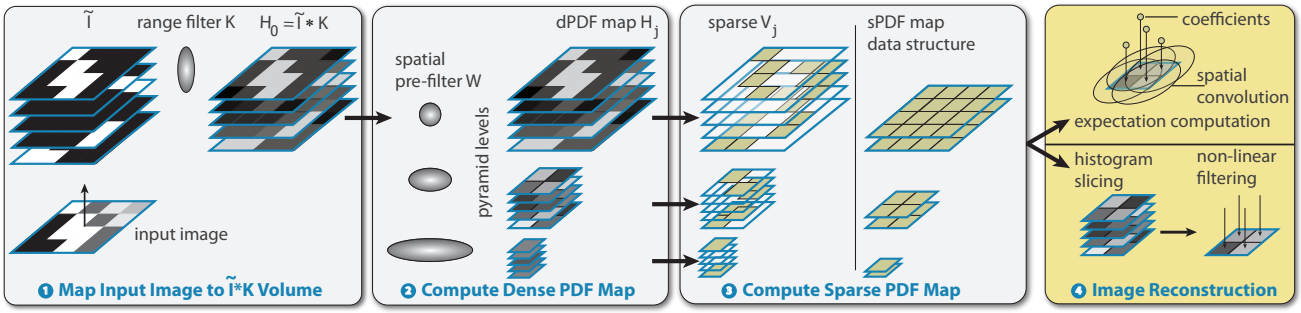

Figure 3: ***Overview of the sPDF-map pipeline.*** *The input image is lifted from the 2D spatial domain into the 3D (space × range) domain. We then apply range smoothing via the filter K. Next, we compute an intermediate dPDF-map H_j, by iteratively applying the spatial pre-filter W and downsampling. The dPDF-map is then transformed into an sPDF-map using Matching Pursuit. Each level of an sPDF-map comprises sPDF-map coefficients stored as a pair of* index *and* coefficient *images. At run time, non-linear operators are evaluated using these coefficients.*

2D spatial domain to the 3D (space × range) domain, and using homogeneous notation for normalization [Paris and Durand 2006]:

$$\left[w_{\boldsymbol{p}} I_{\boldsymbol{p}}^{\text{bilat.}}, w_{\boldsymbol{p}}\right] = \sum_{(\boldsymbol{q},I_b)\in\mathcal{N}_3(\boldsymbol{p},I_{\boldsymbol{p}})} G_{(\boldsymbol{\sigma}_s,\sigma_r)}\big((\boldsymbol{p},I_{\boldsymbol{p}})-(\boldsymbol{q},I_b)\big)\,\tilde{I}\,(\boldsymbol{q},I_b), \quad (2)$$

with quantized intensity bins b ($1 \le b \le B$) with intensities I_b, and $\tilde{I}(\boldsymbol{q},I_b) = [I_{\boldsymbol{q}},1]$, if $I_{\boldsymbol{q}}$ maps to bin b, and $[0,0]$ otherwise. $G_{(\boldsymbol{\sigma}_s,\sigma_r)} = (G_{\sigma_s} \otimes G_{\sigma_s}) \otimes G_{\sigma_r}$, where $\otimes$ is the tensor product.

sPDF-maps also operate in the (space × range) domain. However, they represent a continuous range axis, which we denote as r (instead of I_b), whereas a bilateral grid [Chen et al. 2007] quantizes the range with B bins. In fact, each pyramid level of an sPDF-map can be dynamically sampled into a bilateral grid of any range quantization. Also, the bilateral grid leverages coarse-resolution computations for higher performance, whereas we are interested directly in coarser output images, like display-aware approaches [Jeong et al. 2011].

Histograms are often employed as a compact, quantized representation of distributions. For example, Thompson et al. [2011] use pixel histograms (*hixels*) in the context of representing data uncertainty.

Smoothed local histograms estimate the local distribution of pixel values, which enables a variety of non-linear filters [Kass and Solomon 2010]. Channel smoothing [Felsberg et al. 2006] is a similar robust filter. We compute normalized smoothed local histograms for the initial estimation of sampled pdfs, but our sPDF-maps representation then only stores sparse coefficients that represent continuous pdfs. The smoothed local histogram of a pixel $\boldsymbol{p}$ is defined as:

$$h_{\boldsymbol{p}}(I_b) = \sum_{\boldsymbol{q}\in\mathcal{N}(\boldsymbol{p})} W\,(\boldsymbol{p}-\boldsymbol{q})\,K\,(I_b - I_{\boldsymbol{q}}), \quad (3)$$

where the kernel K sums to one and smoothes the histogram, and the kernel W performs smoothing in the spatial domain [Kass and Solomon 2010]. We will exploit that Eq. 3 can also be considered in the continuous range r, instead of quantized histogram bins b.

3 Method Overview

Basic intuition. Our goal is to apply non-linear image operators at coarse image levels without sacrificing accuracy. Similarly to the linearity of Eq. 2 vs. the non-linearity of Eq. 1, this is possible by pre-filtering and downsampling the 3D (space × range) domain of an image, instead of its 2D spatial domain. We therefore substitute each pixel $\boldsymbol{p}$ by a 1D function $pdf_{\boldsymbol{p}}(r)$ that describes the entire range r.

Basic setup. We consider a given pixel $\boldsymbol{p}$ with neighborhood $\mathcal{N}(\boldsymbol{p})$. We take a probabilistic viewpoint, and consider the pixel value at $\boldsymbol{p}$ to be a continuous random variable $X_{\boldsymbol{p}}$. We describe the distribution of $X_{\boldsymbol{p}}$ by its probability density function $pdf_{\boldsymbol{p}}(r)$, informed by the pixel values in the neighborhood $\mathcal{N}(\boldsymbol{p})$. We then have two basic goals: First, we want to come up with a compact representation for the $pdf_{\boldsymbol{p}}(r)$ of every $\boldsymbol{p}$ that, despite its compactness, allows for easy access. Second, given this representation, we want to use it directly for the evaluation of a variety of non-linear image operations.

Non-linear image operations. Given $pdf_{\boldsymbol{p}}(r)$, we will show that we can compute the result of many non-linear operations as the expectation of a suitably chosen function $t_{\boldsymbol{p}}$ applied to the random variable $X_{\boldsymbol{p}}$, using an appropriate normalization factor $w_{\boldsymbol{p}}$:

$$E\left[t_{\boldsymbol{p}}\left(X_{\boldsymbol{p}}\right)\right] = \frac{1}{w_{\boldsymbol{p}}}\int_0^1 t_{\boldsymbol{p}}(r) pdf_{\boldsymbol{p}}(r)\,dr. \quad (4)$$

With $t_{\boldsymbol{p}}(r) = r$ and $w_{\boldsymbol{p}} = 1$, Eq. 4 simply computes $E\left[X_{\boldsymbol{p}}\right]$, the expected value of the random variable $X_{\boldsymbol{p}}$. In order to gain intuition, we consider evaluating a bilateral filter. We can estimate $pdf_{\boldsymbol{p}}(r)$ as the normalized smoothed local histogram $h_{\boldsymbol{p}}(r)$ (Eq. 3), with the spatial filter $W = (G_{\sigma_s} \otimes G_{\sigma_s})$ (see Eq. 1), and K as the Dirac-delta $\delta_0(r)$. We can then evaluate a bilateral filter using Eq. 4 with:

$$t_{\boldsymbol{p}}(r) = r \cdot G_{\sigma_r}\left(|I_{\boldsymbol{p}} - r|\right), \quad \text{and} \quad w_{\boldsymbol{p}} = \int_0^1 \frac{t_{\boldsymbol{p}}(r)}{r} pdf_{\boldsymbol{p}}(r)\,dr, \quad (5)$$

using the range weight G_{σ_r} from Eq. 1. In this way, the spatial smoothing via G_{σ_s} is contained in $pdf_{\boldsymbol{p}}(r)$, while the range weighting of the bilateral filter is contained in the particular choice of $t_{\boldsymbol{p}}(r)$.

Smoothed histogram volumes. In order to prepare for carrying out operations in 3D instead of in 2D, we define a smoothed histogram volume $H(\boldsymbol{p},r)$ as:

$$H(\boldsymbol{p},r) = \frac{1}{w_{\boldsymbol{p}}} h_{\boldsymbol{p}}(r), \quad \text{with} \quad w_{\boldsymbol{p}} = \int_0^1 h_{\boldsymbol{p}}(r)\,dr, \quad (6)$$

with $h_{\boldsymbol{p}}(r)$ defined by Eq. 3, in the continuous range r, and $w_{\boldsymbol{p}}$ used to guarantee proper normalization.

sPDF-map coefficient volumes. In analogy to the volume $\tilde{I}(\boldsymbol{p},I_b)$ in Eq. 2, we define a sparse volume $V(\boldsymbol{p},r)$ that is non-zero only at specific positions $(\boldsymbol{p}_n,r_n)$, i.e., $V(\boldsymbol{p}_n,r_n) = c_n$ with $c_n \neq 0$. We then represent V solely by the set of tuples $(\boldsymbol{p}_n,r_n,c_n)$, which we

call sPDF-map coefficients (see Fig. 2). We compute a $V(\boldsymbol{p}, r)$ that is as sparse as possible, while approximating $H(\boldsymbol{p}, r)$ well via:

$$H(\boldsymbol{p}, r) \approx \sum_{\substack{(\boldsymbol{q}_n, r_n, c_n) \\ \boldsymbol{q}_n \in \mathcal{N}(\boldsymbol{p})}} (W \otimes K)\big((\boldsymbol{p}, r) - (\boldsymbol{q}_n, r_n)\big) V(\boldsymbol{q}_n, r_n) = V * (W \otimes K), \quad (7)$$

where $*$ denotes a convolution, and $(W \otimes K)$ is a 3D kernel.

A crucial property of our approach is that coefficients influence a neighborhood of pixels $\mathcal{N}(\boldsymbol{p})$ as determined by the spatial kernel W, instead of individual pixels. This reduces the number of required coefficients by exploiting coherence in the (space $\times$ range) domain. Sec. 5 describes how we compute $V(\boldsymbol{p}, r)$. For now, we assume that we know the set of sPDF-map coefficients $(\boldsymbol{p}_n, r_n, c_n)$.

Non-linear operations with sPDF-map coefficients. Instead of computing the $pdf_{\boldsymbol{p}}(r) = H(\boldsymbol{p}, r)$ explicitly from $V(\boldsymbol{p}, r)$, we plug the convolution from Eq. 7 directly into Eq. 4. Also exploiting the separability of $(W \otimes K)$ then allows us to rewrite Eq. 4 as:

$$E[t_{\boldsymbol{p}}(X_{\boldsymbol{p}})] = \frac{1}{w_{\boldsymbol{p}}} \int_0^1 \tilde{t}_{\boldsymbol{p}}(r)\,(V * W)\; dr, \quad \text{with} \quad \tilde{t}_{\boldsymbol{p}} = t_{\boldsymbol{p}} * K. \quad (8)$$

This formulation has moved the range convolution with K into the new function $\tilde{t}_{\boldsymbol{p}}$. Since $V(\boldsymbol{p}, r)$ only consists of sparse coefficients $(\boldsymbol{p}_n, r_n, c_n)$, Eq. 8 can be computed exactly as the sum:

$$E[t_{\boldsymbol{p}}(X_{\boldsymbol{p}})] = \frac{1}{w_{\boldsymbol{p}}} \sum_{\substack{(\boldsymbol{q}_n, r_n, c_n) \\ \boldsymbol{q}_n \in \mathcal{N}(\boldsymbol{p})}} \tilde{t}_{\boldsymbol{p}}(r_n) W(\boldsymbol{p} - \boldsymbol{q}_n)\, V(\boldsymbol{q}_n, r_n). \quad (9)$$

In order to evaluate a non-linear operator, we therefore first define a suitable function $t_{\boldsymbol{p}}$ and convolve it with the range smoothing kernel K to obtain $\tilde{t}_{\boldsymbol{p}}$. Then, we evaluate Eq. 9, which is a simple sum over all sPDF-map coefficients, multiplying each coefficient with the spatial smoothing weight W, times the look-up result from $\tilde{t}_{\boldsymbol{p}}$.

Image pyramids store a pre-computed hierarchy of images smoothed and downsampled in the spatial domain. sPDF-maps are similar pyramids, but additionally smooth and represent the range r of the image. For each pixel $\boldsymbol{p}$, the function $pdf_{\boldsymbol{p}}(r)$ represents the pixel values in the neighborhood of $\boldsymbol{p}$ corresponding to its footprint [Greene and Heckbert 1986] in the original image (Fig. 4).

sPDF-maps pipeline overview. Our pipeline for computing and using sPDF-maps is illustrated in Fig. 3. The smoothing in the range and in the spatial domain, respectively, are computed in two different stages. The initial step computes a smoothed histogram for each pixel in the original image, with a spatial neighborhood of 1×1 pixel and without spatial smoothing. The required quantization rate is determined by the amount of range smoothing applied.

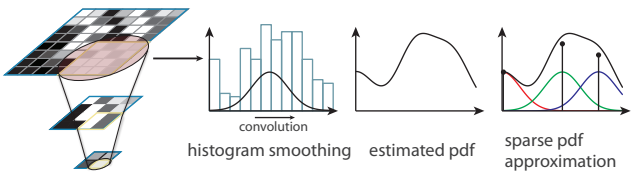

Figure 4: ***Pixel neighborhoods in sPDF-maps and dPDF-maps.*** *The pdf of each pixel represents the neighborhood (*footprint*) of the pixel in the original image. Continuous pdfs are initially estimated as discrete smoothed local histograms, but then represented by sPDF-map coefficients in the combined (space $\times$ range) domain.*

In the next step, an intermediate dense pdf map is computed by iteratively applying a spatial pre-filter and downsampling (Sec. 4). The pre-filter size is determined to match the desired footprint of each pixel (Fig. 4). These two steps together result in smoothed histograms that estimate the $pdf_{\boldsymbol{p}}(r)$ that we require. The dense pdf map is then converted into an sPDF-map level by level, by iteratively computing sPDF-map coefficients $(\boldsymbol{p}_n, r_n, c_n)$ (Sec. 5). After the sPDF-map has been computed, the dense pdf map is discarded.

4 Dense PDF Maps

We call a pyramid of smoothed histogram volumes $H_j(\boldsymbol{p}, r)$, consisting of pyramid levels j, a dense pdf map (dPDF-map). As in most image pyramids, from level j to $j+1$ we decrease the spatial resolution by a factor of two in each axis. However, the range axis r is always sampled with B samples, such that every H_j comprises B image layers. Each layer b corresponds to histogram bin b for all pixels in level j. The choice of B does not have to correspond to the quantization of the intensity axis of the original image. Since we perform range smoothing, a sampling rate that guarantees proper signal representation can be chosen depending on K (Eq. 3).

The dPDF-map is the first step to building an sPDF-map. It is a dense representation of the normalized local histograms. Later, we compute a sparse approximation of it, which is the sPDF-map that we actually use at run time.

We note that each H_j is similar to the homogeneous channel of a bilateral grid [Chen et al. 2007].

4.1 Pixel neighborhoods in dPDF-maps

The spatial filter W in a regular smoothed local histogram (Eq. 3) corresponds to the desired spatial neighborhood size and smoothing, as discussed by Kass and Solomon [2010]. However, for computing a dPDF-map, we define W to act as the spatial pre-filter for coarser pyramid levels. Thus, W defines the neighborhood of each pixel $\boldsymbol{p}$ as the footprint of $\boldsymbol{p}$ in the original image. This is illustrated in Fig. 4. In principle, any spatial pre-filter W can be chosen for dPDF-map computation. Due to their straightforward iterative evaluation, we use the 5×5 Gaussian-like filter of Burt and Adelson [1983], or the 2×2 box filter commonly used in mipmapping [Williams 1983].

4.2 Iterative dPDF-map computation

In principle, every pyramid level j of a dPDF-map can be computed directly from the original image. However, it is more efficient to compute the pyramid iteratively, i.e., computing each level j with $j > 0$ from $(j-1)$, where $j = 0$ corresponds to the original image resolution. This is analogous to the iterative computation of mipmaps and Gaussian pyramids. However, instead of smoothing and downsampling a regular image in each level j, the computation of a dPDF-map requires doing this for each image layer b.

Separable smoothing in space and range. Instead of computing the spatial smoothing (W) and the range smoothing (K) together as in Eq. 3, we perform these two steps separately, using the approach depicted in Fig. 3 ① and ②. We first compute the smoothed histogram volume for level $j = 0$, i.e., $H_0(\boldsymbol{p}, r)$, using Eq. 3 for each pixel $\boldsymbol{p}$. However, we do not perform spatial smoothing in this step, i.e., we set W to a 1×1 box filter. We perform range smoothing using a Gaussian kernel, i.e., $K = G_{\sigma_r}$. In the next step, we compute all downsampled levels j with $j > 0$ iteratively. Each $H_j(\boldsymbol{p}, r)$ is computed from $H_{j-1}(\boldsymbol{p}, r)$, by applying the spatial smoothing kernel W to each of the B image layers, followed by downsampling by a factor of two in each spatial axis.

Formally, we lift the input image $I(\boldsymbol{p})$ into a volume $\tilde{I}(\boldsymbol{p}, r)$ with $\tilde{I}(\boldsymbol{p}, r) = 1$ when $r = I_{\boldsymbol{p}}$, and zero otherwise, and compute H_j as:

$$H_0 = \tilde{I} * K, \quad (10)$$

$$H_j = 2{\downarrow}\left(H_{j-1} * W\right) \quad \text{for all } j > 0, \quad (11)$$

where $2{\downarrow}$ performs spatial downsampling by a factor of two.

5 Sparse PDF Maps

Storing gigapixel images as dPDF-maps is not feasible, because each H_j consists of B image layers. Therefore, our goal is to compute a sparse pdf map (sPDF-map) from the dPDF-map, and discard the dPDF-map afterward. The sPDF-map also consists of pyramid levels j, comprising the sparse volumes $V_j(\boldsymbol{p}, r)$.

sPDF-map pyramid levels and coefficients. In contrast to a dPDF-map, each pyramid level of an sPDF-map solely consists of the sparse set of sPDF-map coefficients $(\boldsymbol{p}_n, r_n, c_n)$ introduced in Sec. 3. The set of coefficients of level j can be used directly with Eq. 9 in order to evaluate non-linear image operations in level j without accessing other pyramid levels. We compute each level j of the desired sPDF-map directly from its corresponding level j in the input dPDF-map. Each level is completely independent of all other pyramid levels, which facilitates out-of-core approaches.

sPDF-map computation. For each pyramid level j, our goal is to compute the sparse volume $V_j(\boldsymbol{p}, r)$ represented by the smallest set of sPDF-map coefficients $(\boldsymbol{p}_n, r_n, c_n)$ that well approximate a pre-scribed dPDF-map level $H_j(\boldsymbol{p}, r)$ in the L_2 sense, when H_j is computed using Eq. 7. We thus quantify similarity using the L_2 distance between each dPDF-map level H_j, and its approximation based on the corresponding coefficients $(\boldsymbol{p}_n, r_n, c_n)$:

$$E(V_j) = \|H_j - V_j * (W_j \otimes K)\|, \quad (12)$$

where $(W_j \otimes K)$ acts like $(W \otimes K)$ in Eq. 7. However, in this context we will call the spatial kernel W_j the *reconstruction filter*. $(W_j \otimes K)$ is the basis function used for fitting, which is then used later for image reconstruction. Note that W_j does not have to match the pre-filter W used in the dPDF-map computation (Eq. 11). In principle, W_j can also be chosen independently for each sPDF-map level j, but we currently use the same Gaussian kernel for all levels.

5.1 Computation of sPDF-map coefficients

In order to compute the sparse volume $V_j(\boldsymbol{p}, r)$ of each sPDF-map level j, we use the Matching Pursuit algorithm of Mallat and Zhang [1993]. Matching Pursuit is a greedy iterative algorithm that, in our framework (Algorithm 1), finds a position $(\boldsymbol{p}_n, r_n)$ at each step of the iteration, such that the inner product of the basis function $(W_j \otimes K)$ centered at $(\boldsymbol{p}_n, r_n)$ with H_j is maximized:

$$(\boldsymbol{p}_n, r_n) = \underset{(\boldsymbol{q},i)\in\mathcal{D}(H_j)}{\operatorname{argmax}} \langle H_j(\boldsymbol{s}, r), (W_j \otimes K)\left((\boldsymbol{s}, r) - (\boldsymbol{q}, i)\right)\rangle. \quad (13)$$

Finding each $(\boldsymbol{p}_n, r_n)$ requires an exhaustive search over all possible $(\boldsymbol{q}, i)$ in the domain of $H_j(\boldsymbol{p}, r)$. The corresponding c_n is then computed as given in Algorithm 1. In practice, we only try a discrete set of range positions i for each $\boldsymbol{q}$, and sample the L_2 errors at even fewer range positions r. Denoting the input quantization by B (e.g., $B = 256$ for 8-bit images), our implementation considers B different i, and computes the L_2 errors at $B/16$ different r.

Algorithm 1 Data fitting via Matching Pursuit

Input: dense histogram volume H_j
Output: sparse volume V_j given by a list of triplets $(\boldsymbol{p}_n, r_n, c_n)$

$E = \|H_j\|$;
n = 0; chunk = 1;
while $E > \varepsilon$ **and** chunk $\leq$ max_chunks **do**
  $V_j \leftarrow \emptyset$
  while $n <$ chunk $\cdot\, m_j$ **do**
    find $(\boldsymbol{p}_n, r_n)$ that maximizes the inner product of Eq. 13
    $c_n = \langle H_j(\boldsymbol{s}, r), (W_j \otimes K)\left((\boldsymbol{s}, r) - (\boldsymbol{p}_n, r_n)\right)\rangle$
    $H_j \leftarrow H_j - c_n (W_j \otimes K)(\boldsymbol{p}_n, r_n)$;
    $E \leftarrow \|H_j\|$;
    $V_j \leftarrow V_j \cup (\boldsymbol{p}_n, r_n, c_n)$
    $n \leftarrow n + 1$;
  end while
  V_j(chunk) $\leftarrow V_j$; chunk $\leftarrow$ chunk $+ 1$;
end while

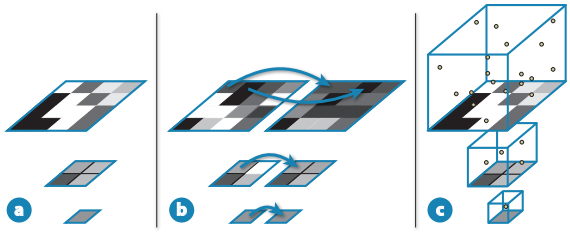

Figure 5: sPDF-maps *vs. image pyramids.* *An image pyramid (a) consists of a single image per level. Each level of an sPDF-map (b) comprises pairs of* index *and* coefficient *images that store the sPDF-map coefficients describing the (space × range) domain (c).*

5.2 sPDF-maps data structure

In order to store the $V_j(\boldsymbol{p}, r)$ efficiently and facilitate GPU implementation, we define the data structure illustrated in Fig. 5b, mimicking the basic geometry of a standard image pyramid (Fig. 5a). In contrast to standard image pyramids, each level j of an sPDF-map consists of one or multiple *coefficient chunks*, each of which is stored as a pair of images: an *index image*, and a *coefficient image*. Fig. 5b depicts a single coefficient chunk, i.e., one such image pair per level.

Coefficient chunks. We define the concept of a coefficient chunk in order to be able to store all data comprising an sPDF-map in images and facilitate parallel image reconstruction on GPUs. Each coefficient chunk is simply a sequence of m_j coefficients $(\boldsymbol{p}_n, r_n, c_n)$, where m_j is the number of image pixels in level j. Therefore, instead of computing a single sequence of coefficients, Algorithm 1 computes coefficients in multiples of coefficient chunks. When only a single chunk is computed, this implies that there is exactly one coefficient per image pixel on average. That is, some pixels might not have any coefficients, whereas other pixels might have multiple coefficients, but the total number of coefficients equals the total number of pixels. When two coefficient chunks are computed instead, there will be two coefficients per pixel on average, and so on.

In principle, coefficients could be represented as a stream of coefficients, but our choice of grouping coefficients into chunks enables storing all coefficients in images. This guarantees a constant memory overhead per pixel and facilitates using 2D textures on GPUs.

Index and coefficient images. The sequence of coefficients computed by Algorithm 1 is sorted in order of descending inner product. However, this order does not facilitate efficient parallel reconstruction. In each chunk, we therefore reorder the coefficients by gathering all coefficients of each pixel $\boldsymbol{p}$ into consecutive memory locations. We then tightly pack (in scanline order) all coefficients

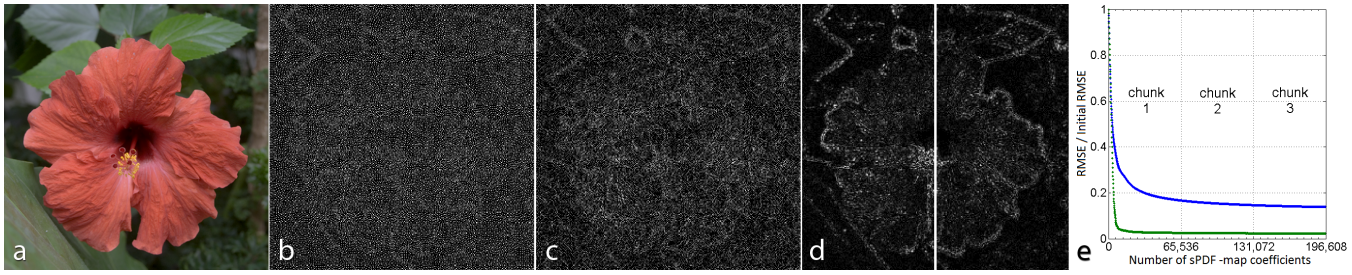

Figure 6: ***Fitting quality of sPDF-maps.*** *(a) Flower image (512×512). We evaluate sPDF-map level 1 (256×256). We show the distribution of coefficients for (b) the first coefficient chunk, and (c) the second chunk (64K coefficients each). White maps to 9 coefficients. We compare $E[X_{\boldsymbol{p}}]$ computed from the sPDF-map with the one from the dPDF-map. (d) shows the scaled difference between these two images with one (left half) and two (right half) coefficient chunks. For one chunk, the RMSE between the two images is 2.39, PSNR 40.57. For two chunks the RMSE is 2.22, PSNR 41.20. (e) the green curve shows how the initial RMSE of 93.75 between the two expected value images decreases as coefficients are added, the blue curve shows how the initial RMSE of 0.0113 between the dPDF-map and its approximation by the sPDF-map decreases. These errors are for the downsampled luminance channel of (a) using a 5×5 Gaussian for downsampling (W) and reconstruction (W_j).*

of all pixels into the coefficient image $C(\cdot)$ of this chunk. In order to remember where the coefficients of each pixel $\boldsymbol{p}$ are stored, we also construct an index image $X(\boldsymbol{p})$ that stores a pair of values $(\mathit{index}, \mathit{count})_{\boldsymbol{p}}$ for each $\boldsymbol{p}$: the position of the first coefficient of $\boldsymbol{p}$ in the coefficient image: $\mathit{index}_{\boldsymbol{p}}$, and the number of coefficients of $\boldsymbol{p}$: $\mathit{count}_{\boldsymbol{p}}$. Each entry in $C(\cdot)$ then simply consists of the pair (r_n, c_n), because $\boldsymbol{p}_n$ from the tuple $(\boldsymbol{p}_n, r_n, c_n)$ is implicit in $X(\boldsymbol{p})$.

The coefficients (r_n, c_n) of pixel $\boldsymbol{p}$ are therefore enumerated as follows: $C\left(X(\boldsymbol{p}).\mathit{index}\right) \,..\, C\left(X(\boldsymbol{p}).\mathit{index} + X(\boldsymbol{p}).\mathit{count} - 1\right)$.

6 Image Reconstruction

Given the set of sPDF-map coefficients $(\boldsymbol{p}_n, r_n, c_n)$, which comprise the sparse volume $V_j(\boldsymbol{p}, r)$, we can reconstruct the output image corresponding to various non-linear operations using Eq. 9. The non-linear operator is determined by the choice of function $t_{\boldsymbol{p}}(r)$ and the normalization $w_{\boldsymbol{p}}$. However, exploiting the properties of different classes of operators, and the corresponding definitions of $t_{\boldsymbol{p}}(r)$, enables different strategies and conceptual optimizations.

6.1 Global functions

The simplest non-linear operator applies the same function t to every pixel $\boldsymbol{p}$, i.e., $t_{\boldsymbol{p}}(r) = t(r)$ for all $\boldsymbol{p}$. A global t is sufficient to apply arbitrary global re-mapping functions, e.g., color maps. In this case, the spatial convolution in Eq. 9 does not need to be computed for every coefficient individually. We can rewrite Eq. 9 as:

$$E\left[t(X_{\boldsymbol{p}})\right] = \frac{1}{w_{\boldsymbol{p}}} \sum_{\boldsymbol{q} \in \mathcal{N}(\boldsymbol{p})} W_j\left(\boldsymbol{p} - \boldsymbol{q}\right) \sum_{\substack{(\boldsymbol{q}_n, r_n, c_n) \\ \boldsymbol{q} = \boldsymbol{q}_n}} \tilde{t}(r_n) V(\boldsymbol{q}_n, r_n). \quad (14)$$

Remember that $V(\boldsymbol{q}_n, r_n) = c_n$. Since t is the same everywhere, the second sum can be computed once per pixel $\boldsymbol{p}$, and then used in all spatial convolutions. This enables evaluating Eq. 14 as follows (setting $w_{\boldsymbol{p}} = 1$). We first compute two images $T(\boldsymbol{p})$ and $N(\boldsymbol{p})$ as:

$$T(\boldsymbol{p}) = \sum_{\substack{(\boldsymbol{p}_n, r_n, c_n) \\ \boldsymbol{p} = \boldsymbol{p}_n}} \tilde{t}(r_n) c_n, \quad \text{and} \quad N(\boldsymbol{p}) = \sum_{\substack{(\boldsymbol{p}_n, r_n, c_n) \\ \boldsymbol{p} = \boldsymbol{p}_n}} \bar{K}(r_n) c_n, \quad (15)$$

where $\bar{K}(r) = \int_0^1 K(x - r)\, dx$ is the integral of K centered at r and clamped to the range $[0, 1]$. We compute $N(\boldsymbol{p})$ in order to enable proper normalization, since the fitting operation of Algorithm 1 does not guarantee pdfs that sum exactly to one. Then, both images are individually convolved with W_j, and the output image is obtained by dividing each pixel in T by the corresponding weight in N:

$$E\left[t(X_{\boldsymbol{p}})\right] = \frac{\left(T * W_j\right)(\boldsymbol{p})}{\left(N * W_j\right)(\boldsymbol{p})}. \quad (16)$$

Color mapping. We can compute the result of anti-aliased color mapping by simply using the color map as $t(r)$ (handling each channel individually). Both $\tilde{t}(r) = t(r) * K$ and $\bar{K}(r)$ can be pre-computed, because they are the same for all pixels. Computationally, we only have to perform a look-up in $\tilde{t}(r)$ and $\bar{K}(r)$ per coefficient, sum the weighted look-ups for each pixel, perform a 2D convolution for T and N each, followed by one division per pixel. Despite its simplicity, this approach accurately evaluates Eq. 8 for any global t.

6.2 Local functions

If the function $t_{\boldsymbol{p}}(r)$ is different for every pixel $\boldsymbol{p}$, the simplification of Eq. 9 described in the previous section cannot be used directly, because the spatial convolution must not mix different $t_{\boldsymbol{p}}$.

Local Laplacian filters can be computed via Eq. 9 by defining the function $t_{\boldsymbol{p}}(r)$ to be the re-mapping function for each pixel $\boldsymbol{p}$ (Paris et al. [2011], Sec. 5.2, Fig. 6). These functions perform local contrast adjustment around each pixel $\boldsymbol{p}$, depending on its intensity $I_{\boldsymbol{p}}$, which we compute as $E[X_{\boldsymbol{p}}]$ (Sec. 3). Apart from the inaccuracies caused by Algorithm 1, no normalization is necessary, i.e., conceptually $w_{\boldsymbol{p}} = 1$ (Eq. 8). However, in practice a normalization factor $w_{\boldsymbol{p}} \neq 1$ must be computed and divided by, with $\bar{K}(r)$ as defined above:

$$w_{\boldsymbol{p}} = \sum_{\substack{(\boldsymbol{q}_n, r_n, c_n) \\ \boldsymbol{q}_n \in \mathcal{N}(\boldsymbol{p})}} \bar{K}(r_n) W_j\left(\boldsymbol{p} - \boldsymbol{q}_n\right) c_n. \quad (17)$$

We note that the computation of each Laplacian coefficient depends on an upsampling step of Gaussian pyramid coefficients, because it is obtained as the difference between $I_{\boldsymbol{p}}$ from level j and the corresponding upsampled intensity from $(j+1)$. It is important that for each upsampled pixel $\boldsymbol{p}$ in pyramid level j, the corresponding neighborhood in level $(j+1)$ is re-mapped with the same $t_{\boldsymbol{p}}(r)$.

Bilateral filters can be computed via Eq. 9 and the definitions of $t_{\boldsymbol{p}}(r)$ and $w_{\boldsymbol{p}}$ given in Sec. 3. However, the spatial kernel W must include both the reconstruction kernel W_j and the spatial neighborhood of the bilateral filter (W_{bl}). Both can either be pre-convolved ($W = W_{bl} * W_j$) or be evaluated in two successive steps. Alternatively, instead of evaluating the bilateral filter directly, it is possible

Image	Resolution	Size [Mpix.]	Pyramid Levels	Overhead [Mpix.]	Pre-Computation [s/Mpix.]
Night Scene	$47,908 \times 7,531$	361	17	120	127
Bellini	$16,898 \times 14,824$	250	16	83	133
NASA Bathymetry	$21,601 \times 10,801$	233	16	77	128
Machu Picchu	$9,984 \times 3,328$	33	15	11	132
Rock	876×584	0.51	11	0.17	139
Beach	800×533	0.43	11	0.14	135
Flower	800×533	0.43	11	0.14	142
Barbara	512×512	0.26	10	0.09	138
Salt and Pepper Noise	320×428	0.14	10	0.05	143

Table 1: ***Images in this paper with corresponding sPDF-map properties and pre-computation times.*** *The overhead column gives the total number of pyramid pixels (in Mpixels) in addition to the original image size. This is the same number of additional pixels required by a regular mipmap or Gaussian pyramid. For each of these additional pyramid pixels, every sPDF-map coefficient chunk requires* 64 *bits of storage per pixel in our current implementation. All our result images use a single coefficient chunk, except where indicated in Fig. 12 and Fig. 13e-m.*

to dynamically compute a bilateral grid [Chen et al. 2007] from an sPDF-map, similar to the histogram slicing process described below. This enables all performance optimizations of the bilateral grid.

6.3 Histogram slicing

Smoothed local histogram filters [Kass and Solomon 2010] are computed from histograms $h_{\boldsymbol{p}}(I_b)$ as defined by Eq. 3. sPDF-maps support this kind of filter by simply sampling the continuous $pdf_{\boldsymbol{p}}(r)$ from the sPDF-map into the $h_{\boldsymbol{p}}(I_b)$ for each pixel $\boldsymbol{p}$. This can be done with an arbitrary sampling rate B without changing the sPDF-map. We call this process *slicing* the sPDF-map into histogram bin images $H_b(\boldsymbol{p})$, each of which contains a bin b, i.e., $h_{\boldsymbol{p}}(I_b)$, for all $\boldsymbol{p}$. Each bin image can be computed identically for all $\boldsymbol{p}$ by evaluating Eq. 8 using the Dirac-delta $t_{\boldsymbol{p}}(r) = t(r) = \delta_{I_b}(r)$ and $w_{\boldsymbol{p}} = 1$.

Because the required function $t(r)$ is the same for all $\boldsymbol{p}$, each bin image can be computed as follows: First, we compute an intermediate slice coefficient image $S_b(\boldsymbol{p})$ corresponding to bin b, as:

$$S_b(\boldsymbol{p}) = \sum_{\substack{(\boldsymbol{p}_n, r_n, c_n) \\ \boldsymbol{p}=\boldsymbol{p}_n}} \big(\delta_{I_b} * K\big)(r_n) V(\boldsymbol{p}_n, r_n) = \sum_{\substack{(\boldsymbol{p}_n, r_n, c_n) \\ \boldsymbol{p}=\boldsymbol{p}_n}} K(I_b - r_n) c_n, \tag{18}$$

which sums up the contributions of all coefficients of pixel $\boldsymbol{p}$, weighted by a look-up into the range kernel K corresponding to I_b. Each $S_b(\boldsymbol{p})$ must then be convolved by W:

$$S_b^*(\boldsymbol{p}) = \big(S_b * W\big)(\boldsymbol{p}), \tag{19}$$

where the desired spatial histogram neighborhood (W_{sh}) can either be computed at the same time ($W = W_{sh} * W_j$), or in two successive convolution passes. In order to obtain a normalized smoothed histogram that represents a pdf, each intermediate bin image must then be normalized by the sum of all bin images. This yields the final histogram bin image $H_b(\boldsymbol{p})$ of each bin b as:

$$H_b(\boldsymbol{p}) = \frac{S_b^*(\boldsymbol{p})}{N(\boldsymbol{p})} \quad \text{with} \quad N(\boldsymbol{p}) = \sum_{b=1}^{B} S_b^*(\boldsymbol{p}). \tag{20}$$

Using different look-up tables for the range kernel K, including derivative and integration kernels as used by Kass and Solomon [2010], using the appropriate normalization, enables all smoothed local histogram filters to be evaluated via an sPDF-map.

7 Implementation and Performance

In order to support the computation of sPDF-maps of gigapixel images using a limited memory footprint, we have implemented the entire pipeline depicted in Fig. 3 in an out-of-core fashion.

7.1 Pre-computation

dPDF-map level $j = 0$. The first step is reading the input image, conceptually computing the corresponding volume $\tilde{I}$ (Sec. 4.2), and applying the range kernel K to compute sPDF-map level 0, i.e., H_0. In order to make this scalable, we read the input image one scanline at a time. For each pixel $\boldsymbol{p}$, we directly compute its contribution to H_0 by applying the range filter K. This results in a plane of H_0 in (space $\times$ range) for each scanline. Instead of writing each plane to disk immediately, we accelerate disk access by avoiding interleaving reads from the input with writes to the output. We forward each plane to a simple caching system that only writes to disk when a user-defined memory limit is exceeded. This guarantees zero overhead for images that can easily be processed in-core.

Reformatting into tiles. We want to store each dPDF-map level as a collection of tiles, with a tile size of $(n + o) \times (n + o)$ pixels, where n denotes the inner area of a tile, and o denotes an overlap region that is replicated amongst neighboring tiles. In order to do this, we collect the corresponding planes of H_0 computed above, and for each tile re-arrange them into a stack of B slice images that sample the range. In our implementation, we are using an inner tile size of $n = 256$, and an overlap of $o = 8$, which facilitates straightforward upsampling in Gaussian pyramids without accessing neighboring tiles. The overlap depends on the size of the spatial filters that must be evaluated. Our relatively small tile size ensures that the tiles fit into the L1 cache of the processor and can also be used directly by the tiled rendering system for image reconstruction.

dPDF-map levels j with $j > 0$. To compute one new level in the dPDF-map, 2×2 tiles are loaded at a time, and downsampled to a single tile by applying the kernel W (Sec. 4.1). This is repeated until all tiles comprising level j have been processed. Instead of writing each downsampled tile directly to disk, it is passed to our cache which is flushed to disk once it exceeds a pre-defined size. In this way, the full pyramid is computed incrementally by applying the same operation to the downsampled tiles until only a single tile is left. Finally, this tile is iteratively downsampled to a single entry.

sPDF-map computation. Next, we compute the sPDF-map from the dPDF-map tile by tile. Because we process each tile independently, our approach naturally scales to images of arbitrary resolution. In every iteration of Algorithm 1, the best position $(\boldsymbol{p}_n, r_n)$ with its corresponding coefficient c_n must be computed by finding the $(\boldsymbol{p}_n, r_n, c_n)$ that reduces the residue E the most. A straightforward but inefficient way of doing this is just going through all possible $(\boldsymbol{p}_n, r_n)$, computing the coefficients and residue decreases, then choosing the position and coefficient with the highest decrease. Another alternative is to keep a priority queue of the residue decreases for all possible positions. However, this queue can quickly

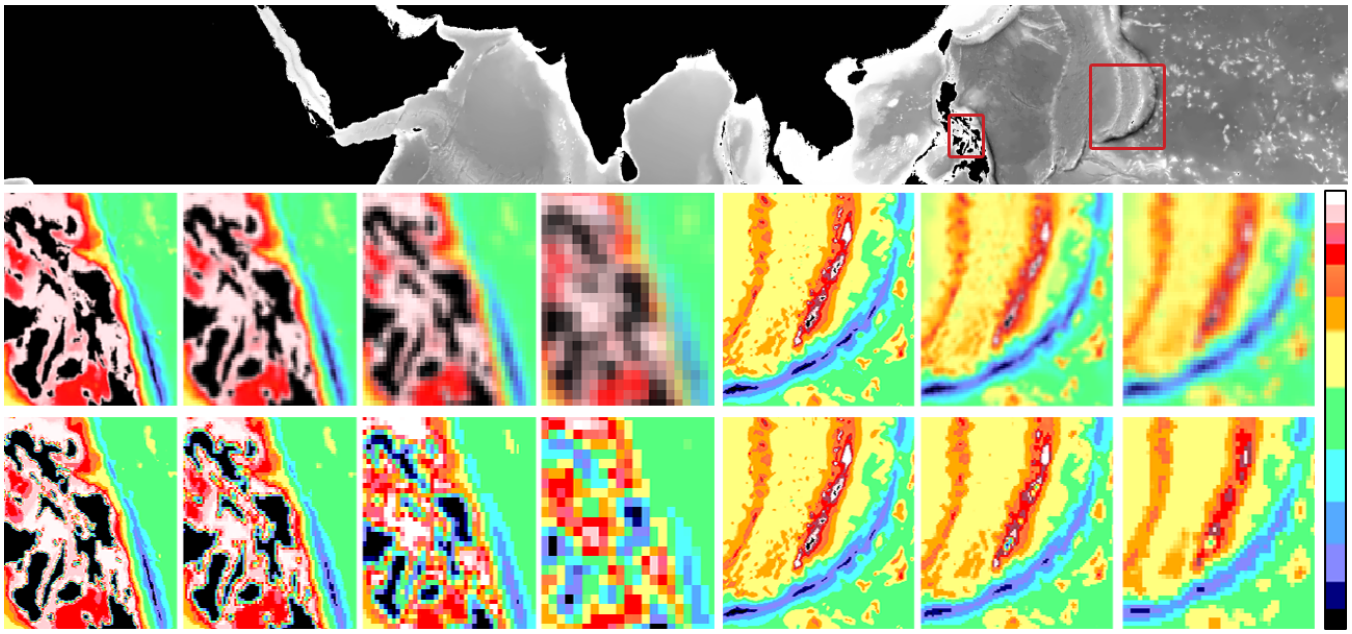

Figure 7: ***Color mapping.*** *(Top row) Gray-scale* $21,601 \times 10,801$ *(233 MPixels) bathymetry image from the NASA Blue Marble collection [NASA 2005]. (Center row) Anti-aliased color mapping computed from the sPDF-map; (Bottom row) Standard pre-filtering and downsampling followed by color mapping: coarser resolutions introduce wrong colors, and whole structures are changing or disappearing.*

become too large to be practical and can be slow to maintain. Instead, we subdivide each tile of the dPDF-map into smaller cubical blocks in (space $\times$ range) and track the optimal positions and residue decreases for each block in a small list. For each iteration, we then perform a parallel reduce step on the list to determine the maximum residue decrease and the corresponding position. We maintain this list continuously, and update all affected entries whenever the residue is modified. To improve efficiency further, we have implemented this approach on the GPU using CUDA.

7.2 Image reconstruction

Our implementation computes the output image $E[t_{\boldsymbol{p}}(X_{\boldsymbol{p}})]$, given various $t_{\boldsymbol{p}}(r)$, from the sPDF-map representation on the GPU using CUDA. We create a thread for every output pixel $\boldsymbol{p}$, and compute the necessary intermediate images, as well as the final image, by evaluating the equations given in Sec. 6. Each thread accesses all coefficients that are required to compute $E[t_{\boldsymbol{p}}(X_{\boldsymbol{p}})]$ for its output pixel $\boldsymbol{p}$ by first accessing the corresponding pixel in the index image $X(\boldsymbol{p})$, and then enumerating and fetching all pairs (r_n, c_n) from the coefficient image $C(\cdot)$ (Sec. 5.2). The required functions $\tilde{t}_{\boldsymbol{p}}(r) = t_{\boldsymbol{p}}(r) * K$ are pre-computed and stored in 1D arrays.

7.3 Performance and memory consumption

We have found that in general our fitting procedure (Algorithm 1) converges quite quickly (see Fig. 6e). It is usually sufficient to compute just one chunk of sPDF-map coefficients. Using more chunks does not dramatically improve the fitting quality for most images. Exceptions are images such as Fig. 12a and Fig. 13e.

The pre-computation time for sPDF-maps, as well as memory consumption, are summarized in Table 1 for various images. On an NVIDIA GTX 580 GPU, we currently require about two minutes to compute an sPDF-map with one coefficient chunk for one Mpixel. This time scales linearly both with image size and number of chunks. While we consider this pre-processing time feasible for large images, we believe using an optimized fitting algorithm could improve performance. In our implementation, each pixel has a constant memory consumption of 64 bits per coefficient chunk. We pack the $(\mathit{index}, \mathit{count})_{\boldsymbol{p}}$ pair of each pixel in the index image $X(\boldsymbol{p})$ into a single 32-bit integer (24 bits for $\mathit{index}_{\boldsymbol{p}}$, 8 bits for $\mathit{count}_{\boldsymbol{p}}$), and use an integer texture for $X(\boldsymbol{p})$. The (r_n, c_n) pair of each pixel in the coefficient image $C(\cdot)$ is stored as two 16-bit floats in a 16-bit float texture. In addition, our implementation incurs the typical overhead for the tiling data structure that enables us to properly deal with gigapixel images. However, this is independent of the sPDF-maps representation and inherent to the use of a tiled out-of-core approach.

The image reconstruction stage of $E[t(X_{\boldsymbol{p}})]$ for a global function t is extremely efficient. On a GTX 580 GPU, we compute anti-aliased color mapping (Sec. 6.1) in 12 ms per Mpixel and coefficient chunk, with a W_j of 5×5. A histogram slice (Sec. 6.3) can be obtained at the same rate. This performance is roughly constant for all our images, and scales linearly with the number of coefficient chunks. For Laplacian filtering, the performance of computing Laplacian pyramid coefficients is independent of the pyramid level that is computed. This is not the case for the original approach of Paris et al. [2011], where the time for computing each coefficient must increase with the original image resolution. The approach by Aubry et al. [2011] is much faster, but depends on pre-computing full-resolution pyramids, which makes its scalability to gigapixel images unclear. Using an sPDF-map, all Laplacian pyramid coefficients for a 1 MPixel image can be computed in around 320 ms, independent of the original image resolution, scaling linearly. Overall performance must also take into account the Laplacian pyramid collapse, which depends on the number of pyramid levels. For a 1024×1024 view showing a zoom-in of level 0 of the $47,908 \times 7,531$ Night Scene image (Fig. 8), the pyramid collapse takes around 5 ms, and overall computation time for the entire 1024×1024 view is below 400 ms.

8 Applications of sPDF-maps

We view the versatility of the sPDF-maps representation as one of its biggest strengths. In this section, we illustrate several non-linear image operators that are all computed from the same data structure.

Figure 8: ***Local Laplacian filtering with an sPDF-map in*** $\mathcal{O}(n)$ ***time.*** *(Top row)* Night scene *of resolution* $47,908 \times 7,531$ *(*361 *Mpixels). The top third of the image is shown with detail enhancement (*$\sigma_r = 0.2$*,* $\alpha = 0.25$*), the center third is the original image, and the bottom third is shown with smoothing (*$\sigma_r = 0.2$*,* $\alpha = 3.0$*). (Bottom row) Images used by Paris et al. [2011]: left-hand image of each pair with detail enhancement (*$\sigma_r = 0.4$*,* $\alpha = 0.25$*), right-hand image with smoothing (*$\sigma_r = 0.2$*,* $\alpha = 2.0$*). RGB color channels were computed separately.*

8.1 Anti-aliased color mapping

Fig. 7 illustrates anti-aliased color mapping vs. first downsampling and then color mapping for a high-resolution gray-scale image from the NASA Blue Marble collection [NASA 2005]. The color map is a standard false color coding often used by scientists to visualize the underwater depth in bathymetry images. The bottom row illustrates the two common problems of color-mapping gigapixel images: (1) In coarser resolution levels, wrong color values are introduced, because the color map is applied after downsampling. This is especially visible around the islands of the Philippines depicted in the left zoom-in. (2) Structures change their shape and/or topology, disappear, or appear from pyramid level to pyramid level. In the left zoom-in, entire islands disappear, whereas in the right zoom-in structures of very large depth in the Mariana Trench successively disappear.

Both of these problems can be avoided by using the same color map as the function $t(r)$ with the corresponding sPDF-map. This is illustrated in the center row of the figure. The output image is computed as described in Sec. 6.1. Each channel of the RGB output is computed separately, but from the same sPDF-map input, using the corresponding channel of the color map as the function $t(r)$.

8.2 Fast local Laplacian filtering

Fig. 8 illustrates different examples of detail enhancement and smoothing, respectively, with the local Laplacian filtering approach of Paris et al. [2011], but evaluated using sPDF-maps. We have implemented an interactive application for gigapixel viewing and filtering that only computes the pixels of image tiles that are currently visible on screen. This is demonstrated in the video. The Laplacian pyramid is only computed and collapsed for the parts actually visible on screen. This approach is greatly facilitated by the fact that sPDF-maps enable direct computations in each pyramid level. We compute the filtering as described in Sec. 6.2. Fig. 9 and Fig. 10 compare our results with the publicly available Matlab implementation of Paris et al. [2011]: Fig. 9 for a zoom-in of the *Bellini* image (Fig. 1), and Fig. 10 for the flower shown in Fig. 6a.

8.3 Smoothed local histogram filtering

Fig. 11 illustrates examples of computing smoothed local histogram filters in image pyramids, and compares the results of standard downsampling and then filtering with sPDF-maps. Evaluating the non-linear filter directly for a downsampled image cannot recover the edges already smoothed away by the linear pre-filter. In contrast, if the non-linear filter uses the sPDF-map, it is conceptually applied to the original image followed by pre-filtering and downsampling, which correctly preserves the non-linearity of the histogram filter.

Fig. 12 illustrates this in more detail for median filtering. The original image (a) contains salt and pepper noise, which is easily removed by median filtering (b). However, when the image is first downsampled (f) and then median-filtered (g), artifacts from the downsampling are visible. In contrast, sPDF-map median filtering is able to remove the noise without artifacts. For this example, we illustrate two different reconstruction kernels W_j, because matching (a) exactly is hard for sPDF-maps. Our standard W_j of 5×5 smoothes the noise (j); 31 dB compared to (b). Using a W_j of 3×3 produces a better match, however at the price of using two coefficient chunks (e, bottom half); 33 dB. If only a single chunk is used, there are not enough coefficients to match the noise (e, top half); 27 dB. However, median filtering quality is good for a W_j of 3×3 and two chunks, as well as for a W_j of 5×5 and one chunk.

8.4 Bilateral filtering

Fig. 13 illustrates bilateral filtering using the sPDF-map representation vs. standard downsampling followed by bilateral filtering. Applying the bilateral filter to a downsampled image loses many of its edge-preserving qualities, because many edges have already been smoothed away by the linear pre-filter. This problem is reduced considerably by applying the bilateral filter to the sPDF-map as described in Sec. 6.2. The top half of Fig. 13 illustrates a zoom-in of the *Bellini* image (level 4), where high-frequency detail is lost in the naive approach. The bottom half of Fig. 13 shows zoom-ins of level 1 and level 2 of the *Barbara* image. This illustrates that applying the bilateral filter to a downsampled image is also problematic for already anti-aliased edges, where using sPDF-maps achieves much better results. Note that due to the many very thin stripes in Fig. 13e, we have computed the corresponding sPDF-map with a W_j of size 3×3 and two coefficient chunks to obtain a good fit.

9 Discussion and Limitations

We envision sPDF-maps as a powerful alternative to regular image pyramids, however enabling filter evaluation to exploit direct access to local distribution functions with a minimal storage overhead.

Figure 9: ***Local Laplacian smoothing.*** *sPDF-map results vs. the original implementation of Paris et al. ($\sigma_r = 0.2$, $\alpha = 2.0$). (a,b) level 0 ($1,744 \times 1,160$); (c,d) level 3 (218×145). (a,c) Paris et al.; (b,d) sPDF-map. Luminance PSNR [dB] between (a,b)* 35*, (c,d)* 36*.*

Approximation quality. sPDF-maps offer surprisingly good approximation quality with very few coefficients, e.g., the same number of coefficients as image pixels, by exploiting the coherence between neighboring pdfs in the 3D (space $\times$ range) domain. This property also enables accurate approximation with Gaussians of constant size. However, it would be interesting to experiment with adaptive (possibly anisotropic) kernels in the future. For most images, a Gaussian reconstruction kernel W_j of size 5×5 with just one coefficient chunk achieves very good quality, both visually and numerically. The only exceptions that we have encountered are images with very high frequency content, e.g., Figs. 12a/13e, for which a smaller W_j of 3×3 and more coefficient chunks can be used. The results of a 3×3 kernel with one chunk are numerically on a par with the naive approach (Fig. 12), and two chunks are clearly better. Furthermore, in our experience the visual quality is often better than the PSNR values suggest, i.e., visually closer to the ground truth than the naive results, even for small PSNR differences. In practice, we use a 5×5 kernel W_j with one chunk, or a 3×3 kernel with two chunks.

Classes of filters. The sPDF-maps representation enables the efficient and accurate evaluation of a wide variety of filters. Essentially, sPDF-maps support any filter that can be evaluated from local 1D pdfs, which includes many important filters used for photo editing. However, filters that require more information cannot be evaluated directly. Examples would be filters that require gradients, such as anisotropic diffusion, or filters that compare exact neighborhoods, such as non-local means filtering. Extending sPDF-maps to higher-dimensional pdfs could enable these kinds of filters in the future.

Apart from these limitations, we think that the biggest current limitation of our method are the pre-computation times required to compute the sPDF-map coefficients. We would like to explore different optimization strategies for better performance in future work.

10 Conclusions

We have introduced the new sPDF-maps representation and data structure to compactly represent pre-computed pdfs of pixel neighborhoods in multi-resolution image pyramids. In order to use sPDF-maps in practice, we have presented an efficient unified method for evaluating a variety of non-linear image operations from the same pre-computed representation, which illustrates the versatility of sPDF-maps. However, we imagine that in the future more operations can make use of the information represented by pixel neighborhood pdfs, using the same sPDF-maps data structure.

Acknowledgements

We would like to thank the anonymous reviewers for helping us improve the paper; Peter Rautek, Thomas Theußl, Thomas Höllt, and Wito Engelke for their help; Justin Solomon for a smoothed local histogram filter implementation, and Sylvain Paris et al. for their local Laplacian filter implementation, which we used for comparisons.

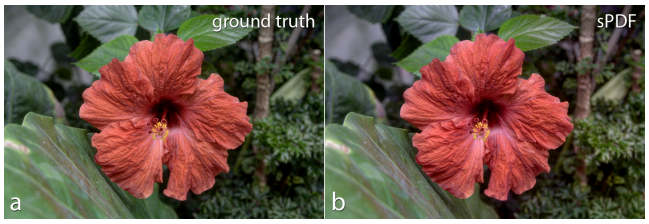

Figure 10: ***Local Laplacian detail enhancement.*** *sPDF-map results vs. the original implementation of Paris et al. ($\sigma_r = 0.2$, $\alpha = 0.5$). Flower (800×533) level 0: (a) Paris et al. (b) sPDF-maps. Luminance PSNR between (a,b)* 37 *dB. Original in Fig. 6a.*

References

AUBRY, M., PARIS, S., HASINOFF, S. W., KAUTZ, J., AND DURAND, F. 2011. Fast and Robust Pyramid-based Image Processing. Tech. rep., MIT. MIT-CSAIL-TR-2011-049.

BURT, P., AND ADELSON, T. 1983. The Laplacian pyramid as a compact image code. *IEEE Transactions on Communications 9*, 4, 532–540.

CARPENTER, L. 1983. The A-buffer, an antialiased hidden surface method. In *Proceedings of SIGGRAPH 1983*, ACM Press / ACM SIGGRAPH, 1–11.

CHEN, J., PARIS, S., AND DURAND, F. 2007. Real-time edge-aware image processing with the bilateral grid. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2007) 26*, 3, 103:1–103:9.

CROW, F. C. 1977. The aliasing problem in computer-generated shaded images. In *Proceedings of SIGGRAPH 1977*, ACM Press / ACM SIGGRAPH, 799–805.

DONNELLY, W., AND LAURITZEN, A. 2006. Variance shadow maps. In *Proceedings of 2006 Symposium On Interactive 3D Graphics and Games*, ACM Press, 161–165.

FATTAL, R., AGRAWALA, M., AND RUSINKIEWICZ, S. 2007. Multiscale shape and detail enhancement from multi-light image collections. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2007) 26*, 3, 51:1–51:9.

FATTAL, R. 2009. Edge-avoiding wavelets and their applications. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2009) 28*, 3, 22:1–22:10.

FELSBERG, M., FORSSEN, P.-E., AND SCHARR, H. 2006. Channel smoothing: Efficient robust smoothing of low-level signal features. *IEEE Transactions on Pattern Analysis and Machine Intelligence 28*, 2, 209–222.

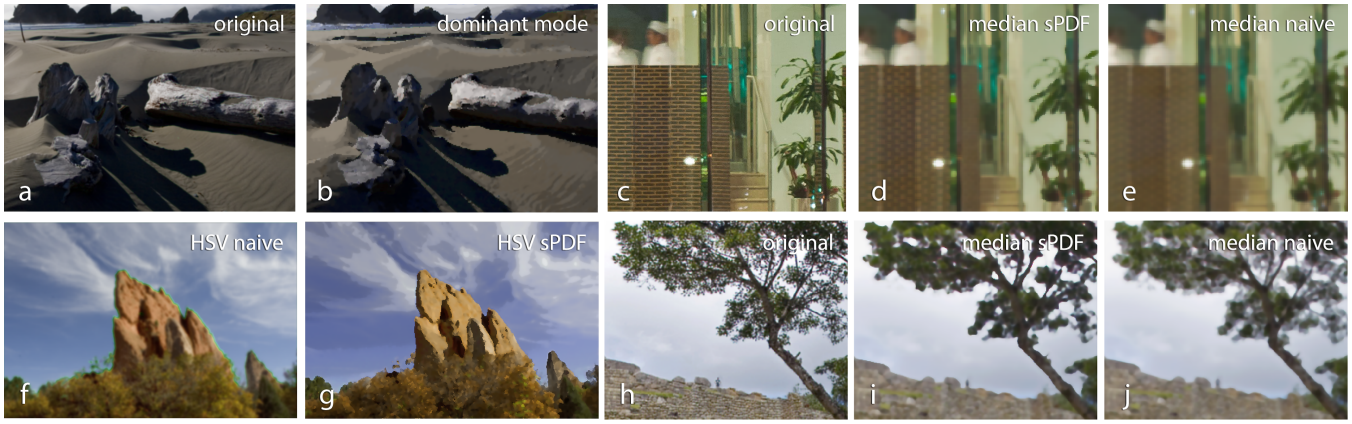

Figure 11: ***Smoothed local histogram filtering.*** *(a,b)* Beach *image in (a) dominant mode-filtered (luminance only) in (b). (f,g)* Rock *image in HSV color model: Standard downsampling introduces strong haloes of the wrong color around the rock (f), whereas dominant mode-filtering the H and V channels correctly preserves the circular domain of the hue channel (g). (c,d,e) and (h,i,j): Median filtering (luminance only) using sPDF-maps vs. downsampling and then filtering. (c,h) Original zoom-ins of the* Night Scene *(c) and the* Machu Picchu *(h) images. (d,i) Median filtering with sPDF-maps prevents over-smoothing by properly preserving the non-linearity of the image operation. (e,j) Median filtering after downsampling introduces strong over-smoothing that cannot be reversed by the median filter applied directly at the coarser resolution.*

GREENE, N., AND HECKBERT, P. 1986. Creating raster omnimax images from multiple perspective views using the elliptical weighted average filter. *IEEE Computer Graphics and Applications 6*, 6, 21–27.

HAN, C., SUN, B., RAMAMOORTHI, R., AND GRINSPUN, E. 2007. Frequency domain normal map filtering. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2007) 26*, 3, 28:1–28:12.

HANIKA, J., DAMMERTZ, H., AND LENSCH, H. 2011. Edge-optimized À-trous wavelets for local contrast enhancement with robust denoising. *Computer Graphics Forum 30*, 7, 1879–1886.

HECKBERT, P. 1989. *Fundamentals of Texture Mapping and Image Warping*. Master's thesis, U.C. Berkeley.

JEONG, W.-K., JOHNSON, M. K., YU, I., KAUTZ, J., PFISTER, H., AND PARIS, S. 2011. Display-aware image editing. In *Proceedings of IEEE International Conference on Computational Photography*.

KASS, M., AND SOLOMON, J. 2010. Smoothed local histogram filters. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2010) 29*, 4, 100:1–100:10.

KAZHDAN, M., AND HOPPE, H. 2008. Streaming multigrid for gradient-domain operations on large images. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2008) 27*, 3, 21:1–21:10.

KOENDERINK, J. J., AND VAN DOORN, A. J. 1999. The structure of locally orderless images. *International Journal of Computer Vision 31*, 2-3 (Apr.), 159–168.

KOPF, J., UYTTENDAELE, M., DEUSSEN, O., AND COHEN, M. F. 2007. Capturing and viewing gigapixel images. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2007) 26*, 3, 93:1–93:10.

MALLAT, S., AND ZHANG, Z. 1993. Matching pursuits with time-frequency dictionaries. *IEEE Transactions on Signal Processing 41*, 12 (Dec), 3397 –3415.

MALLAT, S. 2009. *A Wavelet Tour of Signal Processing: The Sparse Way*, 3rd ed. Academic Press.

NASA, 2005. Blue marble: Next generation. NASA Earth Observatory. http://earthobservatory.nasa.gov/Features/BlueMarble/.

PARIS, S., AND DURAND, F. 2006. A fast approximation of the bilateral filter using a signal processing approach. In *Proceedings of European Conference on Computer Vision (ECCV) 2006*, 568–580.

PARIS, S., HASINOFF, S. W., AND KAUTZ, J. 2011. Local Laplacian filters: Edge-aware image processing with a Laplacian pyramid. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2011) 30*, 4, 68:1–68:12.

REEVES, W., SALESIN, D., AND COOK, R. 1987. Rendering antialiased shadows with depth maps. In *Proceedings of SIGGRAPH 1987*, ACM Press / ACM SIGGRAPH, 283–291.

SUMMA, B., SCORZELLI, G., JIANG, M., BREMER, P.-T., AND PASCUCCI, V. 2010. Interactive editing of massive imagery made simple: Turning Atlanta into Atlantis. *ACM Transactions on Graphics 30*, 2, 7:1–7:13.

TAN, P., LIN, S., QUAN, L., GUO, B., AND SHUM, H. 2008. Filtering and rendering of resolution-dependent reflectance models. *IEEE Transactions on Visualization and Computer Graphics 14*, 2, 412–425.

THOMPSON, D., LEVINE, J., BENNETT, J., BREEMER, P.-T., GYULASSY, A., PEBAY, P., AND PASCUCCI, V. 2011. Analysis of large-scale scalar data using hixels. In *Proceedings of IEEE Symposium on Large-Scale Data Analysis and Visualization (LDAV)*, 23 – 30.

TOMASI, C., AND MANDUCHI, R. 1998. Bilateral filtering for gray and color images. In *Proceedings of International Conference on Computer Vision (ICCV) '98*, 839–846.

WILLIAMS, L. 1983. Pyramidal parametrics. In *Proceedings of SIGGRAPH 1983*, ACM Press / ACM SIGGRAPH, 1–11.

YOUNESY, H., MÖLLER, T., AND CARR, H. 2006. Improving the quality of multi-resolution volume rendering. In *Proceedings of Eurovis 2006*, 251–258.

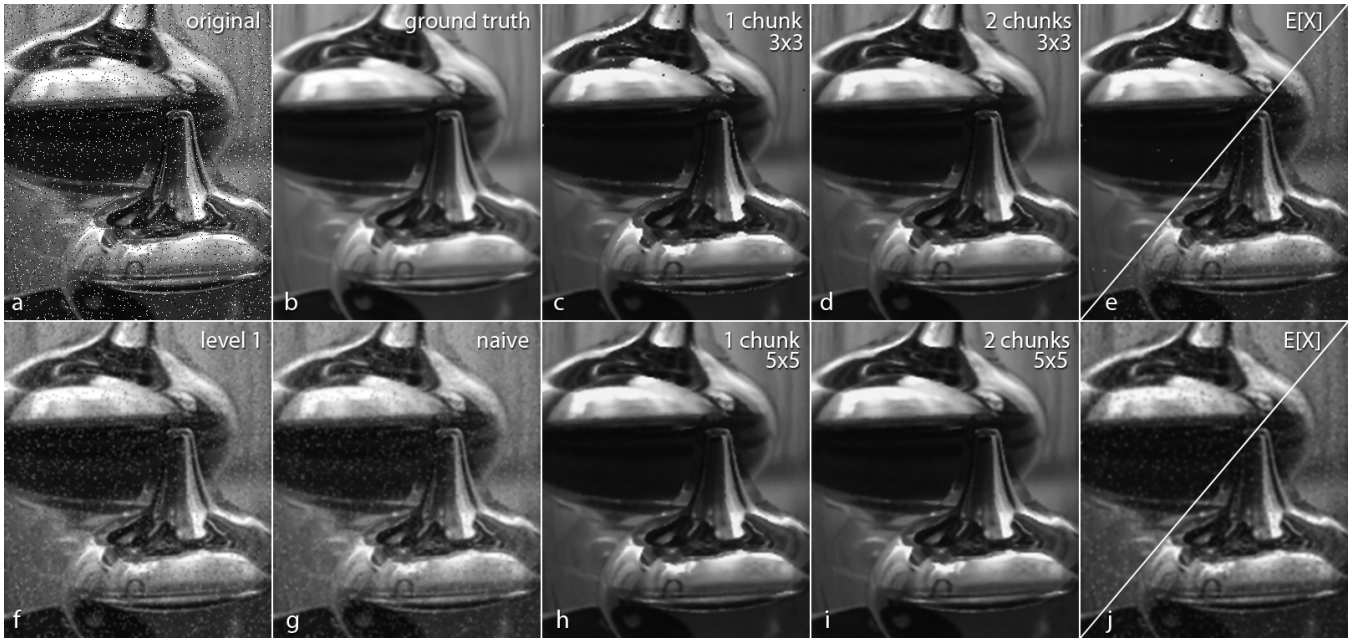

Figure 12: ***Median filtering.*** *(a) Original image (*320×428*) with salt and pepper noise. (b) Ground truth* 5×5 *median applied to level 0, then downsampled to level 1. (g) Naive equivalent median computed in level 1. sPDF-map where* W_j *is a Gaussian of size (c,d,e)* 3×3*, (h,i,j)* 5×5*. Median from sPDF-map with (c,h) 1 coefficient chunk, (d,i) 2 chunks. (f) Gaussian pyramid level 1 (*160×214*). (e,j)* $E[X_{\boldsymbol{p}}]$ *of sPDF-map level 1 (top half: 1 coefficient chunk, bottom half: 2 chunks). PSNR [dB] between (g,b):* 30*, (c,b):* 30*, (d,b):* 36*, (h,b):* 38*, (i,b):* 39*.*

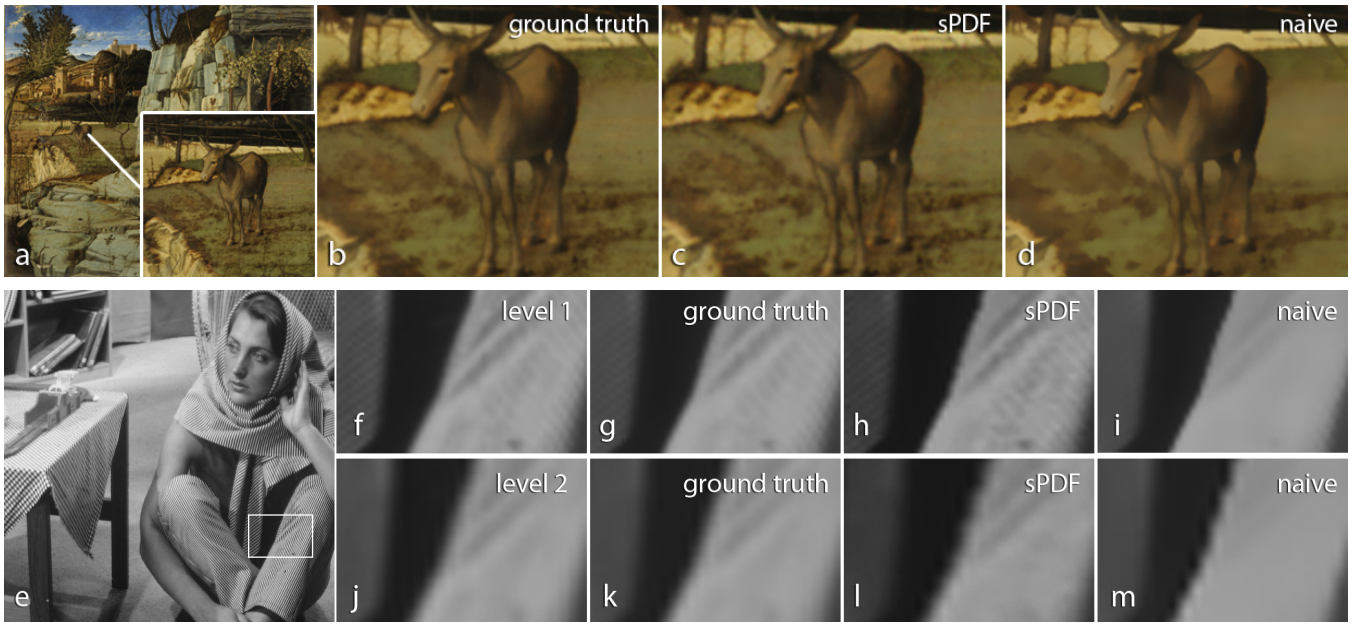

Figure 13: ***Bilateral filtering.*** *(Top row) Zoom-ins of original* $16,898 \times 14,824$ *image (a) at level 4: (b) Ground truth bilateral, (c) sPDF-map bilateral, (d) Naive bilateral. Luminance PSNR [dB] between (c,b)* 43*, (d,b)* 41*. (Bottom row) (e) Original image (*512×512*). The sPDF-map of (e) uses a* W_j *of size* 3×3 *and two coefficient chunks. Zoom-ins from (f,g,h,i) level* 1*, (j,k,l,m) level* 2*. (f,j) Downsampled image, no bilateral filtering. (g,k) Ground truth bilateral. (h,l) sPDF-map bilateral. (i,m) Naive bilateral. PSNR [dB] between (h,g)* 37*, (i,g)* 35*, (l,k)* 38*, (m,k)* 37*.*