Transform Domain Robust Filtering

Hila Nachlieli, Carl Staelin, Mani Fischer,
Doron Shaked, Renato Keshet, Pavel Kisilev
HP Laboratories Israel
HPL-2005-185(R.1)
August 3, 2007*

denoising, image processing, over complete transform

Image quality is one of the main requirements for imaging devices, such as cameras, scanners, printers, etc. In particular, one is interested in obtaining sharp images that contain as little noise as possible. While the level of noise in captured images is determined mainly by the quality of the hardware (e.g., the CCD in cameras), it can usually be reduced by image processing tools. This report presents an image processing scheme for reducing noise in images.

The proposed scheme consists of using robust-estimation filtering methodology in the context of Donoho's state-of-the-art translation-invariant soft-threshold denoising scheme. Specifically, we replace the soft-threshold function in Donoho's scheme with a robust-estimation look-up table, which reduces the blurring common to the approach. Instead of a Wavelet transform (which is usually used in this context), we prefer to use a block-DCT, since it is efficiently implemented in most image capture devices. Extensions and applications of this algorithm for video processing are also proposed.

This approach could be seen as part of a second generation of robust-estimation filters, available today in several HP imaging products.

* Internal Accession Date Only

Transform Domain Robust Filtering

Hila Nachlieli, Carl Staelin, Mani Fischer,
Doron Shaked, Renato Keshet, Pavel Kisilev

Hewlett-Packard Laboratories Israel

July 26, 2007

Abstract

Image quality is one of the main requirements for imaging devices, such as cameras, scanners, printers, etc. In particular, one is interested in obtaining sharp images that contain as little noise as possible. While the level of noise in captured images is determined mainly by the quality of the hardware (e.g., the CCD in cameras), it can usually be reduced by image processing tools. This report presents an image processing scheme for reducing noise in images.

The proposed scheme consists of using robust-estimation filtering methodology in the context of Donoho's state-of-the-art translation-invariant soft-threshold denoising scheme. Specifically, we replace the soft-threshold function in Donoho's scheme with a robust-estimation look-up table, which reduces the blurring common to the approach. Instead of a Wavelet transform (which is usually used in this context), we prefer to use a block-DCT, since it is efficiently implemented in most image capture devices. Extensions and applications of this algorithm for video processing are also proposed.

This approach could be seen as part of a second generation of robust-estimation filters, available today in several HP imaging products.

1 Introduction

Robust-estimation filters are based on robust-estimation theory, and rely on the idea that the filtering process should discard, or reduce, the influence of edges or structured details, which are considered outliers of the process. The outliers themselves are then preserved by the noise removal process. As a consequence, resulting robust-estimation filters tend to remove noise without blurring edges/details, and/or sharpen edge/details without increasing image noise.

The burden on these filters has been increasing lately, as the following trends develop. First, pixels are getting smaller in digital cameras, which significantly increases the level of noise in images. Second, as resolution of acquired images increases, device optics do not always improve at the same rate, and therefore images often become locally blurrier, and outliers become harder to detect. Moreover, when spatially variant enhancement (such as Retinex or variable gamma correction) is applied to high dynamic range images, noise is often boosted on low light areas. Finally, low quality cameras and scanners, with higher noise levels, are becoming more and more pervasive.

A second generation of robust-estimation filters is needed to deal with the above developing trends, and this report proposes one approach to this end. We report a method for extending first-generation robust filters, by combining them with the translation-invariant soft-threshold denoising algorithm [1, 2]. We review the first-generation robust filter in Section 2, and the translation-invariant soft-threshold scheme in Section 3. Our proposal to combine those two approaches is described in Section 4. Section 5 discusses the relationship of the proposed approach to a few alternatives, and its relationship to the first-generation pre-selective filter. Section 6 describes the results, and in Section 7 an extension to video (using three dimensional neighborhoods) is presented. The conclusions are presented in Section 9.

2 The First-Generation Robust Filters

The main idea behind the first generation robust filters [3] is that large differences in luminance values are likely to indicate image features, particularly edges, and conversely small differences represent mostly noise. Therefore, large differences (features) should be enhanced, while small differences (noise) should be suppressed. This principle is applied in a weighted averaging fashion over neighboring pixels. The weights are chosen so to increase the averaging influence of similar-valued neighbors on a given pixel, and to enhance the contrasting influence of neighbors belonging to different image features.

This is accomplished by adding a photometric weight to the standard convolution (filtering) process. The standard convolution operation is given by:

$$Out(i) = \sum_{j \in N(i)} h(j-i) In(j), \tag{1}$$

where In is the input image, i is the processed pixel, $Out(i)$ is the value of the pixel i in the returned image, $N(i)$ is a pre-defined pixel neighborhood, and $h(\cdot)$ is a normalized filter kernel, satisfying $\sum_k h(k) = 1$. In (1), $h(j-i)$ weights a pixel j in the neighborhood $N(i)$ according to its spatial distance to i.

Notice that (1) can be rewritten as follows:

$$Out(i) \;=\; In(i) + \sum_{j \in N(i)} h(j-i) \cdot (In(j) - In(i)) \, . \tag{2}$$

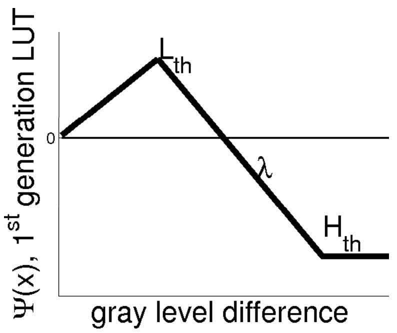


Figure 1: The robust LUT. Only positive input values are shown; the function is anti-symmetric.

The first kind of robust filters, which we call *pre-selective* filter, is given by:

$$Out(i) = In(i) + \sum_{j \in N(i)} h(j-i) \cdot \Psi\left[In(j) - In(i)\right] \tag{3}$$

where $\Psi(\cdot)$ is a pre-defined non-linear function.

Notice how the modification of $In(j) - In(i)$ by Ψ is the only difference between (2) and (3). As illustrated in Figure 1, $\Psi(x)$ is equal to (or near to) the identity function for values of x below a threshold L_{th}, but gradually becomes negative for x above L_{th}. As a consequence, small input differences $[I(j) - I(i)]$ cause the overall operation in (3) to behave as smoothing, whereas large differences cause the operation to perform sharpening. The function is bounded (for input differences higher of a second threshold H_{th}) to avoid over-sharpening artifacts. The non-linear function Ψ is scalar, and hence can be implemented efficiently as a look-up table (LUT). It will be referred to as “the robust LUT” throughout this paper.

An alternative approach is given by the *post-selective* filter:

$$Out(i) = In(i) - \Psi\left[In(i) - \sum_{j \in N(i)} h(j-i) In(j)\right], \tag{4}$$

where the robust LUT is applied to the difference between the original pixel value and its value after a linear filtering operation. As before, small differences represent mostly noise, in which case the filtered value should be taken as output. In contrast, large differences represent mostly features, and are enhanced rather than filtered out.

Notice that, in the pre-selective filter, the LUT operation is applied “before” the summation takes place, whereas, in the post-selective filter, the LUT comes “after” the summation. This is the reason for their names. Since the LUT is used several times per-pixel in the pre-selective filter, but only once per-pixel in the post-selective filter, the latter is faster, but usually the former produces better quality results.

The pre- and post-selective schemes can be combined, by taking the output of the summation of the former to be the input of the latter:

$$Out(i) = In(i) - \Psi^{Post}\Big(\sum_j h(j-i)\, \Psi^{Pre}\big(In(j) - In(i)\big)\Big). \tag{5}$$

The resulting scheme is called "bi-selective filter," and combines the higher selectivity of the pre-selective with the better output control of the post-selective.

3 The Translation-Invariant Soft-Threshold Scheme

Donoho's translation-invariant soft-thresholding algorithm [1, 2] is one of the leading state-of-the-art denoising schemes in the technical literature. This algorithm consists of: 1) representing a given image in a transform domain (e.g., the Wavelet domain), 2) applying a soft-thresholding, or shrinkage operation (reviewed next) to a subset of transform coefficients, and 3) transforming the new coefficients back to the original domain, obtaining an enhanced image. The above procedure is applied to several shifted versions of the image; the resulting enhanced images are shifted back and averaged. This shift-averaging scheme significantly reduces artifacts related to the lack of shift invariance of a Wavelet transform. Shift averaging using block Discrete Cosine Transform (DCT) for JPEG artifact reduction was implemented by Nosratinia [4], and improved by Samadani *et al.* [5]. *Soft-thresholding*

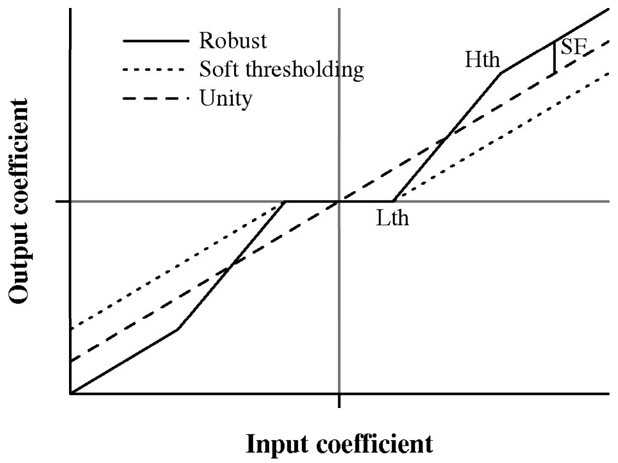


Figure 2: The proposed LUT: output coefficient values vs. input coefficient values. Dashed line is the identity line, Dotted line is the soft threshold LUT and the solid line is the Robust LUT. *Lth* is the low threshold under which coefficients are mapped to zero. *Hth* is the high threshold above which the Robust LUT becomes parallel to identity, with a distance called "sharpening factor" (SF) between them.

(illustrated by dotted line in Figure 2) is the scalar operation given by:

$$\mathrm{SoftTh}\,(y,T) = \begin{cases} y - \mathrm{sign}(y)\cdot T & \text{for} \quad |y| > T \\ \\ 0 & \text{for} \quad |y| \leq T \end{cases} \tag{6}$$

Since it is a scalar operation, which can be implemented with an LUT, we refer to it as "the soft-threshold LUT."

Several authors [2, 6, 7, 8] proved the optimality of the soft threshold algorithm for image denoising, in the sense of $argmin_g\{||f-g||_2 + \lambda||g||_B\}$, where f is the given noisy image, g is the reconstructed image and $||g||_B$ stands for the Besov smoothness norm. In [9], the optimality was also proven for the case where $||g||_B$ is the log of a Laplacian prior function.

The above optimality proofs are enlightening, and therefore we summarize them here in the Appendix sections; The Bayesian approach in [9] is shown in Appendix A, whereas the Besov-norm approach is summarized in Appendix C. These summaries provide good theoretical insight, and explore interesting concepts, related to noise removal.

4 The Proposed method

Like other smoothing algorithms, the translation-invariant soft-threshold one tends to slightly blur images. We improve translation-invariant soft-threshold by adding sharpening to the algorithm, thereby giving more visually pleasing results.

The core of the proposed scheme is identical to that of the translation-invariant soft-threshold algorithm. Next, we characterize this algorithm mathematically. Assume that the input image is a $N \times N$ matrix, and that N is multiple of an integer L. The output image Out of the translation-invariant soft-threshold algorithm is given by:

$$Out = \frac{1}{K}\sum_{k\in G_k} S_{-k}\mathbf{T}^{-1}\mathbf{\Phi}\left(\mathbf{T}S_k In\right), \tag{7}$$

where S_k is the image translation by a given vector k, K is the number of translations in G_k, and $\mathbf{T}$ is a block-transform, where the block size is $L \times L$. Moreover, the block operator $\mathbf{\Phi}$ applies the scalar LUT Φ to each element of a transformed image, except for the DC elements, i.e., if $((\cdot))_L$ denotes the remainder after the integer division by L, then

$$\mathbf{\Phi}([a_{mn}]) = \begin{cases} \Phi_{mn}(a_{mn}), & ((m))_L \neq 0 \text{ or } ((n))_L \neq 0, \\ a_{mn}, & \text{otherwise.} \end{cases} \tag{8}$$

In the original scheme, the soft-thresholding LUT is used for Φ. Our main contribution is to replace this LUT by one that also performs sharpening, as seen in the next subsection.

Proposition 1 *Assuming that* $\mathbf{T}$ *is separable, and generated by a* $L \times L$ *1D transform* T*, then the value* $Out(i,j)$ *of* Out *at a specific pixel* $i = (i_1, i_2)$ *away from the image border is given by:*

$$Out(i_1, i_2) = \frac{1}{K} \sum_{(k_1,k_2) \in G_k} \left(T^{-1}\right)_{((i_1+k_1))_L} \mathbf{\Phi}\left(T \cdot B^{i_1,i_2}_{k_1,k_2}(In) \cdot T^t\right) \left[\left(T^{-1}\right)_{((i_2+k_2))_L}\right]^t, \tag{9}$$

where $(A)_j$ *denotes the* j^{th} *row of a matrix* A*, and* $B^{i_1,i_2}_{k_1,k_2}(In)$ *is the* $L \times L$ *block of the image given by:*

$$\begin{aligned} B^{i_1,i_2}_{k_1,k_2}(In) &= \begin{pmatrix} In(b_1, b_2) & \cdots & In(b_1, b_2 + L - 1) \\ \vdots & \vdots & \vdots \\ In(b_1 + L - 1, b_2) & \cdots & In(b_1 + L - 1, b_2 + L - 1) \end{pmatrix}, && (10) \\ b_1 &= L\lfloor (i_1 + k_1)/L \rfloor - k_1, && (11) \\ b_2 &= L\lfloor (i_2 + k_2)/L \rfloor - k_2. && (12) \end{aligned}$$

Above, $\lfloor \cdot \rfloor$ *denotes the integer truncation, and the signal processing index convention,* $i = 0, \ldots, N-1$*, is used.*

Proof To keep the proof short, we prove here for 1D signals only. The proof for 2D is a simple algebraic extension. In the 1D case, (9) becomes:

$$Out(i) = \frac{1}{K} \sum_{k \in G_k} \left(T^{-1}\right)_{((i+k))_L} \mathbf{\Phi}\left(T \cdot B^i_k(In)\right), \tag{13}$$

with

$$\begin{aligned} B^i_k(In) &= \begin{pmatrix} In(b) & \cdots & In(b + L - 1) \end{pmatrix}^t, && (14) \\ b &= L\lfloor (i + k)/L \rfloor - k, && (15) \end{aligned}$$

and this is what we prove now.

Notice that, in the 1D case, $\mathbf{T}$ can be represented by the $N \times N$ matrix obtained by the Kroenecker product between the $N/L \times N/L$ identity matrix and the transform matrix T.

We define $\mathbf{1}_i$ as the $1 \times N$ vector containing zeros everywhere, except for a 1 at the $(i+1)^{th}$ position. When i is sufficiently away from the border (so that $0 \leq i + k < N$), then $\mathbf{1}_i S_{-k} = \mathbf{1}_{i+k}$, and therefore (7) becomes:

$$\begin{aligned} Out(i) &= \frac{1}{K} \sum_k \mathbf{1}_i S_{-k} \mathbf{T}^{-1} \mathbf{\Phi}\left(\mathbf{T} S_k In\right), \\ &= \frac{1}{K} \sum_k \mathbf{1}_{i+k} \mathbf{T}^{-1} \mathbf{\Phi}\left(\mathbf{T} S_k In\right). && (16) \end{aligned}$$

Since we have

$$\mathbf{1}_{i+k}\mathbf{T}^{-1} = \left[\underbrace{0\ldots0}_{L\lfloor(i+k)/L\rfloor} \left| \left(T^{-1}\right)_{((i+k))_L} \right| \underbrace{0\ldots0}_{N-L-L\lfloor(i+k)/L\rfloor} \right], \tag{17}$$

then

$$\begin{aligned} Out(i) &= \frac{1}{K}\sum_k \left(T^{-1}\right)_{L\lfloor(i+k)/L\rfloor} \left[\mathbf{\Phi}\left(\mathbf{T}S_k In\right)\right]_{L\lfloor(i+k)/L\rfloor+L-1}^{L\lfloor(i+k)/L\rfloor} \\ &= \frac{1}{K}\sum_k \left(T^{-1}\right)_{L\lfloor(i+k)/L\rfloor} \mathbf{\Phi}\left(\left[\mathbf{T}S_k In\right]_{L\lfloor(i+k)/L\rfloor+L-1}^{L\lfloor(i+k)/L\rfloor}\right) \\ &= \frac{1}{K}\sum_k \left(T^{-1}\right)_{L\lfloor(i+k)/L\rfloor} \mathbf{\Phi}\left(T\cdot\left[S_k In\right]_{L\lfloor(i+k)/L\rfloor+L-1}^{L\lfloor(i+k)/L\rfloor}\right) \\ &= \frac{1}{K}\sum_k \left(T^{-1}\right)_{L\lfloor(i+k)/L\rfloor} \mathbf{\Phi}\left(T\cdot\left[In\right]_{L\lfloor(i+k)/L\rfloor+L-1-k}^{L\lfloor(i+k)/L\rfloor-k}\right). \\ &= \frac{1}{K}\sum_k \left(T^{-1}\right)_{L\lfloor(i+k)/L\rfloor} \mathbf{\Phi}\left(T\cdot B_k^i\left[In\right]\right). \end{aligned} \tag{18}$$

$\circ$

Note that the above characterization of $Out(i)$ does not imply that (9) yields a good algorithm for its computation. In fact, (7) is more effective.

4.1 Replacing the soft threshold LUT with the Robust LUT

We propose a simultaneous denoising and sharpening procedure in the transform domain. Following the soft thresholding framework, we suppress the noise by mapping transform coefficients that are smaller than a given constant, Lth, to zero. In addition, we keep the overall sharpness of the image by magnifying the transform coefficients with larger magnitudes. The sharpening effect is bounded for coefficient values above a second threshold Hth to avoid over-sharpening. The described look-up table, named *Robust LUT* (solid line in Figure 2), is given by:

$$\Phi(x) = \text{sign}(x)\begin{cases} 0 & |x| < L_{th} \\ \lambda\left(|x| - L_{th}\right) & L_{th} \le |x| \le H_{th} \\ SF + |x| & |x| > H_{th} \end{cases} \tag{19}$$

where λ is the slope of the line connecting $(L_{th}, 0)$ and $(H_{th}, H_{th}{+}SF)$, and $SF = (\lambda - 1)\,H_{th} - \lambda\cdot L_{th}$. The robust LUT can be viewed as robust sharpening combined with the soft thresholding LUT of Donoho (dotted line in Figure 2).

4.1.1 Choosing the transform

The block DCT transform seems to be the most appropriate for practical uses. One reason is its property of dispersing the errors in a visually less-distracting way. Other arguments in favor of the block DCT transform are its theoretic resemblance to the Karhunen-Loeve transform, in which transform artifacts are minimal [10], and its efficient implementation in image capture products as part of the JPEG compression pipeline. Working in block DCT transform enables us to combine the suggested algorithm with Samadani's JPEG artifact removal algorithm [5].

When working with the block DCT transform, we suggest utilizing the separability of the DCT to dramatically reduce the computational overhead associated with computing the various shifted forms of the DCT and inverse DCT transforms, by re-using intermediate results across shifts.

4.1.2 Weighted average of the enhanced shifted-back images

The closer the pixel is to the edges of the block, the more likely it is to contain transform artifacts [11]. To further decrease transform artifacts we use *weighted* average of the enhanced shifted-back images, where the weights are a function of the distance from the edge of the block.

5 Discussion

5.1 Alternative Approaches

The reader may ask: "Why to embed sharpening within the denoising procedure, instead of just apply a standard sharpening step after soft-thresholding?" First, the proposed scheme is significantly faster and requires much less buffer memory, because it requires only one pass over the image data, as opposed to two, as required by post-sharpening, and because the embedding of sharpening costs precisely nothing (it is just the modification of the values of a look-up table). Furthermore, sharpening within the context of the a pre-selective filter inherits itself the benefits of selectivity; for instance, it adapts itself to the local configuration of the image. As a result, the quality of the embedded scheme is higher.

The option of merging image enhancement and denoising by increasing the wavelet coefficient instead of decreasing them was mentioned by Jung and Scharcanski [12]. Jung and Scharcanski calculate the probability that each wavelet coefficient is either an edge or noise, based on the neighboring wavelet coefficients and the corresponding wavelet coefficients in different levels, and treat each wavelet coefficient differently, as a function of those parameters. This suggestion is much more complicated and computationally demanding than our approach. A different suggestion for bringing sharpening or de-blurring to the same framework is brought by Berkner *et al.* [13]. Berkner *et al.* use the Besov space framework which leads to multiplying the transform coefficients by a constant, which is calculated from prior knowledge of the sharpness of the entire image.

5.2 Why it is Second Generation

In the Introduction, we mention that we consider the proposed scheme to be a second-generation robust filter. We discuss the reason for that in this sub-section.

First, consider the following proposition.

Proposition 2 *In the 1D case, when the block size is 2 pixels, the proposed operator with $LUT\ \Phi(x)$ is equivalent to the pre-selective filter with $LUT\ \Psi(x) = x - \Phi(x)$.*

Proof According to Proposition 1, the output value for the i^{th} pixel is given by:

$$Out(i) = \frac{1}{2}\sum_{k=0}^{1}\left(T^{-1}\right)_{((i+k))_2} \mathbf{\Phi}\left(T \cdot B^{(2)}_{2\lfloor (i+k)/2\rfloor - k}(In)\right). \tag{20}$$

Similarly to most common transforms (including the DFT, Haar DWT, discrete Haddamard transform, etc.), the 2-point 1D DCT transform and its inverse are given, respectively, by the matrices:

$$T = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \text{ and } T^{-1} = \frac{1}{2}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}. \tag{21}$$

Notice that, since any arbitrary integer n satisfies $n = L\lfloor n/L\rfloor + ((n))_L$, we can express (18) as:

$$Out(i) \;=\; \frac{1}{K}\sum_{k}\left(T^{-1}\right)_{L\lfloor (i+k)/L\rfloor} \mathbf{\Phi}\left(T \cdot [In]^{i-((i+k))_L}_{i-((i+k))_L+L-1}\right). \tag{22}$$

Fixing the index i, it can be ether $((i))_2 = 0$ or $((i))_2 = 1$. In both cases, for $k = 0, 1$, (22) becomes:

$$\begin{aligned} Out(i) &= \frac{1}{2}\left[(T^{-1})_0 \Phi T \begin{pmatrix} In(i) \\ In(i+1) \end{pmatrix} + (T^{-1})_1 \Phi T \begin{pmatrix} In(i-1) \\ In(i) \end{pmatrix}\right] \\ &= \frac{1}{2}\left[\frac{1}{2}\left[In(i) + In(i+1) + \Phi(In(i) - In(i+1))\right]\right. \\ &\quad \left. + \frac{1}{2}\left[In(i-1) + In(i) - \Phi(In(i-1) - In(i))\right]\right]. \end{aligned} \tag{23}$$

And because Φ is antisymmetric, we get

$$Out(i) = In(i) + \frac{1}{4}\sum_{j=i\pm 1}\left[(In(j) - In(i)) - \Phi\left(In(j) - In(i)\right)\right].$$

Setting $\Psi(x) = x - \Phi(x)$, $N(i) = \{i-1, i, i+1\}$, and $h(k) \equiv 1/4$, for $k = 0, 1$, leads to the pre-selective filter (3).

$\circ$

In other words, in the simplest case of a 2×1 neighborhood, the proposed scheme is in fact a robust filter, namely, the pre-selective one. This fact alone indicates that the proposed filter belongs to the family of robust filters.

On the other hand, as the size of the neighborhood increases, the proposed filters departs from the first-generation robust filter, and becomes more complex. For instance, Appendix D presents the case when the block size is 2x2 pixels, which is quite different from the 2×2 pre-selective filter.

It can be shown that the first-generation filters are most appropriate to deal with images that can be well-approximated by a piece-wise constant function, in which case it removes noise without distorting edges. This is usually the case for small images, such as 256×256-pixel pictures or smaller, where most edges are typically very sharp.

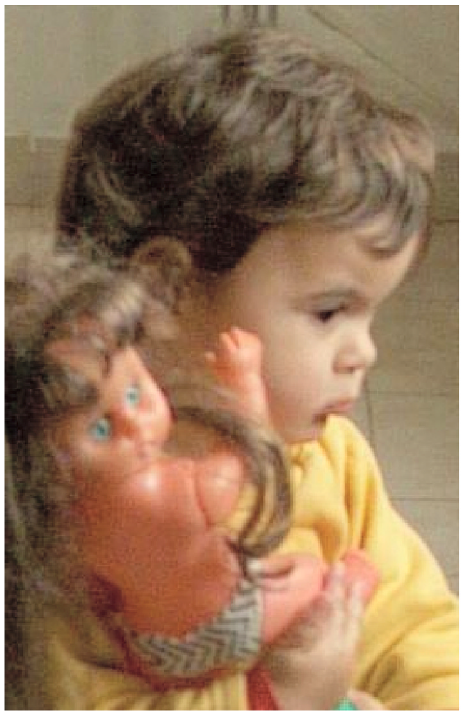
For larger images, such as those acquired by a high-mega-pixel digital camera, or scanned at a large LPI setting, the image is usually significantly better approximated by harmonic functions. In this case, a first-order robust filter tends to over-sharpen the edges, which often creates a posterization effect. Here, a second-generation algorithm, such as the proposed filter, leads to better (more visually pleasing) results.

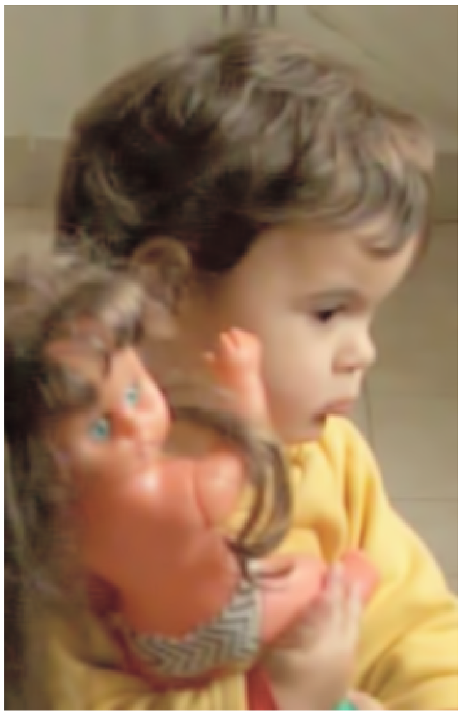
6 Results

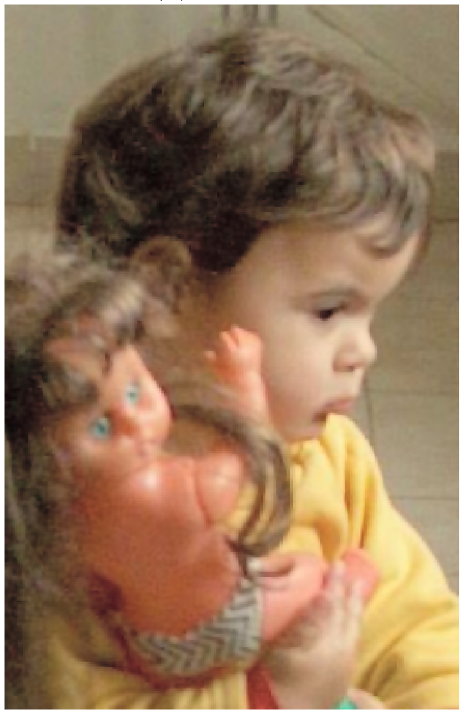
Simulations comparing the proposed scheme to the soft-thresholding and two kinds of first-generation robust filters were conducted. Three images were selected for display here, each of a different nature: Fig. 3(a) is an enhanced version (using HP's adaptive lighting) of an image captured by a digital camera; Fig. 4(a) is a 400-dpi scan from a silver-halide negative; and Fig. 5(a) is a standard low-resolution image, which has been artificially noised by us.

The comparison to the soft-thresholding and the pre-selective filter are shown in Figs. 3, 4, and 5. For each image, the parameters of the algorithms were tuned to obtain the visually best result. As expected, the application of the proposed use of the robust LUT in all cases produces a sharper image than the one denoised by soft-thresholding, while their noise level is equivalent. As for the first-generation pre-selective filter, although improved, there is still a large amount of remaining noise, as compared to the proposed and the soft-thresholding schemes.

Figs. 6 and 7 compare the proposed algorithm to the post-selective filter (4), both using the same kernel size of 8×8. The chosen kernel is an extension of a Laplacian-like 3×3 kernel. As seen in the figures, when viewed in a low scale, the overall sharpness and noise removal


(a) Original


(b) Soft-Threshold


(c) First generation robust

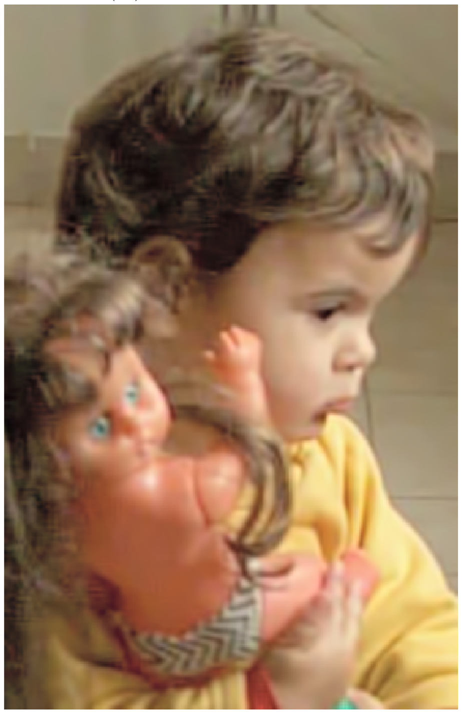
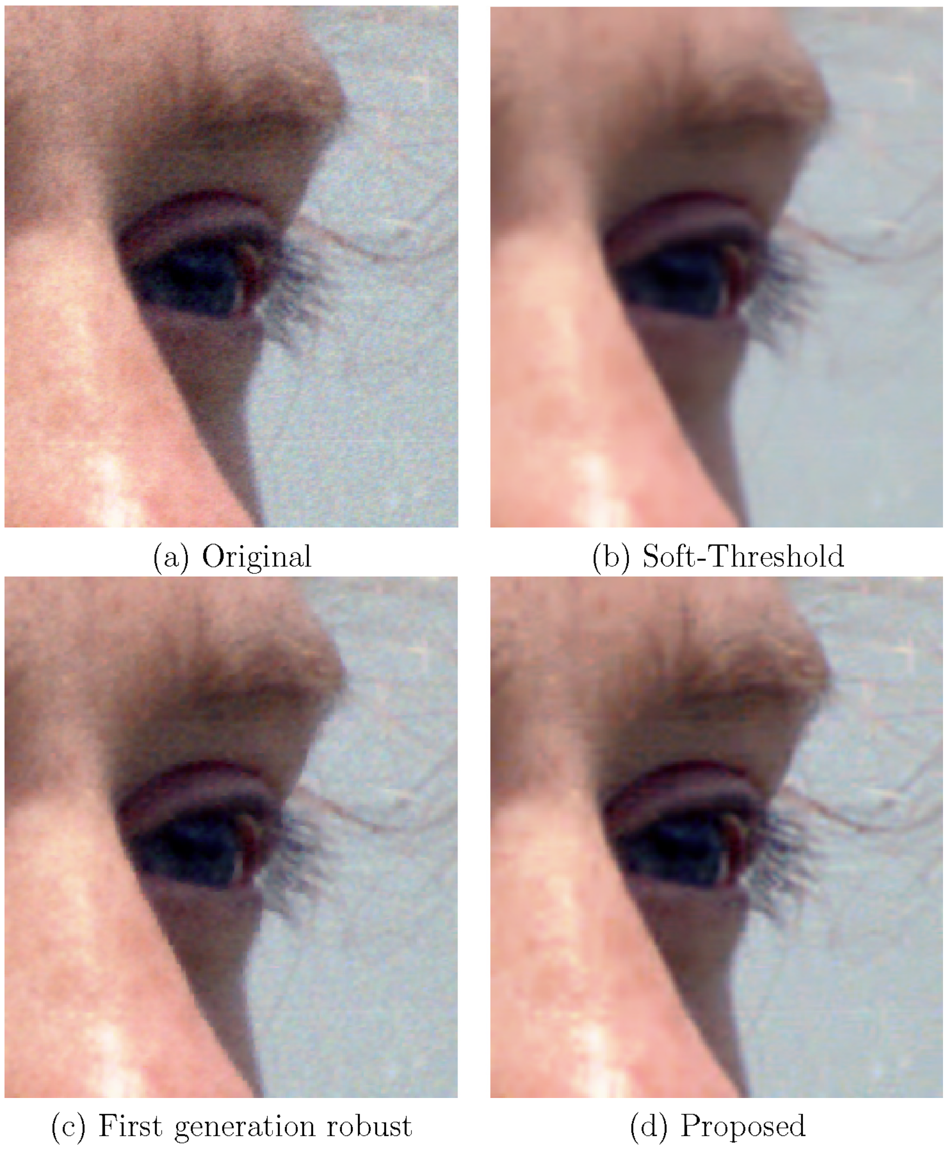
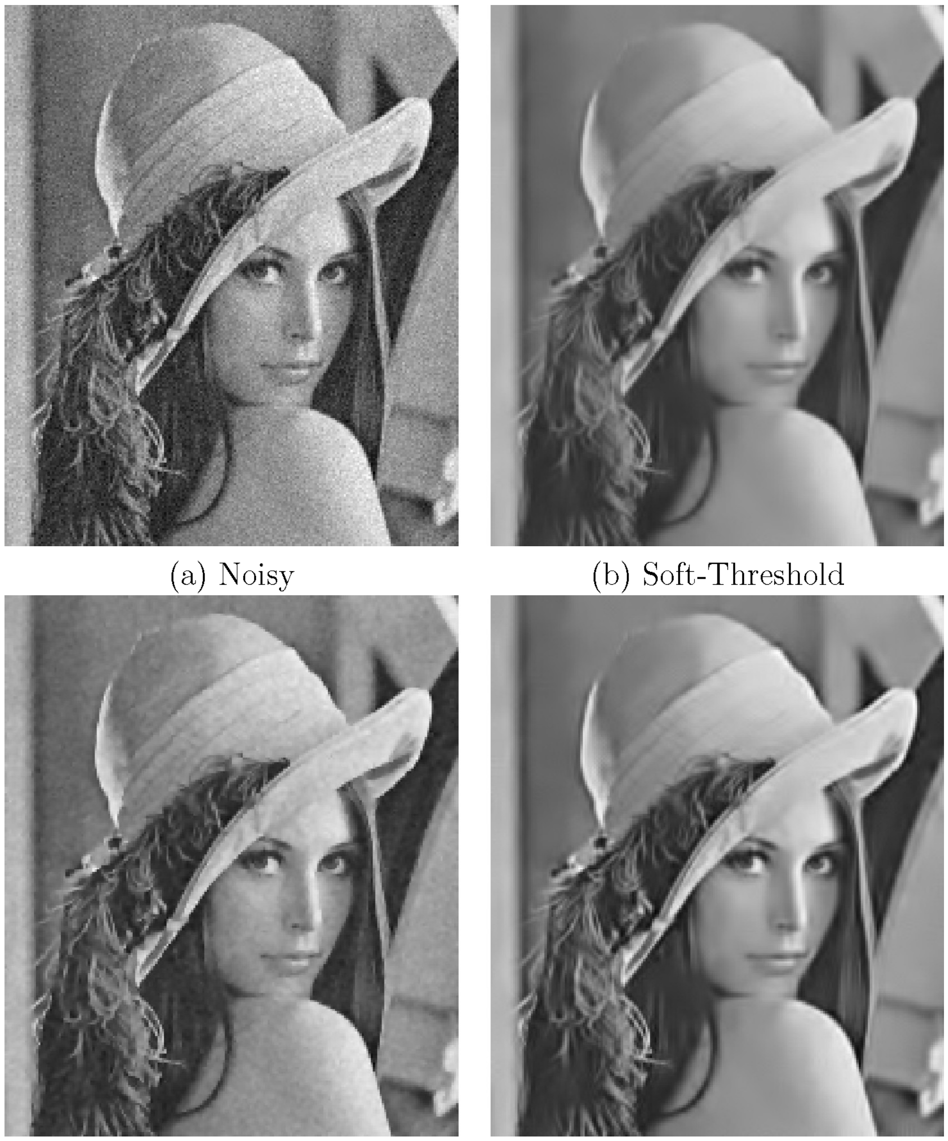

(d) Proposed

Figure 3: (a) original digital camera image, enhanced by adaptive lightning, (b) the visually best ($L_{th} = 16$) result obtained using the shift-invariant soft-threshold algorithm with the 8x8 Block DCT transform and weighted average, (c) pre-selective filter, with $L_{th} = 16$, $H_{th} = 32$, and $SF = 16$, and (d) proposed robust LUT with the same parameters as in (c).


(a) Original (b) Soft-Threshold

(c) First generation robust (d) Proposed

Figure 4: (a) Original noisy image, scanned from a film photograph at 4000 dpi, and down-scaled, (b) shift-invariant soft-threshold algorithm with $L_{th} = 15$, the 8x8 Block DCT transform, and weighted average, (c) pre-selective filter with $L_{th} = 15$, $H_{th} = 30$, and $SF = 15$, and (d) the proposed robust-LUT scheme, with the same parameters as in (c).



(a) Noisy (b) Soft-Threshold

(c) First generation robust (d) Proposed

Figure 5: (a) Original Lena image, corrupted by Gaussian noise with standard deviation of 10 gray levels, (b) shift-invariant soft-threshold algorithm with $L_{th} = 25$, the Block DCT transform, and weighted average, (c) pre-selective filter with $L_{th} = 25$, $H_{th} = 100$, and $SF = 25$, and (d) the proposed robust-LUT scheme, with the same parameters as in (c).

of the two algorithms look about equal; in fact the post-selective result might look cleaner and sharper to some. However, when viewed at a larger scale, one can see that, in fact, the post-selective result is full of artifacts, whereas the proposed scheme is significantly cleaner. These same artifacts of the post-selective scheme are what gives a sharpness impression to the low-scale images; so this sharpness impression is in fact an illusion.

7 Extension to video denoising

We propose to extend this image enhancement algorithm to video denoising, by using a three dimensional transform instead of the two dimensional one. Extending the algorithm from two dimensions to three dimensions increases the data available to the smoothing filter, which results in better denoising and more stable edges.

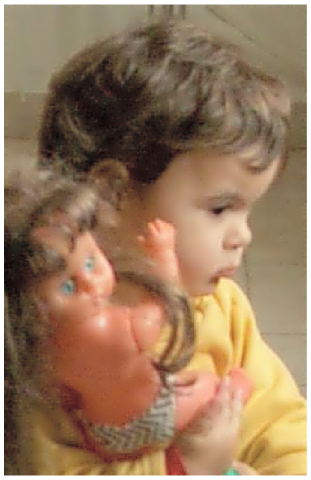
Similar attempts were done by Boo and Bose [14], and Selesnick and Li [15], for slow varying scenes. Our proposal is also valid for abrupt changes, because of the selectivity of the robust LUT. This feature of the robust LUT enables smoothing flat regions in the image, while avoiding smoothing over edges, resulting in noise removal while maintaining image sharpness. As a result, artifacts due to fast moving objects and changing scenes are prevented. The Robust LUT bypasses the need to follow the objects in the scene, as is done in many optical flow based video denoising algorithms, and avoids the related cumbersome calculations.

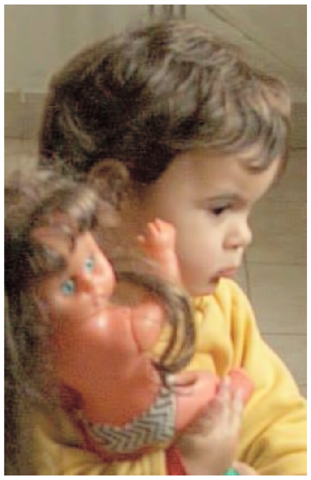
An additional important advantage of our video denoising algorithm is compression-artifact reduction. As shown by Donoho [2] in the two dimensional case, over-sampled image transforms tend to overcome quantization artifacts, which do occur with critically-sampled transforms. Samadani *et al.* [5] showed that this approach can be used to reduce previously introduced JPEG artifacts. Our video denoising algorithm extends the compression artifact reduction property into the three dimensional case, where it shows good MPEG-artifact reduction.

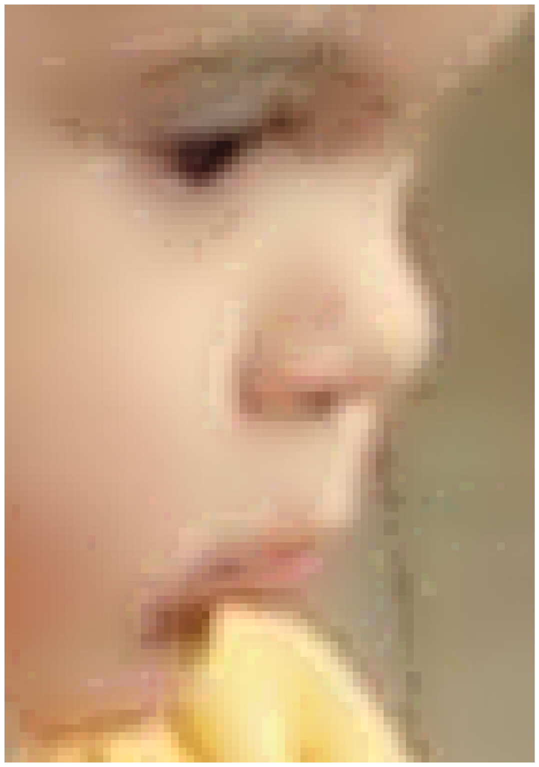
7.1 The proposed algorithm

The algorithm can best be described using the pseudo-code shown in Fig. 8[1]. For the purposes of this example, the variable *video* is an array of size $height \times width \times frames$ containing the whole video data stream. We iterate over the dataset, and for each pixel we take its 8x8x8 neighborhood (line 5) and transform to the frequency domain using a 3D transform (line 6). The actual denoising and artifact reduction happens in line 7, where the coefficients values are modified. We prefer to use the selective sharpening and smoothing modification shown in Figure 2 using an LUT, which was described in section 4. The dashed line in Figure 2 is the identity line, the dotted line is the soft threshold LUT and the solid line is the Robust LUT. *Lth* is the low threshold under which coefficients are mapped to zero. *Hth* is the high threshold above which the Robust LUT becomes parallel to identity, with a distance called

[1]The actual implementations differ from this example in order to constrain the memory requirements.


(a) Results of the first generation filter


(b) Results of the proposed algorithm


(c) Results of the first generation filter

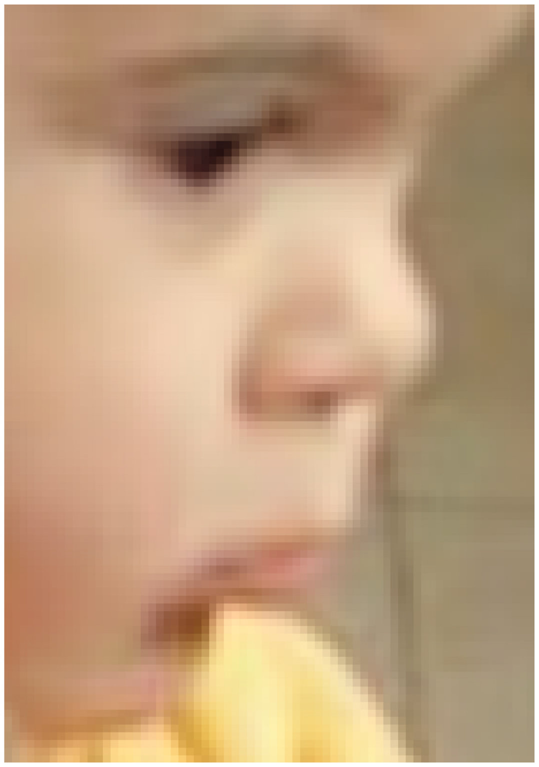
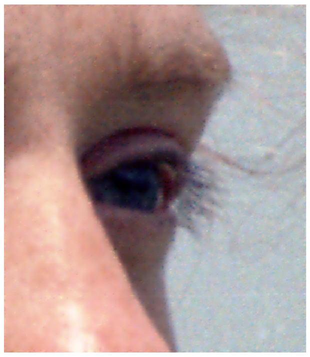
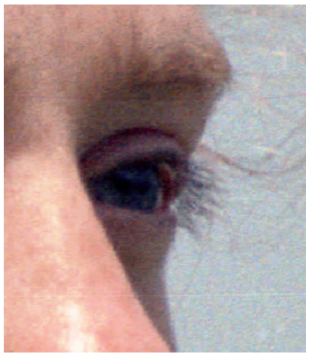
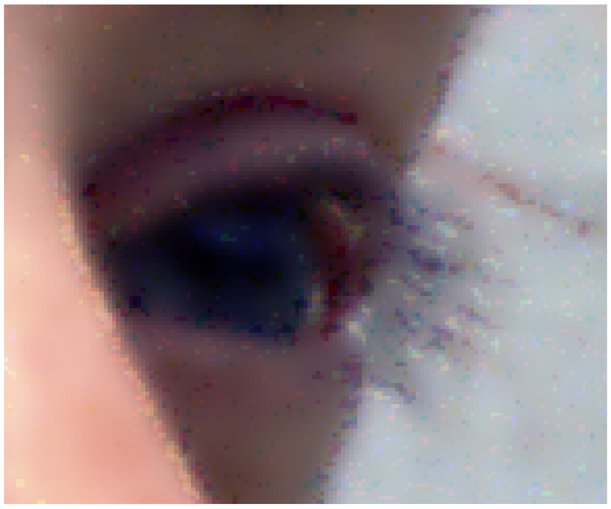

(d) Results of the proposed algorithm

Figure 6: A comparison between the proposed algorithm to the post-selective filter for the image in Fig. 3(a). (a)-(b) Result of the post-selective filter and the proposed scheme, respectively, at a low scale. (c)-(d) Result of the same algorithms at a larger scale, where images have been cropped, in order for the details to be visible.


(a) Results of the first generation filter


(b) Results of the proposed algorithm


(c) Results of the first generation filter

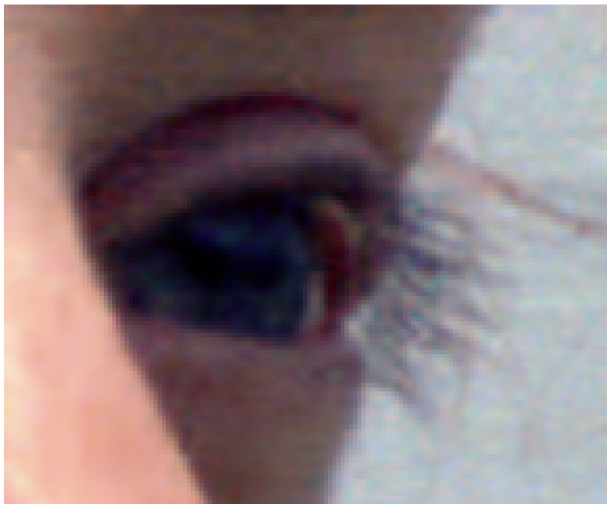

(d) Results of the proposed algorithm

Figure 7: A comparison between the proposed algorithm to the post-selective filter for the image in Fig. 4(a). (a)-(b) Result of the post-selective filter and the proposed scheme, respectively, at a low scale. (c)-(d) Result of the same algorithms at a larger scale, where images have been cropped, in order for the details to be visible.

"sharpening factor" (SF) between them. Any one of a number of methods, such as hard thresholding or JPEG-like quantization can be used instead this LUT. We then transform back to the spatial domain and take the average (lines 8–9).

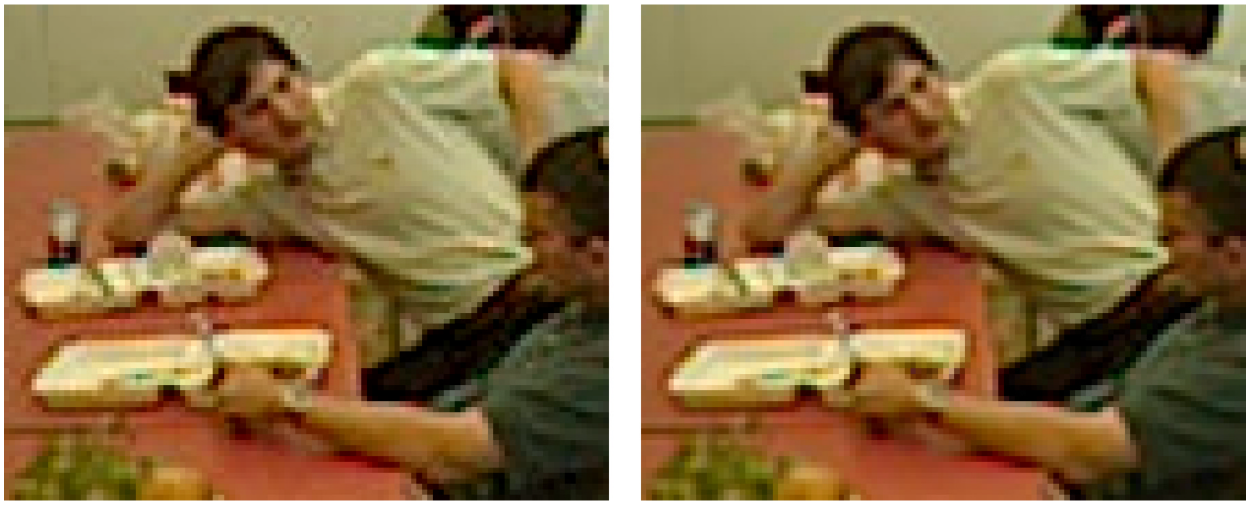
```
 1  output = zeros(height, width, frames);
 2  for f=1:frames-8,
 3        for y=1:height-8,
 4              for x=1:width-8,
 5                     block = video(y:y+8, x:x+8, f:f+8);
 6                     block = 3Dtransform(block);
 7                     block = modify(block);
 8                     block = 3DInversetransform(block) / (8 * 8 * 8);
 9                     output(y:y+8, x:x+8, f:f+8) = output(y:y+8, x:x+8, f:f+8) + block;
10              end;
11        end;
12  end;
```

Figure 8: Matlab-like pseudo code

7.2 Results

Figure 9 shows one specific frame from a video sequence: (a) is the original frame and (b) is denoised using the proposed algorithm. It is apparent that we were able to remove most of the noise while keeping most of the image original sharpness.

The obvious alternative to 3D video denoising is to simply use the 2D algorithm on each frame independently. However, tests show that this may result in flickering between frames, and other artifacts, because slight differences in adjacent image frames can result in different denoising results, which in turn can lead to temporal artifacts, particularly near edges. By using a three dimensional transform, the algorithm makes far fewer mistakes of this form, as the information from the adjacent frames is included in the transform, so its results near edges are more stable in time, and there are far fewer of these sorts of artifacts. Figure 10 demonstrate the difference between denoising a video using the 2D algorithm on each frame independently (sub-figures a and c) and using the 3D algorithm (sub-figures b and d). Both algorithms use the same Robust LUT, with Lth=15, Hth=45 and Sco=15. Figures 10(a) and (b) are obtained from Figure 9(a), by using the proposed 2D and 3D algorithms respectively. One can see that the results of the 3D denoising are sharper, as seen in the writing on the white-board, and in the texture of the basket (bottom left of each image). Figures 10(c) and (d) demonstrate the flickering in the 2D (c) and 3D (d) denoising algorithms by showing the differences between the denoised frame brought in 10(a) and (b), and it's sequential frame. One can see that there are far more differences between sequential frames when using the Robust LUT on each frame independently, which results in flickering.



(a) Noisy input image (b) Output image

Figure 9: Sample digital camera video frames

8 Acknowledgments

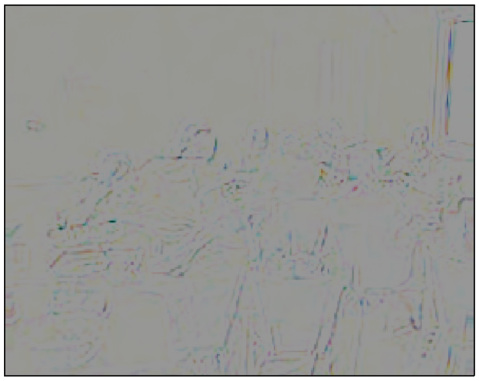
We gratefully acknowledge the advice and feedback of Ramin Samadani and Niranhan Damera-Venkata.

9 Conclusions

An image enhancement algorithm based on Donoho's translation-invariant soft-threshold denoiser for grayscale images is presented. Our algorithm overcomes the blurring effect typical to this and other denoising algorithms by replacing the soft threshold LUT with the Robust LUT. This Robust LUT suppresses transform coefficients representing mostly the noise, and enhances transform coefficients representing mostly the noise-free image features. We adapt the algorithm to practical constrains by using the block DCT, which is efficiently implemented in most image capture devices. The framework was extended to three dimensions for video denoising.

(a) 2D transform denoising

(b) 3D transform denoising



(c) Difference map: 2D denosing

(d) Difference map: 3D denosing

Figure 10: The results obtained when applying the 2D algorithm on each frame independently (a) and when applying the proposed 3D denoising (b). Both algorithms use the same Robust LUT, with Lth=15, Hth=45 and Sco=15. (c) and (d) are the difference between sequential frames, or rather 151 minus the absolute difference between the two frames, where 151 is the maximal difference for the 2D denoising. The maximal difference when using the 3D algorithm is smaller: 124 gray levels.

10 References

[1] R. R. Coifman and D. L. Donoho, “Translation-invariant de-noising,” in *Wavelet and Statistics*, ser. Lecture Notes in Statistics, A. Antoniadis and G. Oppenheim, Eds. Berlin: Springer, 1995, pp. 125 – 150, http://www-stat.stanford.edu/~donoho/reports/1995/TIDeNoise.pdf.

[2] D. L. Donoho, “De-noising by soft-thresholding,” *IEEE Transactions on Information Theory*, vol. 41, no. 3, pp. 613–627, May 1995.

[3] R. Maurer, “Selective smoothing and sharpening of images by generalized unsharp masking,” Patent number 6665448, December 16 2003.

[4] A. Nosratinia, “Enhancement of JPEG-compressed images by re-application of JPEG,” *Journal of VLSI Signal Processing*, vol. 27, pp. 69–79, 2001.

[5] R. Samadani, A. Sundararajan, and A. Said, “Deringing and deblocking dct compression artifacts with efficient shifted transforms,” in *Proceedings of the International Conference on Image Processing (ICIP)*, vol. 3, October 2004, pp. 1799–1802.

[6] D. L. Donoho and I. M. Johnstone, “Minimax estimation via wavelet shrinkage,” *Annals of Statistics*, vol. 26, no. 3, pp. 879–921, 1998. [Online]. Available: citeseer.nj.nec.com/donoho92minimax.html

[7] N. L. A. Chambolle, R. DeVore and B. Lucier, “Nonlinear wavelet image processing: Variational problems, compression, and noise removal through wavelet shrinkage,” *IEEE Tran. Image Proc.*, vol. 7, no. 3, pp. 319–333, 1998. [Online]. Available: citeseer.nj.nec.com/chambolle96nonlinear.html

[8] J. Kalifa and S. Mallat, “Thresholding estimators for linear inverse problems,” *Annals of Statistics*, vol. 31, no. 1, pp. 58–109, February 2003.

[9] E. P. Simoncelli, *Bayesian inference in wavelet based models*, ser. Lecture Notes in Statistics. New York, NY: Springer-Verlag, 1999, vol. 141, ch. 18, pp. 291–308. [Online]. Available: www.cns.nyu.edu/pub/eero/Simoncelli98e.pdf

[10] A. K. Jain, *Fundamentals of Image Processing.* Prentice-Hall, 1989.

[11] O. G. Guleryuz, “Subspaces of quantization artifacts for image transform compression,” in *Proceedings of the International Conference on Image Processing*, vol. 3, September 2000, pp. 865–868. [Online]. Available: citeseer.nj.nec.com/288612.html

[12] C. R. Jung and J. Scharcanski, “Adaptive image denoising and edge enhancement in scale-space using the wavelet transform,” *Pattern Recognition Letters*, vol. 24, no. 7, pp. 965–971, April 2003.

[13] K. Berkner, M. J. Gormish, and E. L. Schwartz, "Multiscale sharpening and smoothing in Besov Spaces with applications to image enhancement," *Applied and Computational Harmonic Analysis*, vol. 11, pp. 2–31, 2001.

[14] K. J. Boo and N. K. Bose, "A motopm compensated spatio-temporal filter for image sequences with signal dependent noise," *IEEE transactions on circuits and systems for video technology*, vol. 8, no. 3, 1998.

[15] I. W. Selesnick and K. Y. Li, "Video denoising using 2d and 3d dual-tree complex wavelet transforms," in *Proc 48'th SPIE on Wavelets X*. San Diego USA: SPIE, 3-8 August 2003.

[16] M. L. Green, "Statistics of images, the TV algorithm of Rudin-Osher-Fatemi for image denoising and an improved denoising algorithm," UCLA CAM, Tech. Rep. 02-55, October 2002.

[17] J. Huang, "Statistics of natural images and models," Ph.D. dissertation, Division of Applied Mathematics at Brown University, Providence, RI, 2000.

[18] E. P. Simoncelli and E. H. Adelson, "Noise removal via Bayesian wavelet coring," in *Proceedings 3rd IEEE International Conference on Image Processing*, vol. 1, 1996, pp. 379–382.

[19] J. L. P. Moulin, "Analysis of multiresolution image denoising schemes using generalized gaussian and complexity priors," *Information Theory, IEEE Transactions on*, vol. 45, pp. 909 – 919, 1999.

[20] M. Mitrea, "A note on Besov regularity of layer potentials and solutions of elliptic PDE's," *Proceedings American Math. Soc.*, vol. 130, pp. 2599–2607, 2002. [Online]. Available: http://www.math.missouri.edu/˜marius/betterB4.pdf

[21] J. Bergh and J. Lofstrom, *Interpolation Spaces, An Introduction*. New York: Springer-Verlag, 1976.

[22] Y. Meyer, *Wavelets and Operators*, ser. Cambridge Studies in Advanced Mathematics. Cambridge University Press, 1993, vol. 37.

[23] R. Devore, B. Jawerth, and B. J. Lucier, "Image compression through wavelet transform coding," *IEEE Transactions on Information Theory*, vol. 38, no. 2, pp. 719–746, March 1992.

[24] S. Mallat, *A Wavelet tour of signal processing*. Academic Press, 1998.

[25] R. Devore and V. Popov, "Interpolation of besov spaces," *Transactions of the american methematical society*, vol. 305, no. 1, pp. 397–414, January 1988.

Appendices

A Optimality of Soft-Thresholding: Bayesian approach

Given a natural image X, the distribution of any zero mean projection coefficients on X can be described by a generalized Laplacian distribution [16, 17, 18]. Given an image Y that was created from X by adding i.i.d. zero mean Gaussian noise N, the white noise remains a Gaussian white noise under any linear transform. We can estimate X from the corrupted observation Y by maximizing the a-posteriori (MAP) estimator. We estimate the value for the original noise free pixel by:

$$\hat{x}(y) = argmax_x \left(P\left(x|y\right)\right)$$

Applying Bayes rule leads to:

$$\hat{x}(y) = argmax_x \left(P(y/x)\frac{P(x)}{P(y)}\right)$$

and since the value of y is constant for the argmax operation, we have:

$$\begin{aligned}\hat{x}(y) &= argmax_x \left(P(y/x)P(x)\right) \\ &= argmax_x \left(P((x+n)/x)P(x)\right) \\ &= argmax_x \left(P(n)P(x)\right) \\ &= argmax_x \left(log\left(P(n)\right) + log\left(P(x)\right)\right)\end{aligned}$$

assuming a generalized Laplacian noise distribution:

$$P_n(x) = K_n \cdot e^{-\left|\frac{x}{s_n}\right|^{p_n}} \tag{24}$$

and a generalized Laplacian prior for the transform coefficients,

$$P_s(x) = K_s \cdot e^{-\left|\frac{x}{s_s}\right|^{p_s}} \tag{25}$$

where p_n, and p_s are the respective generalized Laplacian distribution parameters, such that for example, $p = 1$ results in a Laplacian distribution, and $p = 2$ results in a Gaussian distribution. K_n and K_s are the normalization factors

$$K_n = \frac{p_n}{2s_n\Gamma\left(\frac{1}{p_n}\right)}, \qquad K_s = \frac{p_s}{2s_s\Gamma\left(\frac{1}{p_s}\right)}$$

And

$$s_s = \sigma_x\sqrt{\frac{\Gamma\left(\frac{1}{p_s}\right)}{\Gamma\left(\frac{3}{p_s}\right)}}, \qquad s_n = \sigma_n\sqrt{\frac{\Gamma\left(\frac{1}{p_n}\right)}{\Gamma\left(\frac{3}{p_n}\right)}} \tag{26}$$

where σ_n, and σ_s are the standard deviations of the noise and transform coefficients respectively. When image prior and noise distribution are described by generalized Laplacian (equations 24,25), the estimated value for the original noise free pixel is:

$$\hat{x}(y) = argmin_x \left\{ \left| \frac{y-x}{s_n} \right|^{p_n} + \left| \frac{x}{s_s} \right|^{p_s} \right\}. \tag{27}$$

Figure 11 illustrates some of the LUTs obtained when solving Equation 27 for a Gaussian

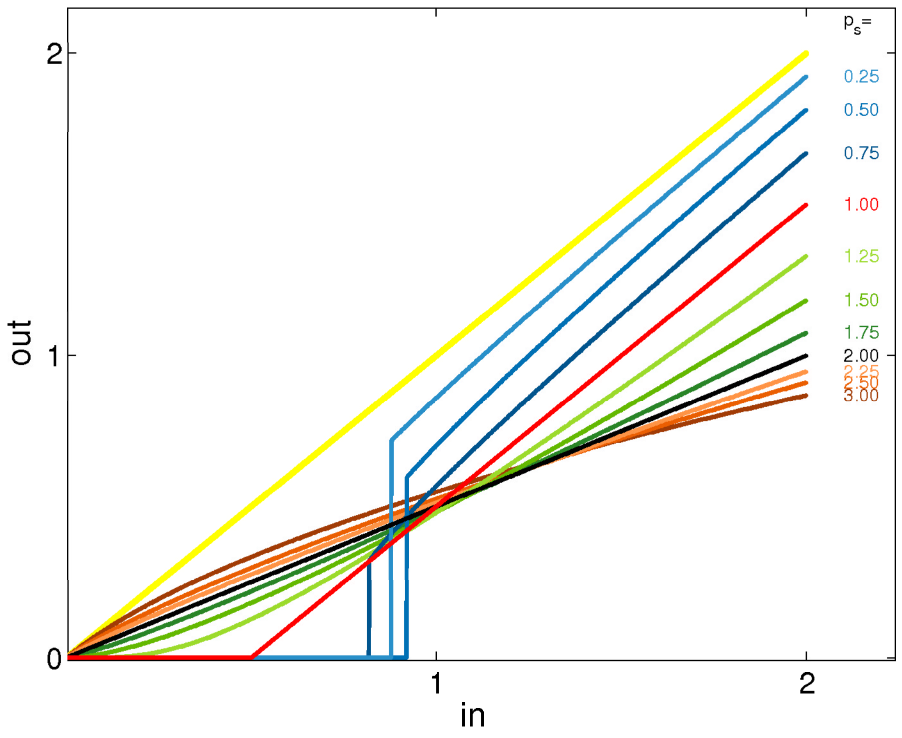


Figure 11: The estimated noise-free pixel value given the noisy value, for a Gaussian noise ($p_n = 2, s_n = 1, s_s = 1$) for several image priors: The blue colored lines represents $p_s = 0.25, 0.5, 0.75$, from light to dark, accordingly. Following is the soft threshold ($p_s = 1$) in red, $p_s = 1.25, 1.5, 1.75$ in darkening green, $p_s = 2$ yields a linear line in black, and priors which are wider then the noise, $p_s = 2.25, 2.5, 3$, where the noise is sparser than image prior, yields the LUTs drawn in darkening orange colors. The yellow line is the identity line.

noise ($p_n = 2, s_n = 1, s_s = 1$) for various values of p_s. An estimation for the LUT curve for a general p_s value is suggested in [19]. We are not aware of a general closed analytical solution for this equation.

For the simple case of i.i.d. Gaussian noise ($p_n = 2$,$s_n = \sigma_n/\sqrt{2}$) and a Laplacian prior

$(p_s = 1, s_s = \sigma_x/\sqrt{2})$, the distributions are:

$$P(n) = \frac{e^{-\frac{n^2}{2\sigma_n^2}}}{\sqrt{2\pi}\sigma_n}, \qquad P(x) = \frac{e^{-\frac{\sqrt{2}|x|}{\sigma_x}}}{\sqrt{2}\sigma_x}$$

and Equation 27 reduces to:

$$\hat{x}(y) \;=\; argmax_x \left(-\frac{(y-x)^2}{2\sigma_n^2} - \frac{\sqrt{2}\,|x|}{\sigma_x} \right)$$

which is computed analytically in Appendix B to obtain:

$$\hat{x}(y) = \mathrm{SoftTh}(y, \frac{\sqrt{2}\sigma_n^2}{\sigma_x}) \tag{28}$$

with $\mathrm{SoftTh}(y, T)$ as in Equation 6.

B Finding the minimum of $E = (y - x/a)^2 + \lambda|x|$

We are looking for the minimum of:

$$E = (y - x/a)^2 + \lambda\,|x|$$

Taking the derivative of E with respect to x we obtain:

$$\frac{\partial E}{\partial x} = -\frac{2}{a}\left(y - \frac{x}{a}\right) + \lambda \mathrm{sign}\,(x)$$

the second derivative is:

$$\frac{\partial^2 E}{\partial x^2} = \frac{2}{a^2} > 0$$

regardless of x, and hence E is a convex function.

Let us assume that y is positive. Taking x to be zero we obtain:

$$x_0 = 0, \qquad E_0 = E(x_0) = y^2$$

where $E(x < 0) > E_0$. Taking

$$x_{ay} = ay, \qquad E_{ay} = E(x_{ay}) = \lambda a\,|y|\,,$$

and $E(x > ay) > E_{ay}$, and hence we have $0 \le x_m \le ay$ where x_m is the value of x for which E obtains its minimal value. Requiring $\frac{\partial E}{\partial x} = 0$ we obtain:

$$x_m = ay - \frac{a^2\lambda}{2}\mathrm{sign}\,(y)$$

and:

$$E_m = \frac{a^2\lambda^2}{4} + a\lambda\left||y| - \frac{a\lambda}{2}\right|$$

It is easy to see that:

$$\begin{array}{lclcrcl} E_{ay} & < & E_0 & \text{if and only if} & a\lambda & < & |y| \\ E_m & < & E_0 & \text{if and only if} & \frac{1}{2}a\lambda & < & |y| \\ E_m & < & E_{ay} & \text{if and only if} & \frac{3}{8}a\lambda & < & |y| \end{array}$$

and hence the x for which the minimal value of E is obtained is:

$$\tilde{x} = \begin{cases} a\left(y - \frac{1}{2}a\lambda \,\text{sign}\,(y)\right) & \text{for} \quad |y| > \frac{1}{2}a\lambda \\ 0 & \text{for} \quad |y| \leq \frac{1}{2}a\lambda \end{cases} \tag{29}$$

For $\lambda = 2\frac{\sqrt{2}\sigma_n^2}{\sigma_x}$ and $a = 1$, Equation 28 is obtained.

C Soft-threshold as derived using Besov Norms.

This appendix summarize the derivation of the soft threshold algorithm in Besov spaces. For more details refer to [20, 21, 22, 23, 24, 7].

C.1 The definition of Besov norm

Let us intemperate an image as a function f defined on the unit square $I = [0,1]^2$. Let us limit the discission to functions f in $L^p(I)$, meaning functions with a finite L^p norm:

$$||f||_{L^p(I)} = \left(\int_I (f(x))^p dx\right)^{1/p} < \infty. \tag{30}$$

The definition of the Besov norm is as follows:
let us define:

$$\Delta_h^0 f(x) := f(x)$$

and

$$\Delta_h^k f(x) := \Delta_h^{k-1} f(x+h) - \Delta_h^{k-1} f(x)$$

for positive integer k, which sums up to:

$$\Delta_h^k f(x) = \sum_{l=0}^{k} \left(\begin{array}{c} k \\ l \end{array}\right) (-1)^l f(x + lh).$$

$\Delta_h^k (f(x))$ is a functional of the signal f which describes differences in f for scale h. Let us define a modulus of smoothness $\omega_r (f,t)_p$ as the sup of the p-norm of the difference functional, over all scales smaller than, or equal to, t:

$$\omega_r (f,t)_p := \begin{array}{c} sup \\ |h| \leq t \end{array} \left(\int_{I_{rh}} |\Delta_h^r f(x)|^p \, dx \right)^{1/p}$$

where $I_{rh} := \{x \in I | x + rh \in I\}$. The modulus of smoothness estimates maximal fluctuation: If there is a scale, smaller than t, in which the signal fluctuates, the modulus of smoothness will be at least as big as the L^p norm of the difference functional of the signal in that scale. Weighting the scales:

$$|f|_{B_q^{\alpha,r}(L^p(I))} \quad := \quad \left(\int_0^\infty \left[t^{-\alpha} \omega_r (f,t)_p \right]^q \frac{dt}{t} \right)^{1/q}$$

or, for $q \to \infty$:

$$|f|_{B_\infty^{\alpha,r}(L^p(I))} \quad := \quad \sup_{t>0} \left[t^{-\alpha} \omega_r (f,t)_p \right]$$

we obtain the besov norm, which is the sum of the L^p norm of the signal and the weighted L^q norm of the modulus of smoothness:

$$||f||_{B_q^{\alpha,r}(L^p(I))} := ||f||_{L^p(I)} + |f|_{B_q^{\alpha,r}(L^p(I))}$$

A function f is in $B_q^{\alpha,r} (L^p(I))$ only if $||f||_{B_q^{\alpha,r}(L^p(I))} < \infty$, and hence $B_q^{\alpha,r} (L^p(I))$ is contained in $L^p(I)$.

For each $r', r > \alpha$, $B_q^{\alpha,r'} (L^p(I)) = B_q^{\alpha,r} (L^p(I))$ (see [23]) and hence one defines:

$$||f||_{B_q^{\alpha}(L^p(I))} \equiv ||f||_{B_q^{\alpha,r}(L^p(I))} \text{ for any } r > \alpha.$$

remarks:

1. if $q < 1$ or $p < 1$, the above definition is a quasi-norm, and not a real norm, since it does not satisfy the triangle inequality. There exists, however, a constant C such that for all $f, g \in B_q^{\alpha} (L^p(I))$,

$$||f+g||_{B_q^{\alpha}(L^p(I))} \leq C \left(||f||_{B_q^{\alpha}(L^p(I))} + ||g||_{B_q^{\alpha}(L^p(I))} \right)$$

2. If $\alpha_1 < \alpha_2$, $B_q^{\alpha_1} (L^p(I)) \subset B_q^{\alpha_2} (L^p(I))$.

C.1.1 Example: A step function

Working with the definition of the Besov norm is not recommended, and the equivalent wavelet-based definition, presented the next section, is preferable. However, in order to gain some intuition, we will explore the example of using the definition of the Besov norm on a step function.

Let us look at the image f, defined on $I := [0,1]^2$:

$$f(x_1,x_2) = \begin{cases} 0 & \text{for} \quad x_1 < 1/2 \\ 1 & \text{for} \quad x_1 \geq 1/2 \end{cases},$$

or at the one dimensional function describing the same discontinuity,

$$f(x) = \begin{cases} 0 & \text{for} \quad x < 1/2 \\ 1 & \text{for} \quad x \geq 1/2 \end{cases}$$

defined on $I_1 := [0,1]$. For each order r, $(1+1)^r > \begin{pmatrix} r \\ k \end{pmatrix}$, where $0 \leq k \leq r$, $sup|f(x)| = 1$, and hence the difference functional is bounded by

$$\Delta_h^r f(x) = \sum_{k=0}^{r} (-1)^{r-k} \begin{pmatrix} r \\ k \end{pmatrix} f(x+kh) \leq 2^r sup|f(x)| \leq 2^r.$$

Far from the edge, when $|x - 1/2| > r|h|$, The difference function is zero: $|\Delta_h^r f(x)| = 0$.

Near the edge, for $|x - 1/2| < rt$, for $h = t$, $|\Delta_h^r f(x)| \geq 1$, but bounded by 2^r.

Close to the edge, for small t,

$$\begin{aligned} \omega_r(f,t)_p &= \sup_{|h|<t} \left(\int_{I_{rh}} |\Delta_h^r f(x)|^p \, dx \right)^{1/p} \\ &\approx \tilde{C}(r) \left(\int_{-rt+1/2}^{rt+1/2} 1 \, dx \right)^{1/p} \\ &\approx C(r)\, t^{1/p}. \end{aligned}$$

Far from the edge, for $rt \geq r|h| > max(|x - \frac{1}{2}|) = \frac{1}{2}$ the one-dimensional image is constant or not defined, and hence for $t > \frac{1}{2r}$ the modulus of smoothness is zero or constant. Substituting the modulus of smoothness into the norm definition for both cases we obtain:

$$\begin{aligned} |f|_{B_q^{\alpha,r}(L^p(I))} &:= \left(\int_0^\infty \left[t^{-\alpha} \omega_r(f,t)_p \right]^q \frac{dt}{t} \right)^{1/q} \\ &= \left(\int_0^{1/2r} \left[t^{-\alpha} C t^{1/p} \right]^q \frac{dt}{t} \right)^{1/q} + \left(\int_{1/2r}^\infty \left[t^{-\alpha} C_0 \right]^q \frac{dt}{t} \right)^{1/q} \end{aligned}$$

While the second integral, away from the edge, is always finite as α and q are always positive. The first integral is finite only for $-\alpha q + q/p - 1 + 1 > 0$, and hence the image is contained only in besov spaces with $\alpha < 1/p$.

C.1.2 Orthogonal wavelet basis

A wavelet ψ is a function of zero average:

$$\int_{-\infty}^{\infty} \psi(t)dt = 0$$

ψ is usually non-zero on a bounded neighborhood only, which means it has both positive and negative values on a small neighborhood, and hence convolving ψ with a signal gives spatial-difference-related information at each location. Setting:

$$\psi_{j,k}(x) = 2^{k/2}\psi(2^k x - j)$$

To be the scaled,by $2^{k/2}$ and translated (by $j/2^k$) of the original function ψ, one obtains an orthogonal basis $\Psi = \{\psi_{j,k}\}_{j,k\in Z}$ on $L_2(\Re)$, so that defining the wavelet coefficients as:

$$c_{j,k} := \int_{\Re} f(x)\psi_{j,k}(x)dx$$

the function $f(x)$ is presented by:

$$f(x) = \sum_{j,k\in Z} c_{j,k}\psi_{j,k}$$

and:

$$||f||^p_{L_p(\Re)} = \sum_{j,k\in Z} c^p_{j,k}$$

C.1.3 Wavelet dependent Besov Norm equivalent

Mayer [22] showed, that as long as $p \in [1,\infty]$ and $f \in L^n[0,1]$, the besov norm $B^\alpha_q(L^p(I))$ is equivalent to

$$|f|_{B^{\alpha,r}_q(L^p(I))} \asymp \left(\sum_{k=1}^{\infty}\left(\sum_{j,\psi} 2^{k\alpha p}\, 2^{nk(\frac{p}{2}-1)}\, |c_{jk\psi}|^p\right)^{q/p}\right)^{1/q} \tag{31}$$

where n is the dimention of the space (for an image $I \subset \Re^2$, and hence $n = 2$), and $A(f) \asymp B(f)$ means that there exists c_1, c_2 such that:

$$c_1 A(f) \leq B(f) \leq c_2 A(f)$$

for all f, and hence the term 31 converges if and only if f has a finite Besov Norm.

C.2 Denoising induced by wavelet dependent Besov Norm equivalent

Let f be an image defined on $I = [0,1]^2$. One would like to approximate f by a smoother function, g. Let us define the error of the approximation in an X norm, $||f - g||_X$, e.g. L_1 or L_2, and the smoothness $||g||_Y$, using smoothness norm Y, such as Bounded Variation or a Bosov norm $B_p^\alpha\left(L^p(I)\right)$ (Notice the choice $q = p$).

We are looking for:

$$\inf_{g \in Y}\left(||f-g||_X + \lambda\, ||g||_Y\right) \tag{32}$$

We can write the expansion of both functions f and g in the chosen basis:

$$f = \sum_{0 \le k} \sum_{j \in Z_k^2} \sum_{\psi \in \Psi_k} \quad c_{j,k,\psi}\psi_{jk} \tag{33}$$

$$\text{where} \qquad c_{j,k,\psi} := \int_I f(x)\psi_{j,k}(x)dx$$

$$g = \sum_{0 \le k} \sum_{j \in Z_k^2} \sum_{\psi \in \Psi_k} \quad d_{j,k,\psi}\psi_{jk} \tag{34}$$

$$\text{where} \qquad d_{j,k,\psi} := \int_I g(x)\psi_{j,k}(x)dx$$

expressing the X and Y norms in terms of the wavelet coefficients $c_{j,k,\psi}$ and $d_{j,k,\psi}$. Substituting Equation 31 (for $p = q$),

$$|f|^p_{B_q^{\alpha,r}(L^p(I))} \quad = \quad \sum_1^\infty \sum_{j,\psi} 2^{k\alpha p}\, 2^{k(p-2)}\, |c_{jk\psi}|^p\,,$$

Equation 32 becomes:

$$\sum_{0 \le k} \sum_{j \in Z_k^2} \sum_{\psi \in \Psi_k} \left\{ \left(c_{j,k,\psi} - d_{j,k,\psi}\right)^2 + \lambda 2^{k\alpha p}\, 2^{k(p-2)}\, |d_{jk\psi}|^p \right\} \tag{35}$$

from which we can derive the coefficients $d_{j,k,\psi}$ of the smoothed function g as a function of the coefficients of the given function $c_{j,k,\psi}$, as shown in the following examples.

C.2.1 In Sobolev space: $Y = W^\alpha\left(L_2(I)\right)$

The Besov space with $p = q = 2$ is known as the Sobolev space,

$$W^2\left(L_2(I)\right) := B_2^\alpha\left(L^2(I)\right).$$

For that space, the wavelet dependent Besov norm equivalent 31 becomes:

$$||g||_{W^2(L_2(I))} = \left(\sum_{k=0}^{\infty}\left(\sum_{j,\psi} 2^{2k\alpha}\left|d_{jk\psi}\right|^2\right)\right)^{1/2}.$$

One seeks for:

$$\inf_{g\in Y}\left(||f-g||_{L_2(I)} + \lambda ||g||^2_{W^2(L_2(I))}\right)$$

which, after expressing the $L_2(I)$ and $W^2\left(L_2(I)\right)$ norms in the chosen wavelet basis becomes:

$$\sum_{0\leq k}\sum_{j\in Z_k^2}\sum_{\psi\in\Psi_k}\left\{\left(c_{j,k,\psi} - d_{j,k,\psi}\right)^2 + \lambda 2^{2k\alpha}\left|d_{j,k,\psi}\right|^2\right\}$$

The wavelet basis being a basis, one can find the infimum by looking at each dimension separately, and obtain:

$$d_{j,k,\psi} = \frac{c_{j,k,\psi}}{\lambda 2^{2k\alpha}+1} \tag{36}$$

which is a linear smoothing, illustrated by a black line in Figure 11.

C.2.2 $Y = B_1^1\left(L_1(I)\right)$

Let us concentrate on Besov spaces $B_p^\alpha\left(L_p(I)\right)$ embedded in $L_2(I)$. We are looking for the largest space among them, the space which contains as many sample functions as possible (i.e. has *minimal smoothness*). As shown by De Vore and Popov [25], those minimal smoothness spaces obey $1/p = \alpha/2 + 1/2$. Among those spaces, the only space with positive integer values of p, and a positive value of α, is $B_1^1\left(L_1(I)\right)$. $B_1^1\left(L_1(I)\right)$ is included in the Bounded Variation space BV(I):

$$B_1^1\left(L_1(I)\right) \subset BV(I) \subset B_\infty^1\left(L_1(I)\right).$$

In $Y = B_1^1\left(L_1(I)\right)$ the wavelet dependent Besov norm equivalent (Equation 31) becomes:

$$||g||_{B_1^1(L_1(I))} = \left(\sum_{1}^{\infty}\left(\sum_{j,\psi}\left|d_{jk\psi}\right|\right)\right). \tag{37}$$

One seeks for:

$$\inf_{g\in Y}\left(||f-g||_{L_2(I)} + \lambda ||g||_{B_1^1(L_1(I))}\right)$$

which, after expressing the $L_2(I)$ and $B_1^1\left(L_1(I)\right)$ norms in the chosen wavelet basis becomes:

$$\sum_{0\leq k}\sum_{j\in Z_k^2}\sum_{\psi\in\Psi_k}\left\{\left(c_{j,k,\psi} - d_{j,k,\psi}\right)^2 + \lambda\left|d_{j,k,\psi}\right|\right\}$$

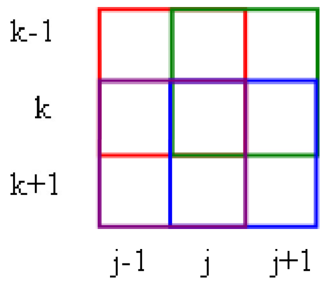


Figure 12: Illustration of the blocks that contains the pixel (i,k).

The wavelet basis being a basis, one can find the infimum by looking at each dimension separately (appendix B, with $a = 1$), and obtain:

$$d_{j,k,\psi} = sign(c_{j,k,\psi}) \begin{cases} |c_{j,k,\psi}| - \lambda/2 & |c_{j,k,\psi}| > \lambda/2 \\ 0 & |c_{j,k,\psi}| \leq \lambda/2 \end{cases} \tag{38}$$

known as soft threshold or wavelet shrinkage by Donoho and Johnstone [2, 6]. The red line in Figure 11 shows this operation: coefficients that are lower than a threshold $\lambda/2$ are being erased, those which are bigger are being diminished.

D An example of the Bi-Selective Filter.

While the one dimensional bi-selective filter unites with the first generation robust filter, the two dimensional bi-selective filter is slightly more complicated, even in the simple case of $N = 2$, since the pre-LUT terms are a function of four pixels, instead of two in the first generation robust filter. This case is bought here as an example of applying the Bi-selective filter in transform domain. The calculation is done using the 2x2, block DCT transform, though other transforms (e.g. Haar) have the same form for $N = 2$. The per-pixel expression obtained in this calculation is long, but most of it's terms are re-used when calculating the neighboring pixels. As expressed in Equation 39, we calculate the two dimensional block DCT transform for all of the blocks that contain the pixel (j, k), as shown in Figure 12. The Robust LUT is applied on the non-DC Block DCT coefficients, which are back transformed to the image domain, and averaged:

$$\begin{aligned} Out(j,k) = \frac{1}{4} \quad & \left\{ T^{-1}\Phi\left(T \begin{bmatrix} In(j,k) & In(j,k+1) \\ In(j+1,k) & In(j+1,k+1) \end{bmatrix} T^t \right) T^{-t} \right\}_{(1,1)} + \\ & \left\{ T^{-1}\Phi\left(T \begin{bmatrix} In(j,k-1) & In(j,k) \\ In(j+1,k-1) & In(j+1,k) \end{bmatrix} T^t \right) T^{-t} \right\}_{(1,2)} + \end{aligned}$$

$$\left\{T^{-1}\Phi\left(T\begin{bmatrix} In(j-1,k) & In(j-1,k+1) \\ In(j,k) & In(j,k+1)\end{bmatrix}T^t\right)T^{-t}\right\}_{(2,1)} +$$
$$\left\{T^{-1}\Phi\left(T\begin{bmatrix} In(j-1,k-1) & In(j-1,k) \\ In(j,k-1) & In(j,k)\end{bmatrix}T^t\right)T^{-t}\right\}_{(2,2)} \tag{39}$$

substituting $T = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1\end{pmatrix}$, $\widetilde{\Phi}\left(x\right) = \Phi\left(\frac{1}{\sqrt{2}}x\right)$, applying the LUT only at the non-DC components, and remembering that $\widetilde{\Phi}\left(-x\right) = -\widetilde{\Phi}\left(x\right)$, we obtain:

$$\begin{aligned}
= \frac{1}{4\sqrt{2}}\Big\{ & \quad 4\cdot In(j,k) \\
+ & \quad 2\left(In(j,k+1) + In(j+1,k) + In(j,k-1) + In(j,k+1)\right) \\
+ & \quad In(j+1,k+1) + In(j+1,k-1) + In(j-1,k+1) + In(j-1,k-1) \\
- & \quad \widetilde{\Phi}\left(In(j+1,k) + In(j+1,k+1) - In(j,k+1) - In(j,k)\right) \\
- & \quad \widetilde{\Phi}\left(In(j+1,k) - In(j+1,k+1) + In(j,k+1) - In(j,k)\right) \\
- & \quad \widetilde{\Phi}\left(-In(j+1,k) + In(j+1,k+1) + In(j,k+1) - In(j,k)\right) \\
- & \quad \widetilde{\Phi}\left(In(j,k-1) + In(j+1,k-1) - In(j+1,k) - In(j,k)\right) \\
- & \quad \widetilde{\Phi}\left(In(j,k-1) - In(j+1,k-1) + In(j+1,k) - In(j,k)\right) \\
- & \quad \widetilde{\Phi}\left(-In(j,k-1) + In(j+1,k-1) + In(j+1,k) - In(j,k)\right) \\
- & \quad \widetilde{\Phi}\left(In(j-1,k) + In(j-1,k+1) - In(j,k+1) - In(j,k)\right) \\
- & \quad \widetilde{\Phi}\left(In(j-1,k) - In(j-1,k+1) + In(j,k+1) - In(j,k)\right) \\
- & \quad \widetilde{\Phi}\left(-In(j-1,k) + In(j-1,k+1) + In(j,k+1) - In(j,k)\right) \\
- & \quad \widetilde{\Phi}\left(In(j-1,k-1) + In(j-1,k) - In(j,k-1) - In(j,k)\right) \\
- & \quad \widetilde{\Phi}\left(In(j-1,k-1) - In(j-1,k) + In(j,k-1) - In(j,k)\right) \\
- & \quad \widetilde{\Phi}\left(-In(j-1,k-1) + In(j-1,k) + In(j,k-1) - In(j,k)\right) \quad \Big\}
\end{aligned}$$