TAMPERE UNIVERSITY OF TECHNOLOGY

FEDOR KUDASOV

CONTENT-BASED IMAGE FILTERING

Master's thesis

Examiners: Professor Karen Egiazarian
Examiners and topic approved by the Faculty Council of the Faculty of Computing and Electrical Engineering on 07 December 2012.

PREPHASE

This work is a master thesis for the Master of Science programme in the Tampere University of Technology. This is a result of two years of my study in Tampere. I express my gratitude to everyone who provided me an ability to improve my education: from the teaching staff of Saint-Petersburg State University where I got my Bachelor's to the staff of TUT who accepted my application in 2011.

I would like to thank my supervisor professor Karen Egiazarian who believed in my abilities, provided me funding and created comfortable atmosphere for performing the research work. Also, I would like to thank him for his incredible patience and for his permanent support. I appreciate help of Aram Danielyan, Eugeniy Belyaev and everyone in the signal processing department group who provided me very important notes concerning my thesis work.

Moreover, I would like to thank my family. My mother and father for supporting me during the "hard times". My grandfather for being interested in my research and for giving me useful advices. Finally, I especially appreciate the contribution of my girlfriend, Katya, who gave me priceless comments on the thesis text and who helped me to find the strength to finish it.

18.08.2013
Fedor Kudasov

ABSTRACT

TAMPERE UNIVERSITY OF TECHNOLOGY
Master's Degree Programme in Information Technology
FEDOR KUDASOV : Content-based image filtering
Master of Science Thesis, 50 pages, 6 Appendix pages
August 2013
Major: Signal Processing
Examiner: Professor Karen Egiazarian
Keywords: Denoising, classification, DCT, image filtering, Gaussian noise

This paper presents an adaptive content-based image denoising technique. This technique uses image area classification for two purposes: perform more precise filtering and decrease computation complexity compared to modern filters of the same quality performance.

Overview of several top image filtering techniques was made. Spatial domain (LPA-ICI), transform domain (SW-DCT) and combined filters (SA-DCT and BM3D) were studied in order to understand basic principles of image denoising. Image area classification which gives reasonable division into classes with clearly distinguishable properties for image filtering was observed. We have chosen block-wise classification that maps each block to "Texture", "Smooth" and "Edge" classes. Performance of discussed filters on image area classes was shown. Adaptive free parameters choise for filtering quality improvement was analysed. It was shown that for some classes best parameters set differs from the best parameter set for the entire image.

Methods to improve denoising algorithms speed which we were using in our adaptive solution were proposed. The most suitable algorithms with appropriate parameters set for each image area class were chosen. Modified classification algorithm applied to noisy images was developed. Whereupon, a modified BM3D-based adaptive denoising algorithm was proposed. Finally, multiple tests were performed and verification of speed and quality performances improvement compared to a baseline BM3D algorithm was obtained.

CONTENTS

TERMS AND ABBREVIATIONS

BM3D	(Block Matching 3D) a recent state-of-the-art denoising algorithm which is based on grouping of similar 2D blocks into 3D structure and performing linear decorrelating transforms
DCT	(Discrete Cosine Transform) a linear transform on real number which is widely used for signal decorrelation
DFT	(Discrete Fourier Transform) a linear transform on complex number which is widely used for signal decorrelation
FFT	(Fast Fourier Transform) a fast implementation of DFT
KLT	(Karhunen-Loeve Transform) a linear transform which optimally compacts the signal energy, is calculated individually for a particular signal
LPA-ICI	(Local polynomial approximation - intersection of confidence intervals) an image denoising algorithms based on weighted averaging of local pixel neighborhood
MSE	(Mean Square Error) a measure of difference between two signals is calculated as an average of squared point-wise signal difference
PSNR	(Peak Signal to Noise Ratio) a measure of image quality is calculated as $10 \cdot \log_{10} \frac{255^2}{MSE(I,I_n)}$, where I is an original image and I_n is degraded image
SA-DCT	(Shape adaptive DCT) an image denoising algorithm which is based on 2D linear transforms applied to complex shape image areas
SW-DCT	(Sliding window DCT) an image denoising algorithm which is based on 2D block-wise linear transforms

LIST OF TABLES

LIST OF FIGURES

1. INTRODUCTION

Image processing methods and, in particular, image filtering, becomes very important nowadays. In the last decades we can see a rapid development of digital technology, widespread use of gadgets with photo and video cameras, such as smartphones, increase in computational power of embedded devices. Whereupon, we get an ability to provide to the end-user such services which in past demanded stationary computation complexes of highly specialized equipment but now can be performed real-time, e. g. high definition video recording or on-line video filtering.

With the growth of the computational abilities there are also qualitative improvements in filtering algorithms which, unfortunately, cost more and more computational resources and storage memory. Additionally, even negligible increase in PSNR might dramatically slowdown the filtering process. Note also that often theoretical studies differ from real implementation: even though in theory we may achieve PSNR improvement, practically it either might not provide us visual enhancement or an improvement might be insignificant compared to increase in computational complexity. Hence, in order to be able to use recent achievements in image filtering one should find a way to simplify algorithms allowing an insignificant drop in quality but keeping computational complexity low which would be appropriate for practical implementations.

Image filtering in digital cameras is a very broad concept. Generally, it means an image enhancement by a proper tuning of the following parameters: white balance, gamma correction, brightness level etc. Hereinafter in this work by image filtering we understand image denoising – suppression of additive zero mean white Gaussian noise. This mathematical model of noise has several important attributes. It has good mathematical properties (e. g. flat frequency spectrum). Poisson distribution can be converted to a normal one. Poisson noise in turn along with Gaussian noise are the main components of noise in image sensors (photo and video cameras) [1, Chapter 4.6], [2]. Additionally, white Gaussian noise describes some natural noises (heat) behaviour. Therefore, consideration of only that kind of noise allows us to solve highly demanded problem and use extensive achievements from mathematical tools since normal distribution is a well-studied mathematical model.

Even though recent developments in image filtering yield great results in performance, there is still some room for improvements. Complex algorithms, which usually have several free parameters are applied to the entire image with the same settings or even the same settings are used for every image to be processed. One way of algorithm improvement, which is applied in [3], is to use an adaptive window size and therefore perform different filtering for areas with different pixel similarity. More complex approach utilizes image patches along with adaptive window size to calculate weighted sum of estimates with adaptive weights [4].

Another way to improve existing denoising methods is to use image segmentation/classification and apply particular filter with particular settings according to the classification on each image part individually. Separate handling of areas with different content and, therefore, different properties (e. g. smooth and texture) can not only improve filtering, but also significantly decrease computational complexity of filtering since some areas of image may not require advanced techniques for its denoising. The main goals of this work are as follows:

1. Analyze basic principles and features of existing denoising algorithms.
2. Embed classification approach in the filtering process.
3. Use classification and features of different filters for denoising quality improvements.

Along with the mentioned objectives we aim to apply if possible some methods to decrease algorithms computational complexity without sensible drop in denoising quality.

2. THEORETICAL BACKGROUND

In order to understand in what way we can utilize difference in image content in filtering, we should study basics of modern filtering techniques and recent state-of-the-art approaches in image filtering. Since that we will start our overview with spatial domain filters.

2.1 Spatial domain filters

According to [5, Chapter 3] spatial domain is related to the image plane itself which means that filtering is based on direct manipulation with image pixels. Filtering in spatial domain can be represented as:

$$g(x, y) = T[f(x, y)]$$

where $f(x, y)$ is the input image, $g(x, y)$ is the filtered image and T is an operator on f which processes pixels in some neighborhood of (x, y). Neighborhood is traditionally a square or rectangular shape subimage area centered at (x, y). This center of subimage is moved from a pixel to pixel and operator T is applied to the neighborhood at each location to yield the output g (filtered image).

One can notice two key elements in spatial domain filtering: selection of the neighborhood and selection of the operator. Generally, knowledge of an image degradation type facilitates the selection of the proper operator. For example, median filter is used in suppressing impulsive noise, such as "salt & pepper" [5, Chapter 3]. White Gaussian noise will be considered as a main degradation source in this work. It can be suppressed by the use of the local (weighted) averaging filter. The choice of an optimal neighborhood (window) in this case becomes one of the primary objectives.

From the mathematical point of view in filtering process there is a trade-off between bias and variance [6, Chapter 2.3.1]: by taking large window size we are greatly suppressing the noise, but loosing most of the image details; while using small window for filtering we preserve details, but keep noise less suppressed. Minimum window size (only the original pixel value) guarantees no bias but keeps variance at its initial level, while the window of all the image size will yield extremely small variance and huge bias that blurs all the image content. Thus, as a window size grows the variance decreases and a bias component increases. Therefore, as the point-wise MSE that we want to decrease can be represented as a sum of two terms depending on bias and variance, selection of window size depends on variability of pixels taken from undistorted image in the window [12]. As long as pixels are close in value, we can increase a window size to decrease a variance without growth of bias and when pixels start to differ much we should choose small window not to loose details because of the bias.

The importance of right bias/variance balance determination can be observed in the following example. In Figures 2.1(a) and 2.1(b) two parts of “Barbara” image with a smooth content and with a texture are shown. These subimages are degraded with a white Gaussian noise having standard deviation of $\sigma = 30$ (see Figures 2.1(c), 2.1(d)). They are filtered with averaging filters with different size and form of windows. For both “Smooth” and “Texture” images square form of the windows (Figure 2.2, usual choice in image processing) was used, additionally, for the texture image we have used windows of the form of diagonal segments (Figure 2.3). This choice is motivated by the fact that the texture image pattern has diagonal stripes, thus it is interesting to study how the form of the window resembling image pattern improves the filtering. Figure 2.4 demonstrates the dependence between window sizes and PSNR for each of the cases described above. From this naive observation of filtering results we can see that:

1. With the averaging filter we can obtain higher PSNR improvement with the “Smooth” image than that with the “Texture” image.

2. “Optimal” size of the window for a “Smooth” image is much larger than that for a “Texture” one.

3. The better form of a window resembles the texture pattern the better filtering results can be obtained.

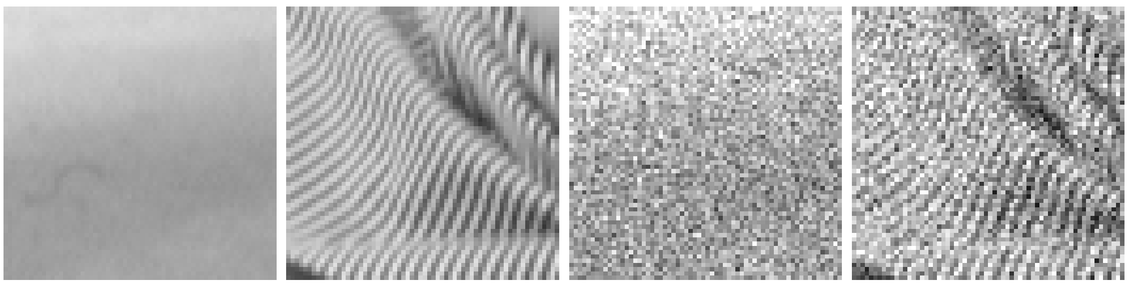

(a) “Smooth” (b) “Texture” (c) “Noisy “Smooth” (d) “Noisy “Texture”

Figure 2.1: Test images

Particular examples of the filtered images for 3×3 and 9×9 windows and for diagonal windows of length 3 and 9 for “Texture” image (see Figure 2.5) are shown

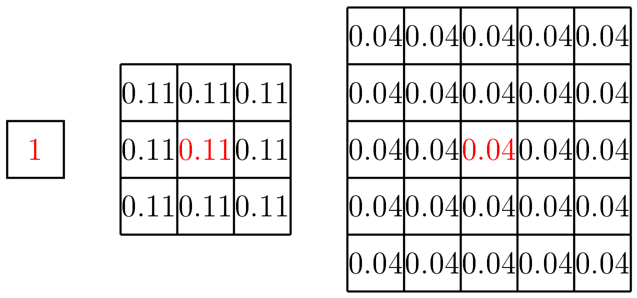


Figure 2.2: Square averaging windows.

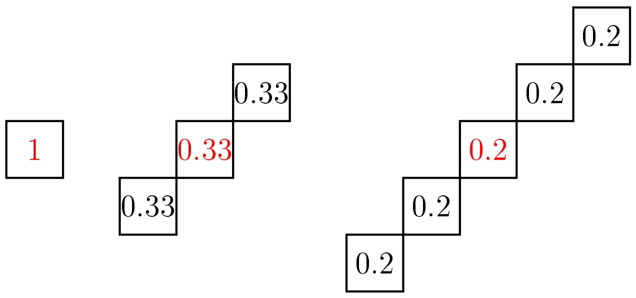


Figure 2.3: Diagonal averaging windows.

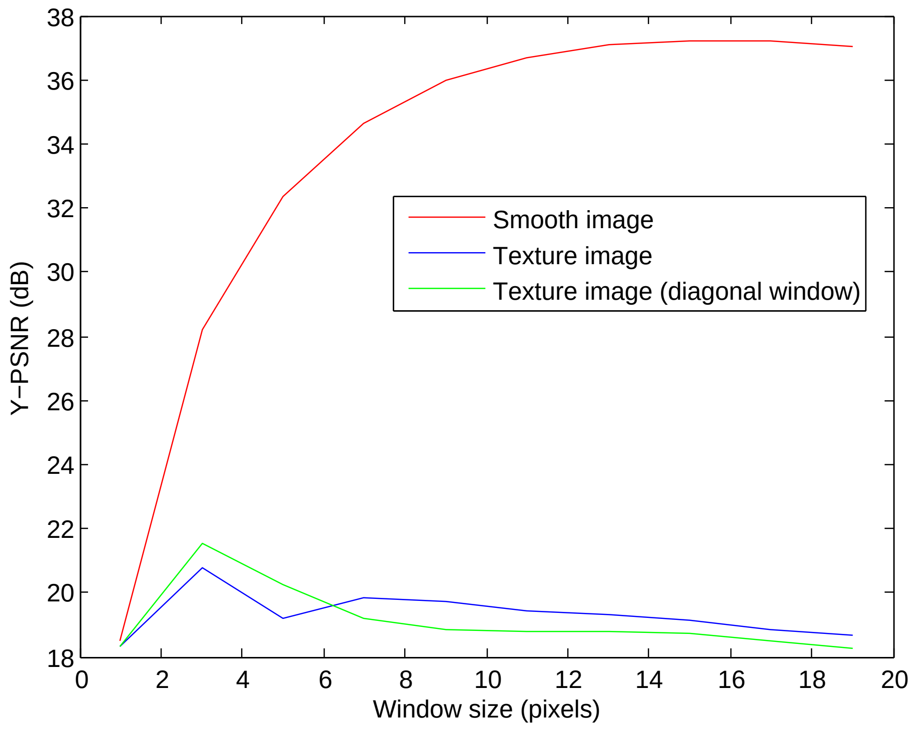
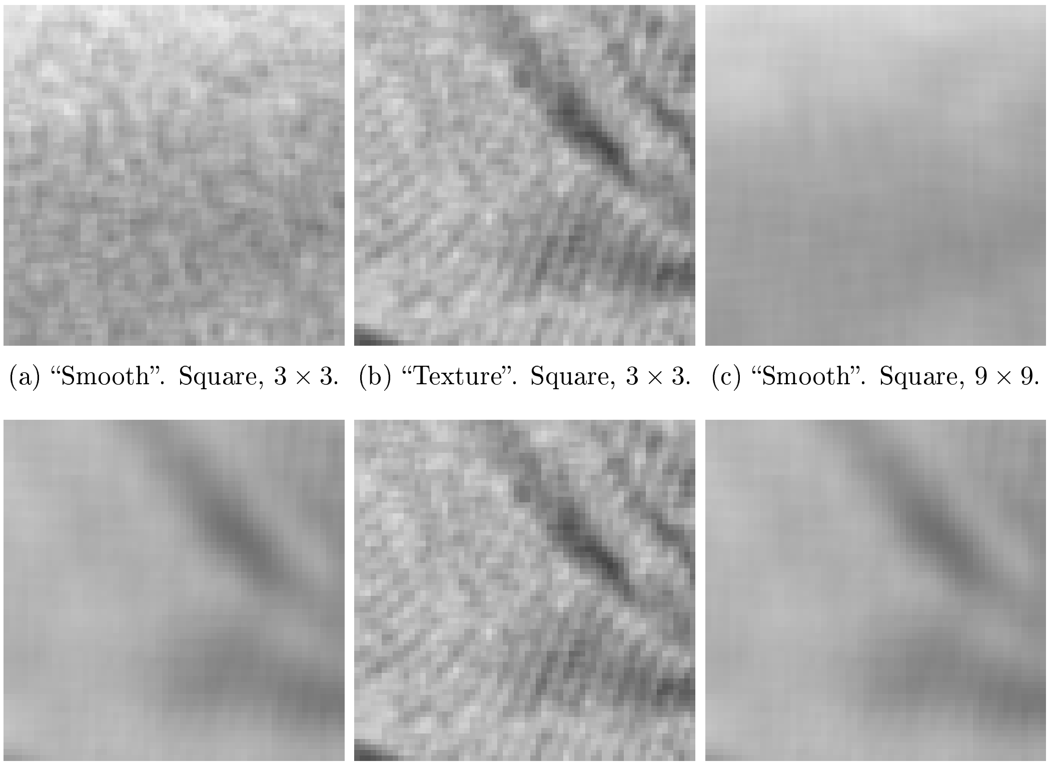


Figure 2.4: PSNR for smooth and texture images filtered with different square windows.

in order to demonstrate such a phenomena. Noisy “Smooth” image after filtering becomes closer to the undistorted one with the growth of the windows size, while a “Texture” image looses most details (we can see that only the fold of the tissue is left but the the pattern is generally lost).


(a) “Smooth”. Square, 3×3. (b) “Texture”. Square, 3×3. (c) “Smooth”. Square, 9×9.

(d) “Texture”. Square, 9×9. (e) “Texture”. Diagonal, 3×3. (f) “Texture”. Diagonal, 9×9.

Figure 2.5: Images filtered with different window shapes and sizes.

As it was mentioned above, an important task is the right choice in bias/variance trade-off. Since image pattern is usually not uniform this choice should be done individually for every image location. As it was shown, a much noise suppression with higher PSNR can be achieved in “Smooth” regions than in the area of edges. Hence, to get better filtering one should modify a window for every pixel in a way that averaging (weighted averaging) is performed only among “similar” pixels. The recently developed technique called Anisotropic LPA-ICI (local polynomial approximation – intersection of confidence intervals) is based on this idea [6]. Consideration of this approach is the first step in our study of determination of proper filtering parameter values based on the image content.

2.1.1 Anisotropic LPA-ICI

Let $X \subset \mathbb{Z}^2$ be an image domain, $y(x)$ $(x \in X)$ is an undistorted signal, $z(x) = y(x) + \sigma \cdot \eta(x)$ is a noisy observation of the signal $y(x)$ $(\eta(x) \sim \mathcal{N}(0,1))$. The goal of denoising is to determine the estimate $\hat{y}(x)$ of signal $y(x)$ which is as close to the undistorted signal as possible (e. g. in MSE sense). We mentioned earlier that this problem can be solved by using averaging filter. Additionally, we have shown that filtering using windows which comprise only similar pixels from the pixel neighborhood yield better estimate of the origin pixel compared to filtering using the general shape windows (squares). This can be explained by lower bias component of estimate obtained from window of similar pixels.

Anisotropic LPA for each pixel determines its own neighborhood of similar pixels which is used as a window for averaging [8]. Ideally, the optimal window should:

1) comprise as much as possible similar pixels to decrease variance of the estimate;

2) keep pixels similarity at a high level to decrease bias of the estimate.

It is hard to reach both of these conditions and obtain an optimal window which comprises only desired pixels. Practically, optimal window is replaced by its approximation which has some shape restrictions (e. g. start-shaped)[14].

These restrictions allow to obtain approximation of optimal window as a union of windows of some particular shape. For example, neighborhood of the pixel can be divided into conical sectors (origin pixel – their apex) or as a set of rays (origin pixel – their initial point). The task of optimal window determination can be reduced to the separate determinations of the optimal windows in particular directions, union of which comprises approximation of the optimal window (see Figure 2.6). These windows can be characterized by their linear size or scale parameter [7]. Further simplification of this process can be achieved by approximation of the optimal scale parameter with one picked from a fixed set of scales. To determine which scale corresponds to the optimal window (for a particular direction) so-called ICI-rule was developed [13].

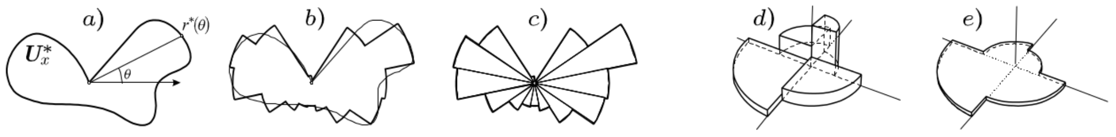


Figure 2.6: Approximation of optimal support and aggregation of directional estimates.

The idea of the ICI method is in following: for every scale (denoted by h) from a scale set with some probability p true value $y(x)$ is located in the vicinity of its estimate $\hat{y}_h(x)$. This vicinity depends on the scale and is called as a confidence interval (denoted by $\mathcal{D}$). It can be shown [6] that the radius of the confidence interval is $\sigma_{\hat{y}_h(x)} \cdot \Gamma$ ($\mathcal{D} = [\hat{y}_h(x) - \sigma_{\hat{y}_h(x)} \cdot \Gamma, \hat{y}_h(x) + \sigma_{\hat{y}_h(x)} \cdot \Gamma]$), where $\sigma_{\hat{y}_h(x)}$ is a standard deviation of an estimate $\hat{y}_h(x)$ and Γ is some parameter that depends on the probability p and a pixel similarity in the signal y.

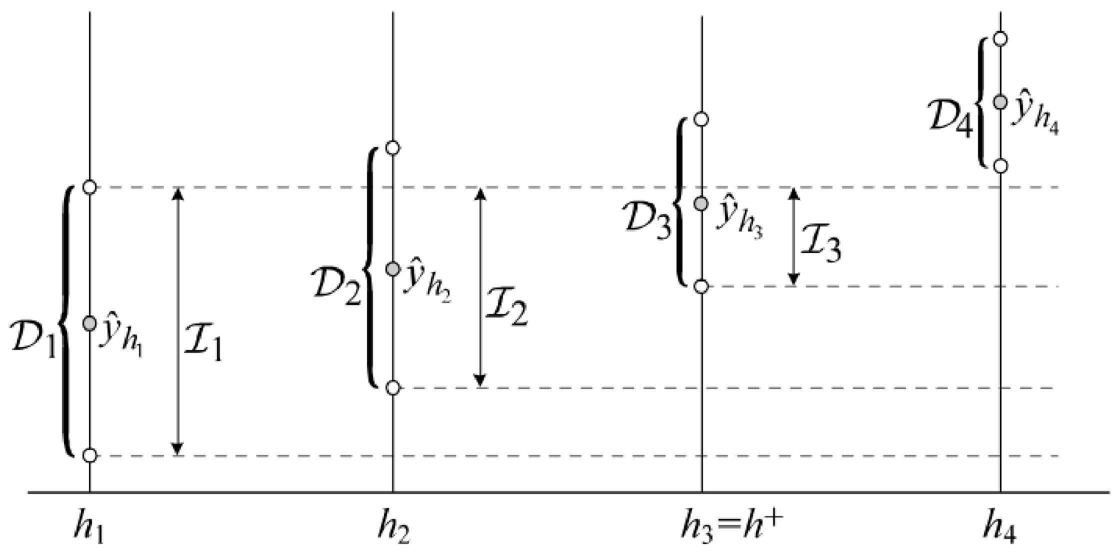


Figure 2.7: The ICI rule.

Confidence intervals may intersect. This intersection with some probability contains true value of $y(x)$. As scale h increases, corresponding window size increases and $\sigma_{\hat{y}_h(x)}$ decreases. Therefore, radius of the confidence interval decreases with the growth of h. ICI-rule states that as an optimal scale one should choose the largest scale for which intersection of all confidence intervals corresponding to the scales less or equal than this is not empty (see Figure 2.7). In other words, if $H = \{h_1, h_2, ...h_N\}$ is a set of the scales in the ascendant order, then an optimal scale is defined as:

$$h_{opt} = \max_i \{h_i : \bigcap_{j \leq i} \mathcal{D}_j \neq \varnothing\}. \tag{2.1}$$

Thus, the optimal scales for each direction are found, therefore, an optimal window for each direction can be found and from the union of these windows the approximation of the optimal window can be obtained.

Anisotropic LPA-ICI is an example of one of the most powerful spatial domain image filters. Even though it is an adaptive filter which means that the window for each pixel is selected individually and the selection depends on image content, it still has a free parameter Γ – the source of our further investigation and which impact we will see in the Chapter 3.

2.2 Transform domain filters

Transform (frequency) domain filters constitute another class of filters. The key feature in transform domain filtering is that before any enhancement image (or its part) pixels are transformed. The transform is done to perform a decorrelation of image pixels. First order Markov process with a correlation parameter close to 1 is a well established model for pixels in a small size image blocks. Since that neighboring image pixels usually have very high correlation, decorrelation helps us to separate very effectively a signal from noise. This correlation of pixels is local, hence a best processing is achieved by dividing image into subimages and performing filtering of each subimage independently. These subimages are usually called blocks (for simplicity they have a rectangular shape). Often block is of 8 or 16 pixel width since many transforms with the power of 2 length have fast implementation algorithms despite of those of arbitrary length (e.g. FFT instead of DFT).

Evidently, the best decorrelating transform matrix depends on the particular data and is known as a Karhunen-Loeve Transform (KLT) [15, Chapter 1]. The great drawback of KLT is that its matrix should be computed every time for every image block. The transform that is usually used instead of image-dependent KLT in image processing is a Discrete Cosine Transform (DCT). DCT transform matrix has:

1) a fixed structure;

2) very good decorrelation properties assuming first order Markov process model (approximates well KLT matrix);

3) fast computational algorithm [19].

The simplest but at the same time very powerful technique of transform domain filtering is a sliding window DCT filtering ([16], [17]).

2.2.1 Sliding window DCT filtering

Sliding window DCT (SW-DCT) performs denoising of the image by block-wise filtering. Blocks are of the fixed size and may overlap. If an image pixel belongs to several blocks, the final estimate of it is calculated from estimates of this pixel in all the blocks it belongs to. Usually, the more blocks contain the pixel, the better final estimate for this pixel we can obtain. This can be explained by more complete use of a pixel neighborhood.

Filtering of a block is performed as follows. Block (B) is transformed by 2D DCT and a block of DCT coefficients is obtained($B_T = DCT_{2D}(B)$). Usually, 2D-DCT is replaced by the consequent application of 1D DCT to block rows and to block columns, which is called 2D separable DCT transform.

As we stated above chosen degradation model can be represented as follows: $z(x) = y(x) + n(x)$, where x represents 2D coordinates and $n(x) \sim \mathcal{N}(0, \sigma^2)$. This model takes place for every image block since it takes place for the entire image. Hence, noise is distributed in transform coefficients uniformly (flat spectrum) while useful information is decorrelated and is mostly localized in the “low frequency” coefficients. Therefore, the transform coefficients which absolute value are below some level contain mainly noise. They are suppressed (set to zero) and thereby, some noise components are suppressed as well.

Thus, in the next step transform coefficients are processed with the hard thresholding operation ($\hat{B}_T = HT(B_T)$):

$$HT(a) = \begin{cases} a, & \text{if } |a| \geq T \\ 0, & \text{otherwise} \end{cases}, \text{where } T \text{ is a threshold.}$$

Threshold depends (linearly) on the noise standard deviation level ($\lambda \cdot \sigma$). The factor λ before σ characterizes how hard the noise is suppressed. It was shown that, in general, the best threshold level is around $2.7 \cdot \sigma$ [18].

Hard thresholding is a main part of denoising itself. After that, inverse 2D-DCT is applied to the block of filtered coefficients: $\hat{B} = IDCT_{2D}(\hat{B}_T)$. As an output the filtered image block in spatial domain is obtained.

Since the filter is applied block-wise, there are have two possible choices:

1. Divide an image into non-overlapping blocks and obtain filtered image as a set of filtered blocks.

2. Filter all possible blocks (or some of them) so that for each pixel there is at least one estimate (there is at least one block that contains that pixel).

The first choice provides poorly reconstructed image, but no aggregation is needed. Non-overlapping block processing in 2D DCT domain resembles the principle of JPEG [27] image compression. Second option demands some aggregation to combine estimates of the same pixel from different blocks into the final estimate. Two types of aggregation are usually used:

1. Simple averaging of all estimates.

2. Weighted averaging where a weight for an estimate from each block equals to the reciprocal of amount of non-zero DCT coefficients in this block after hard thresholding.

The second aggregation type corresponds to the maximum likelihood solution [22] (see Appendix A). This aggregation type gives a larger weight to the homogeneous blocks. Thus, e. g. blocks with edges have smaller weight than those around the edge. Therefore, the border effect of edge denoising can be reduced.

Worth to mention that instead of DCT transform one can use other transforms for sliding window filtering. For example, Haar or wavelet transforms may be used to save the computational time or perform better decorrelation if image pixels model is different.

There are several parameters in SW-DCT algorithm: block size, block shift (or more generally distribution of the blocks within the image) and thresholding coefficient λ. Each of them has different impact on filtering quality depending on image content. We will study this dependence in Chapters 3 and 4.

2.2.2 Empirical Wiener filter

Empirical Wiener filter is a modification that can be applied to SW-DCT filtering. In most of the cases it gives filtered image quality improvement. Empirical Wiener filter is based on the idea of Wiener filter.

Suppose, we have a signal $y(t)$ and an observation of this signal degraded with an additive noise $n(t)$:

$$z(t) = y(t) + n(t) \tag{2.2}$$

Assume, we would like to get a signal $g(t)$ which being convolved with the degraded signal $z(t)$ yields an estimate of the original signal $y(t)$:

$$\hat{y}(t) = g(t) \star z(t) \tag{2.3}$$

Wiener filter allows to determine optimal signal $g(t)$ that minimizes MSE of the estimate. Assuming that, $N(f)$ is the noise signal and $Y(f)$ is the original signal in the transform domain, we can get by the formula for $g(t)$ in transform domain as follows:

$$G(f) = \frac{1}{1 + \frac{|N(f)|^2}{|Y(f)|^2}}. \tag{2.4}$$

Therefore, the estimate in the transform domain is:

$$\hat{Y}(f) = G(f) \cdot Z(f). \tag{2.5}$$

Signal $g(t)$ is usually applied in the frequency domain since it has very simple form there, but it can be applied in the spatial domain too. An image can be considered as a locally stationary signal[20]. Hence, the signal $g(t)$ can be windowed and convolution $g(t) \star z(t)$ which provides an optimal estimate of signal $y(t)$ can be performed locally. Therefore, filtering can be performed locally in the frequency domain: determine $G(f)$ for each local part of a signal (block) and apply it in frequency domain.

Since we do not have an original signal ($y(t)$), empirical approach implies a use of some estimate of this signal. In application to SW-DCT as an original signal we use an estimate obtained from SW-DCT filtering [21].

Empirical Wiener filter can be considered as a more advanced way of thresholding. In hard thresholding coefficients are either totally suppressed (set to zero) or remain unchanged. In the case of Wiener filter every coefficient is multiplied by some factor ≤ 1 which performs suppression more accurately.

In the previous section of this thesis we were studying spatial and frequency domain filters. Each of these types of filters use different information from the image and even though frequency domain filters generally perform better, they do not take into account spatial similarity of image pixels coming from different blocks. Most of modern filters are actually combined (SA-DCT[22], BM3D[25]). In these algorithms filtering is performed by a modification of transform domain hard thresholding. But the data for filtering are collected from the spatial domain considering pixel similarity. This is an important step which helps to decorrelate data in a better way and, therefore, to filter it more qualitatively. In the following sections we will briefly review features of SA-DCT and BM3D.

2.3 SA-DCT filtering

Shape adaptive DCT (SA-DCT) filter for better denoising uses complex-shaped windows which comprise neighborhood of similar pixels [23, Chapter 3]. It lets to achieve stronger pixel decorrelation which in turn yields better noise suppression. SA-DCT can be described as a combination of elements from LPA-ICI and SW-DCT. Complex-shaped blocks are filtered with 2D DCT hard thresholding. It is performed as two 1D transforms row-wise and column-wise as in SW-DCT. Blocks for processing are determined pixel-wise via LPA-ICI algorithm.

In the considered implementation [22] block of similar pixels is determined for each pixel using several LPA-ICI runs for rays in (eight) different directions. All the ray start at the original pixel. Determination of best ray length is performed according to ICI rule, which uses Γ parameter as well as anisotropic LPA-ICI algorithm. End positions of rays form a (possibly concave) polygon (octagon). This polygon is considered as a region of similar pixels.

Compared to SW-DCT there are several modifications in the block denoising phase in SA-DCT. The length of the rows generally may be different. Since that, coefficients obtained after first 1D transform are aligned. Alignment is performed in a way that coefficients representing similar frequencies are processed together during the second 1D transform along the columns[24, Chapter 2].

Block in SA-DCT is not rectangular, therefore order of execution of 1D transforms affects the result. If the longest column is longer that the longest row, columns are processed first, otherwise rows are processed first.

Additionally, different length of rows and columns may cause so called “mean weighting defect" [33]. To prevent this phenomenon before separable 2D DCT an average value of the block is subtracted from all its pixels.

Since block in SA-DCT has a variable size, hard thresholding coefficient in SA-DCT depends on the amount of pixels in a block (denoted by N). In general λ is calculated by the following formula: $\lambda = \alpha\sqrt{2\log N + 1}$. Coefficient α represents the strength of the noise suppression. In the considered implementation of SA-DCT it has a constant value: $\alpha = 0.77$.

The entire process is illustrated in Figure 2.8([22]).

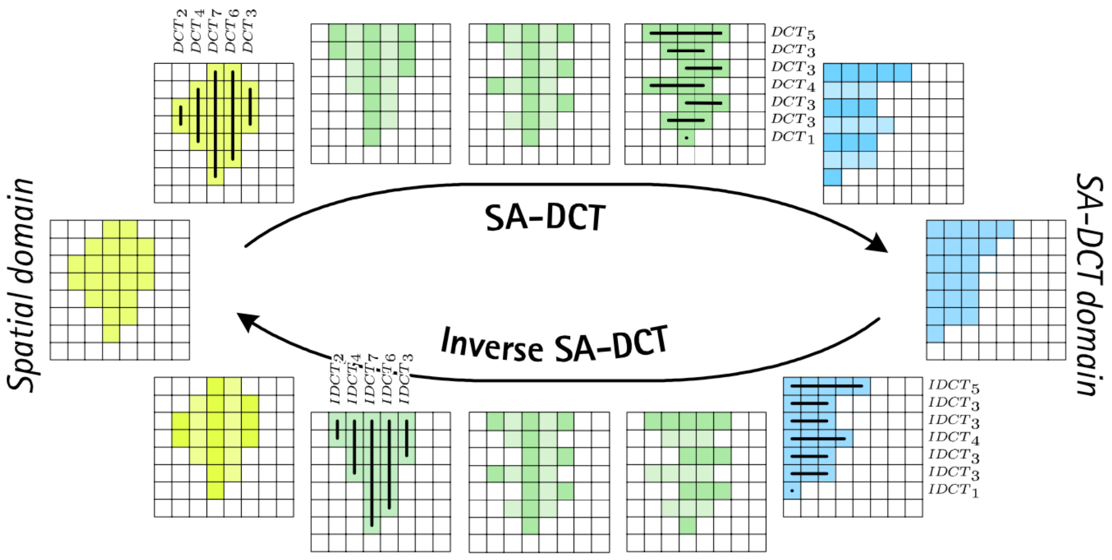

SA-DCT
Inverse SA-DCT
Spatial domain
SA-DCT domain
DCT2 DCT4 DCT7 DCT6 DCT3
DCT5 DCT3 DCT3 DCT4 DCT3 DCT3 DCT1
IDCT2 IDCT4 IDCT7 IDCT6 IDCT3
IDCT5 IDCT3 IDCT3 IDCT4 IDCT3 IDCT3 IDCT1

Figure 2.8: SA-DCT scheme.

There are two parameters which have to be set for filtering: Γ and α. As we studied in LPA-ICI, the first parameter is responsible for bias/variance trade-off, i. e. in this case it is a criterion of pixel similarity in the block. The second parameter is a thresholding coefficient very similar to the one used in SW-DCT but since in SA-DCT the shape and size of the block vary and correlation of pixels is generally higher than in SW-DCT, there are some feature of this coefficient. The influence of these parameters will be studied in the Chapter 3.

Finally, we will briefly overview BM3D algorithm as a state-of-the-art image denoising algorithm.

2.4 Block Matching 3D filtering

The basic idea of Block Matching 3D filtering (BM3D) is a collaborative filtering. Similarly to SW-DCT, BM3D is performed block-wise and it uses blocks of fixed size. But for better decorrelation similar 2D blocks are grouped up to 3D structures, which are thresholded in transform domain. After that each block from a filtered 3D structure is aggregated with some weight to form a final image estimate.

When filtering a block, BM3D searches for similar blocks in some neighborhood of the current one. Similarity is usually determined by MSE difference. If the MSE is below some threshold block is considered to be “similar”, otherwise it is not. MSE can be also calculated between blocks of 2D transform (DCT) coefficients. Several most similar block along with the current one are grouped up into a 3D structure. Size of the 3rd dimension is determined depending on how many “similar” blocks can be found in the neighborhood.

Each block in a 3D structure is transformed with a 2D transform. In addition to the 2D transforms, a 3rd dimension transform is applied to reduce between-block correlations. Transform in the 3rd dimension may differ from 2D transform used. It may be a simpler transform (e. g. Haar).

After filtering of a 3D structure, each of the obtained filtered blocks is placed to the accumulative buffer to its corresponding position multiplied by a corresponding weight. Weight is determined according to one of the techniques described in the previous chapter (SW-DCT). The scheme of BM3D is shown in the Figure 2.9([25]).

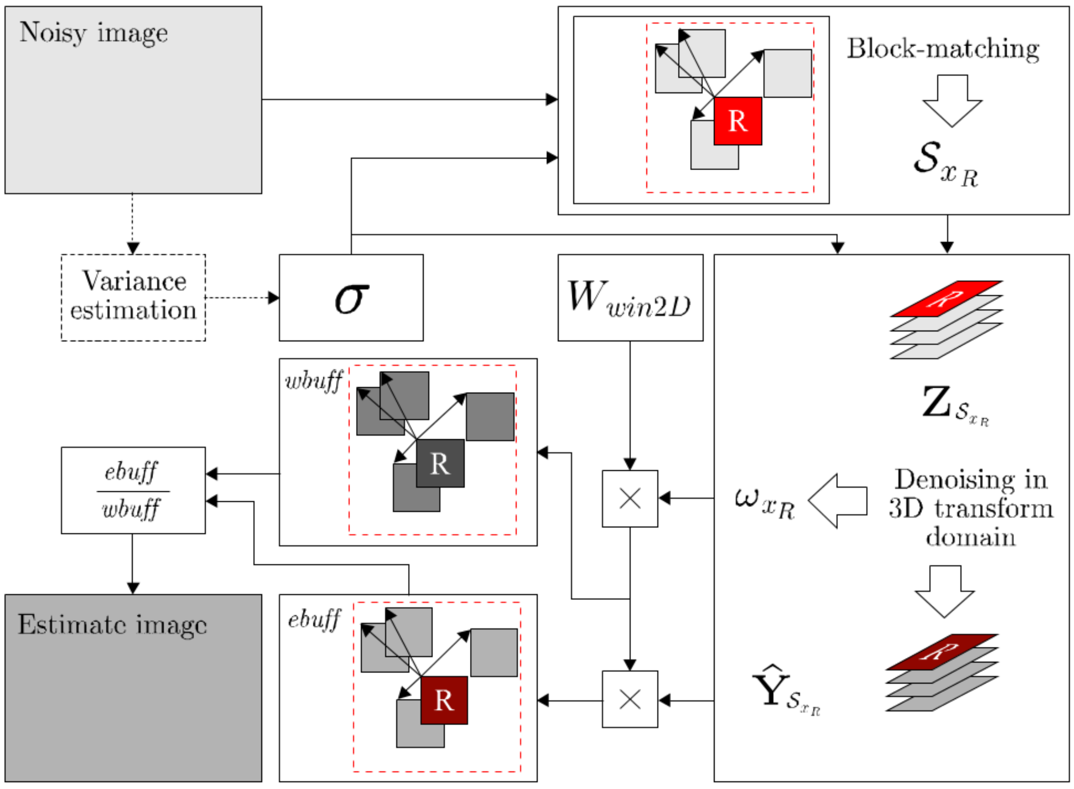


Figure 2.9: BM3D scheme.

BM3D algorithm has several parameters, which are fixed during the filtering process. It inherits all parameters from SW-DCT and, additionally, has settings responsible for block matching. Examples of these parameters are: range of search and thresholding MSE (which limits the maximum MSE for the blocks similarity).

BM3D is the last denoising algorithm that we review in this chapter. It is currently one of the best image noise suppression approaches [34, Chapter 7.2] and in out study we will mostly focus on the classification-based improvement of this algorithm.

Speaking about image areas we were using only empirical criteria for its definition. However, for automatic adaptive denoising one should use some classification algorithm to separate image areas. Hence, before we start the analysis we have to introduce an image segmentation tool.

2.5 Image segmentation

Image pixel classification algorithm is very important part of our study since smart classification allows us to utilize features of particular class for better filtering, while inappropriate classification won't provide us any benefits. Even though there are many image segmentation techniques it is very hard to find an appropriate one. Some of them provide division of an image only into two classes [10], others demand long time processing [11]. In this work, we use classification algorithm based on processing of 2D DCT image blocks [9].

In this algorithm three types of image areas are distinguished:

1. Plain areas. We can utilize high pixels similarity of these regions, using larger windows or suppressing noise harder with larger thresholding coefficients.

2. Texture areas. Pixel similarity is very low, thus estimates based on averaging will have high bias. Complex filtering approaches should be used to preserve image details.

3. Edge areas. One can utilize strict shape of an edge to decrease computational time (e. g. simplify block matching in BM3D) or improve quality (preserve edges).

For the classification of image pixels block classification is used. It is not so accurate as a classification of each pixel independently, but fast enough not to influence filtering algorithm speed which is important. Additionally, BM3D and SW-DCT need 2D DCT blocks for their processing, hence overhead for classification is even more negligible in case one uses these algorithms for filtering.

For the classification, an image is divided into 8×8 blocks and 2D DCT is calculated for each of these blocks. After that, in each block all AC coefficients are divided into three groups: low frequencies, high frequencies and those coefficients corresponding to edges (Figure 2.10). For each group, sum of the absolute values of the coefficients is calculated (three numbers). Based on these sums, all pixels belonging to a block are classified to one of three classes. Several conditions which indicate belonging to the particular class were experimentally determined. In details, let's denote the sum of absolute values of edge coefficients as E and sum of absolute values of low and high frequency coefficients as L and H, respectively. Then, a classification algorithm can be written as in the following pseudo-code:

```
mu_1 = 125
mu_2 = 900
alpha_1 = 2.3
alpha_2 = 1.4
beta_1 = 1.6
beta_2 = 1.1
gamma = 4
kappa = 290

IF( E+H > mu_1 ) THEN
    IF( E+H > mu_2 ) THEN
        IF( ( ( L/E >= alpha_2 ) AND ( (L+E)/H >= beta_2 ) ) OR
            ( ( L/E >= beta_2 ) AND ( (L+E)/H >= alpha_2 ) ) OR
            ( (L+E)/H >= gamma ) ) THEN
            EDGE
        ELSE
            TEXTURE
    ELSE
        IF( ( (L/E >= alpha_1 ) AND ( (L+E)/H >= beta_1 ) ) OR
            ( (L/E >= beta_1 ) AND ( (L+E)/H >= alpha_1 ) ) OR
            ( (L+E)/H >= gamma ) ) THEN
            EDGE
        ELSEIF( E+H > kappa ) THEN
            TEXTURE
        ELSE
            SMOOTH
ELSE
    SMOOTH
```

Using this classification we can perform adaptive processing schemes for the blocks. Separation to smooth and texture areas helps us to obtain the right value for the coefficients which represent bias/variance trade-off. Therefore, one can achieve either better noise suppression or more accurate details preservation. Determination of edge regions may help to pay more attention to the color change preservation. Additionally, one can use it to decrease a computational time for similar blocks

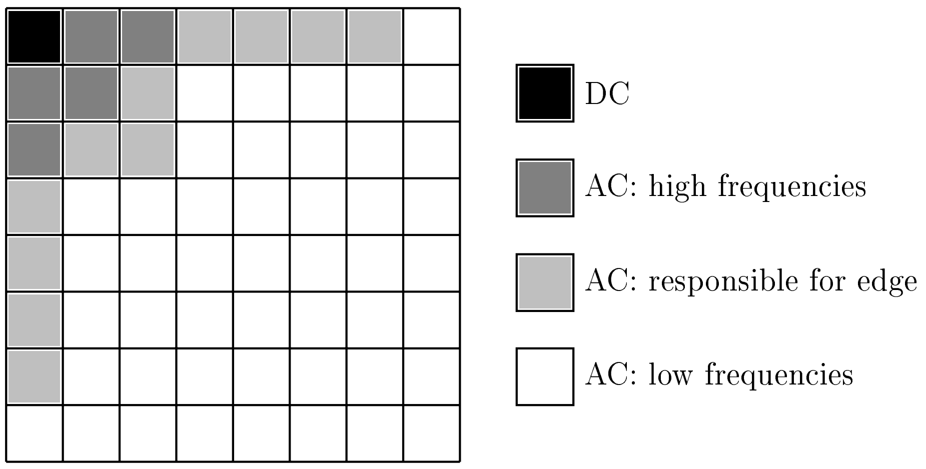


Figure 2.10: Coefficients separation.

determination (e. g. in BM3D algorithm).

3. IMAGE FILTERING WITH CLASSIFICATION

In the previous chapter we reviewed basic principles of several filtering techniques of different types (spatial domain, transform domain, combined spatial and transform domain). Additionally, we reviewed a segmentation tool that provides us an ability to analyze separately features of different image content classes, namely what filters and what settings are the most suitable for each class.

In this chapter we will study dependence of the filtering quality (measured in PSNR) on different filter parameters. We want to show optimal (from PSNR of filtered image point of view) values of parameters individual for each content class. Therefore, content-dependent filtering can improve a quality of the filtered image compared to the baseline approach, where the same settings are applied to the entire image.

For this purpose we will perform image filtering with the range of values for its parameters. After that, we will observe change of the PSNR for the entire image and for each content class separately. From these observations we can determine optimal values of the parameters for the entire image and compare them with the optimal values of the parameters for each of the content classes. Knowing optimal parameters for each class we can perform filtering with adaptive parameter application. From this application we expect to get more qualitative filtering.

For the experiments we will use test image set of widely used images: “Barbara”, “Cameraman” and “Lenna” (Figure 3.1). Their content classification which is used further in this chapter is shown in Figure 3.2. We perform our study of the algorithms in the same order as in the previous chapter: from simple to more complex ones. Hence our first step is LPA-ICI.

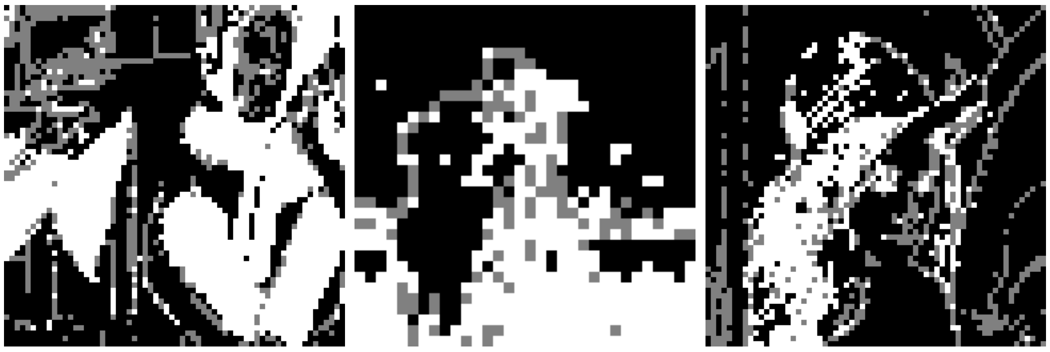
3.1 LPA-ICI with classification

As it was mentioned in the previous chapter, parameter Γ in the LPA-ICI algorithm is responsible for the pixel neighborhood size. Γ controls neighborhood size by controlling the pixel similarity in the pixel neighborhood, because if we accept less similar pixels the size of the neighborhood can be larger. Since that, the choice of this parameter value depends on the bias/variance trade-off and its optimal value should be different for different image content classes.

In order to determine optimal Γ's we observed dependence of calculated PSNR

(a) Barbara (b) Cameraman (c) Lenna

Figure 3.1: Test images



(a) Barbara (b) Cameraman (c) Lenna

Figure 3.2: Test image content classification [9]: black – “smooth”, gray – “edge”, white – “texture”

on this parameter. We degraded each image with white Gaussian noise of different magnitudes and after that we filtered degraded images with the range of Γ values from 0.5 to 2.0 with the step 0.05. Dependences of PSNR on Γ for the entire image and content classes can be seen in Figure 3.6.

In the original article it is recommended to use $\Gamma = 1.05$[6, Chapter 2.4.4]) as the optimal value for the entire image. Optimal Γ for the entire image in our experiments agrees with this value for the low noise. Heavy noise usually demands higher values for the optimal filtering. Dependences of calculated PSNR on Γ are show in Figure 3.6. Optimal Γ are the argument values corresponding to the peaks of the shown curves. For convenience the exact values are shown in Table 3.1. Ideally, each of the classes should be filtered with the corresponding optimal Γ value. We can get two important observations from the consideration of separate content classes filtering:

1. The result of the filtering of smooth areas gives much higher PSNR than that of texture and edge areas of the image.

2. Optimal Γ's for the “Texture” and “Smooth” areas differ from the optimal Γ

	Γ			
Image	Entire	Smooth	Edge	Texture
Barbara, $\sigma = 20$	1.0	1.25	1.10	0.9
Cameraman, $\sigma = 20$	1.05	1.30	1.05	1.0
Lenna, $\sigma = 20$	1.1	1.25	1.1	0.95
Barbara, $\sigma = 40$	1.05	1.30	1.10	0.95
Cameraman, $\sigma = 40$	1.10	1.45	1.0	1.0
Lenna, $\sigma = 40$	1.2	1.35	1.1	1.0
Barbara, $\sigma = 60$	1.15	1.35	1.15	1.1
Cameraman, $\sigma = 60$	1.1	1.5	1.0	1.05
Lenna, $\sigma = 60$	1.2	1.4	1.15	1.1

Table 3.1: Optimal Γ values for LPA-ICI filtering.

for the entire image.

The first observation means that from PSNR point of view error of estimate for the "Smooth" class is much lower than that for the "Texture" class. Therefore, improvement in "Texture" class is more important than that in the "Smooth" class. Hence, drop in quality in the "Smooth" class impacts less on the overall image PSNR than the same drop in "Texture" class.

The second observation is consistent with the assumption about profitability of a content-dependent parameter usage. As it was mentioned, in general the optimal value for the filtering is $\Gamma = 1.05$. Therefore, one can see that texture regions demand lower than average Γ, while smooth regions require greater than average Γ. Thus, by using separate values for Γ for different classes we get PSNR gain in the combined image for the cost of classification. Let's denote an image, which is a union of pixels from three classes each of which is filtered with its own optimal Γ, as a combined image. Table 3.2 shows that combined image has better PSNR than the image which is entirely filtered with the same Γ value ($\Gamma = 1.05$).

Figure 3.3 demonstrates visual comparison between LPA-ICI and LPA-ICI with adaptive Γ selection. One can see that fragments with adaptive approach have been clearly filtered and have more pleasant looking smooth areas with less noise.

Thereby, a use of additional information such as image area classification can improve image filtering quality for the spatial domain filtering. This result will be considered further in SA-DCT filtering. We will see later that restoration quality of LPA-ICI is very poor compared to other filtering algorithms and its processing does not suit well for our block-wise classification. The next step is to study possible improvements for transform domain algorithms.

	PSNR	
Image	LPA-ICI	Combined LPA-ICI
Barbara, $\sigma = 20$	27.47	27.64
Cameraman, $\sigma = 20$	29.28	29.42
Lenna, $\sigma = 20$	30.71	30.87
Barbara, $\sigma = 40$	23.83	24.02
Cameraman, $\sigma = 40$	25.52	25.79
Lenna, $\sigma = 40$	27.50	27.73
Barbara, $\sigma = 60$	22.54	22.59
Cameraman, $\sigma = 60$	23.30	23.62
Lenna, $\sigma = 60$	25.72	25.95

Table 3.2: Comparison of calculated PSNR for images filtered with LPA-ICI with the same Γ for the entire image and combined image.

3.1.1 SW-DCT with classification

Optimal value of thresholding coefficient (λ) of SW-DCT may depend on the image content as well as the value of Γ in LPA-ICI. It is recommended to use fixed value of thresholding coefficient ($\lambda = 2.7$) which is usually an optimal value for the entire image (yields highest PSNR). But, as it was mentioned above, the optimal value can differ for different types of image content. To test this hypothesis we made series of denoising experiments with different λ value from 1.3 to 4.0 with step 0.1 applied to test images, which were degraded with white Gaussian noise with several levels of standard deviations.

Results of the experiments are shown in Figure 3.7. Table 3.3 presents arguments of curve peaks from Figure 3.7. One can see that for plane region the optimal hard thresholding coefficient λ is higher than default one and for the texture region it is lower. Moreover, the diversity of the optimal value for the same noise and the same region type for different images is low (± 0.1). Therefore, e. g. for noise $\sigma = 20$ we can use separate values for λ: 2.4 for texture and around 3.0 for smooth areas. Optimal λ for the edge region varies from image to image and single value can not be determined. The last observation can be explained by the fact that edge area is heterogeneous and two different areas meet there: smooth and smooth area, smooth and texture area or texture and texture area. Therefore, in average it does not have any special properties except the pixel color intensity leap.

Comparison of PSNR for images which are entirely filtered with the same value of λ and images with own λ value for each region is shown in Table 3.4.

Visual comparison of SW-DCT and SW-DCT with adaptive λ selection (Figure 3.4) shows that adaptive approach allows to suppress more noise in the image. In a fragment filtered with simple SW-DCT one can notice some pattern left by the unsuppressed high frequency coefficients on the flat areas. In the adaptive SW-DCT

	λ			
Image	Entire	Smooth	Edge	Texture
Barbara, $\sigma = 20$	2.6	3.0	2.7	2.4
Cameraman, $\sigma = 20$	2.5	3.1	2.5	2.3
Lenna, $\sigma = 20$	2.7	3.0	2.7	2.4
Barbara, $\sigma = 40$	2.6	3.2	2.8	2.4
Cameraman, $\sigma = 40$	2.6	3.4	2.4	2.5
Lenna, $\sigma = 40$	2.8	3.4	2.7	2.5
Barbara, $\sigma = 60$	2.6	3.4	2.8	2.5
Cameraman, $\sigma = 60$	2.7	3.5	2.5	2.5
Lenna, $\sigma = 60$	2.9	3.5	2.8	2.6

Table 3.3: Optimal λ values.

	PSNR	
Image	LPA-ICI	Combined SW-DCT
Barbara, $\sigma = 20$	30.07	30.23
Cameraman, $\sigma = 20$	29.61	29.99
Lenna, $\sigma = 20$	32.15	32.34
Barbara, $\sigma = 40$	26.21	26.50
Cameraman, $\sigma = 40$	26.22	26.56
Lenna, $\sigma = 40$	28.80	29.14
Barbara, $\sigma = 60$	24.16	24.43
Cameraman, $\sigma = 60$	24.29	24.63
Lenna, $\sigma = 60$	26.82	27.20

Table 3.4: Comparison of calculated PSNR for images filtered with SW-DCT with the same λ for the entire image and combined image.

this pattern is weaker and the image seems to be filtered clearer.

In case of SW-DCT image blocks classification demands less computations than that in the spacial domain case (since forward 2D DCT is calculated anyway). At the same time, we can make an important observation: classification provides an information that improves filtering.

Additionally, classification is not only helping us to increase filtering quality but also may speed up filtering. Larger thresholding coefficient for smooth regions leaves less DCT coefficients and, therefore, the inverse DCT can be performed faster in a reduced way (e. g. if all coefficient except the DC value are zeros, all the values in the block after the inverse DCT are the same).

SW-DCT showed acceptable filtering quality performance, presence of ways for its acceleration and convenience for classifier use. Here we finish our study of pure filters and move to the combined ones. The first algorithm to analyze is a Shape-adaptive DCT.

	Γ			
Image	Entire	Smooth	Edge	Texture
Barbara, $\sigma = 20$	3.0	1.6	3.0	3.0
Cameraman, $\sigma = 20$	1.4	1.5	1.6	1.4
Lenna, $\sigma = 20$	1.8	1.6	1.7	2.2
Barbara, $\sigma = 40$	3.0	1.5	2.0	3.0
Cameraman, $\sigma = 40$	1.2	1.5	1.1	1.1
Lenna, $\sigma = 40$	1.6	1.6	1.4	2.4
Barbara, $\sigma = 60$	3.0	1.5	2.4	3.0
Cameraman, $\sigma = 60$	1.4	1.6	1.1	1.1
Lenna, $\sigma = 60$	1.6	1.6	1.4	1.7

Table 3.5: Optimal Γ values.

	α			
Image	Entire	Smooth	Edge	Texture
Barbara, $\sigma = 20$	0.75	0.85	0.8	0.75
Cameraman, $\sigma = 20$	0.75	0.9	0.8	0.7
Lenna, $\sigma = 20$	0.8	0.85	0.8	0.75
Barbara, $\sigma = 40$	0.75	0.9	0.8	0.7
Cameraman, $\sigma = 40$	0.75	0.95	0.75	0.75
Lenna, $\sigma = 40$	0.8	0.9	0.8	0.75
Barbara, $\sigma = 60$	0.8	0.9	0.8	0.75
Cameraman, $\sigma = 60$	0.8	0.95	0.7	0.75
Lenna, $\sigma = 60$	0.85	0.95	0.8	0.75

Table 3.6: Optimal α values.

3.2 SA-DCT with classification

In this section we will observe impact of Γ and α parameters on SA-DCT filtering. Parameter Γ is responsible for the size of the block and homogeneity of pixels in the block. While α is used in hard thresholding. From [22] we know that in average the best values for the entire image are $\alpha = 0.77$, $\Gamma = 1.5$. In our analysis we fix one of these parameters to the optimal value for the entire image and vary the other one.

In the first experiment we fix $\alpha = 0.75$ and vary Γ from 0.5 to 5.0 with the step 0.05. In the second experiment we fix $\Gamma = 1.5$ and vary α from 0.5 to 1.2 with the step 0.05. The results for both of these experiments are shown in Figures 3.8 and 3.9 respectively. Arguments for the curve peaks for these figures are presented in Tables 3.6 and 3.5.

From the Figure 3.8 one can see that, unfortunately, we can not get improvement from adaptive Γ parameter use. For the smooth and edge areas the best value of Γ is close to one used for the entire image. The optimal value for the Γ in texture areas can be both larger and less than the optimal value for the entire image. Large value of optimal Γ for the “Texture” in the “Barbara” image is caused by “phenomenon of trousers pattern” This phenomenon we will observe in the next chapter about SW-DCT (Table 4.1). Thus, we can not utilize different values of Γ since we do not have a tool that can determine such kind of texture, which demands large window size for its filtering.

Figure 3.9 depicts the same behaviour for α as for λ in SW-DCT: smooth areas should be filtered with larger than average (0.77) values of hard thresholding coefficient, while smaller values should be used for the texture areas. Unfortunately, in case of SA-DCT an adaptive approach yield very low PSNR increase (less than 0.1 dB) and low improvement of visual quality (indistinguishable) compared to usual SA-DCT. Since that we will not use it in our future development.

From comparison in Figures 3.7 and 3.9 one can notice that the value of PSNR in “Smooth” areas filtered with SA-DCT is higher than that in the smooth areas filtered with SW-DCT. From PSNR point of view one should use SA-DCT for smooth region instead of SW-DCT. But in terms of time consumption in SA-DCT we pay for the block shape determination (the process which is absent in SW-DCT). Additionally, in block processing we cannot use only fast transforms since block size is not fixed. Moreover, PSNR improvement in smooth areas is not as significant as in others. The PSNR there is higher than in any other areas, hence, the contribution made to the entire image from this area improvement is minor. Therefore, we will not use SA-DCT for “Smooth” areas.

We nearly finish our analysis of possible contributions, which can be made by the classification being applied to the filtering process. The last algorithm, which is left to analyse is a powerful BM3D.

3.3 BM3D filtering with classification

As in SW-DCT case, separate use of hard thresholding coefficient λ for different classes may significantly improve the quality of filtering. It is directly responsible for the noise suppression. Thus, we will study it first. As in the SW-DCT, range for λ for the experiments was chosen from 1.3 to 4 with the step 0.1. The results of these experiments are shown in Figure 3.10 and the arguments for the curve peaks (optimal values of λ) from this figure are shown in the Table 3.7.

From the graphs one can see that the optimal thresholding coefficients are lower in “Texture” areas than the coefficient for the entire image and higher in “Smooth”. We observed the same behaviour in the cases of SW-DCT and SA-DCT. Particular values for λ in BM3D case differ from λ in SW-DCT because the possibility to suppress noise depends on pixels correlation. The higher correlation the larger λ should be used. Here the additional decorrelation is obtained from 3rd dimension, therefore, optimal thresholding coefficients for corresponding image classes are higher in

	λ			
Image	Entire	Smooth	Edge	Texture
Barbara, $\sigma = 20$	2.8	3.2	3.0	2.6
Cameraman, $\sigma = 20$	2.6	3.5	2.6	2.4
Lenna, $\sigma = 20$	2.9	3.2	2.9	2.6
Barbara, $\sigma = 40$	2.8	3.2	3.0	2.6
Cameraman, $\sigma = 40$	2.7	3.8	2.6	2.5
Lenna, $\sigma = 40$	3.0	3.7	2.9	2.6
Barbara, $\sigma = 60$	2.9	3.7	3.0	2.7
Cameraman, $\sigma = 60$	2.9	4.0	2.7	2.7
Lenna, $\sigma = 60$	3.2	3.9	3.0	2.8

Table 3.7: Optimal λ values.

	PSNR	
Image	BM3D	Combined BM3D
Barbara, $\sigma = 20$	31.31	31.37
Cameraman, $\sigma = 20$	30.08	30.34
Lenna, $\sigma = 20$	32.56	32.67
Barbara, $\sigma = 40$	27.19	27.36
Cameraman, $\sigma = 40$	26.63	26.93
Lenna, $\sigma = 40$	29.02	29.35
Barbara, $\sigma = 60$	25.19	25.45
Cameraman, $\sigma = 60$	24.86	25.20
Lenna, $\sigma = 60$	26.97	27.56

Table 3.8: Comparison of calculated PSNR for images filtered with BM3D with the same λ for the entire image and combined image.

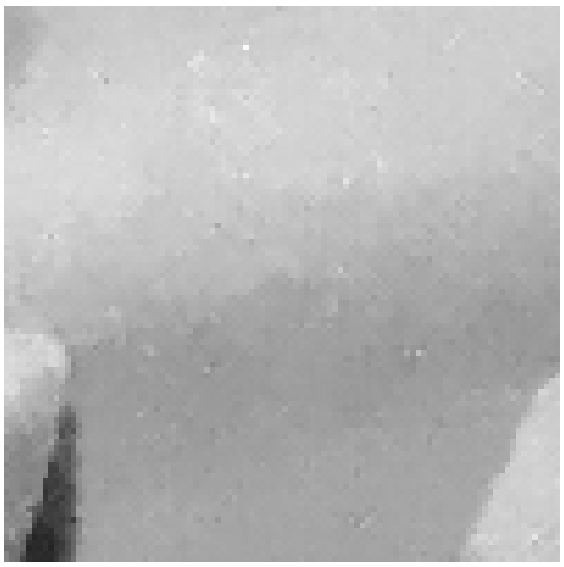
BM3D than in SW-DCT (compare Figures 3.7 and 3.10).

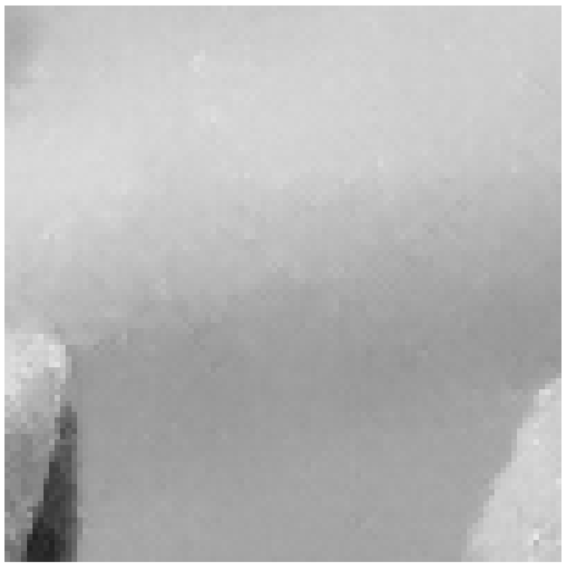
Comparison of quality measured in PSNR for combined images (where λ is chosen individually for each content class) and for images where the same λ for the entire image is presented in Table 3.8. Visual comparison is shown in Figure 3.5. One can notice that fragments related to BM3D with content-dependent λ selection are clearer and, therefore, better filtered than the corresponding fragments filtered with BM3D.

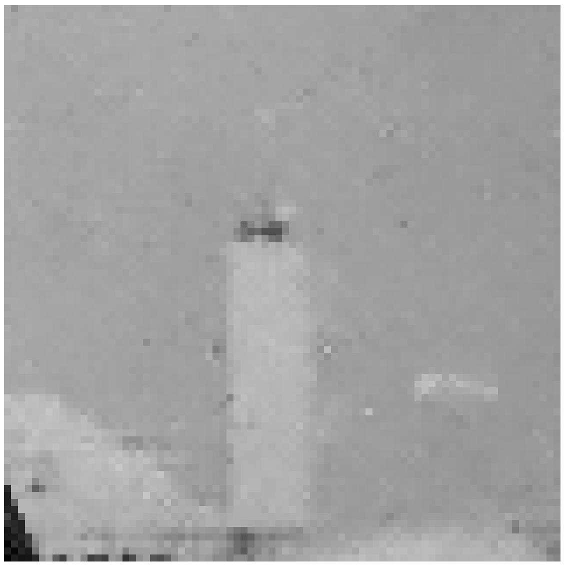
Since we are interested in decrease of computational complexity we will also consider if it is possible to decrease similar blocks search range without loosing quality. This parameter is crucial for BM3D speed performance since block searching implies MSE calculation between source block and all blocks within some range. Since amount of MSE to be computed is approximately $4 \cdot R^2$ for each block, where R is a search range, this part takes most of the algorithm processing time. One can see that halving the R causes reduction of computational complexity by a factor of 4. To determine a dependence of calculated PSNR of search range we have performed experiments with variable search range for the numbers from a set: $\{0, 1, 2, 4, 8, 16, 32\}$. Worth to be noticed that BM3D degenerates to SW-DCT when the search range is set to zero. Results of experiments are shown in Figure 3.11.

From the latter Figure one can see that usually the best value for search range is 16, but it is also noticeable that for smooth area PSNR for 16 and 0 does not differ much. Therefore, we can use simple SW-DCT in the smooth area and greatly decrease computational complexity. With that simplification quality does not drop much. Additionally, possible lose is located in the area where quality is high enough anyway. In the other areas significant drop in PSNR can be observed for the search range bellow 8, while PSNR for 8 and 16 does not differ much. Hence, we can use value 8 instead of 16 for these areas and reduce computations almost by a factor of 4.

One more modification which decreases computational complexity is fast block-matching algorithm. During the processing full block-matching search is performed even though search range is decreased to the local region. As it was mentioned earlier, this process takes a lot of computational resources and therefore time. Several modifications such as two-dimensional logarithmic search [29], diamond search [30] and hexagon search [31] were developed and were proved to perform fast and accurate block matching without significant MSE growth. These techniques can further speed-up the algorithm and they are already used in several BM3D-like implementation [32]. In this work we do not focus on the improvement of these algorithms and none of the fast search techniques is used in the implementations.

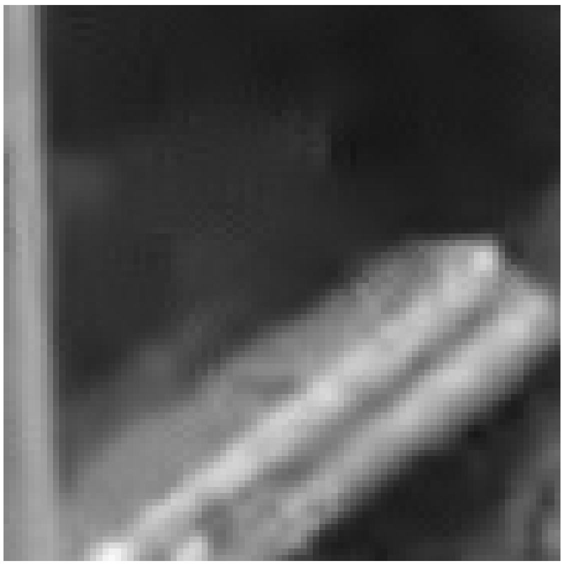

(a) Barbara fragment. LPA-ICI

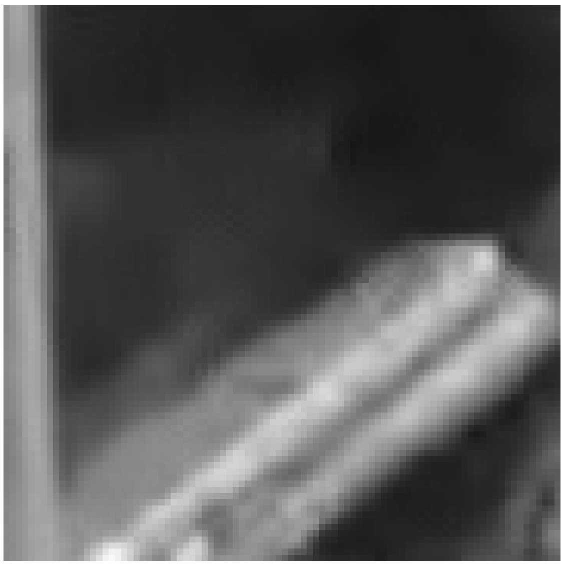

(b) Barbara fragment. Content-dependant LPA-ICI

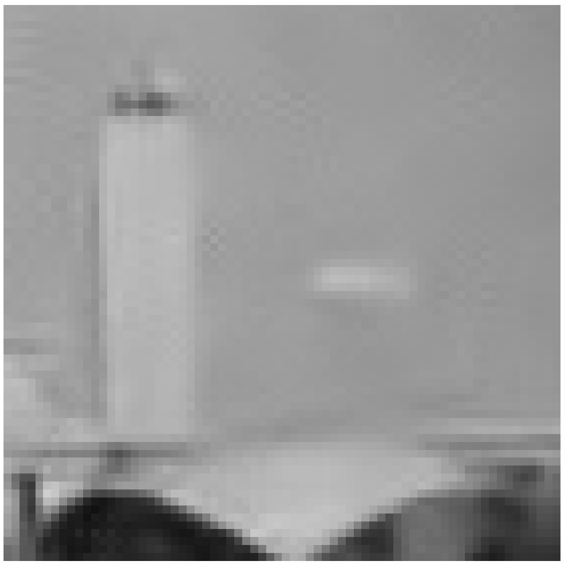

(c) Cameraman fragment. LPA-ICI

(d) Cameraman fragment. Content-dependant LPA-ICI

Figure 3.3: Visual comparison between image filtered with LPA-ICI and content-dependant LPA-ICI.


(a) Barbara fragment. SW-DCT


(b) Barbara fragment. Content-dependant SW-DCT


(c) Cameraman fragment. SW-DCT

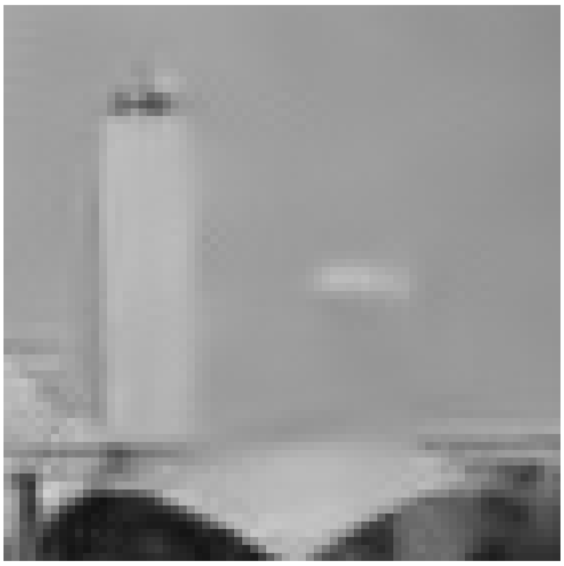
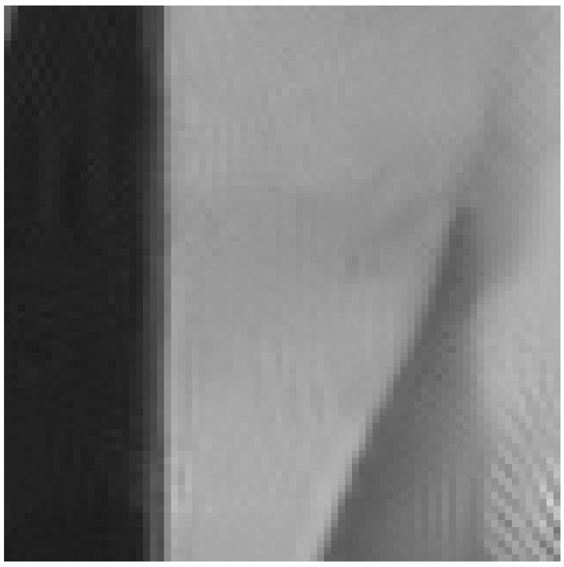
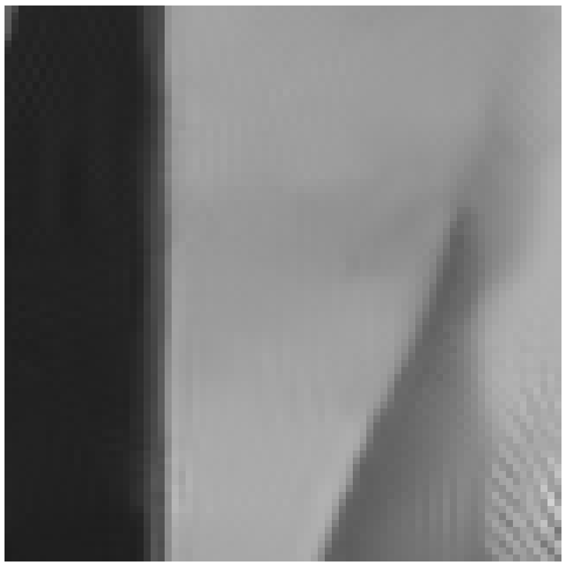
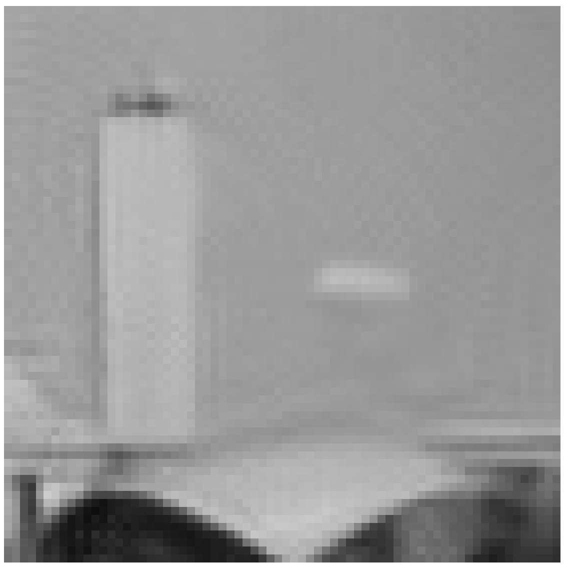

(d) Cameraman fragment. Content-dependant SW-DCT

Figure 3.4: Visual comparison between image filtered with SW-DCT and content-dependant SW-DCT.


(a) Barbara fragment. BM3D


(b) Barbara fragment. Content-dependent BM3D


(c) Cameraman fragment. BM3D

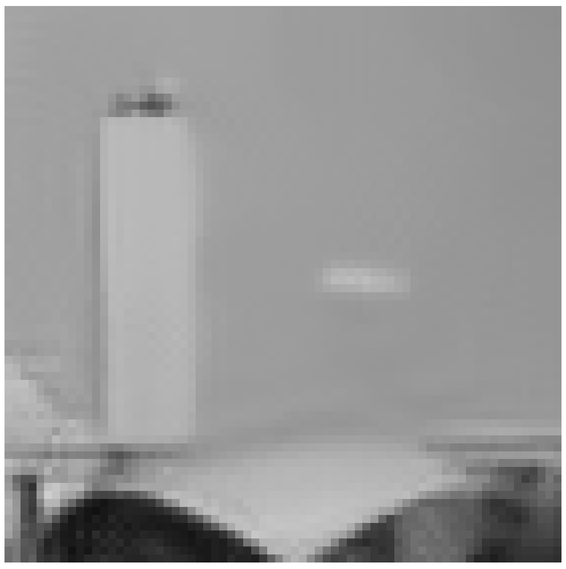
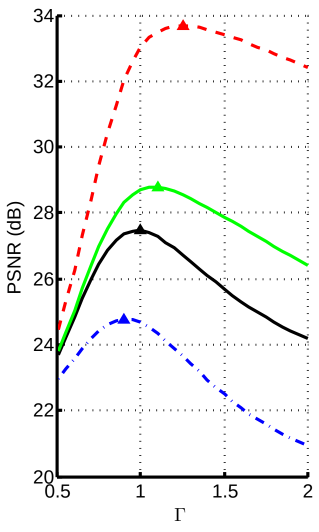
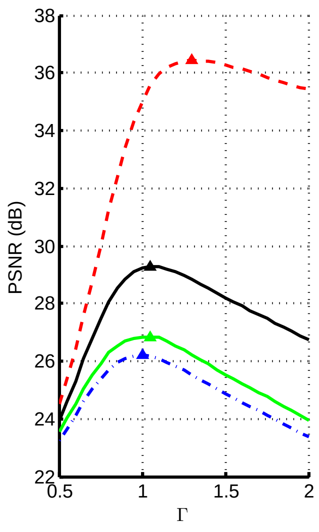

(d) Cameraman fragment. Content-dependent BM3D

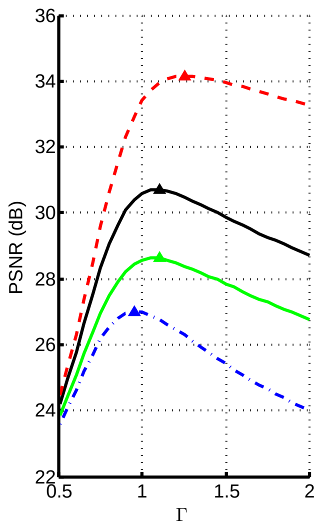
Figure 3.5: Visual comparison between image filtered with SW-DCT and content-dependent SW-DCT.



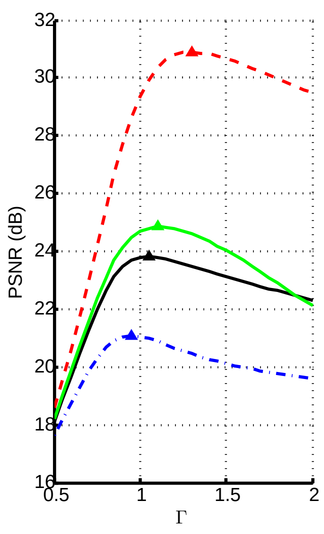
(a) “Barbara”. $\sigma = 20$.



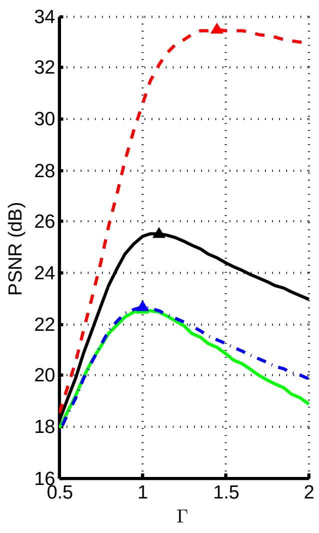
(b) “Cameraman”. $\sigma = 20$.



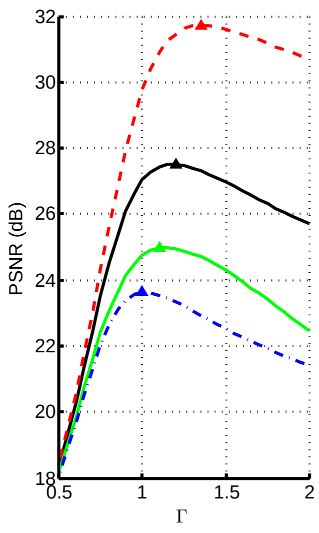
(c) “Lena”. $\sigma = 20$.



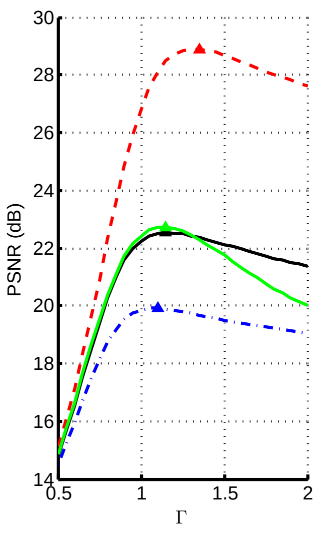
(d) “Barbara”. $\sigma = 40$.



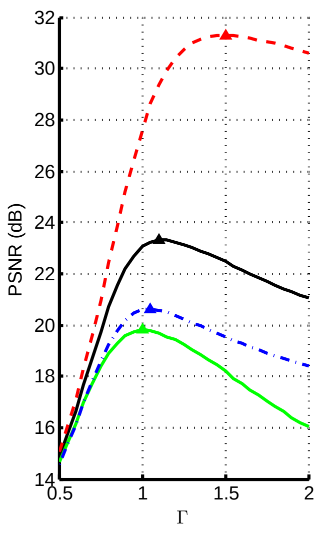
(e) “Cameraman”. $\sigma = 40$.



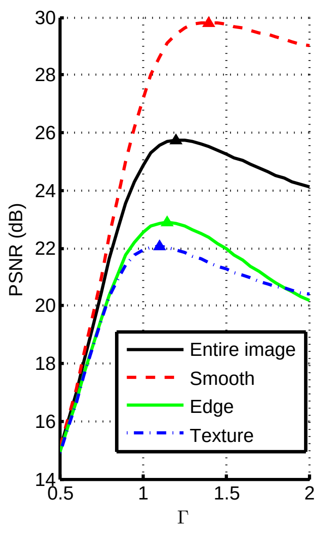
(f) “Lena”. $\sigma = 40$.



(g) “Barbara”. $\sigma = 60$.

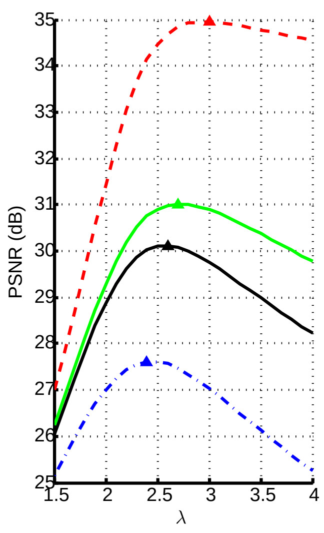


(h) “Cameraman”. $\sigma = 60$.

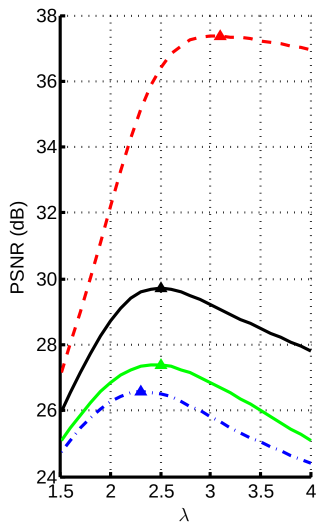


(i) “Lena”. $\sigma = 60$.

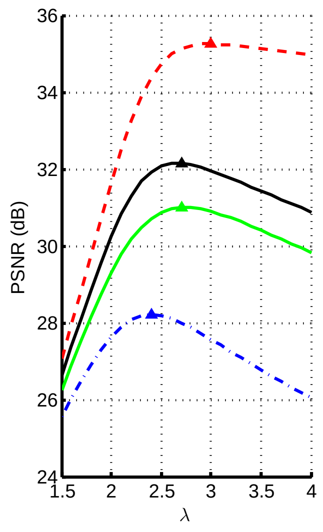
Figure 3.6: Dependence of calculated PSNR on Γ for LPA-ICI filtering.



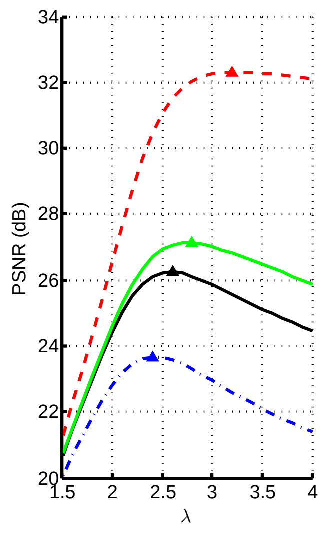
(a) “Barbara”. $\sigma = 20$.



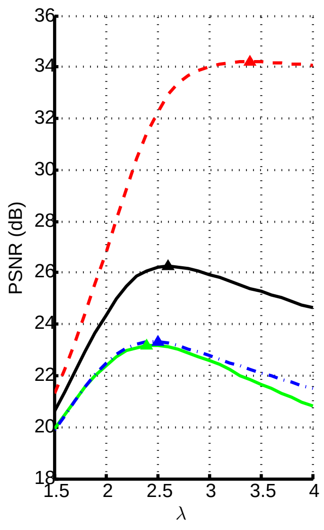
(b) “Cameraman”. $\sigma = 20$.



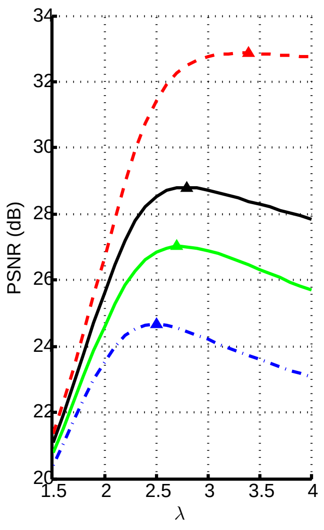
(c) “Lena”. $\sigma = 20$.



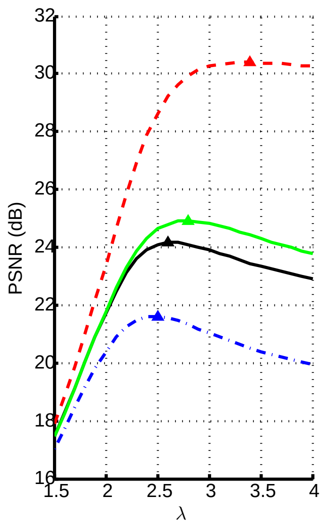
(d) “Barbara”. $\sigma = 40$.



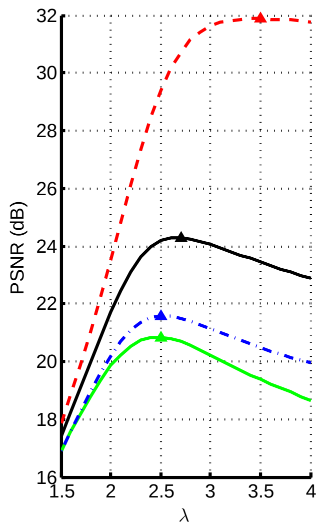
(e) “Cameraman”. $\sigma = 40$.



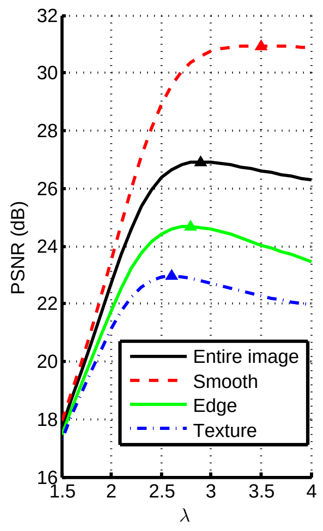
(f) “Lena”. $\sigma = 40$.



(g) “Barbara”. $\sigma = 60$.

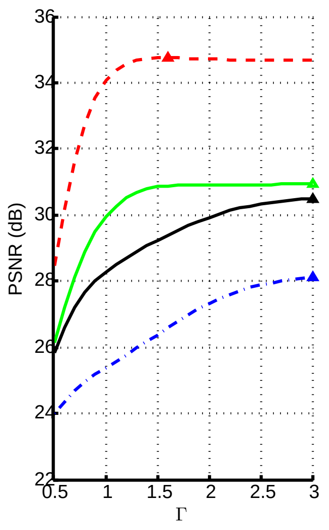


(h) “Cameraman”. $\sigma = 60$.

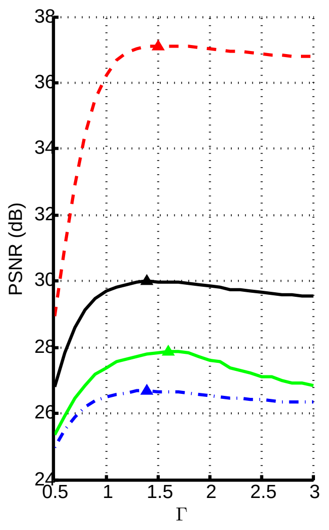


(i) “Lena”. $\sigma = 60$.

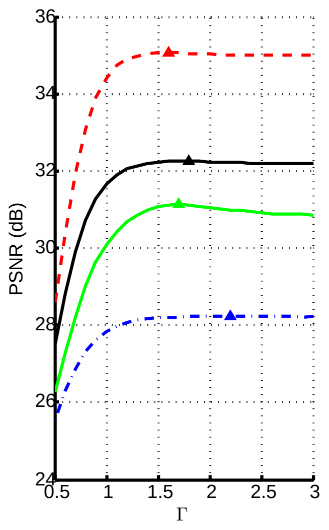
Figure 3.7: Dependence of calculated PSNR on λ for SW-DCT filtering.



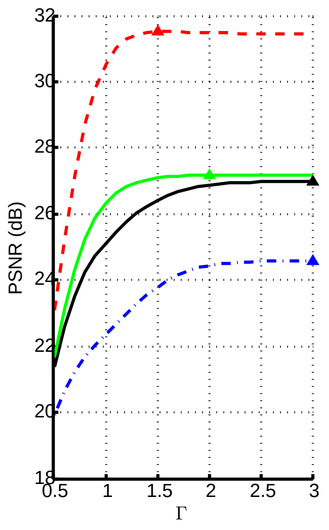
(a) "Barbara". $\sigma = 20$.



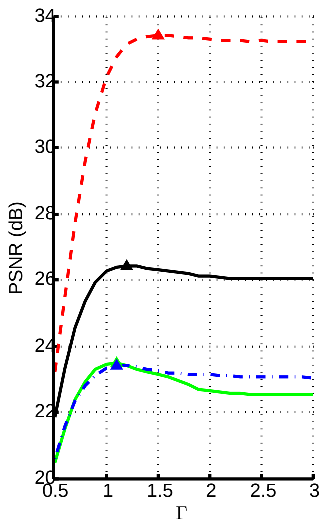
(b) "Cameraman". $\sigma = 20$.



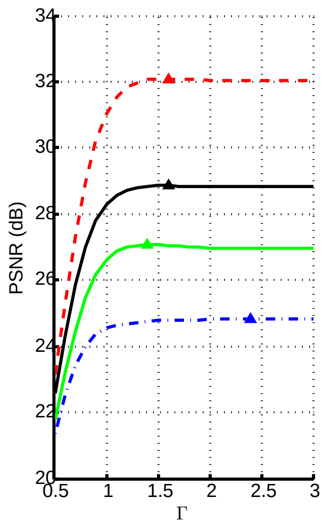
(c) "Lena". $\sigma = 20$.



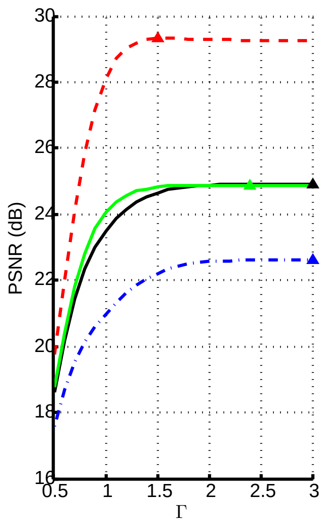
(d) "Barbara". $\sigma = 40$.



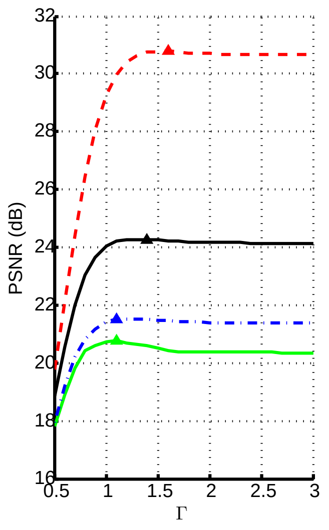
(e) "Cameraman". $\sigma = 40$.



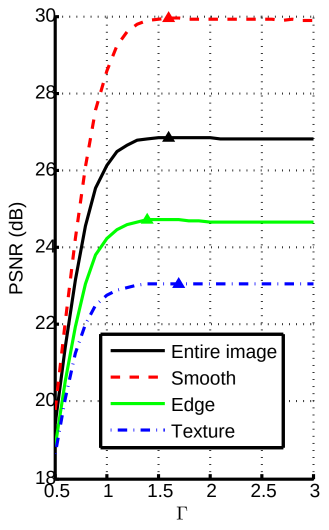
(f) "Lena". $\sigma = 40$.



(g) "Barbara". $\sigma = 60$.

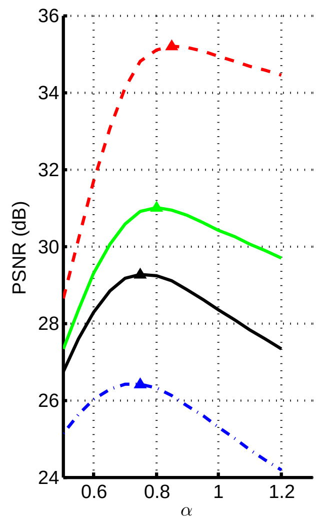


(h) "Cameraman". $\sigma = 60$.

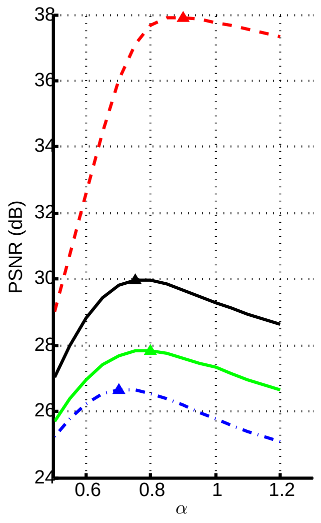


(i) "Lena". $\sigma = 60$.

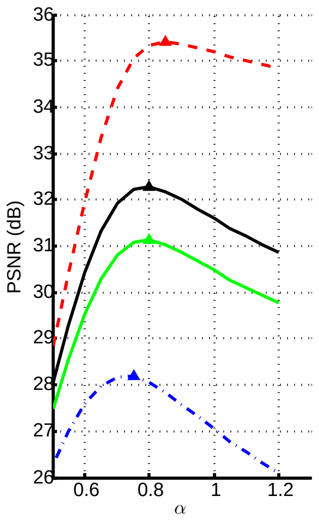
Figure 3.8: Dependence of calculated PSNR on Γ for SA-DCT filtering.



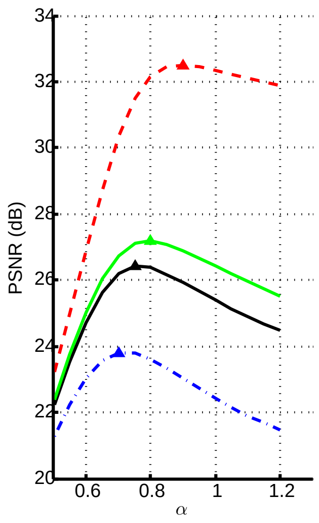
(a) “Barbara”. $\sigma = 20$.



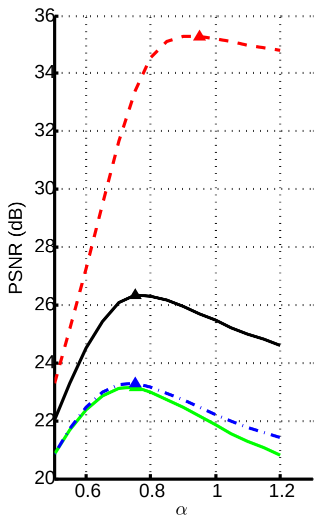
(b) “Cameraman”. $\sigma = 20$.



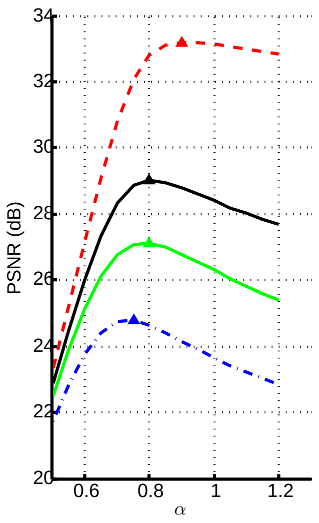
(c) “Lena”. $\sigma = 20$.



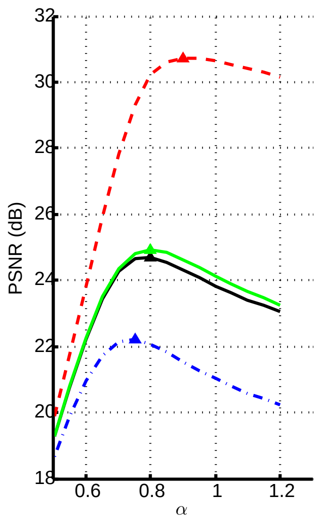
(d) “Barbara”. $\sigma = 40$.



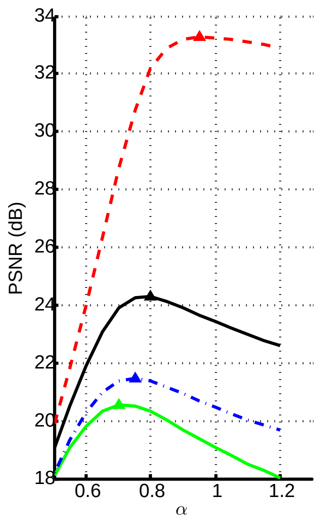
(e) “Cameraman”. $\sigma = 40$.



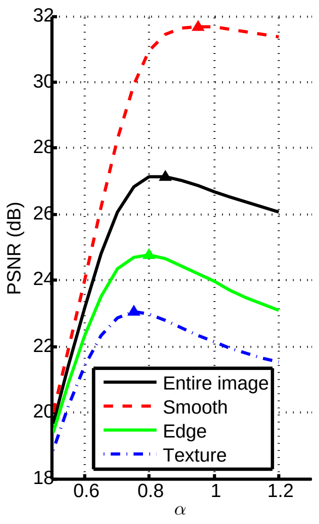
(f) “Lena”. $\sigma = 40$.



(g) “Barbara”. $\sigma = 60$.

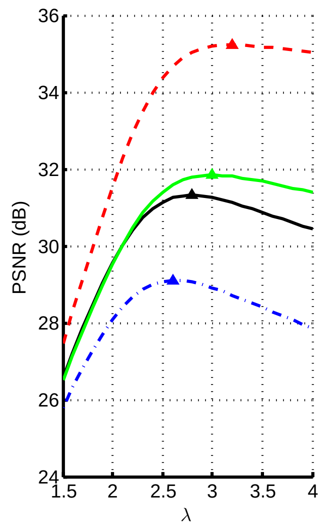


(h) “Cameraman”. $\sigma = 60$.

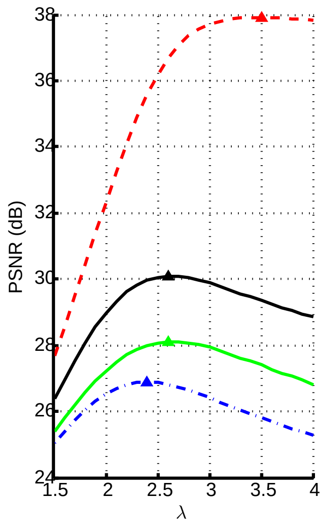


(i) “Lena”. $\sigma = 60$.

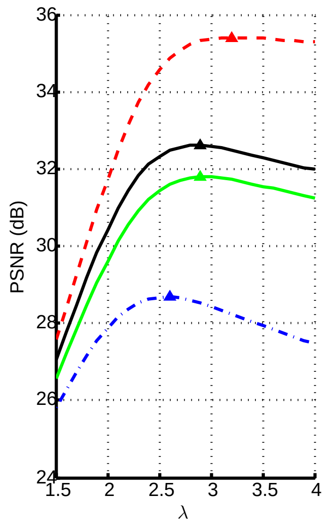
Figure 3.9: Dependence of calculated PSNR on α for SA-DCT filtering.



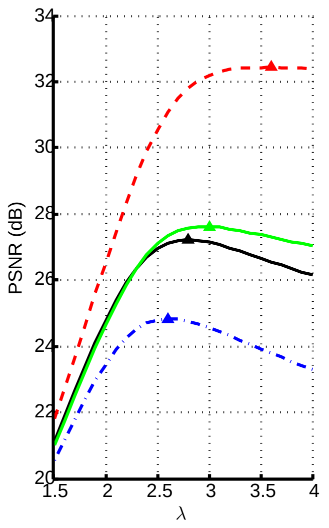
(a) “Barbara”. $\sigma = 20$.



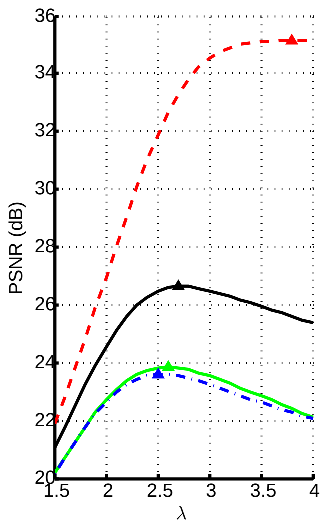
(b) “Cameraman”. $\sigma = 20$.



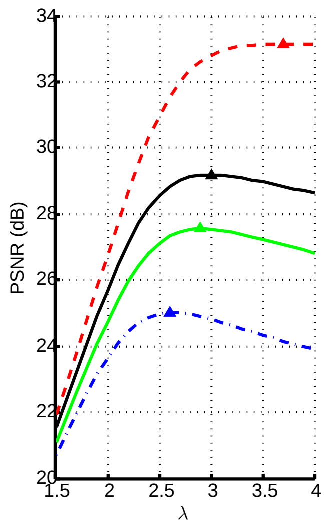
(c) “Lena”. $\sigma = 20$.



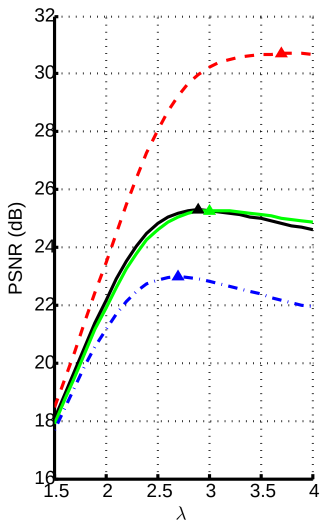
(d) “Barbara”. $\sigma = 40$.



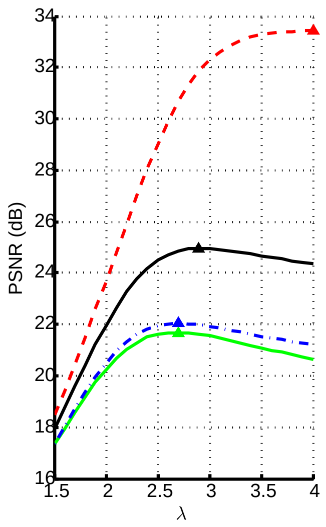
(e) “Cameraman”. $\sigma = 40$.



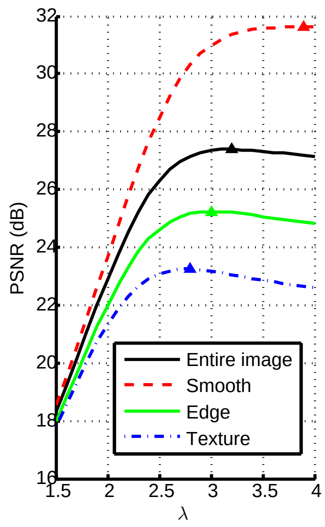
(f) “Lena”. $\sigma = 40$.



(g) “Barbara”. $\sigma = 60$.

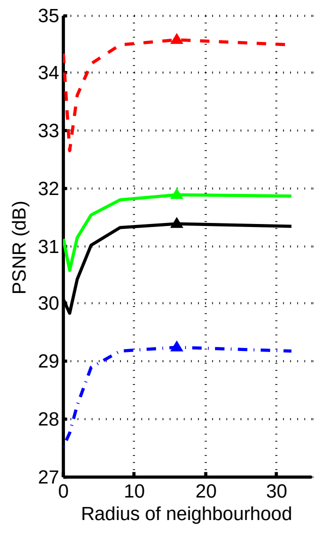


(h) “Cameraman”. $\sigma = 60$.

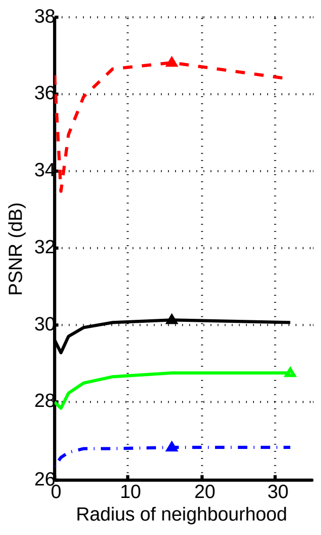


(i) “Lena”. $\sigma = 60$.

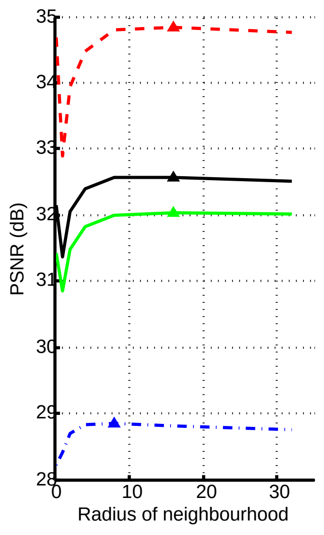
Figure 3.10: Dependence of calculated PSNR on Γ for BM3D filtering.



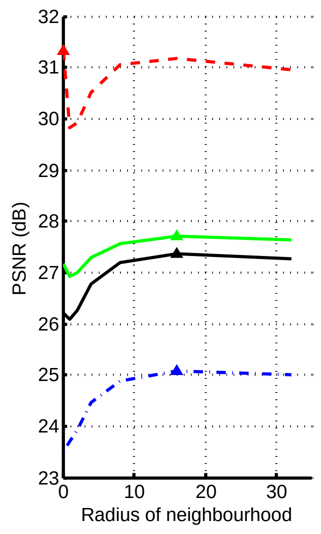
(a) “Barbara”. $\sigma = 20$.



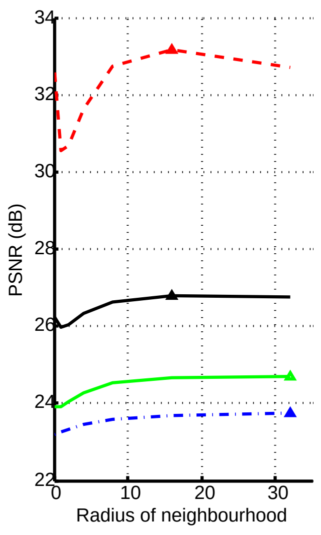
(b) “Cameraman”. $\sigma = 20$.



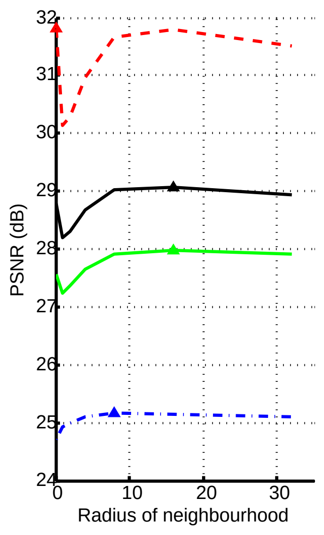
(c) “Lena”. $\sigma = 20$.



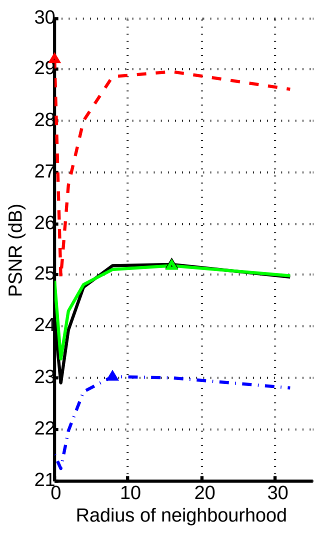
(d) “Barbara”. $\sigma = 40$.



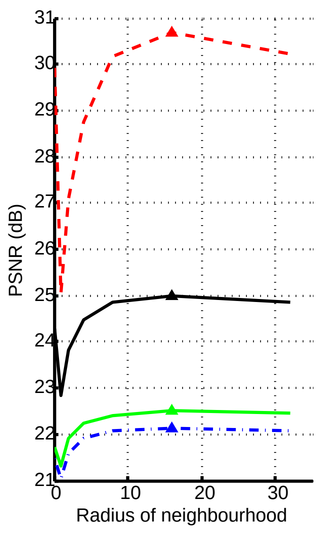
(e) “Cameraman”. $\sigma = 40$.



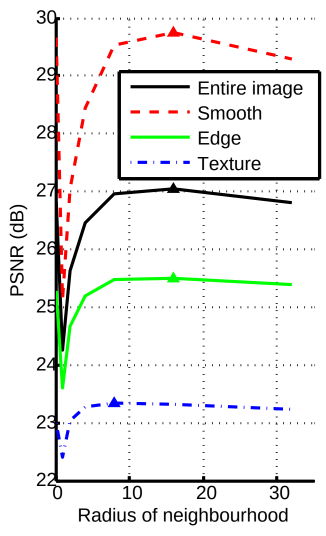
(f) “Lena”. $\sigma = 40$.



(g) “Barbara”. $\sigma = 60$.



(h) “Cameraman”. $\sigma = 60$.



(i) “Lena”. $\sigma = 60$.

Figure 3.11: Dependence of calculated PSNR on radius of neighbourhood for BM3D filtering.

4. SW-DCT MODIFICATIONS

In the previous chapter while considering SW-DCT we were using its version with the following options:

1) floating point calculation;
2) weighted aggregation;
3) 8×8 block size
4) no Wiener filter.

In this chapter we will consider how block size, aggregation type, Wiener filter and transition to the integer variable calculation impacts on filtering quality and speed performance.

4.1 Block size

We start SW-DCT analysis with a study of block size, aggregation and Wiener filter contribution to the filtering quality. For that purpose our test image set was corrupted with white Gaussian noise with different levels and filtered with SW-DCT with different settings.

For filtering we used 4×4, 8×8 and 16×16 block sizes and two types of aggregation (average and weighted average), additionally Wiener post-filtering was applied to weighted average aggregation. Results for each image are shown in Figure 4.1. For better visualization, all the graphs are drown relatively to 8×8 SW-DCT with weighted average aggregation, which is shown as the x-axis.

From these graphs we can make several conclusions:

1. Generally, 8×8 is the best in quality/complexity block size for the filtering. The exception in Barbara image (16×16 is better than 8×8) is caused by "trousers texture" (Table 4.1), which is very specific texture pattern. Additionally, 16×16 DCT is at least $\frac{16}{8} \cdot \log_8 16 = \frac{8}{3}$ times slower than 8×8 one since fast implementations of DCT usually have time complexity proportional to $N \cdot \log_2 N$, where N is the window size. One more drawback of 16×16 block is that necessity to operate with 4 times larger block may cause delays related to memory processing. Thus, we will further consider 8×8 as a standard block size for the filtering.

	PSNR			
Block size	Overall	Smooth	Edge	Texture
8×8	30.01	**34.85**	**30.96**	27.38
16×16	**30.55**	34.84	30.89	**28.17**

Table 4.1: SW-DCT filter performance for different "Barbara" image areas depending on block size.

2. Wiener filter can yield up to 0.5 dB better quality. But it requires repetition of the filtering process and, therefore, doubles the time. Thus, we will not use it further for experiments and implementations.

3. Generally, weighted average aggregation performs better than simple average aggregation (up to 0.5 dB in "Cameraman" image). The source of this advantage is "edges" in the image where weighted average aggregation increases impact of smooth blocks around the "edges". For that reason we will use weighted average aggregation further. However, in case of very fast implementations simple averaging will be performed, because it can be computed with the help of less multiplication and division operations.

From the comparison of Figures 3.6 and 3.7 we can also infer that SW-DCT, which represents frequency domain filtering, shows better quality performance compared to LPA-ICI filtering. The drawback of the SW-DCT method is the amount of the complex computations per pixel in average (computation of forward and inverse DCT's and thresholding). Our goal is not only improvement of quality but also decrease of complexity. We aim to use all possibilities to reduce computational complexity and study what is the cost for such accelerations from a quality point of view. Since that, we will consider integer version of SW-DCT.

4.1.1 Integer pseudo-DCT

The replacement of real number calculations with integer ones can lead to worse PSNR due to rounding errors. At the same time it increases speed of computations for processor architectures where floating point operations are performed slower than integer ones. Basic algorithms for integer transforms were taken from H.264/AVC standard [26].

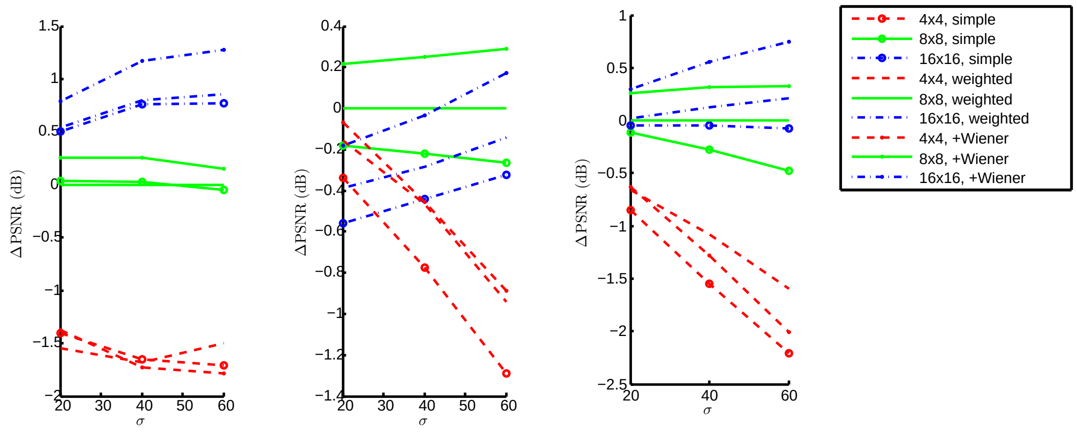
The main benefit of the integer transform is that some processors (especially in mobile devices) perform calculations with integer variables faster than with floating point. Another advantage of integer pseudo-DCT is that it does not use multiplications and divisions in transforms themselves. Multiplications are used in normalization step (one multiplication per pseudo-DCT coefficient) and in aggregation (if we use weighted average aggregation). Divisions are used only in aggregation step (one division per image pixel).

Integer pseudo-DCT in addition to forward and inverse transforms (similar to the real number DCT) have normalization step. Normalization step is introduced because forward and inverse pseudo-DCT transforms are not orthonormal. In this step pseudo-DCT coefficients are multiplied by coefficients so that after inverse pseudo-DCT values close to the original are obtained.

In real valued DCT hard thresholding step is performed right after forward transform. In integer case it can be performed after forward pseudo-DCT but threshold in hard thresholding is different for each coefficient since they are not normalized. These individual thresholds can be computed ones before the processing. Proposed threshold modification allows to decrease computational complexity since normalization step (which includes multiplication operation) can be performed after hard thresholding for non-zero (non-suppressed) coefficients only. For the detailed description see Appendix B.

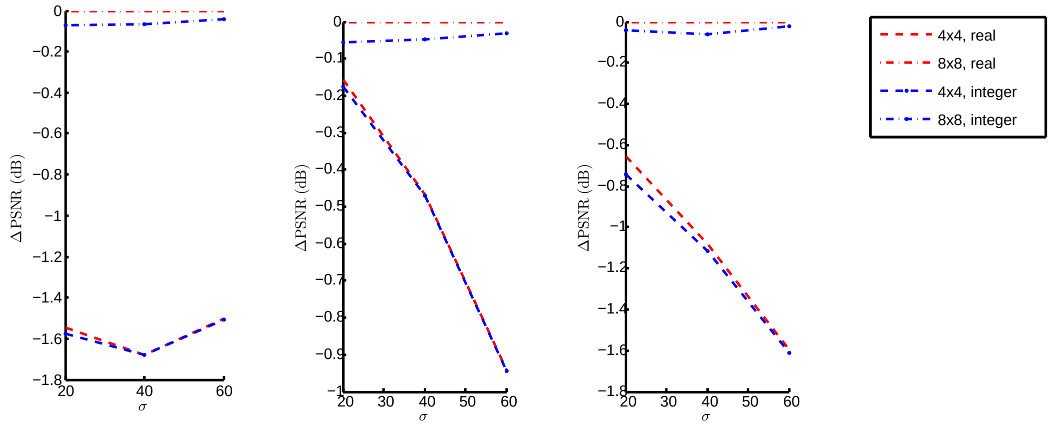
As it was already mentioned, switch from real-valued to integer-valued arithmetic may lead to worse filtering PSNR. Depending on the architecture of the processing unit one may switch to integer processing if the speed performance gain is more substantial than minor drop in quality. Figure 4.2 demonstrates that the quality of integer processing filtering is negligibly lower. Drop in PSNR is very small (at most 0.1 dB for the considered noise levels), some curves corresponding to the PSNR of real and integer filtering of the same images are even indistinguishable. Thus, integer pseudo-DCT is a good alternative to the real-valued DCT and can be used in our implementations.

Thus, in most of the cases we can change floating point computations to approximate integer ones with pseudo transforms and decrease computational time (especially in mobile devices, where floating point operations sometimes is a “bottle neck”). In some cases this change can be done without any drop in quality.



(a) “Barbara” (b) “Cameraman” (c) “Lena”

Figure 4.1: Dependence of calculated ΔPSNR on the noise level σ for different parameters. All the values are shown with respect to the result of 8×8 block SW-DCT with weighted average aggregation. “Weighted” means weighted average aggregation, “+Wiener” – use of Wiener filtering, “simple” – simple average aggregation.



(a) “Barbara” (b) “Cameraman” (c) “Lena”

Figure 4.2: Comparison of SW-DCT algorithms real and integer DCTs. All values are shown with respect to the result of 8×8 block real valued SW-DCT.

5. CONTENT-BASED IMAGE FILTERING

After performance analysis of the filters described in the previous chapters, in this chapter we propose an algorithm that uses different filters for different image content classes with individual optimal parameters for each class. Image classification not only helps to get an improved image quality, but also it is crucial in saving the computational time (Chapter 3, BM3D serach range). Examples of both quality and time improvements we will see further in this chapter.

In the previous chapters we saw that the “Texture” region should be processed as accurate as possible, since in this area there is a low pixel similarity. In general, the highest PSNR for “Texture” class among all of the presented algorithms can be achieved by use of BM3D. Thereby, the best decision of filtering for “Texture” is BM3D with corresponding optimized parameters (hard thresholding coefficient etc.).

The “Smooth” region does not need very accurate and detailed filtering algorithm as “Texture” does. Therefore, the best choice is to use either simple SW-DCT or SA-DCT. The latter filtering approach demands more computational resources than SW-DCT but provides better filtering quality. This can be explained by the fact that pixel similarity in the block is much higher and can be controlled by Γ coefficient. One certain advantage of SW-DCT is that chosen image content classification approach is based on 8×8 DCT coefficients blocks which are used in SW-DCT as well. Blocks in SA-DCT are adaptive and rarely of the square 8×8 shape. Thus, we cannot simplify classification calculation while using SA-DCT.

The “Edge” region is generally a region of low pixel similarity as well (except the case when two flat surfaces meet), but filtering with maximum possible accuracy of this area is very computationally expensive process. In this case, we can use the idea that the most similar blocks to the original “Edge” block are situated only along some direction (namely, along the edge detected). Thereby, the block search can be performed only in the vicinity of the edge itself and the BM3D processing time can be reduced.

Considered image classification algorithm provides an accurate method of noiseless image segmentation to three regions with different pixel correlation behaviour. For test purposes the segmentation of noise-free images was sufficient. However, since we are aiming to perform real image filtering without having an “oracle” which provides the noiseless image, presence of (probably heavy) noise in images to be filtered is a drawback of this algorithm. Thus, we have to present a modified version, that can handle segmentation for noisy images as well.

5.1 Noisy image segmentation

In this section we propose a modified algorithm for image segmentation. The generic classes of the image regions ("Smooth", "Edge" and "Texture") in the modified algorithm remain the same. This division is very convenient and natural from filtering point of view for the reasons explained earlier. But the exact content of the classes in modified classification slightly changes. The trend of these changes is as follows.

First, some blocks with large low frequency coefficients which originally are classified as "Texture" can be included in the "Edge" class. These large low frequency coefficients represent some slow but significant changes in pixels intensity level and can be visually noticed. Thus, in the neighboring blocks along some direction large low frequency coefficients should remain large. This happens due to continuity of real life images.

Second, with the growth of the noise some area that is considered to be "Texture" on the noise-free image can not be determined as such in the noisy one. In other words, texture pattern is weak and noise completely destroys it. Therefore, the pixels from this area can be included in the "Smooth" class.

The modifications proposed above might decrease filtered image PSNR (but not necessarily, since false block classification may decrease PSNR as well). However, from computational complexity point of view, it improves the algorithm, since "Smooth" and "Edge" block processing according to the technique described earlier is faster than that for the "Texture".

Decision on the belonging of image pixels to the particular class is made block-wise based on 2D DCT coefficients blocks. Block-wise processing reminds one describes in [9], we divide coefficients into groups (see Figure 5.1). But in this case we focus only on 2 groups: high frequency coefficients and low frequency coefficients. From the analysis of 2D DCT coefficients of multiple true edge blocks we decided that coefficients marked as "low frequency" are mostly responsible for the presence of the edge.

Thus, first, we want to have robust to moderate noise edge detection. There are two ideas of the selected determination approach:

1. Among low frequency coefficients should be those which have large enough value, so that visually an edge can be recognized.

2. Values of these coefficients should exceed high frequency coefficient values (otherwise a result of classification is "Texture").

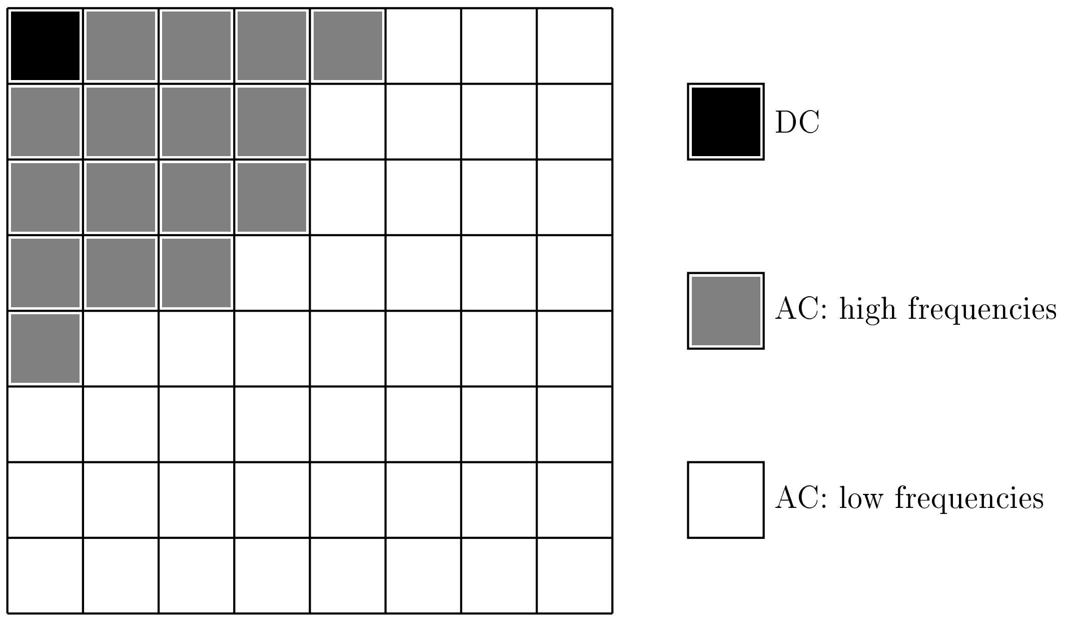
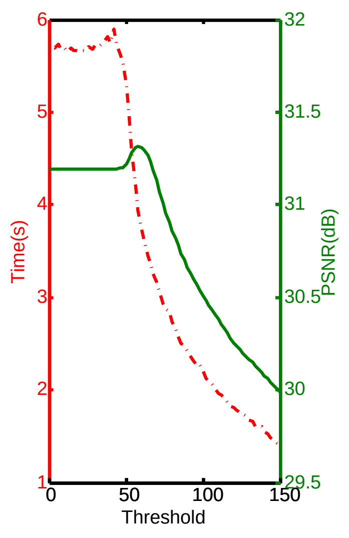


Figure 5.1: Coefficients separation.

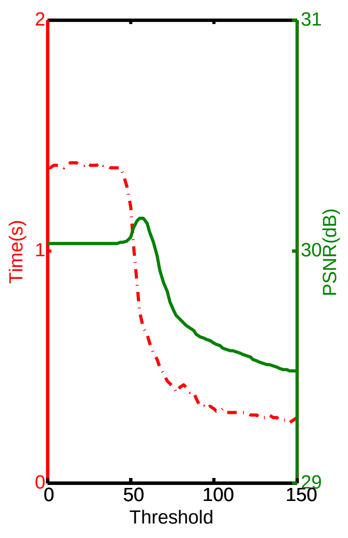
Therefore, it can be formally written as: $\max\{L_A\} \geq T_E$ and $\max\{L_A\} \geq C_{LH} \cdot \max\{H_A\}$, where L_A is a set of absolute values of low frequency coefficients, H_A is a set of absolute values of high frequency coefficients, T_E is some threshold and C_{LH} is a coefficient. From the series of experiments values for T_E and C_{LH} which provide reasonable visible edge determination were found.

Second, we want to have a clear segmentation of "Smooth" and "Texture" regions. AC coefficients indicate presence of some texture pattern in a block. Therefore, if the absolute value of some coefficients is large enough the block can be classified as "Texture". Additionally, noise linearly enhances AC coefficients, thus, the threshold which indicates "large enough" coefficients has to be dependent on the noise level. Hence, we decide that pixels belonging to a "Texture" class can be determined with a formula: $\max\{\max\{L_A\}, \max\{H_A\}\} \geq f(\sigma)$, where f is a function of σ.

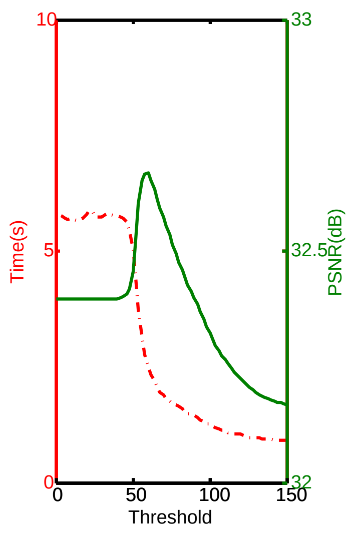
The exact formula for function $f(\sigma)$ is unknown. Since that we study impact of the threshold (value of f for a particular noise) on the filtering results. Too low value of the threshold leads to misclassification that assigns most of the image pixels to the "Texture" area. Therefore, most of the image will be filtered with BM3D with hard thresholding coefficient optimal for "Texture" class. This coefficient is slightly lower than that for the entire image because we consider such for the "Texture". Otherwise, if the value of f is too large the filtering degenerates to the SW-DCT with over-suppressed noise. Noise is over-suppressed since optimal hard thresholding coefficient for "Smooth" area is larger. Figure 5.2 shows what happens with filtering speed and quality performances with varying $f(\sigma)$ for images from testing set for noise levels $\sigma = 20, 40, 60$.



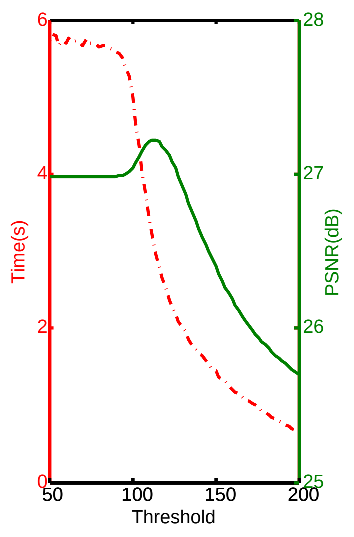
(a) “Barbara”. $\sigma = 20$.



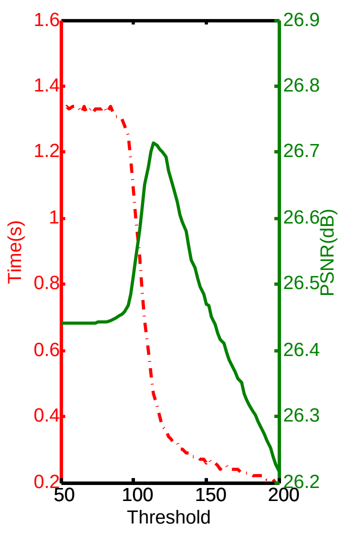
(b) “Cameraman”. $\sigma = 20$.



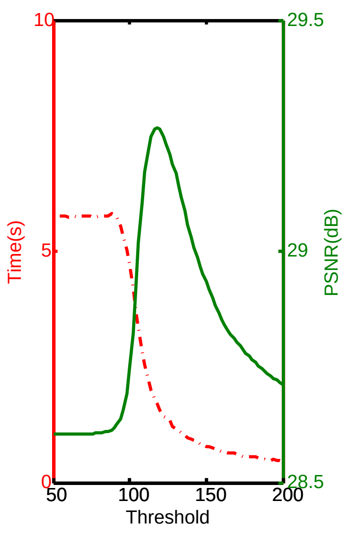
(c) “Lena”. $\sigma = 20$.



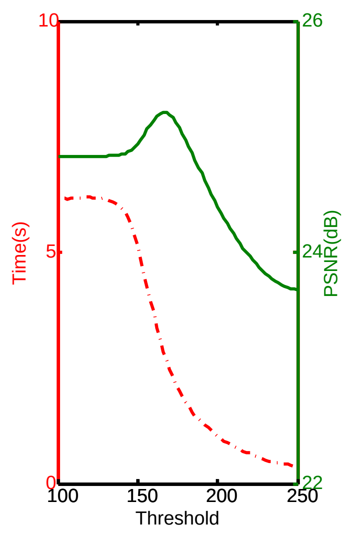
(d) “Barbara”. $\sigma = 40$.



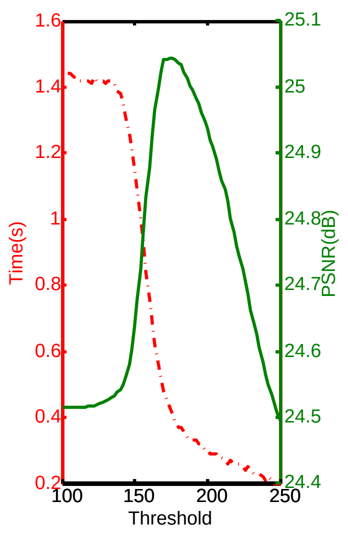
(e) “Cameraman”. $\sigma = 40$.



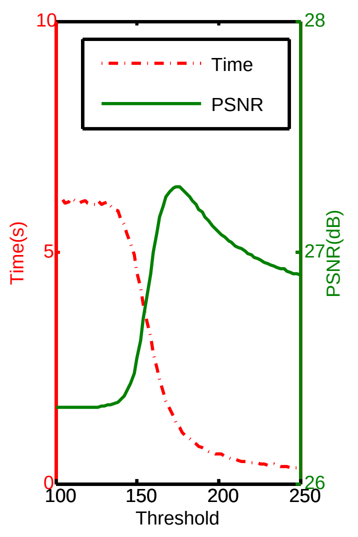
(f) “Lena”. $\sigma = 40$.



(g) “Barbara”. $\sigma = 60$.



(h) “Cameraman”. $\sigma = 60$.



(i) “Lena”. $\sigma = 60$.

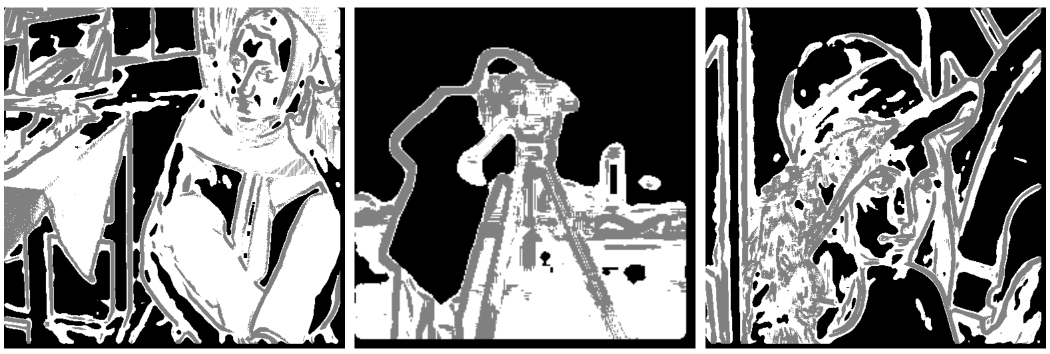
Figure 5.2: Dependence of filtering speed and quality performances on the threshold level for $\sigma = 20, 40, 60$ noise levels.

One can see that in each of the graphs shown there is always lower limit for f below which we do not gain neither speed nor quality improvement. The values of threshold below this limit are useless. The values above this limit is the area where one can find the trade-off of quality/complexity. One can see that starting from this lower limit with the growth of the threshold the value of PSNR goes up while the time graph falls down. This can be explained by the fact that more blocks are classified as "Smooth" and in these blocks BM3D is being replaced by SW-DCT which is faster. At some value of $f(\sigma)$ PSNR graph reaches its peak but with further growth of the threshold it starts to fall down.

By using values larger than that corresponding to the peak we can either achieve quality improvement with a small speed-up or any speed-up (up to the speed of SW-DCT) with some drop in quality. In this work for "Smooth"/"Texture" thresholding we were using values slightly higher than those corresponding to the peaks, since we wanted to show speed-up without quality loss.

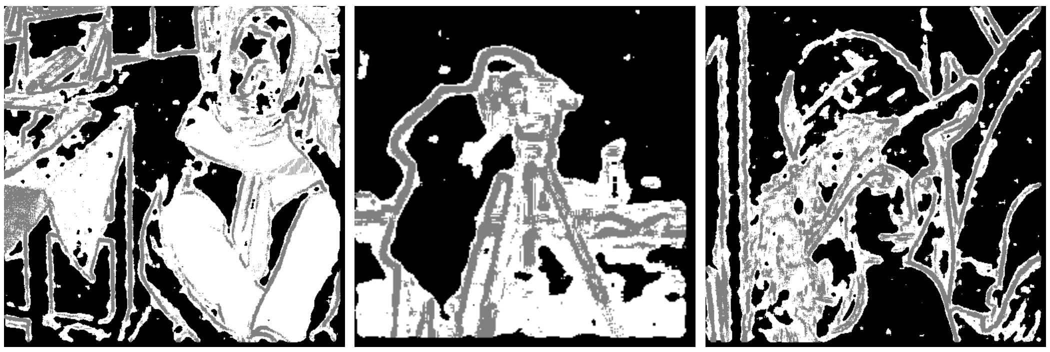
In practical implementation where processing time is a crucial parameter from the graph similar one to shown in Figure 5.2 one can find the value for the threshold which yields desired time and approximately determines possible PSNR drop. In our experiments, e. g. for $\sigma = 20, 40, 60$, we used thresholds $60, 120, 176$, respectively.

From the observations described above we developed a classification technique, which provides a satisfactory image segmentation. Briefly, algorithm works as follows: we check a block for being "Edge", if it is not an "Edge" then it is either "Texture" (high AC) or "Smooth". Since neighboring blocks most probably belong to the same class and noise introduces high probability of misclassification, we additionally perform median filtering of block classification for "Smooth" and "Texture" classes. Some examples of segmentation of images from test image set can be seen in Figures 5.3 and 5.4.



(a) "Barbara" (b) "Cameraman" (c) "Lena"

Figure 5.3: Classification of blocks for noiseless images.



(a) "Barbara" (b) "Cameraman" (c) "Lena"

Figure 5.4: Classification of blocks for images with noise (σ=20).

5.1.1 Processing of "Edge" blocks

As it was stated earlier, in the vicinity of the edges one can improve block matching speed performance by simplification of the block search. The algorithm of the search optimization is simple and uses spatial features of edge.

Optimization is done as follows. We start the search in the 5×5 square (neighborhood range = 2). At the border of this square we detect the block which gives the lowest MSE with the initial block (the most similar one). This block roughly denotes the direction of the edge. Since we believe that the edge is locally straight, the opposite block to the detected one is considered to denote the direction of the edge as well. From this moment we have two blocks each of which denotes edge direction. These blocks have unprocessed neighboring blocks. For each neighborhood of two blocks the block with the least MSE to the initial one is independently determined. Both these determined blocks now denote the edge direction and the process of this "edge distribution" is repeated. Amount of these recursive steps can be bounded by the same value as was used for neighborhood range search.

An example of the first step is shown in Figure 5.5. Each pixel denotes the upper left corner of a block. Red – initial block. Dark gray – border blocks. Black – block with the minimum MSE and opposite to it (blocks which denote the edge direction). Light gray – unprocessed neighbours of blocks which denote edge direction.

Examples of the complete search for some blocks of the test images is shown in Figure 5.6. A colored pixel denotes an upper-left corner of a block. Green pixel – initial block, blue – blocks processed by the original full-search algorithm, magenta – blocks which were processed by optimized search. The speed-up of the search is the ratio of blue and magenta areas.

Since all the parts are complete (classification and processing for each class), we can perform filtering of the entire images with the proposed algorithm and make comparisons. Quality and speed performances of BM3D and BM3D with content-

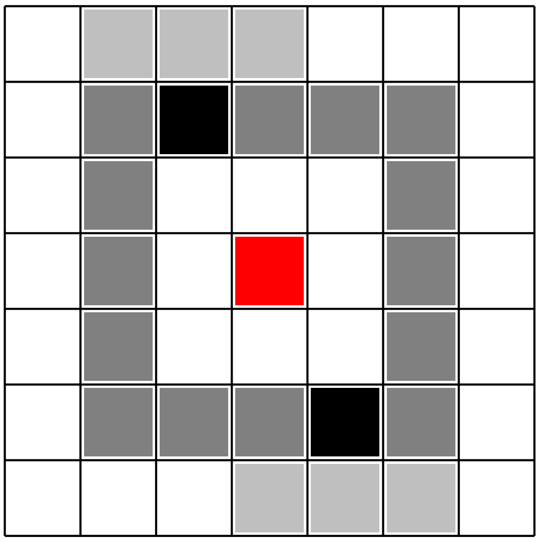


Figure 5.5: First step of “edge distribution”.

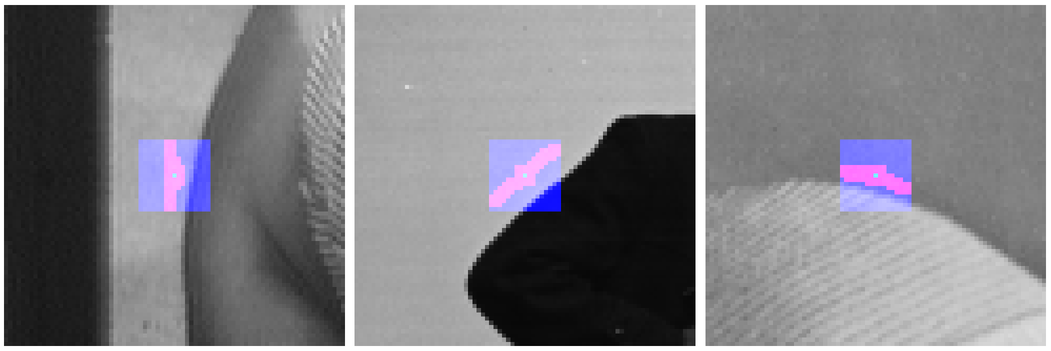

(a) “Barbara” (b) “Cameraman” (c) “Lenna”

Figure 5.6: Edge block-matching.

dependent optimizations for several images are shown in Tables 5.1 and 5.2. One can observe that the quality of filtered images in average remains the same. Time spent for filtering is always less for an adaptive scheme. The speed-up is $1.5 - 2.5$ times depending on the ratio of smooth/edge/texture area of the particular image.

For visual comparison of BM3D and content-dependent BM3D some examples of filtered fragments from “Barbara” and “Cameraman” images are shown in Figure 5.7. One can see that fragments filtered with the proposed algorithm have less noise artefacts.

5.2 Adaptive integer BM3D

Another way to decrease a computational complexity of BM3D is to substitute floating point variables and operations to integer ones. In the previous chapter such a substitution for SW-DCT was shown. BM3D compared to SW-DCT has 2 more operations: block matching and transform for 3rd dimension. Integer block

Algorithm	Time spent for images (seconds)						
	Barbara	Cameraman	Lenna	Hill	Couple	House	Peppers
BM3D	8.13	1.80	8.14	8.30	8.21	1.85	1.81
CD-BM3D	5.36	0.80	3.48	4.69	5.04	0.76	1.09
CD-intBM3D	4.68	0.75	3.10	4.07	4.42	0.65	0.98

Table 5.1: Comparison of speed performance.

Algorithm	PSNR for images (dB)						
	Barbara	Cameraman	Lenna	Hill	Couple	House	Peppers
BM3D	31.24	30.03	32.54	30.36	30.24	33.28	30.84
CD-BM3D	31.19	30.02	32.62	30.31	30.27	33.28	30.91
CD-intBM3D	31.21	29.68	32.61	30.28	30.19	33.13	30.70

Table 5.2: Comparison of quality performance.

matching is performed by an analogy to a floating point one. The feature that one should take into account is that values of integer 2D pseudo-DCT blocks are larger than those of usual floating point 2D DCT blocks. Therefore, MSE threshold for similarity determination should be larger as well. Integer form of Haar transform which is used as the third dimension transform in BM3D is very similar to the usual Haar. Therefore, with minor changes we can convert our floating point BM3D to an integer one.

Our adaptive scheme uses a classification which in turn uses 2D DCT blocks. However, using integer transforms we calculate only pseudo DCT. Fortunately, these pseudo DCT coefficients are very close to the real DCT coefficients multiplied by some factor (see B.1). Therefore, we can use pseudo DCT coefficient blocks for classification and save computational time in the same manner as in floating point case.

We performed comparisons of PSNR and time performance for the proposed content-dependant integer BM3D filter for the extended image set too (see Tables 5.1 and 5.2). Integer transform yield lower PSNR while being used in filtering compared to the floating point one. However, in all cases we get improvement in time performance.

Since tests were performed on a desktop computer with Intel® processor we did not obtain significant speed-up related to the transition from floating to integer transforms. However, for processors used in mobile devices this transition speed-ups computations dramatically. Additionally, most of the modern mobile devices use ARM® architecture which supports NEON™[36] instruction set. It can further decrease computational time up to a factor of 4 for computationally demanding parts. Nevertheless, further computational improvements are beyond our main scope (adaptive approach) and may be considered in the future work.

Algorithm	Average PSNR (dB)		
	σ=20	σ=40	σ=60
BM3D	28.94	25.59	24.00
Content-dependent BM3D	29.03	25.65	24.09

Table 5.3: Comparison of quality performance on filtering of Berkley Segmentation Dataset.

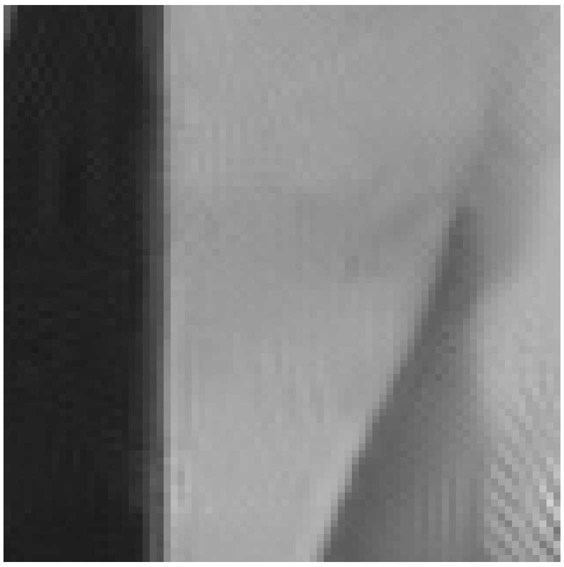
	σ=20	σ=40	σ=60
Speed-up	1.7292	2.8750	3.6800

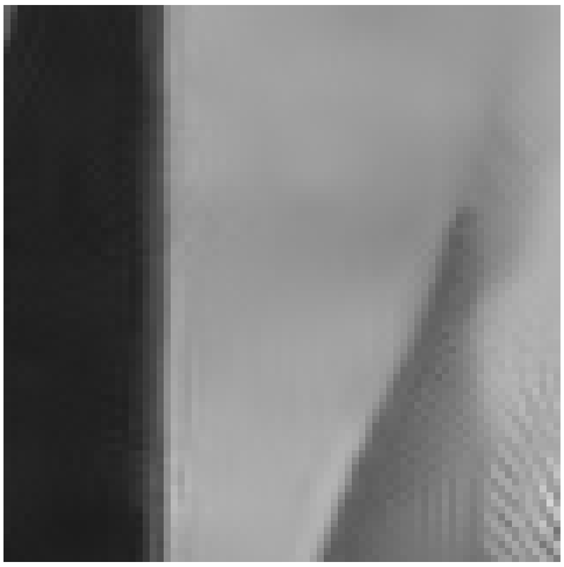
Table 5.4: Average speed-up of content-dependent BM3D compared to BM3D.

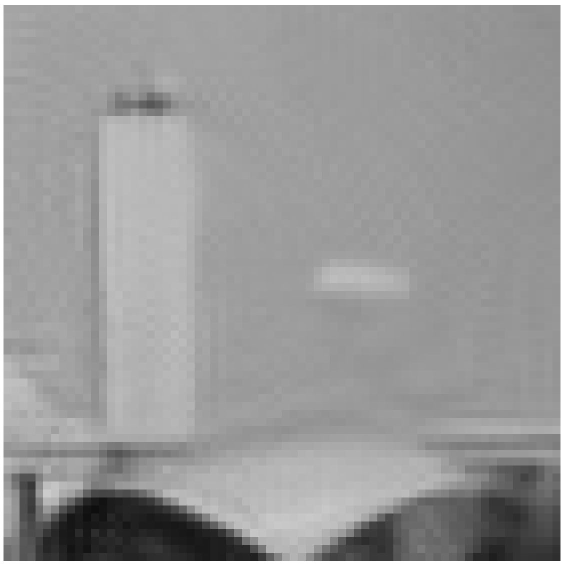
5.3 Testing

All the previous comparison results between BM3D and Adaptive BM3D were made only on small set of test images. In order to understand the real changes in quality and time consumption we have performed experiments on a large set of real-life images. Test images were taken from Berkeley Segmentation Dataset [35] (300 images). All the images are of the same size (481 $\times$ 321 pixels), since that one can perform averaging in terms of time and MSE to calculate mean values among all images. Each image was filtered independently, however, in addition to individual time and MSE we also calculated average values.

Comparison of average results which were obtained from filtering of Berkley Segmentation Dataset degraded with different noise levels is shown in Tables 5.3 and 5.4. We can see that in average on all noise levels content-dependent BM3D yields images with better PSNR than simple BM3D. Additionally, content-dependent approach speeds-ups filtering. With the growth of noise level speed-up is larger. Thus, content-based approach helps us to achieve better filtering quality and at the same time highly increase filtering speed.


(a) Barbara fragment. BM3D


(b) Barbara fragment. Proposed algorithm


(c) Cameraman fragment. BM3D

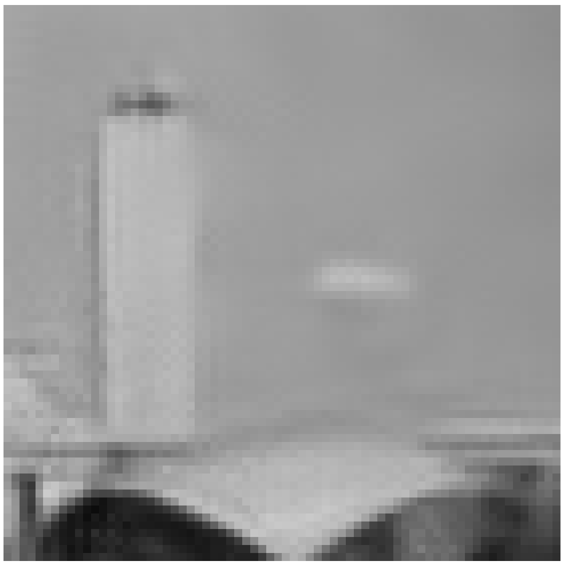

(d) Cameraman fragment. Proposed algorithm

Figure 5.7: Visual comparison between image filtered with BM3D and proposed algorithm.

6. CONCLUSION

In this work, we have shown an adaptive approach, that uses image classification and existing image denoising filters for more accurate and time-saving denoising.

At first, analysis of modern state-of-the-art filtering algorithms which utilize various techniques for image denoising was performed. The features of these algorithms and mathematical principles which are used to filter out obstructive part of the signal were studied. An image area classification method was introduced. Based on classification, performance of denoising techniques on different classes was investigated and various comparisons were made. For each class desired filtering algorithms properties were determined.

Afterwards, several algorithm optimizations and simplifications were tested. Computational and quality performance results obtained from these changes were compared individually for each image class. Useful modifications were taken into account for the implementation. A modified block-based image classification tool was developed, which allows to save computations by embedding into filtering process.

Finally, a solution that uses image content classification to perform advanced filtering was presented. Several performance tests were made and the increase in both quality and computational performances compared to baseline filters was shown. Additionally, an integer solution which is more appropriate for mobile platforms with further development prospects related to practical implementations was proposed.

Therefore, we performed all the tasks stated at the beginning of this work. We developed competitive algorithm that outstrips its analogs in several parameters. However, we left various ways of possible improvements, thus, opened a niche for further study and developments.

A. MAXIMUM LIKELIHOOD SOLUTION

Let's assume that estimate from each block is unbiased: $\hat{y}_i(x) \sim \mathcal{N}(y(x), \sigma^2_{i,loc})$. The formula for local variance after thresholding is $\sigma^2_{i,loc} = \frac{N_{i,T} \cdot \sigma^2}{BS^2}$, where BS – side of the block size, $N_{i,T}$ – number of nonzero coefficients after thresholding. Log-likelihood:

$$L = \ln \prod_i (2\pi\sigma^2_{i,loc})^{-\frac{1}{2}} e^{-\frac{1}{2\sigma^2_{i,loc}}(\hat{y}_i(x)-y(x))^2} =$$

$$= -\frac{1}{2} \sum_i \frac{1}{\sigma^2_{i,loc}} (\hat{y}_i(x) - y(x))^2 + \ln 2\pi\sigma^2_{i,loc}$$

To maximize L we should solve $\frac{\delta L}{\delta y} = 0$ and we obtain maximum likelihood solution $\hat{y}^{ML}$:

$$0 = \sum_i \frac{1}{\sigma^2_{i,loc}} (\hat{y}_i(x) - \hat{y}^{ML}(x))^2$$

$$\hat{y}^{ML}(x) = \frac{\sum_i \frac{1}{\sigma^2_{i,loc}} \hat{y}_i(x)}{\sum_i \frac{1}{\sigma^2_{i,loc}}} = \frac{\sum_i \frac{1}{N_{i,T}} \hat{y}_i(x)}{\sum_i \frac{1}{N_{i,T}}}$$

B. INTEGER PSEUDO-DCT TRANSFORM

In this appendix we observe features of integer transforms which were used in this work. In our experiments we used pseudo-DCT of 4×4 and 8×8 block sizes. The matrices for these transforms were taken from the H.264/AVC standard [26]. Since these matrices are not orthonormal in H.264/AVC coefficients obtained by the forward transform are processed by 2-stage quantization-normalization process: in the transmitter and in the receiver. This process in accordance with its name is performed for two purposes:

1. Normalization of pseudo-DCT coefficients since forward and inverse transforms are not orthonormal.

2. Quantization of the coefficients to compress them for transport means.

In the H.264/AVC standard there are several quantization-normalization matrices characterized quantization parameter for different compression powers. All these pairs of matrices (one for transmitter and one for receiver) yield the approximately (because of the integer values) the same matrix after element-wise multiplication. The difference is that the larger the *quantization parameter* the harder the first matrix quantizes the coefficients. In our work we do not have the data transmission. Since that for normalization purpose we can use only one matrix which is the element-wise product of pair of quantization-normalization matrices. In order to demonstrate principles of this pseudo-DCT transform we will show examples for 4×4 and 8×8 case with matrices for $\textit{quantization parameter} = 0$.

B.1 Integer 4×4 pseudo-DCT

We start with 4×4 pseudo-DCT. Basic steps implemented in H.264/AVC are:

1. *Forward integer transform.* For 4×4 case each row and then each column of the block, that we denote as x is multiplied by matrix T[28], in other words: $X = T \cdot x \cdot T^{\top}$, where:

$$T = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & -1 & -2 \\ 1 & -1 & -1 & 1 \\ 1 & -2 & 2 & -1 \end{pmatrix}$$

2. *Forward pseudo-quantization.* For 4×4 case it means element-wise multiplication of block transformed by previous step by pseudo-quantization matrix $X_{Q_1} = X \oplus Q_1$, where:

$$Q_1 = \begin{pmatrix} 13107 & 8066 & 13107 & 8066 \\ 8066 & 5243 & 8066 & 5243 \\ 13107 & 8066 & 13107 & 8066 \\ 8066 & 5243 & 8066 & 5243 \end{pmatrix} \cdot \frac{1}{2^{15}}$$

Multiplication by $\frac{1}{2^15}$ is performed by right shift.

3. *Inverse pseudo-quantization.* For 4×4 case it means element-wise multiplication of block transformed by previous step by pseudo-quantization matrix $X_{Q_2} = X_{Q_1} \oplus Q_2$, where:

$$Q_2 = \begin{pmatrix} 10 & 13 & 10 & 13 \\ 13 & 16 & 13 & 16 \\ 10 & 13 & 10 & 13 \\ 13 & 16 & 13 & 16 \end{pmatrix} \cdot \frac{1}{2^4}$$

4. *Inverse integer transform.* For 4×4 case each row and then each column of the block, that we denote as x is multiplied by matrix H, in other words: $x = H \cdot X \cdot H^\top$, where:

$$H = \begin{pmatrix} 1 & 1 & 1 & \frac{1}{2} \\ 1 & \frac{1}{2} & -1 & -1 \\ 1 & -\frac{1}{2} & -1 & 1 \\ 1 & -1 & 1 & -\frac{1}{2} \end{pmatrix}$$

As we have no need in implementation forward and inverse normalizations-quantizations separately, forward and invcerse normalization-quantization matrices were replaced with one pseudo-normalization matrix. Its coefficients were recalculated in a following way: $Q_3 = Q_2 \oplus Q_1$. Thus:

$$Q_3 = \begin{pmatrix} 131072 & 104858 & 131072 & 104858 \\ 104858 & 83886 & 104858 & 83886 \\ 131072 & 104858 & 131072 & 104858 \\ 104858 & 83886 & 104858 & 83886 \end{pmatrix} \cdot \frac{1}{2^{19}}$$

This merge not only decreases amount of multiplications but also slightly increases accuracy (one inaccurate operation instead of two).

Thresholding step that is necessary for filtering is done not after quantization but before it, therefore for all the coefficients with zero value after thresholding we do not need to do quantization and therefore we save computational time. Of course we need to have different thresholds for different coefficients as quantization operation is not uniform, but this task is easily solved by storing precalculated matrix of thresholds, where each coefficient is calculated as:

$$Thres[i,j] = 2.7 \cdot \sigma \cdot Q[i,j]$$

Where σ is noise standard deviation and Q is a matrix that performs normalization of transform T:

$$Q = \begin{pmatrix} 2 \\ 2\sqrt{2.5} \\ 2 \\ 2\sqrt{2.5} \end{pmatrix} \cdot \begin{pmatrix} 2 & 2\sqrt{2.5} & 2 & 2\sqrt{2.5} \end{pmatrix} = \begin{pmatrix} 4 & 4\sqrt{2.5} & 4 & 4\sqrt{2.5} \\ 4\sqrt{2.5} & 10 & 4\sqrt{2.5} & 10 \\ 4 & 4\sqrt{2.5} & 4 & 4\sqrt{2.5} \\ 4\sqrt{2.5} & 10 & 4\sqrt{2.5} & 10 \end{pmatrix}$$

Matrix Q is calculated from the product of column and row, those are formed from the reciprocals of the coefficients which are to be used to normalize rows of matrix T.

B.2 Integer 8×8 pseudo-DCT

Integer 8×8 DCT was taken from H.264/AVC standard as well as 4×4 one. Nearly the same procedures of simplification were made except that numerators and denominators of quantization coefficients were divided by 2, as otherwise they would cause 32-bit integer bit overflow. As in 4×4 case forward transform is simply multiplication of each row and then column of the block by matrix: $X = T \cdot x \cdot T^{\top}$, where matrix T:

$$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ \frac{3}{2} & \frac{5}{4} & \frac{3}{4} & \frac{3}{8} & -\frac{3}{8} & -\frac{5}{4} & -\frac{5}{4} & -\frac{3}{2} \\ 1 & \frac{1}{2} & -\frac{1}{2} & -1 & -1 & -\frac{1}{2} & \frac{1}{2} & 1 \\ \frac{5}{4} & -\frac{3}{8} & -\frac{3}{2} & -\frac{3}{4} & \frac{3}{4} & \frac{3}{2} & \frac{3}{8} & -\frac{5}{4} \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ \frac{3}{4} & -\frac{3}{2} & \frac{3}{8} & \frac{5}{4} & -\frac{5}{4} & -\frac{3}{8} & \frac{3}{2} & -\frac{3}{4} \\ \frac{1}{2} & -1 & 1 & -\frac{1}{2} & -\frac{1}{2} & 1 & -1 & \frac{1}{2} \\ \frac{3}{8} & -\frac{3}{4} & \frac{5}{4} & -\frac{3}{2} & \frac{3}{2} & -\frac{5}{4} & \frac{3}{4} & -\frac{3}{8} \end{pmatrix}$$

Inverse transform is again multiplication of each row and then column of the block by matrix: $x = H \cdot X \cdot H^\top$, where matrix H:

$$\begin{pmatrix}
1 & \frac{3}{2} & 1 & \frac{5}{4} & 1 & \frac{3}{4} & \frac{1}{2} & \frac{3}{8} \\
1 & \frac{5}{4} & \frac{1}{2} & -\frac{3}{8} & -1 & -\frac{3}{2} & -1 & -\frac{3}{4} \\
1 & \frac{3}{4} & -\frac{1}{2} & -\frac{3}{2} & -1 & \frac{3}{8} & 1 & \frac{5}{4} \\
1 & \frac{3}{8} & -1 & -\frac{3}{4} & 1 & \frac{5}{4} & -\frac{1}{2} & -\frac{3}{2} \\
1 & -\frac{3}{8} & -1 & \frac{3}{4} & 1 & -\frac{5}{4} & -\frac{1}{2} & \frac{3}{2} \\
1 & -\frac{3}{4} & -\frac{1}{2} & \frac{3}{2} & -1 & -\frac{3}{8} & 1 & -\frac{5}{4} \\
1 & -\frac{5}{4} & \frac{1}{2} & \frac{3}{8} & -1 & \frac{3}{2} & -1 & \frac{3}{4} \\
1 & -\frac{3}{2} & 1 & -\frac{5}{4} & 1 & -\frac{3}{4} & \frac{1}{2} & -\frac{3}{8}
\end{pmatrix}$$

Original pseudo-quantization matrices Q_1 and Q_2 are:

$$\begin{pmatrix}
13107 & 12222 & 16777 & 12222 & 13107 & 12222 & 16777 & 12222 \\
12222 & 11428 & 15481 & 11428 & 12222 & 11428 & 15481 & 11428 \\
16777 & 15481 & 20972 & 15481 & 16777 & 15481 & 20972 & 15481 \\
12222 & 11428 & 15481 & 11428 & 12222 & 11428 & 15481 & 11428 \\
13107 & 12222 & 16777 & 12222 & 13107 & 12222 & 16777 & 12222 \\
12222 & 11428 & 15481 & 11428 & 12222 & 11428 & 15481 & 11428 \\
16777 & 15481 & 20972 & 15481 & 16777 & 15481 & 20972 & 15481 \\
12222 & 11428 & 15481 & 11428 & 12222 & 11428 & 15481 & 11428
\end{pmatrix} \cdot \frac{1}{2^{16}}$$

$$\begin{pmatrix}
20 & 19 & 25 & 19 & 20 & 19 & 25 & 19 \\
19 & 18 & 24 & 18 & 19 & 18 & 24 & 18 \\
25 & 24 & 32 & 24 & 25 & 24 & 32 & 24 \\
19 & 18 & 24 & 18 & 19 & 18 & 24 & 18 \\
20 & 19 & 25 & 19 & 20 & 19 & 25 & 19 \\
19 & 18 & 24 & 18 & 19 & 18 & 24 & 18 \\
25 & 24 & 32 & 24 & 25 & 24 & 32 & 24 \\
19 & 18 & 24 & 18 & 19 & 18 & 24 & 18
\end{pmatrix} \cdot \frac{1}{2^{6}}$$

And recalculated matrix for pseudo-quantization is $Q_3 = Q_2 \cdot Q_1$:

$$\begin{pmatrix} 131070 & 116109 & 209713 & 116109 & 131070 & 116109 & 209713 & 116109 \\ 116109 & 102852 & 185772 & 102852 & 116109 & 102852 & 185772 & 102852 \\ 209713 & 185772 & 335552 & 185772 & 209712 & 185772 & 335552 & 185772 \\ 116109 & 102852 & 185772 & 102852 & 116109 & 102852 & 185772 & 102852 \\ 131070 & 116109 & 209712 & 116109 & 131070 & 116109 & 209713 & 116109 \\ 116109 & 102852 & 185772 & 102852 & 116109 & 102852 & 185772 & 102852 \\ 209713 & 185772 & 335552 & 185772 & 209713 & 185772 & 335552 & 185772 \\ 116109 & 102852 & 185772 & 102852 & 116109 & 102852 & 185772 & 102852 \end{pmatrix} \cdot \frac{1}{2^{21}}$$

And thresholding is performed as in previous 4×4 case:

$$Thres[i,j] = 2.7 \cdot \sigma \cdot Q[i,j],$$

where

$$Q = \begin{pmatrix} 2\sqrt{2} \\ \sqrt{9.03125} \\ \sqrt{5} \\ \sqrt{9.03125} \\ 2\sqrt{2} \\ \sqrt{9.03125} \\ \sqrt{5} \\ \sqrt{9.03125} \end{pmatrix} \cdot \begin{pmatrix} 2\sqrt{2} & \sqrt{9.03125} & \sqrt{5} & \sqrt{9.03125} & 2\sqrt{2} & \sqrt{9.03125} & \sqrt{5} & \sqrt{9.03125} \end{pmatrix} =$$

$$= \begin{pmatrix} 8 & \sqrt{72.25} & \sqrt{40} & \sqrt{72.25} & 8 & \sqrt{72.25} & \sqrt{40} & \sqrt{72.25} \\ \sqrt{72.25} & 9.03125 & \sqrt{45.15625} & 9.03125 & \sqrt{72.25} & 9.03125 & \sqrt{45.15625} & 9.03125 \\ \sqrt{40} & \sqrt{45.15625} & 5 & \sqrt{45.15625} & \sqrt{40} & \sqrt{45.15625} & 5 & \sqrt{45.15625} \\ \sqrt{72.25} & 9.03125 & \sqrt{45.15625} & 9.03125 & \sqrt{72.25} & 9.03125 & \sqrt{45.15625} & 9.03125 \\ 8 & \sqrt{72.25} & \sqrt{40} & \sqrt{72.25} & 8 & \sqrt{72.25} & \sqrt{40} & \sqrt{72.25} \\ \sqrt{72.25} & 9.03125 & \sqrt{45.15625} & 9.03125 & \sqrt{72.25} & 9.03125 & \sqrt{45.15625} & 9.03125 \\ \sqrt{40} & \sqrt{45.15625} & 5 & \sqrt{45.15625} & \sqrt{40} & \sqrt{45.15625} & 5 & \sqrt{45.15625} \\ \sqrt{72.25} & 9.03125 & \sqrt{45.15625} & 9.03125 & \sqrt{72.25} & 9.03125 & \sqrt{45.15625} & 9.03125 \end{pmatrix} \tag{B.1}$$

All the threshold matrices can be computed from Q matrices before filtering knowing σ only once and then can be used in each block thresholding step. Additionally, in practice, multiplication by pseudo-transform matrices is implemeneted only with the help of sum and shift operations, which are simple and fast operations.

BIBLIOGRAPHY

[1] Charles Boncelet, Alan C. Bovik, “Image Noise Models”, *Handbook of Image and Video Processing. Academic Press. 2005.*

[2] F. Luisier, Blu, T., and Unser, M., “Image Denoising in Mixed Poisson-Gaussian Noise”, *IEEE Transactions on Image Processing, vol. 20, no. 3, p. 696-708, 2011.*

[3] Karen Egiazarian, Vladimir Katkovnik and Jaakko Astola, “Local transform-based image de-noising with adaptive window size selection”, *Proceedings of SPIE Vol. 4170 (2001) 2001 SPIE*

[4] Charles Kervrann and Jérôme Boulanger, “Optimal Spatial Adaptation for Patch-Based Image Denoising”, *IEEE Transactions on Image Processing, vol. 15, no. 10, October 2006*

[5] Rafael C. Gonzalez and Richard E. Woods, “Digital Image Processing”, *Pearson Education, second edition, 2000.*

[6] Alessandro Foi, “Anisotropic nonparametric image processing: Theory, algorithms and applications”, *Tesi, Dipartimento di Matematica, Politecnico di Milano, Italy, 2005.*

[7] S. Mallat, “A wavelet tour of signal processing, second edition”, *Academic Press, New York, 1999.*

[8] V. Katkovnik, A. Foi, K. Egiazarian and J. Astola, “Directional varying scale approximations for anisotropic signal processing”, *Proceedings of XII European Signal Processing Conference, EUSIPCO 2004, Vienna, Austria, 6-10 Sept. 2004, 101-104.*

[9] Tong, H.H.Y. and Venetsanopoulos, A.N., “A perceptual model for JPEG applications based on block classification, texture masking, and luminance masking”, *ICIP 1998, Chicago, IL, USA, 4-7 Oct 1998, pp.428-432.*

[10] Chee Sun Wont, Kyungsuk Pyun, and Robert M. Gray, “Automatic object segmentation in images with low depth of field”, *IEEE Proc. Of Image Processing, vol III, pp.805-808, 2002.*

[11] Breiman, L., Friedman, J. H., Olshen, R. A., Stone, C. J. (1984), “Classification and regression trees”, *Monterey, CA: Wadsworth & Brooks/Cole Advanced Books & Software.*

[12] Katkovnik, V., K. Egiazarian, and J. Astola, "Adaptive window size image denoising based on intersection of confidence intervals (ICI) rule", *J. of Math. Imaging and Vision, vol. 16, no. 3, pp. 223-235, 2002.*

[13] Katkovnik, V., "A new method for varying adaptive bandwidth selection", *IEEE Trans. on Signal Proc., vol. 47, issue 9, pp. 2567-2571, 1999.*

[14] Alessandro Foi, Vladimir Katkovnik, Karen Egiazarian and Jaakko Astola "A novel anisotropic local polynomial estimator based on directional multiscale optimizations", *Proc. 6th Int. Conf. Mathematics in Signal Processing, Cirencester, UK, pp. 79-82, 2004.*

[15] K . R . Rao and P . C . Yip, "The Transform and Data Compression Handbook", *Boca Raton, CRC Press LLC, 2001.*

[16] R. Öktem, L. Yaroslavsky, K. Egiazarian and J.Astola, "Transform domain image restoration methods: review, comparison and interpretation", *TICSP, 9, pp. 15, December 2000.*

[17] R. Öktem, L. Yaroslavsky and K. Egiazarian, "Signal and Image Denoising in Transform Domain and Wavelet Shrinkage: A Comparative Study", *Eusipco'98, Signal Processing IX, Theories and Applications, Island of Rhodes, Greece, 4, pp. 2269-2272, 8-11 September 1998.*

[18] R. Öktem, K. Egiazarian, V. V. Lukin, N. N. Ponomarenko, O. V. Tsymbal, "Locally Adaptive DCT Filtering for Signal-Dependent Noise Removal", *EURASIP Journal on Advances in Signal Processing, Vol. 2007, Article ID 42472, 10 p.*

[19] Christoph Loeffler, Adriaan Lieenberg, and George S. Moschytz, "Practical dast 1-D DCT algorithms with 11 multiplications", *Acoustics, Speech, and Signal Processing, 1989. ICASSP-89, 1989. pp 988-991.*

[20] F. Jin, P. Fieguth, L. Winger and E. Jernigan, "Adaptive wiener filtering of noisy images and image sequences", *ICIP 2003, vol. 3, pp 349-52.*

[21] Hyeokho Choi and Richard Baraniuk, "Analysis of wavelet-domain wiener filters", *Proceedings of SPIE, San Diego, 1997.*

[22] Foi, A., "Pointwise shape-adaptive DCT image filtering and signal-dependent noise estimation", *Tampere University of Technology, Publication 710, ISBN 978-952-15-1922-2, December 2007.*

[23] Alessandro Fio, Kostadin Dabov, Vladimir Katkovnik, Karen Egiazarian, "Shape-Adaptive DCT for Denoising and Image Reconstruction", *Proc. SPIE*

Electronic Imaging 2006, Image Processing: Algorithms and System V, San Jose, 2006.

[24] Alessandro Foi, Vladimir Katkovnik and Karen Egiazarian, "Pointwise shape-adaptive DCT as an overcomplete denoising tool", *In The International Workshop on Spectral Methods and Multirate Signal Processing, number 5, June 2005.*

[25] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian, "Image denoising with block-matching and 3D filterin", *Proc. SPIE Electronic Imaging '06, no. 6064A-30, San Jose, California, USA, January 2006.*

[26] ISO/IEC 14496-10 and ITU-T Rec. H.264, Advanced Video Coding, 2003.

[27] ISO/IEC 10918-1 and ITU-T Rec. T.81, Coding of still pictures, 1992.

[28] Iain E. G. Richardson, "H.264 and MPEG-4 Video Compression: Video Coding for Next-generation Multimedia", *John Wiley and Sons Publisher, 2003.*

[29] Jaswant R. Jain, Anil K. Jain, "Displacement measurement and its application in interframe image coding", *IEEE Transactions on Communications, Volume COM-29, Number 12, pp 1799-1808, December 1981.*

[30] Shan Zhu and Kai-Kuang Ma, "A new diamond search algorithm for fast block matching motion estimation", *Proceedings of International Conference on Communications and Signal Processing, 1:292-296, 1997.*

[31] Ce Zhu, Xiao Lin and Lap-Pui Chau, "Hexagon-based search pattern for fast block motion estimation", *IEEE Transactions on Circuits and Systems for Video Technology, 13(7):614-619, July 2003.*

[32] Junsheng Fu, "A real-time rate-distortion oriented joint video denoising and compression algorithm", *Master of Science Thesis, Tampere University of Technology, 2011.*

[33] Peter Kauff and Klaas Schüür, "An Extension of Shape-Adaptive DCT (SA-DCT) Towards DC Separation and ΔDC Correction", *Proc. of 1997 Picture Coding Symposium, pp. 647-652, Sep. 1997.*

[34] Marc Lebrun, "An Analysis and Implementation of the BM3D Image Denoising Method", *Image Processing On Line 2012 (2012).*

[35] `http://www.eecs.berkeley.edu/Research/Projects/CS/vision/bsds/BSDS300/html/images/plain/normal/gray/`

[36] `http://www.arm.com/products/processors/technologies/neon.php`