information

MDPI

Article

Quantitative Evaluation of Dense Skeletons for Image Compression

Jieying Wang [1,*], Maarten Terpstra [1], Jiří Kosinka [1] and Alexandru Telea [2]

[1] Bernoulli Institute, University of Groningen, 9747 AG Groningen, The Netherlands; maartenlterpstra@gmail.com (M.T.); j.kosinka@rug.nl (J.K.)

[2] Department of Information and Computing Sciences, Utrecht University, 3584 CC Utrecht, The Netherlands; a.c.telea@uu.nl

* Correspondence: jieying.wang@rug.nl

Received: 15 April 2020; Accepted: 15 May 2020; Published: 20 May 2020

Abstract: Skeletons are well-known descriptors used for analysis and processing of 2D binary images. Recently, dense skeletons have been proposed as an extension of classical skeletons as a dual encoding for 2D grayscale and color images. Yet, their encoding power, measured by the quality and size of the encoded image, and how these metrics depend on selected encoding parameters, has not been formally evaluated. In this paper, we fill this gap with two main contributions. First, we improve the encoding power of dense skeletons by effective layer selection heuristics, a refined skeleton pixel-chain encoding, and a postprocessing compression scheme. Secondly, we propose a benchmark to assess the encoding power of dense skeletons for a wide set of natural and synthetic color and grayscale images. We use this benchmark to derive optimal parameters for dense skeletons. Our method, called Compressing Dense Medial Descriptors (CDMD), achieves higher-compression ratios at similar quality to the well-known JPEG technique and, thereby, shows that skeletons can be an interesting option for lossy image encoding.

Keywords: medial descriptors; skeletonization; image compression; benchmarking

1. Introduction

Images are created, saved and manipulated every day, which calls for effective ways to compress such data. Many image compression methods exist [1], such as the well-known discrete cosine transform and related mechanisms used by JPEG [2]. On the other hand, binary shapes also play a key role in applications such as optical character recognition, computer vision, geometric modeling, and shape analysis, matching, and retrieval [3]. Skeletons, also called medial axes, are well-known descriptors that allow one to represent, analyze, but also simplify such shapes [4–6]. As such, skeletons and image compression methods share some related goals: a compact representation of binary shapes and continuous images, respectively.

Recently, Dense Medial Descriptors (DMD) have been proposed as an extension of classical binary-image skeletons to allow the representation of grayscale and color images [7]. DMD extracts binary skeletons from all threshold sets (luminance, hue, and/or saturation layers) of an input image and allows the image to be reconstructed from these skeletons. By simplifying such skeletons and/or selecting a subset of layers, DMD effectively acts as a dual (lossy) image representation method. While DMD was applied for image segmentation, small-scale detail removal, and artistic modification [7–9], it has not been used for image compression. More generally, to our knowledge, skeletons have never been used so far for lossy compression of grayscale or color images.

In this paper, we exploit the simplification power of DMD for image compression, with two contributions. First, we propose Compressing Dense Medial Descriptors (CDMD), an adaptation of

DMD for lossy image compression, by searching for redundant information that can be eliminated, and also by proposing better encoding and compression schemes for the skeletal information. Secondly, we develop a benchmark with both natural and synthetic images, and use it to evaluate our method to answer the following questions:

- What kinds of images does CDMD perform on best?
- What is CDMD's trade-off between reconstructed quality and compression ratio?
- Which parameter values give best quality and/or compression for a given image type?
- How does CDMD compression compare with JPEG?

The joint answers to these questions, which we discuss in this paper, show that CDMD is an effective tool for both color and grayscale image compression, thereby showing that medial descriptors are an interesting tool to consider, and next refine, for this task.

The remainder of the paper is organized as follows. Section 2 introduces DMD, medial descriptors, and image quality metrics. Section 3 details our proposed modifications to DMD. Section 4 describes our evaluation benchmark and obtained results. Section 5 discusses our results. Finally, Section 6 concludes the paper.

2. Related work

2.1. Medial Descriptors and the DMD Method

We first introduce the DMD method (see Figure 1). To ease presentation, we consider only grayscale images here. However, DMD can also handle color images by considering each of the three components of an Luv or RGB space in turn (see next Section 4). Let $I : \mathbb{R}^2 \to [0, 255]$ be an 8-bit grayscale image.

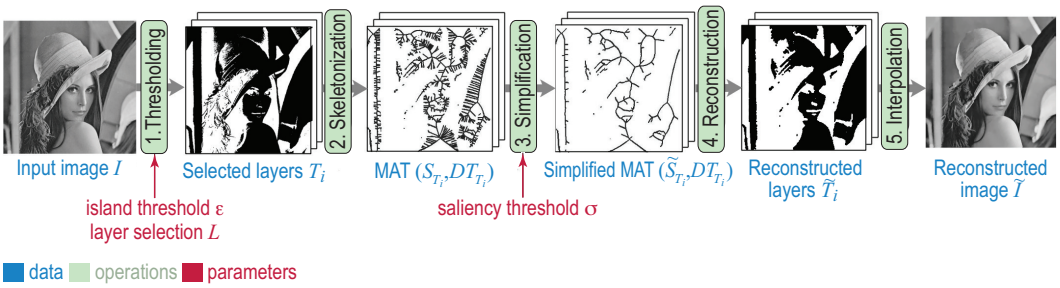

Figure 1. Dense medial descriptor (DMD) computation pipeline.

The key idea of DMD is to use 2D skeletons to efficiently encode isoluminant structures in an image. Skeletons can only be computed for binary shapes, so I is first reduced to n (256 for 8-bit images) threshold sets (see Figure 1, step 1) defined as

$$T_i = \left\{ \mathbf{x} \in \mathbb{R}^2 \mid I(\mathbf{x}) \geq i \right\}, \quad 0 \leq i \leq n-1. \tag{1}$$

Next, a binary skeleton is extracted from each T_i. Skeletons, or medial axes, are well-known shape descriptors, defined as the locus of centers of maximal disks contained in a shape [10–12]. Formally, for a binary shape $\Omega \in \mathbb{R}^2$ with boundary $\partial\Omega$, let

$$DT_\Omega(\mathbf{x} \in \Omega) = \min_{\mathbf{y} \in \partial\Omega} \|\mathbf{x} - \mathbf{y}\| \tag{2}$$

be its distance transform. The skeleton S_Ω of Ω is defined as

$$S_\Omega = \{\mathbf{x} \in \Omega \mid \exists \mathbf{f}_1, \mathbf{f}_2 \in \partial\Omega, \mathbf{f}_1 \neq \mathbf{f}_2 \colon \|\mathbf{f}_1 - \mathbf{x}\| = \|\mathbf{f}_2 - \mathbf{x}\| = DT_\Omega(\mathbf{x})\}, \tag{3}$$

where $\mathbf{f}_1$ and $\mathbf{f}_2$ are the so-called feature points of skeletal point $\mathbf{x}$ [13]. The pair (S_Ω, DT_Ω), called the Medial Axis Transform (MAT), allows an exact reconstruction of Ω as the union of disks centered at $\mathbf{x} \in S_\Omega$ having radii $DT_\Omega(\mathbf{x})$. The output of DMD's second step is hence a set of n MATs (S_{T_i}, DT_{T_i}) for all the layers T_i (Figure 1, step 2). For a full discussion of skeletons and MATs, we refer to [4].

Computing skeletons of binary images is notoriously unstable and complex [4,5]. They contain many so-called spurious branches caused by small perturbations along $\partial\Omega$. Regularization eliminates such spurious branches which, in general, do not capture useful information. Among the many regularization methods, so-called collapsed boundary length ones are very effective in terms of stability, ease of use, and intuitiveness of parameter setting [14–17]. These compute simplified skeletons $\tilde{S}$ by removing from S all points $\mathbf{x}$ whose feature points subtend a boundary fragment of length ρ shorter than a user-given threshold ρ_{min}. This replaces all details along $\partial\Omega$ which are shorter than ρ_{min} by circular arcs. However, this 'rounds off' salient (i.e., sharp and large-scale) shape corners, which is perceptually undesirable. A perceptually better regularization method [13] replaces ρ by

$$\sigma(\mathbf{x}) = \rho(\mathbf{x}) / DT_\Omega(\mathbf{x}). \tag{4}$$

Skeleton points with σ below a user-defined threshold τ are discarded, thereby disconnecting spurious skeletal branches from the skeleton rump. The final regularized $\tilde{S}$ is then the largest connected component in the thresholded skeleton. Note that Equation (4) defines a saliency metric on the skeleton, which is different from existing saliency metrics on the image, e.g., [18,19].

Regularized skeletons and their corresponding MATs can be efficiently computed on the CPU [17] or on the GPU [7]. GPU methods can skeletonize images up to 1024^2 pixel resolution in a few milliseconds, allowing for high-throughput image processing applications [8,20] and interactive applications [21]. A full implementation of our GPU regularized skeletons is available [22].

The third step of DMD (see Figure 1) is to compute a so-called regularized MAT for each layer T_i, defined as $MAT_i = (\tilde{S}_{T_i}, DT_{T_i})$. Using each such MAT, one can reconstruct a simplified version $\tilde{T}_i$ of each layer T_i (Figure 1, step 4). Finally, a simplified version $\tilde{I}$ of the input image I is reconstructed by drawing the reconstructed layers $\tilde{T}_i$ atop each other, in increasing order of luminance i, and performing bilinear interpolation between them to remove banding artifacts (Figure 1, step 5). For further details, including implementation of DMD, we refer to [7].

2.2. Image Simplification Parameters

DMD parameterizes the threshold-set extraction and skeletonization steps (Section 2.1) to achieve several image simplification effects, such as segmentation, small-scale detail removal, and artistic image manipulation [7–9]. We further discuss the roles of these parameters, as they crucially affect DMD's suitability for image compression, which we analyze next in Sections 3–5.

Island removal: During threshold-set extraction, islands (connected components in the image foreground T_i or background $\overline{T}_i$) smaller than a fraction ϵ of $|T_i|$, respectively $|\overline{T}_i|$, are filled in, respectively removed. Higher ϵ values yield layers T_i having fewer small-scale holes and/or disconnected components. This creates simpler skeletons S_{T_i} which lead to better image compression. However, too high ϵ values will lead to oversimplified images.

Layer selection: As noted in [7], one does not need all layers T_i to obtain a perceptually good reconstruction $\tilde{I}$ of the input I. Selecting a small layer subset of $L < n$ layers from the n available ones leads to less information needed to represent $\tilde{I}$, so better compression. Yet, too few layers and/or suboptimal selection of these degrades the quality of $\tilde{I}$. We study how many (and which) layers are needed for a good reconstruction quality in Section 3.1.

Skeleton regularization: The intuition behind saliency regularization (Equation (4)) follows a similar argument as for layer selection: One can obtain a perceptually good reconstruction $\tilde{I}$, using less information, by only keeping skeletal branches above a certain saliency τ. Yet, how the choice of

τ affects reconstruction quality has not been investigated, neither in the original paper proposing saliency regularization [13] nor by DMD. We study this relationship in Section 4.

2.3. Image Compression Quality Metrics

Given an image I and its compressed version $\tilde{I}$, a quality metric $q(I, \tilde{I}) \in \mathbb{R}^+$ measures how perceptually close $\tilde{I}$ is to I. Widely used choices include the mean squared error (MSE) and peak signal-to-noise ratio (PSNR). While simple to compute and having clear physical meanings, they tend not to match perceived visual quality [23]. The structural similarity (SSIM) index [24] alleviates this by measuring, pixel-wise, how similar two images are by considering quality as perceived by humans. The mean SSIM (MSSIM) is a real-valued quality index that aggregates SSIM by averaging over all image pixels. MSSIM was extended to three-component SSIM (3-SSIM) by applying non-uniform weights to the SSIM map over three different region types: edges, texture, and smooth areas [25]. Multiscale SSIM (MS-SSIM) [26] is an advanced top-down interpretation of how the human visual system interprets images. MS-SSIM provides more flexibility than SSIM by considering variations of image resolution and viewing conditions. As MS-SSIM outperforms the best single-scale SSIM model [26], we consider it next in our work.

2.4. Image Compression Methods

Many image compression methods have been proposed in the literature, with a more recent focus on compressing special types of images, e.g., brain or satellite [1,27]. Recently, deep learning methods have gained popularity showing very high (lossy) compression rates and good quality, usually measured via PSNR and/or MS-SSIM [28–32]. However, such approaches require significant training data and training computational effort and can react in hard to predict ways to unseen data (images that are far from the types present during training). Our method, described next, does not aim to compete with the compression rates of deep learning techniques. However, its explicit 'feature engineering' approach offers more control to how images are simplified during compression, is fast, and does not require training data. Separately, technique-wise, our contribution shows, for the first time, that medial descriptors are a useful and usable tool for image compression.

Saliency metrics have become increasingly interesting in image compression [33,34]. Such metrics capture zones in an image deemed to be more important (salient) to humans into a so-called saliency map and use this to drive compression with high quality in those areas. Many saliency map computations methods exist, e.g., [35–38]; for a good survey thereof, we refer to [34]. While conceptually related, our approach is technically different, since (1) we compute saliency based on binary skeletons (Equation (4)); (2) our saliency thresholding (computation of $\tilde{S}$, Section 2.1) both detects salient image areas and simplifies the non-salient ones; and (3) as explained earlier, we use binary skeletons for this rather than analyzing the grayscale or color images themselves.

3. Proposed Compression Method

Our proposed Compressing Dense Medial Skeletons (CDMD) adapt the original DMD pipeline (Figure 1) to make it effective for image compression in two directions: layer selection (Section 3.1) and encoding the resulting MAT (Section 3.2), as follows.

3.1. Layer Selection

DMD selects a subset of $L < n$ layers T_i from the total set of n layers based on a simple greedy heuristic: Let $\tilde{I}_i$ be the reconstruction of image I using all layers, except T_i. The layer T_i yielding the smallest reconstruction error $\min_{1 \leq i \leq n} SSIM(I, \tilde{I}_i)$ is deemed the least relevant and thus first removed. The procedure is repeated over the remaining layers, until only L layers are left. This approach has two key downsides: Removing the least-relevant layer (for reconstruction) at a time does not guarantee that subsequent removals do not lead to poor quality. For an optimal result, one would have to maximize quality over all combinations of L (kept) layers selected from n, which is prohibitively

expensive. Secondly, this procedure is very expensive, as it requires $O((n-L)^2)$ reconstructions and image comparisons to be computed.

We improve layer selection by testing three new strategies, as follows.

Histogram thresholding: We compute a histogram of how many pixels each layer T_i individually encodes, i.e., $|T_i \setminus T_{i+1}|$. Next, we select all layers having values above a given threshold. To make this process easy, we do a layer-to-threshold conversion: given a number of layers L to keep, we find the corresponding threshold based on binary search.

Histogram local maxima: Histogram thresholding can discard layers containing small but visually important features such as highlights. Furthermore, all layers below the threshold are kept, which does not lead to optimal compression. We refine this by finding histogram local maxima (shown in Figure 2b for the test image in Figure 2a). The intuition here is that the human eye cannot distinguish subtle differences between adjacent (similar-luminance) layers [39], so, from all such layers, we can keep only the one contributing the most pixels to the reconstruction. As Figure 2c shows, 15 layers are enough for a good-quality reconstruction, also indicated by a high MS-SSIM score.

Cumulative histogram: We further improve layer selection by using a cumulative layer histogram (see Figure 2d for the image in Figure 2a). We scan this histogram left to right, comparing each layer T_i with layer $T_{j=i+m}$, where m is the minimally-perceivable luminance difference to a human eye (set empirically to 5 [39] on a luminance range of $[0, 255]$). If the histogram difference between layers T_i and T_j is smaller than a given threshold λ, we increase j until the difference is above λ. At that point, we select layer T_j and repeat the process until we reach the last layer. However, setting a suitable λ is not easy for inexperienced users. Therefore, we do a layer-to-threshold conversion by a binary search method, as follows. Let $[r_{min}, r_{max}]$ be the range of the cumulative histogram. At the beginning of the search, this range equals $[0, 1]$. We next set $\lambda = (r_{min} + r_{max})/2$ and compare the number of layers L' produced under this condition with the target, i.e. desired, user-given value L. If $L' = L$, then the search ends with the current value of λ. If $L' < L$, we continue the search in the lower half $[r_{min}, (r_{min} + r_{max})/2]$ of the current range. If $L' > L$, we continue the search in the upper half $[(r_{min} + r_{max})/2, r_{max}]$ of the current range. Since L is an integer value, the search may sometimes oscillate, yielding values L' that swing around, but do not precisely equal, the target L. To make the search end in such situations, we monitor the computed L' over subsequent iterations and, if oscillation, i.e., a non-monotonic evolution of the L' values over subsequent iterations, is detected, we stop the search and return the current λ. Through this conversion, what users need to set is only the desired number of layers, which makes it simple to use by any target group – much like setting the 'quality' parameter in typical JPEG compression. Compared to local maxima selection, the cumulative histogram method selects smoother transition layers, which yields a better visual effect. For example, in Figure 2c, the local details around the shoulder show clear banding effects; the same region is much smoother when cumulative histogram selection is used (Figure 2e). Besides improved quality, cumulative histogram selection is simpler to implement and use, as it does not require complex and/or sensitive heuristics for detecting local maxima.

Figure 3 compares the four layer selection methods discussed above. We test these on a 100-image database with 10 different image types, each having 10 images (see Table 1). The 10 types aim to capture general-purpose imagery (people, houses, scenery, animals, paintings) which are typically rich in details and textures; images having a clear structure, i.e., few textures, sharp contrasts, well-delineated shapes shapes (ArtDeco, cartoon, text); and synthetic images being somewhere between the previous two types (scientific visualization).

Average MS-SSIM scores show that the cumulative histogram selection yields the best results for all image types, closely followed by local maxima selection and next by the original greedy method in DMD. The naive histogram thresholding yields the poorest MS-SSIM scores, which also strongly depend on image type. Besides better quality, the cumulative histogram method is also dramatically faster, 3000 times more than the greedy selection method in [7]. Hence, cumulative histogram is our method of choice for layer selection for CDMD.

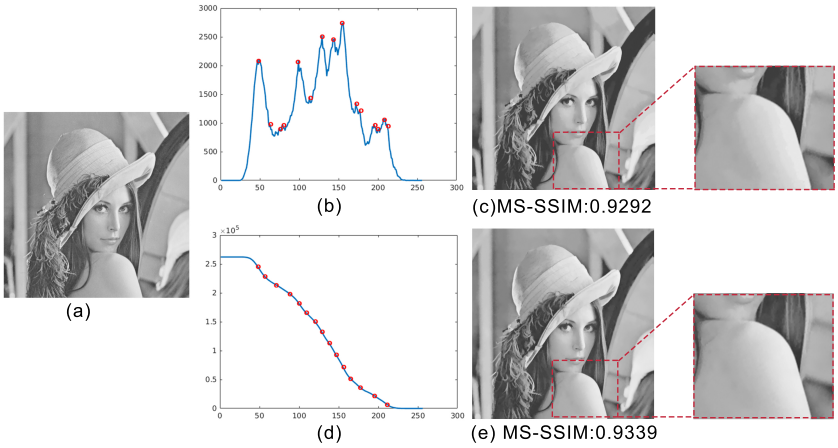

Figure 2. Layer selection methods. (**a**) Original image. (**b**) Histogram of (**a**), with local maxima marked in red. (**c**) Reconstruction of (**a**) using 15 most relevant layers given by (**b**). (**d**) Cumulative histogram of (**a**), with selected layers marked red. (**e**) Reconstruction of (**a**) using the 15 most relevant layers given by (**d**).

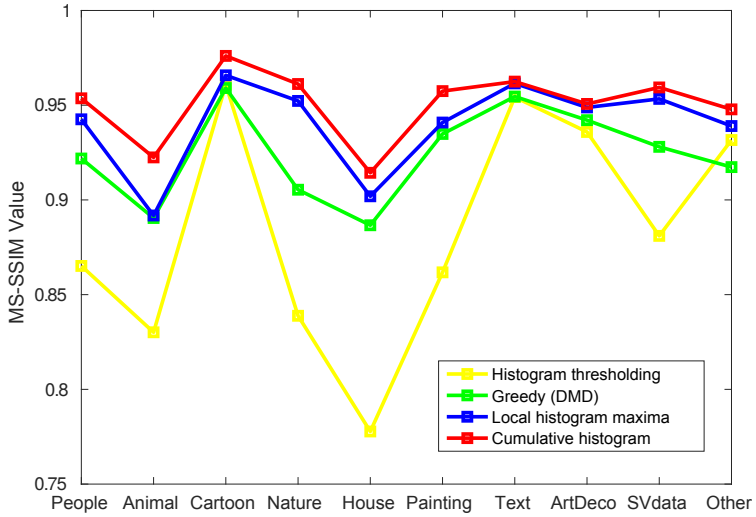

Figure 3. Average MS-SSIM scores for four layer selection methods (30 layers selected) for images in ten different classes. The cumulative histogram method performs the best and is hence used in CDMD.

Table 1. The benchmark of 100 images (available at [40]) used throughout this work for testing CDMD.

Type	Description
animal	Wild animals in their natural habitat
artDeco	Art deco artistic images
cartoon	Cartoons and comic strips
house	Residential homes surrounded by greenery
nature	Panorama landscapes and close-ins of plants
other	Miscellaneous (fruit, planets, natural scenery)
painting	Classical and modern paintings
people	Portrait photos of various people
SVdata	Scientific visualizations (scalar and vector fields)
text	Typography of various styles and scales

3.2. MAT Encoding

MAT computation (Section 2.1) delivers, for each selected layer T_i, pairs of skeletal pixels $\mathbf{x}$ with corresponding inscribed circle radii $r = DT_{T_i}(\mathbf{x})$. Naively storing this data requires two 16-bit integer values for the two components of $\mathbf{x}$ and one 32-bit floating-point value for r, respectively. We propose next two strategies to compress this data losslessly.

Per-layer compression: As two neighbor pixels in a skeleton are 8-connected, their differences in x and y coordinates are limited to $\triangle x, \triangle y \in \{-1, 0, 1\}$, and similarly $\triangle r \in \{-2, -1, 0, 1, 2\}$. Hence, we visit all pixels in a depth-first manner [41] and encode, for each pixel, only the $\triangle x$, $\triangle y$, and $\triangle r$ values. We further compress this delta-representation of each MAT point by testing ten lossless encoding methods: Direct encoding (use one byte per MAT point in which $\triangle x$ and $\triangle y$ take up two bits each, and $\triangle r$ three bits, i.e., 0xxyyrrr); Huffman [42], Canonical Huffman, Unitary [43], Exponential Golomb, Arithmetic [44], Predictive, Compact, Raw, and Move-to-Front (MTF) [45]. To compare the effectiveness of these methods, we use the compression ratio of an image I defined as

$$CR(I) = \frac{|I|}{|MAT(\tilde{I}))|}, \tag{5}$$

where $|I|$ is the byte-size of the original image I and $|MAT(\tilde{I})|$ is the byte-size of the MAT encoding for all selected layers of $\tilde{I}$. Table 2 (top row) compares the 10 tested encoding methods, showing average $CR(I)$ value for the 10 image types in Figure 3, and 12 different combinations of parameters ϵ, L, and τ per compression-run. The highest value in each row is marked in bold.

Inter-layer compression: The inter-layer compression leaves, likely, still significant redundancy in the MATs of different layers. To remove this, we compress the MAT of all layers (each encoded using all 10 lossless methods discussed above) with eight lossless-compression algorithms: Lempel–Ziv–Markov Chain (LZMA) [46], LZHAM [47], Brotli [48], ZPAQ [49], BZip2 [50], LZMA2 [46], BSC [51], and ZLib [52], all available in the Squash library [53]. Figure 4 shows CR boxplots (Equation (5)) for all our 100 test images. Blue boxes show the 25–75% quantile; red lines are medians; black whiskers show extreme data points not considered outliers; outliers are shown by red '+' marks. Overall, ZPAQ is the best compression method, 20.15% better than LZMA, which was used in the original DMD method [7]. Hence, we select ZPAQ for CDMD.

Table 2. Comparison of average compression ratios (Equation (5)) for 10 lossless MAT-encoding methods on 100 images using only per-layer compression (top row) and inter-layer compression (bottom row).

Encoding Method	Direct	Huff-man	Cano-nical	Uni-tary	Exp-Golomb	Arith-metic	Predic-tive	Com-pact	Raw	MTF	40-Case
Per-layer	1.672	2.464	2.464	2.074	1.799	**2.673**	1.865	2.121	2.418	1.865	1.67
Inter-layer	4.083	2.727	2.751	2.912	2.9	1.692	2.874	3.155	2.816	2.46	**4.358**

Table 2 (second row) shows the average CR values after applying inter-layer compression. Interestingly, direct encoding turns to be better than the nine other considered lossless encoding methods. This is because the pattern matching of the inter-layer compressor is rendered ineffective when the signal encoding already approaches its entropy. Given this finding, we further improve direct encoding by considering all combinations among possible values of $\triangle x$, $\triangle y$ and $\triangle r$. Among the $3 \times 3 \times 5 = 45$ combinations, only 40 are possible as the five cases with $\triangle x = \triangle y = 0$ cannot exist in practice. This leads to an information content of $\log_2(40) \approx 5.32$ bits per skeleton pixel instead of $2\log_2(3) + \log_2(5) \approx 5.89$ bits for direct encoding. Table 2 (rightmost column) shows the average CR values with the 40-case encoding, which is 6.74% better than the best in the tested methods after all-layer compression. Hence, we keep this encoding method for CDMD.

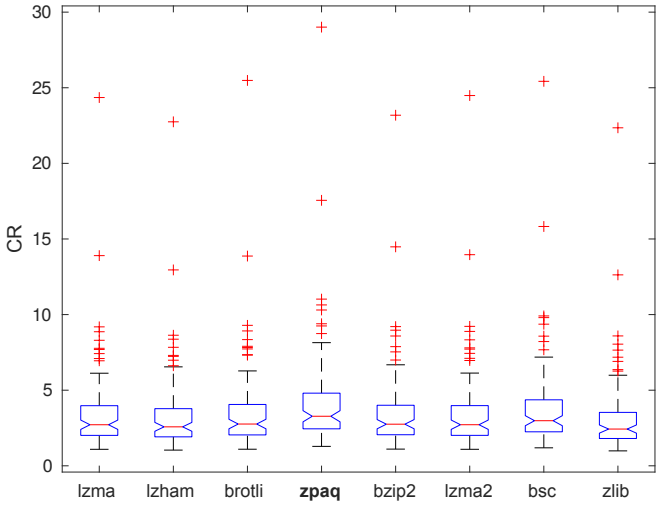

Figure 4. Compression ratio boxplots for eight compression methods run on 100 images.

4. Evaluation and Optimization

Our CDMD method described in Section 3 introduced three improvements with respect to DMD: the cumulative histogram layer selection, the intra-layer compression (40-case algorithm), and the inter-layer compression (ZPAQ). On our 100-image benchmark, these jointly deliver the following improvements:

- Layer selection: 3000 times faster and 3.28% higher quality;
- MAT encoding: 20.15% better compression ratio.

CDMD depends, however, on three parameters: the number of selected layers L, the size of removed islands ϵ, and the saliency threshold τ. Moreover, a compressed image $\tilde{I}$ is characterized by two factors: the visual quality that captures how well $\tilde{I}$ depicts the original image I, e.g., measured by the MS-SSIM metric, and the compression ratio CR (Equation (5)). Hence, the overall quality of CDMD can be modeled as

$$(MS\text{-}SSIM, CR) = CDMD(L, \epsilon, \tau). \tag{6}$$

Optimizing this two-variate function of three variables is not easy. Several commercial solutions exist, e.g., TinyJPG [54] but their algorithms are neither public nor transparent. To address this, we first merge the two dependent variables, *MS-SSIM* and CR, into a single one (Section 4.1). Next, we describe how we optimize for this single variable over all three free parameters (Section 4.2).

4.1. Joint Compression Quality

We need to optimize for both image quality *MS-SSIM* and compression ratio CR (Equation (6)). These two variables are, in general, inversely correlated: strong compression (high CR) means poor image quality (low *MS-SSIM*), and vice versa. To handle this, we combine *MS-SSIM* and CR into a single joint quality metric

$$Q = \frac{f_{MS\text{-}SSIM}(MS\text{-}SSIM) + f_{CR}(\overline{CR})}{2}, \tag{7}$$

where $\overline{CR}$ is the CR of a given image I normalized (divided) by the maximal CR value over all images in our benchmark. The transfer functions $f_{MS\text{-}SSIM}(x) = x^2$ and $f_{CR}(x) = x$ are used to combine (weigh) the two criteria we want to optimize for, namely quality *MS-SSIM* and compression ratio CR. After extensive experimentation with images from our benchmark, we found that *MS-SSIM* perceptually weighs more than CR, which motivates the quadratic contribution of the former vs. linear of the latter. Note that, if desired, $f_{MS\text{-}SSIM}$ and f_{CR} can be set to the identity function, which would imply a joint quality Q defined as the mean of the two.

4.2. Optimizing the Joint Compression Quality

To find parameter values that maximize Q (Equation (7)), we fix, in turn, two of the three free parameters L, ϵ, and τ to empirically-determined average values, and vary the third parameter over its allowable range via uniform sampling. The maximum Q value found this way determines the value of the varied parameter. This is simpler, and faster, than the usual hyper-parameter grid-search used, e.g., in machine learning [55], and is motivated by the fact that our parameter space is quite large (three-dimensional) and thus costly to search exhaustively by dense grid sampling. This process leads to the following results.

Number of layers: To study how L affects the joint quality Q, we plot Q as a function of L for our benchmark images. We sample L from 10 to 90 with a step of 10, following observations in [7] stating that 50–60 layers typically achieve good $SSIM$ quality. The two other free variables are set to $\epsilon = 0.02$ and $\tau = 1$. Figure 5a shows the results. We see that CDMD works particularly well for images of art deco and scientific visualization types. We also see that Q hardly changes for $L > 40$. Figure 5b summarizes these insights, showing that values $L \in \{20, 30, 40\}$ give an overall high Q for all image types.

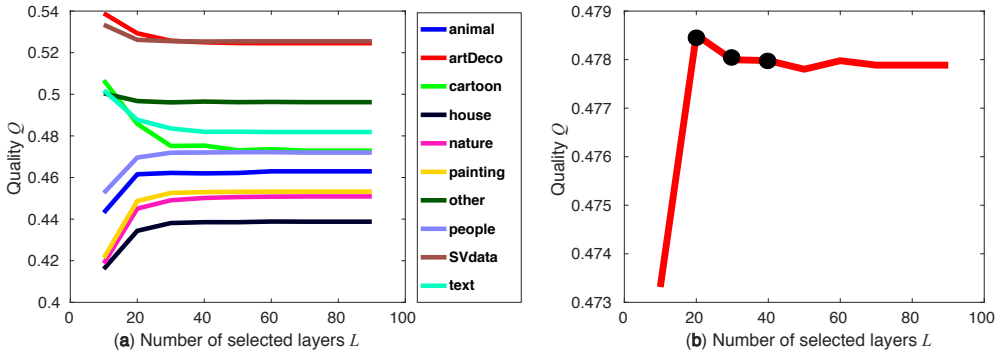

Figure 5. Quality Q as a function of number of layers L. (**a**) Q plots per image type. (**b**) Average Q for all image types. Black dots indicate good L values (20, 30, and 40).

Island size and saliency: We repeat the same evaluation for the other two free parameters, i.e., minimal island size ϵ and skeleton saliency τ, fixing each time the other two parameters to average values. Figure 6 shows how Q varies when changing ϵ and τ over their respective ranges of $\epsilon \in [0, 0.04]$ and $\tau \in [0, 6]$, similar to Figure 5. These ranges are determined by considerations outlined earlier in related work [7–9,13]. Optimal values for ϵ and τ are indicated in Figure 6 by black dots.

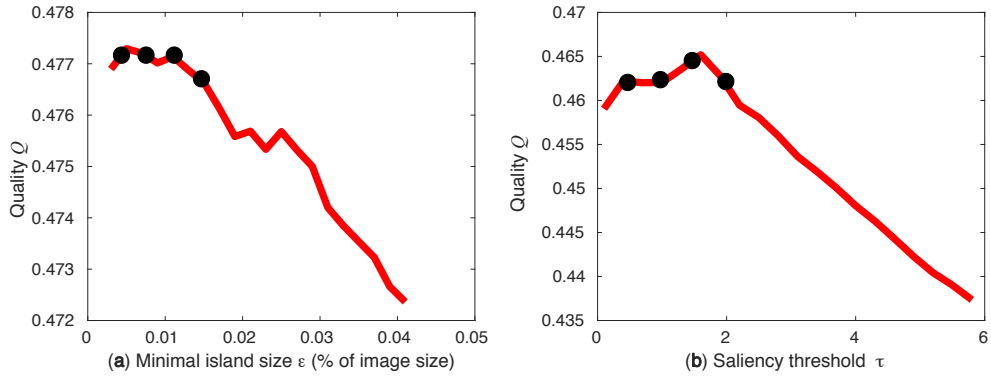

Figure 6. Quality Q as a function of island size ϵ (**a**) and skeleton saliency simplification τ (**a**). Selected optimal parameter values are marked black.

4.3. Trade-Off between MS-SSIM and CR

As already mentioned, our method, and actually any lossy image compression method, has a trade-off between compression (which we measure by CR) and quality (which we measure by MS-SSIM). Figure 7 shows the negative, almost-linear, correlation between CR and MS-SSIM for the 10 house images in our benchmark, with each image represented by a different color. Same-color dots show $3 * 4 * 4 = 48$ different settings of L, ϵ, and τ parameters, computed as explained in Section 4.2. This negative correlation is present for both the color version of the test image (Figure 7b) and its grayscale variant (Figure 7a). However, if we compare a set of same-color dots in Figure 7a, i.e., compressions of a given grayscale image for the 48 parameter combinations, with the similar set in Figure 7b, i.e., compressions of the same image, color variant for the same parameter combinations, we see that the first set is roughly lower and more to the left than the second set. That is, CDMD handles color images compressed better than grayscale ones, i.e., yields higher CR and/or higher MS-SSIM values. Very similar patterns occur for all other nine image types in our benchmark. For full results, we refer to [40].

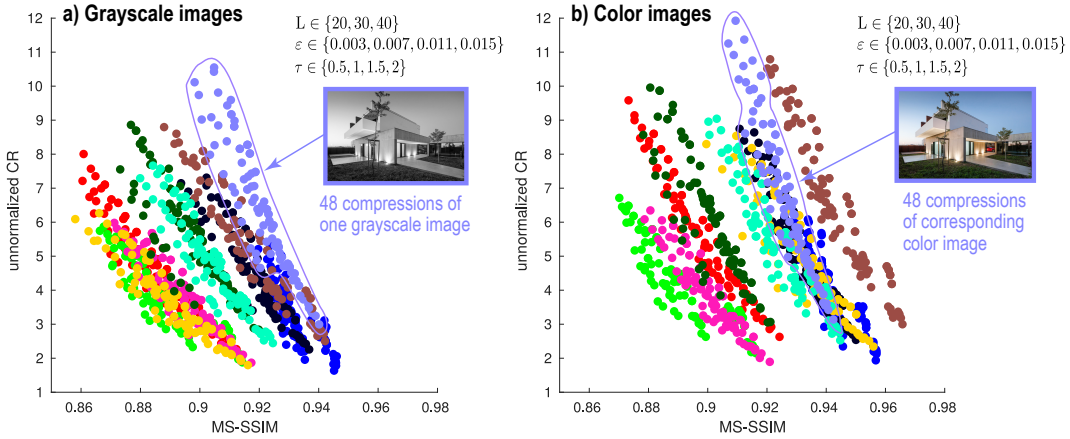

Figure 7. Trade-off between MS-SSIM and CR on 10 grayscale house images (**a**) and their corresponding color versions (**b**). The outlines show the compressions of a single image for 48 parameter combinations.

Besides parameter values, the trade-off between MS-SSIM and CR depends on the image type. Figure 8 shows this by plotting the average MS-SSIM vs CR for all 10 image types in our benchmark. Here, one dot represents the average values of the two metrics for a given parameter-setting over all images in the respective class. We see the same inverse correlation as in Figure 7. We also see that CDMD works best for art decoration (artDeco) and scientific visualization (SVdata) image types.

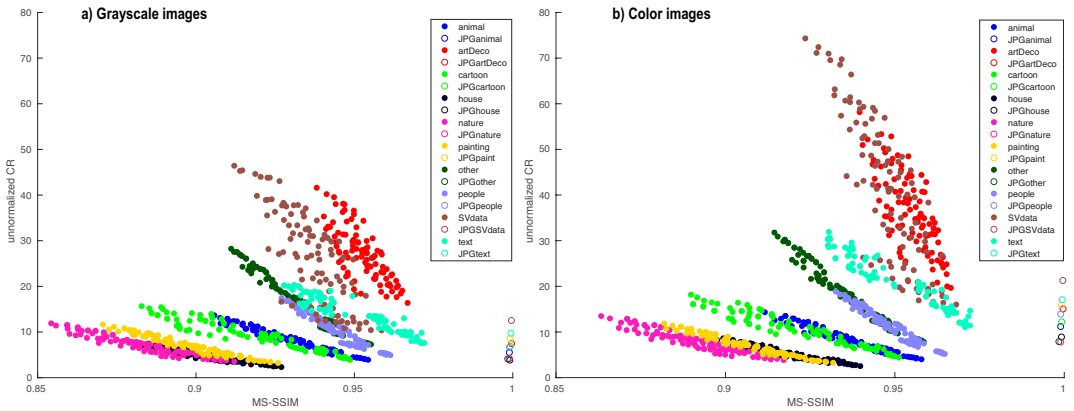

Figure 8. Average MS-SSIM vs. CR for 10 image types for CDMD (filled dots) and JPEG (hollow dots). Left shows results for the grayscale variants of the color images (shown right).

4.4. Comparison with JPEG

Figure 8 also compares the MS-SSIM and CR values of CDMD (full dots) with JPEG (hollow dots) for all our benchmark images, for their grayscale versions (a) and color versions (b), respectively. Overall, JPEG yields higher MS-SSIM values, but CDMD yields better CR values for most of its parameter settings. We also see that CDMD performs relatively better for the color images. Figure 9 further explores this insight by showing ten images, one of each type, from our benchmark, compressed by CDMD and JPEG, and their corresponding CR and MS-SSIM values. Results for the entire 100-image database are available in the supplementary material. We see that, if one prefers a higher CR over higher image quality, CDMD is a better choice than JPEG. Furthermore, there are two image types for which we get both a higher CR than JPEG and a similar quality: Art Deco and Scientific Visualization. Figure 10 explores these classes in further detail, by showing four additional examples, compressed with CDMD and JPEG. We see that CDMD and JPEG yield results which are visually almost identical (and have basically identical MS-SSIM values). However, CDMD yields compression values 2 up to 19 times higher than JPEG. Figure 10(a3–d3) shows the per-pixel difference maps between the compressed images with CDMD and JPEG (differences coded in luminance). These difference images are almost everywhere black, indicating no differences between the two compressions. Minimal differences can be seen, upon careful examination of these difference images, along a few luminance contours, as indicated by the few bright pixels in the images. These small differences are due to the salience-based skeleton simplification in CDMD.

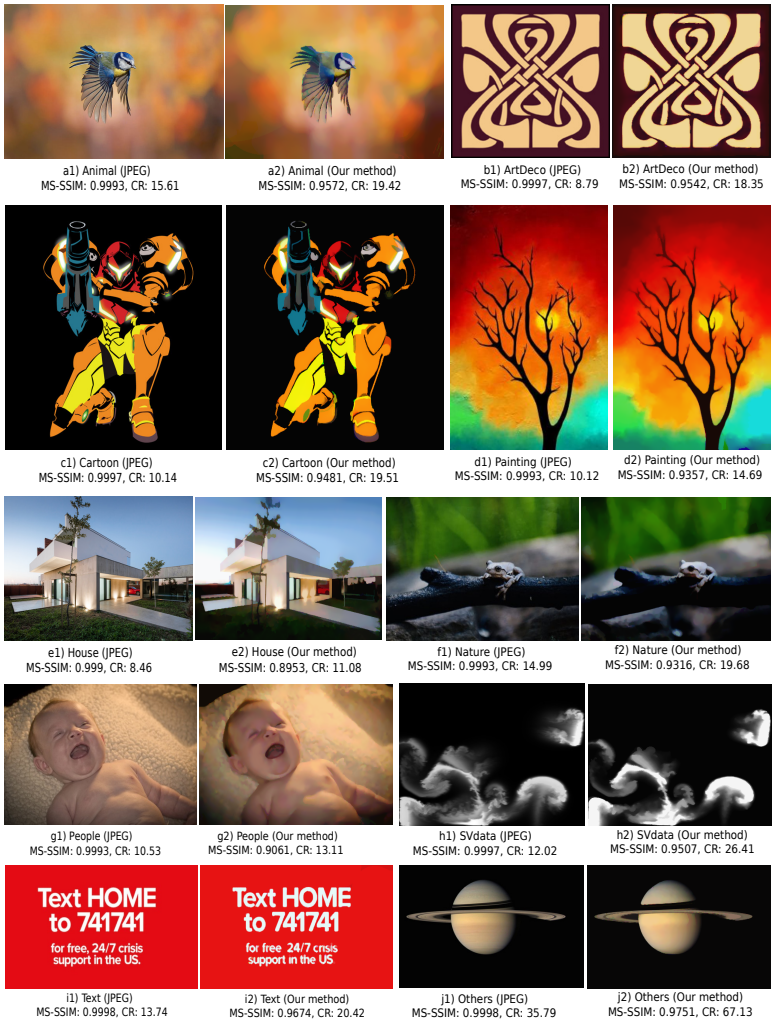

Figure 9. Comparison of JPEG (**a1**–**j1**) with our method (**a2**–**j2**) for 10 image types. For each image, we show the MS-SSIM quality and compression ratio CR.

For a more detailed comparison with JPEG, we next consider JPEG's quality setting q. This value, set typically between 10% and 100%, controls JPEG's trade-off between quality and compression, with higher values favoring quality. Figure 11 compares CDMD for the Scientific Visualization and ArtDeco image types (filled dots) with 10 different settings of JPEG's q parameter, uniformly spread in the $[10, 100]$ interval (hollow dots). Each dot represents the average of MS-SSIM and CR for a given method and image type for a given parameter combination. We see that CDMD yields higher MS-SSIM values, and for optimal parameters, also yields a much high CR value. In contrast, JPEG either yields good MS-SSIM or only high CR, but cannot maximize both.

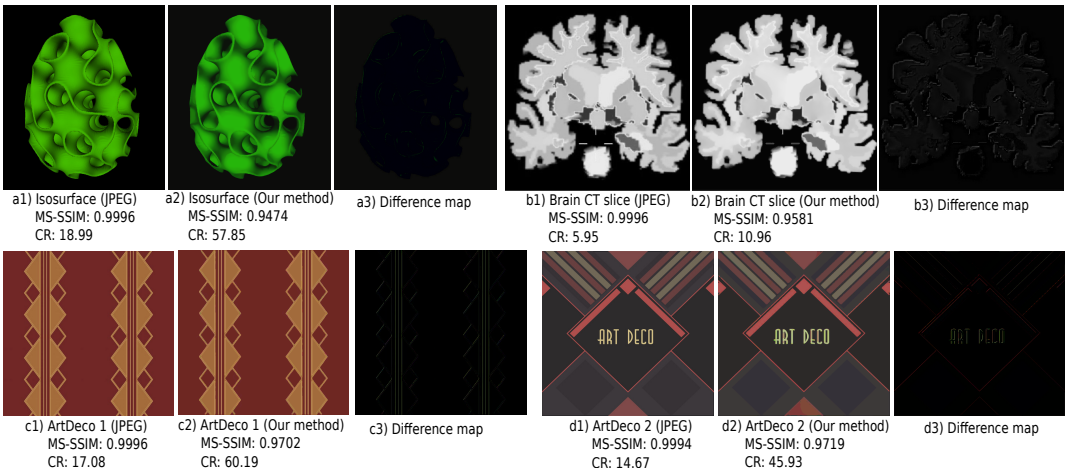

Figure 10. Our method (**a2**–**d2**) yields higher compression than, and visually identical quality with, JPEG (**a1**–**d1**) for two image classes: Scientific Visualization (**a**,**b**) and Art Deco (**c**,**d**)).

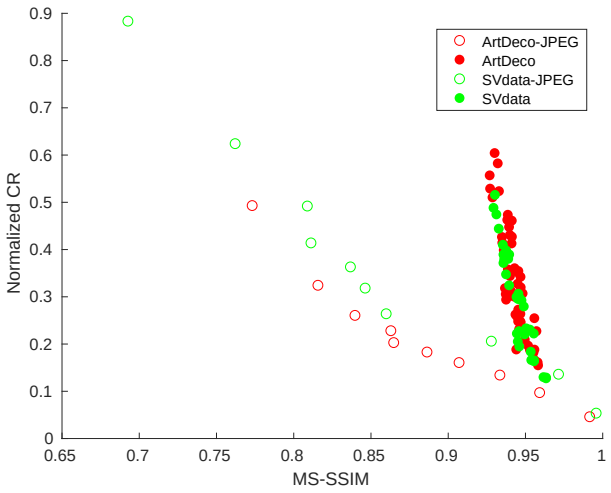

Figure 11. Average MS-SSIM vs. CR for two image classes (Art Deco, Scientific Visualization), for our method (filled dots) and JPEG (hollow dots).

4.5. Handling Noisy Images

As explained in Section 2.2, the island removal parameter ϵ and the saliency threshold τ jointly 'simplify' the compressed image by removing, respectively, small-scale islands and small-scale indentations along the threshold-set boundaries. Hence, it is insightful to study how these parameters affect the compression of images which have high-frequency, small-scale details and/or noise. Figure 12 shows an experiment that illustrates this. An original image was selected which contains high amounts of small-scale high-frequency detail, e.g., the mandrill's whiskers and fur patterns.

The left column shows the CDMD results for four combinations of ϵ and τ. In all cases, we used $L = 30$. As visible, and in line with expectations, increasing ϵ and/or τ has the effect of smoothing out small-scale details, thereby decreasing MS-SSIM and increasing the compression ratio CR. However, note that contours that separate large image elements, such as the red nose from the blue cheeks, or the pupils from the eyes, are kept sharp. Furthermore, thin-but-long details such as the whiskers have a high saliency, and are thus kept quite well.

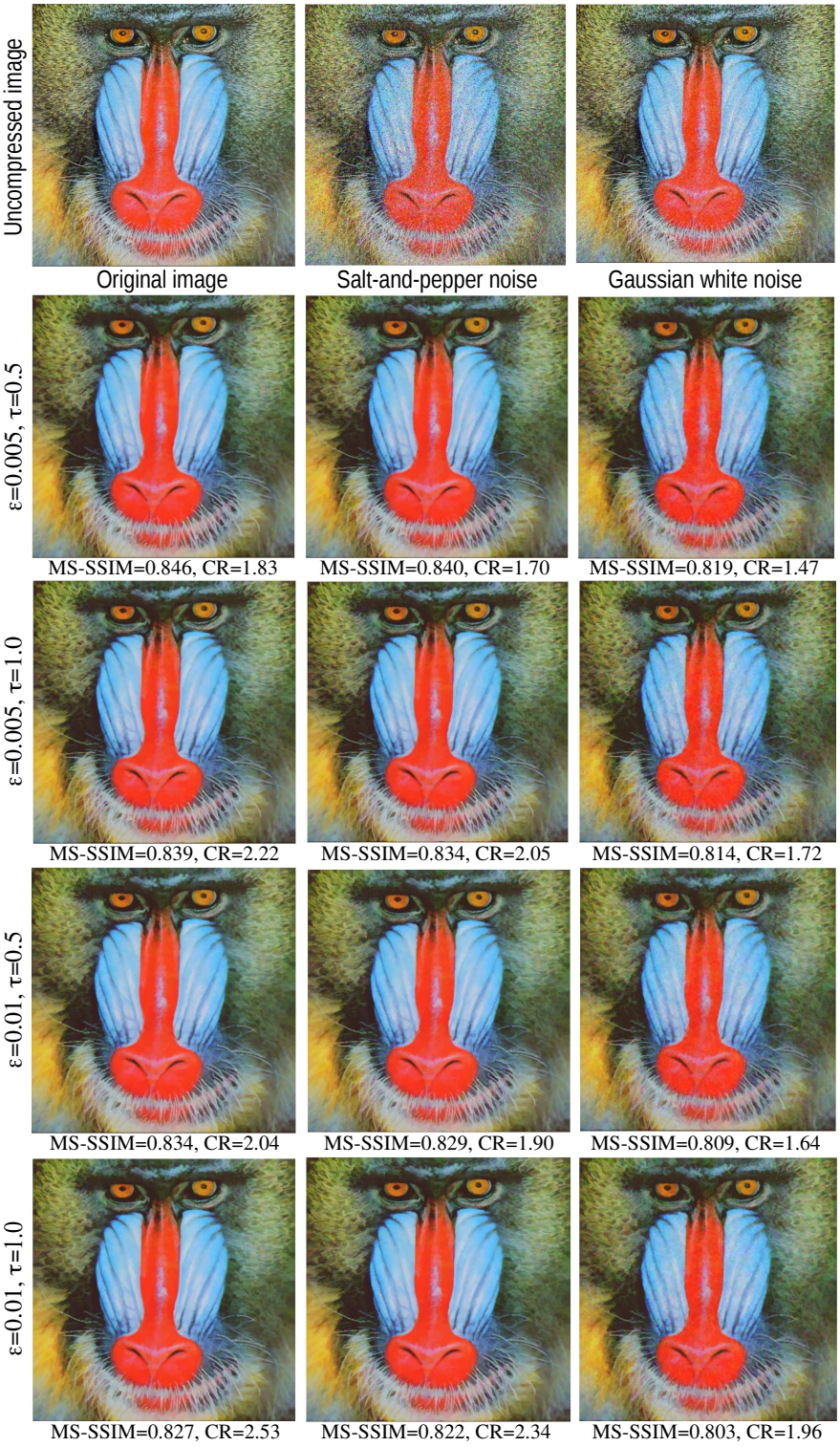

Figure 12. Results of CDMD on an image with fine-grained detail (left column) additionally corrupted by small-scale noise (middle and right columns), for different values of the ϵ and τ parameters.

The middle column in Figure 12 shows the CDMD results for the same image, this time corrupted by salt-and-pepper noise of density 0.1, compressed with the same parameter settings. We see that the noise is removed very well for all parameter values, the compression results being visually nearly identical to those generated from the uncorrupted image. The MS-SSIM and CR values are now slightly lower, since, although visually difficult to spot, the added noise does affect the threshold sets in the image. Finally, the right column in Figure 12 shows the CDMD results for the same image, this time corrupted by zero-mean Gaussian white noise with variance 0.01. Unlike salt-and-pepper noise, which is distributed randomly over different locations and has similar amplitudes, the Gaussian noise has a normal amplitude distribution and affects all locations in an image uniformly. Hence, CDMD does not remove Gaussian noise as well as the salt-and-pepper one, as we can see both from the actual images and the corresponding MS-SSIM and CR values. Yet, even for this noise type, we argue that CDMD does not produce disturbing artifacts in the compressed images, and still succeeds in preserving the main image structures and also a significant amount of the small-scale details.

5. Discussion

We next discuss several aspects of our CDMD image compression method.

Genericity, ease of use: CDMD is a general-purpose compression method for any types of grayscale and color images. It relies on simple operations such as histogram computation and thresholding, as well as on well-tested, robust, algorithms, such as the skeletonization method in [16,17], and ZPAQ. CDMD has three user parameters – the number of selected layers L, island thresholding ϵ, and skeleton saliency threshold τ. These three parameters affect the trade-off between compression ratio and image quality (see Section 4.2). End users can easily understand these parameters as follows: L controls how smooth the gradients (colors or shades) are captured in the compressed image (higher values yield smoother gradients); ϵ controls the scale of details that are kept in the image (higher values remove larger details); and τ controls the scale of corners that are kept in the image (larger values round-off larger corners). Good default ranges of these parameters are given in Section 4.2.

Speed: The most complex operation of the CDMD pipeline, the computation of the regularized skeletons $\tilde{S}$, is efficiently done on the GPU (see Section 2.1). Formally, CDMD's computational complexity is $O(R)$ for an image of R pixels, since the underlying skeletonization is linear in image size, being based on a linear-time distance transform [56]. This is the best that one can achieve complexity-wise. Given this, the CDMD method is quite fast: For images of up to 1024^2 pixels, on a Linux PC with an Nvidia RTX 2060 GPU, layer selection takes under 1 millisecond; skeletonization takes about 1 second per color channel; and reconstruction takes a few hundred milliseconds. Obviously, state-of-the-art image compression methods have highly engineered implementations which are faster. We argue that the linear complexity of CDMD also allows speed-ups to be gained by subsequent engineering and optimization.

Quality vs. compression rate: We are not aware of studies showing how quality and compression rates relate vs. image size for, e.g., JPEG. Still, analyzing JPEG, we see that its size complexity linearly depends on the image size. That is, the compression ratio CR is overall linear in the input image size R for a given, fixed, quality, since JPEG encodes an image by separate 8×8 blocks. In contrast, CDMD's skeletons are of $\sqrt{R}$ complexity, since they are 1D structures. While a formal evaluation pends, this suggests CDMD may scale better for large image sizes.

Color spaces: As explained in Section 2.1, for color images, (C)DMD is applied to the individual channels of these, following representations in various color spaces. We currently tested the RGB and HSV color spaces, following the original DMD method proposal. For these, we obtained very similar compression vs. quality results. We also tested YUV (more precisely, YCbCr), and obtained compression ratios about twice as high as those reported earlier in this paper (for the RGB space). However, layer selection in the YCbCr space is more delicate than in RGB space: While the U and V channels can be described well with just a few layers (which is good for compression), a slightly too aggressive compression (setting a slightly too low L value) can yield strong visual

differences between the original and compressed images. Hence, the method becomes more difficult to control, parameter-wise, by the user. Exploring how to make this control simpler for the end user, while retaining the higher compression rate of the YUV space, is an interesting point for future work.

Best image types: Layer removal is a key factor to CDMD. Images that have large and salient threshold-sets, such as Art Deco and Scientific Visualization, can be summarized by just a few such layers (low L). For instance, the Art Deco image in Figure 10(c1) has only a few distinct gray levels, and large, salient, shapes in each layer. Its CDMD compression (Figure 10(c2)) is of high quality, and is more than 60 times smaller than the original. The JPEG compression of the same image is just 17 times smaller than the original. At the other extreme, we see that CDMD is somewhat less suitable for images with many fine details, such as animal furs and greenery (Figure 9(e2)). This suggests that CDMD could be very well suited (and superior to JPEG) for compressing data-visualization imagery, e.g., in the context of remote/online viewing of medical image databases.

Preprocessing for JPEG: Given the above observation, CDMD and JPEG seem to work best for different types of images. Hence, a valid idea is to combine the two methods rather than let them compete against each other, following earlier work that preprocesses images to aid JPEG's compression [57]. We consider the same idea, i.e., use CDMD as a preprocessor for JPEG. Figure 13 shows three examples of this combination. When using only JPEG, the original images (a1–c1), at 20% quality (JPEG setting q), yield blocking artifacts (a2–c2). When using JPEG with CDMD preprocessing, these artifacts are decreased (a3–c3). This can be explained by the rounding-off of small-scale noise dents and bumps that the saliency-based skeleton simplification performs [13]. Such details correspond to high frequencies in the image spectrum which next adversely impact JPEG. Preprocessing by CDMD has the effect of an adaptive low-pass filter that keeps sharp and large-scale details in the image while removing sharp and small-scale ones. As Figure 13 shows, using CDMD as preprocessor for JPEG yields a 10% to 20% compression ratio increase as compared to plain JPEG, with a limited loss of visible quality.

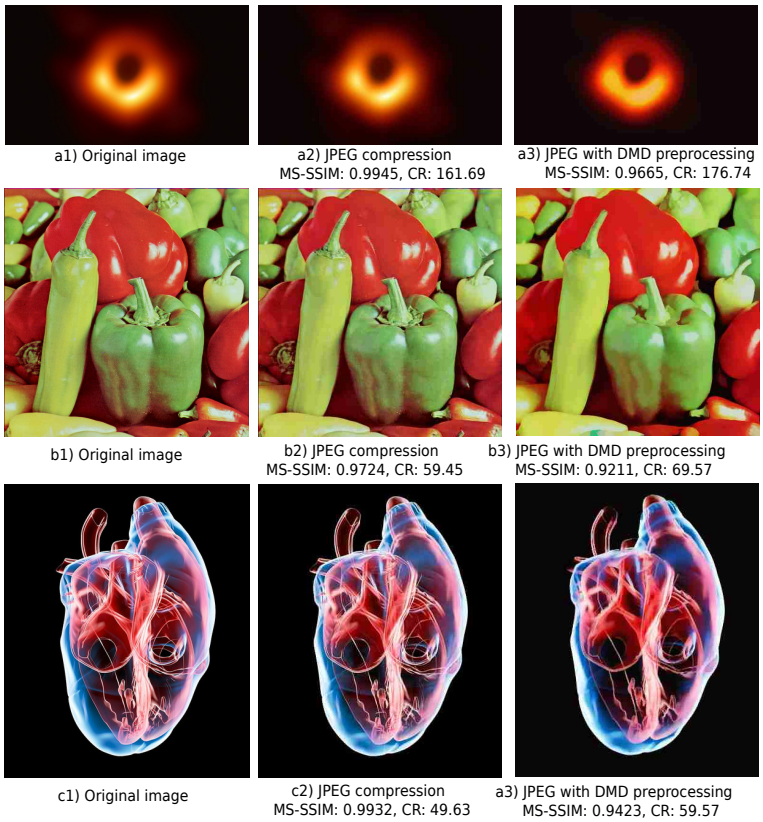

Figure 13. Comparison of plain JPEG (**a2–c2**) with CDMD applied as preprocessor to JPEG (**a3–c3**) for three images.

Limitations: Besides the limited evaluation (on only 100 color images and their grayscale equivalents), CDMD is here only evaluated against a single generic image compression method, i.e., JPEG. As outlined in Section 2.4, tens of other image compression methods exist. We did not perform an evaluation against these since, as already noted, our main research question was to show that skeletons can be used for image compression with good results—something that has not been done so far. We confirmed this by comparing CDMD against JPEG. Given our current positive results, we next aim to improve CDMD, at which point comparison against state-of-the-art image compression methods becomes relevant.

6. Conclusions

We have presented Compressing Dense Medial Descriptors (CDMD), an end-to-end method for compressing color and grayscale images using a dense medial descriptor approach. CDMD adapts the existing DMD method, proposed for image segmentation and simplification, for the task of image compression. For this, we proposed an improved layer-selection algorithm, a lossless MAT-encoding scheme, and an all-layer lossless compression scheme.

To study the effectiveness of our method, we considered a benchmark of 100 images of 10 different types, and did an exhaustive search of the free-parameters of our method, in order to measure and optimize the compression-ratio, perceptual quality, and combination of these two metrics. On a practical side, our evaluation showed that CDMD delivers superior compression to JPEG at a small quality loss; that it delivers both superior compression and quality for specific image types. On a more theoretical (algorithmic) side, CDMD shows, for the first time, that medial descriptors offer interesting and viable possibilities to compress grayscale and color images, thereby extending their applicability beyond the processing of binary shapes.

Several future work directions are possible. First, more extensive evaluations are interesting and useful to do, considering more image types and more compressors, e.g., JPEG 2000, to find the added value of CDMD. Secondly, a low-hanging fruit is using smarter representations of the per-layer MAT: Since skeleton branches are known to be smooth [4], encoding them by higher-level constructs such as splines rather than pixel-chains can yield massive compression-ratio increases with minimal quality losses. We plan to address such open avenues in the near future.

Author Contributions: Conceptualization, A.T.; methodology, J.W. and A.T.; software, M.T. and J.W.; validation, J.W.; analysis, J.W., J.K., and A.T.; investigation, M.T. and J.W.; data curation, J.W.; writing—original draft preparation, J.W.; writing—review and editing, J.K., J.W., and A.T.; visualization, J.W.; supervision, A.T. and J.K. All authors have read and agreed to the published version of the manuscript

Funding: J. Wang acknowledges the China Scholarship Council (Grant number: 201806320354) for financial support.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Shum, H.Y.; Kang, S.B.; Chan, S.C. Survey of image-based representations and compression techniques. *IEEE Trans. Circuits Syst. Video Technol.* **2003**, *13*, 1020–1037. [CrossRef]
2. Wallace, G.K. The JPEG still picture compression standard. *IEEE TCE*. **1992**, *38*, xviii–xxxiv. [CrossRef]
3. Davies, E.R. *Machine Vision: Theory, Algorithms, Practicalities*; Academic Press: London, UK, 2004.
4. Siddiqi, K.; Pizer, S. *Medial Representations: Mathematics, Algorithms and Applications*; Springer: New York, NY, USA, 2008.
5. Saha, P.K.; Borgefors, G.; di Baja, G.S. A survey on skeletonization algorithms and their applications. *Pattern Recognit. Lett.* **2016**, *76*, 3–12. [CrossRef]
6. Saha, P.K.; Borgefors, G.; di Baja, G.S. *Skeletonization—Theory, Methods, and Application*; Academic Press: London. 2017.
7. Van Der Zwan, M.; Meiburg, Y.; Telea, A. A dense medial descriptor for image analysis. In Proceedings of the International Conference on Computer Vision Theory and Applications(VISAPP-2013), Barcelona, Spain, 21–24 February 2013; pp. 285–293.

8. Koehoorn, J.; Sobiecki, A.; Boda, D.; Diaconeasa, A.; Doshi, S.; Paisey, S.; Jalba, A.; Telea, A. Automated Digital Hair Removal by Threshold Decomposition and Morphological Analysis. In Proceedings of the International Symposium on Mathematical Morphology and Its Applications to Signal and Image (ISMM), Reykjavik, Iceland, 27–29 May 2015.
9. Sobiecki, A.; Koehoorn, J.; Boda, D.; Solovan, C.; Diaconeasa, A.; Jalba, A.; Telea, A. A New Efficient Method for Digital Hair Removal by Dense Threshold Analysis. In Proceedings of the 4th World Congress of Dermoscopy, Vienna, Austria, 21 April 2015.
10. Blum, H. A transformation for extracting new descriptors of shape. In *Models for the Perception of Speech and Visual Form*; Dunn, W.W., Ed.; MIT Press: Cambridge, UK, 1967; pp. 362–381.
11. Blum, H.; Nagel, R. Shape description using weighted symmetric axis features. *Pattern Recognit.* **1978**, *10*, 167–180. [CrossRef]
12. Sethian, J.A. A fast marching level set method for monotonically advancing fronts. *Proc. Natl. Acad. Sci. USA* **1996**, *93*, 1591–1595. [CrossRef]
13. Telea, A. Feature Preserving Smoothing of Shapes Using Saliency Skeletons. In *Visualization in Medicine and Life Sciences II (VMLS)*; Springer: Basel, Switzerland, 2012; pp. 153–170.
14. Ogniewicz, R.L.; Kubler, O. Hierarchic Voronoi skeletons. *Pattern Recognit.* **1995**, *28*, 343–359. [CrossRef]
15. Costa, L.; Cesar, R. *Shape Analysis and Classification*; CRC Press: New York, NY, USA, 2000.
16. Falcão, A.; Stolfi, J.; Lotufo, R. The image foresting transform: Theory, algorithms, and applications. *IEEE Trans. Pattern Anal. Mach. Intell.* **2004**, *26*, 19–29. [CrossRef]
17. Telea, A.; van Wijk, J.J. An Augmented Fast Marching Method for Computing Skeletons and Centerlines. In Proceedings of the 2002 Joint Eurographics and IEEE TCVG Symposium on Visualization, VisSym, Barcelona, Spain, 27–29 May 2002.
18. Kadir, T.; Brady, M. Saliency, Scale and Image Description. *Int. J. Comput. Vis.* **2001**, *45*, 83–105. [CrossRef]
19. Battiato, S.; Farinella, G.M.; Puglisi, G.; Ravi, D. Saliency-based selection of gradient vector flow paths for content aware image resizing. *IEEE Trans. Image Process.* **2014**, *23*, 2081–2095. [CrossRef]
20. Ersoy, O.; Hurter, C.; Paulovich, F.; Cantareiro, G.; Telea, A. Skeleton-based edge bundles for graph visualization. *IEEE Trans. Vis. Comput. Graph.* **2011**, *17*, 2364–2373. [CrossRef]
21. Zhai, X.; Chen, X.; Yu, L.; Telea, A. Interactive Axis-Based 3D Rotation Specification Using Image Skeletons. In Proceedings of the GRAPP, Valletta, Malta, 27–29 February 2020.
22. Telea, A. Real-Time 2D Skeletonization Using CUDA. Available online: http://www.cs.rug.nl/svcg/Shapes/CUDASkel (accessed on 1 May 2019).
23. Wang, Z.; Bovik, A.C. Mean squared error: Love it or leave it? A new look at Signal Fidelity Measures. *IEEE Signal Proc. Mag.* **2009**, *26*, 98–117. [CrossRef]
24. Wang, Z.; Bovik, A.; Sheikh, H.; Simoncelli, E. Image quality assessment: from error visibility to structural similarity. *IEEE Trans. Image Process.* **2004**, *13*, 600–612. [CrossRef] [PubMed]
25. Li, C.; Bovik, A.C. Content-weighted video quality assessment using a three-component image model. *J. Electron. Imaging* **2010**, *19*, 110–130.
26. Wang, Z.; Simoncelli, E.P.; Bovik, A.C. Multiscale structural similarity for image quality assessment. In Proceedings of the Thrity-Seventh Asilomar Conference on Signals, Systems Computers, Pacific Grove, CA, USA, 9–12 November 2003; Volume 2, pp. 1398–1402.
27. Zhang, C.; Chen, T. A survey on image-based rendering—Representation, sampling and compression. *Signal Process Image* **2004**, *19*, 1–28. [CrossRef]
28. Toderici, G.; O'Malley, S.; Hwang, S.J.; Vincent, D.; Minnen, D.; Baluja, S.; Covell, M.; Sukthankar, R. Variable Rate Image Compression with Recurrent Neural Networks. *arXiv* **2016**, arXiv:1511.06085.
29. Ballé, J.; Laparra, V.; Simoncelli, E. End-to-end Optimized Image Compression. *arXiv* **2017**, arXiv:1611.01704.
30. Toderici, G.; Vincent, D.; Johnston, N.; Hwang, S.J.; Minnen, D.; Shor, J.; Covell, M. Full Resolution Image Compression with Recurrent Neural Networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017.
31. Prakash, A.; Moran, N.; Garber, S.; DiLillo, A.; Storer, J. Semantic Perceptual Image Compression using Deep Convolution Networks. In Proceedings of the Data Compression Conference (DCC), Snowbird, UT, USA, 4–7 April 2017.
32. Stock, P.; Joulin, A.; Gribonval, R.; Graham, B.; Jégou, H. And the Bit Goes Down: Revisiting the Quantization of Neural Networks. *arXiv* **2019**, arXiv:1907.05686.

33. Guo, C.; Zhang, L. A novel multi resolution spatiotemporal saliency detection model and its applications in image and video compression. *IEEE Trans. Image Process.* **2010**, *19*, 185–198.
34. Andrushia, A.D.; Thangarjan, R. Saliency-Based Image Compression Using Walsh-Hadamard Transform (WHT). In *Biologically Rationalized Computing Techniques For Image Processing Applications*; Springer: Cham, Switzerland, 2018; pp. 21–42.
35. Itti, L.; Koch, C.; Niebur, E. A model of saliency-based visual attention for rapid scene analysis. *IEEE Trans. Pattern Anal. Mach. Intell.* **1998**, *20*, 1254–1259. [CrossRef]
36. Imamoglu, N.; Lin, W.; Fang, Y. A saliency detection model using low-level features based on wavelet transform. *IEEE Trans. Multimed.* **2013**, *15*, 96–105. [CrossRef]
37. Lin, R.J.; Lin, W.S. Computational visual saliency model based on statistics and machine learning. *J. Vis.* **2014**, *14*, 1–18. [CrossRef] [PubMed]
38. Arya, R.; Singh, N.; Agrawal, R. A novel hybrid approach for salient object detection using local and global saliency in frequency domain. *Multimed. Tools Appl.* **2015**, *75*, 8267–8287. [CrossRef]
39. Hecht, S. The visual discrimination of intensity and the Weber-Fechner law. *J. Gen. Physiol.* **2003**, *7*, 235–267. [CrossRef] [PubMed]
40. Wang, J. CDMD-Benchmark. Available online: https://github.com/WangJieying/CDMD-benchmark (accessed on 1 May 2020).
41. Cormen, T.H.; Stein, C.; Rivest, R.L.; Leiserson, C.E. *Introduction to Algorithms*, 3rd ed.; MIT Press: London, UK, 2001; pp. 540–549.
42. Geelnard, M. Basic Compression Library. Available online: github.com/MariadeAnton/bcl/blob/master/src (accessed on 14 January 2015).
43. Roy, A.; Scott, A.J. Unitary designs and codes. *Des. Codes Cryptogr.* **2009**, *53*, 13–31. [CrossRef]
44. Langdon, G.G. An Introduction to Arithmetic Coding. *IBM J. Res. Dev.* **1984**, *28*, 135–149. [CrossRef]
45. Bentley, J.L.; Sleator, D.D.; Tarjan, R.E.; Wei, V.K. A Locally Adaptive Data Compression Scheme. *Commun. ACM* **1986**, *29*, 320–330. [CrossRef]
46. Pavlov, I. LZMA SDK (Software Development Kit). Available online: http://www.7-zip.org/sdk.html (accessed on 1 May 2019).
47. Geldreich, R. LAHAM. Available online: https://code.google.com/archive/p/lzham/ (accessed on 1 March 2020).
48. Alakuijala, J.; Szabadka, Z. Brotli Compressed Data Format. Available online: https://tools.ietf.org/html/rfc7932 (accessed on 1 March 2020).
49. Mahoney, M. The Zpaq Compression Algorithm. Available online: http://mattmahoney.net/dc/zpaq_compression.pdf (accessed on 1 March 2020).
50. Seward, J. Bzip2. Available online: http://en.wikipedia.org/wiki/Bzip2 (accessed on 1 March 2020).
51. Grebnov, I. Libbsc: A High Performance Data Compression Library. Available online: https://github.com/IlyaGrebnov/libbsc (accessed on 1 March 2020).
52. Deutsch, P.; Gailly, J. ZLIB Compressed Data Format Specification Version 3.3. Available online: https://datatracker.ietf.org/doc/rfc1950 (accessed on 1 March 2020).
53. Nemerson, E. Squash Library. Available online: http://quixdb.github.io/squash (accessed on 1 March 2020).
54. TinyJPG. Smart JPEG and PNG Compression. Available online: https://tinyjpg.com (accessed on 1 March 2020).
55. Bergstra, J.; Bengio, Y. Random Search for Hyper-Parameter Optimization. *J. Mach. Learn. Res.* **2012**, *13*, 281–305.
56. Cao, T.T.; Tang, K.; Mohamed, A.; Tan, T.S. Parallel banding algorithm to compute exact distance transform with the GPU. In Proceedings of the 2010 ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games, Washington, DC, USA, 19–21 February 2010.
57. Tushabe, F.; Wilkinson, M.H.F. Image preprocessing for compression: Attribute filtering. In Proceedings of International Conference on Signal Processing and Imaging Engineering (ICSPIE'07), San Francisco, CA, USA, 24–26 October 2007; pp. 1411–1418.