NONLOCAL TEXTURE FILTERING WITH EFFICIENT TREE STRUCTURES AND INVARIANT PATCH SIMILARITY MEASURES

Oliver Kleinschmidt[1]*, Thomas Brox*[2]*, and Daniel Cremers*[1]

[1]Computer Vision Group, University of Bonn
Römerstr. 164, 53117 Bonn, Germany, {kleinsch,dcremers}@cs.uni-bonn.de

[2] Department of Computer Science, University of Dresden
Nöthnitzer Str. 46, 01187 Dresden, Germany, brox@inf.tu-dresden.de

ABSTRACT

In this work, we propose a study of efficient and optimal texture denoising methods based on the nonlocal means filter. In particular, we present efficient implementations of nonlocal filtering using highly adapted data structures such as cluster trees, spill trees and cluster forests. A comparative study of computational speed indicates that cluster forests are superior to alternative methods. Moreover, we introduce several extensions of the original nonlocal means filter which introduce invariance with respect to variations in brightness, scale, and rotation.

1. NONLOCAL SMOOTHING

Image enhancement and noise removal is a classical task in image processing with a long history and many methodologies. Especially discontinuity preserving filters, such as nonlinear diffusion filters [1], the ROF filter [2], and certain types of wavelet shrinkage [3], have been very successful in removing noise from images while preserving their most relevant structures. However, despite their success in a wide field of applications, they reveal a significant shortcoming when it comes to highly oscillatory structures, as they naturally appear in textural patterns. Such structures are confused with the noise and are erroneously removed.

When considering denoising of an image $I : \Omega \to \mathbb{R}^D$, nonlinear diffusion filters and related methods are spatially local in the sense that at each location $x \in \Omega$ the update of the evolving image $u : \Omega \to \mathbb{R}^2 \times [0, T]$, $u(x, 0) = I(x)$ is determined only by derivatives of u at that same location x. A class of image filters which adaptively takes into account intensity information from more distant locations are the Yaroslavsky neighbourhood filters [4]:

$$u(x) = \frac{\int K(x, y) I(y)\, dy}{\int K(x, y)\, dy}. \tag{1}$$

Here the smoothed image $u(x)$ is stated as the weighted average of pixels of the original image $I(x)$. The weights are determined by a nonnegative kernel function K, which decays with the distance $d^2(x, y) = \gamma|x - y|^2 + |I(x) - I(y)|^2$. A typical choice is the Gaussian kernel $K(x, y) = \frac{1}{(2\pi h^2)^{D/2}} \exp\left(-\frac{d^2(x,y)}{2h^2}\right)$ with kernel width h and dimensionality of the data D. This filter assigns large weights to pixels y and their intensities $I(y)$ which are similar in the sense that they are close to $(x, I(x))$ in space and in intensity. The parameter γ allows to adjust the relative importance of spatial and tonal similarity. Neighbourhood filters are also known as local M-smoothers [5, 6]. These filters can also be iterated, which results in the bilateral filter [7, 8]. Relations between such neighbourhood filters and nonlinear diffusion filters have been investigated in [9, 10, 11].

It turns out that even though these semi-local filters[1] substantially increase the number of candidate pixels for averaging compared to diffusion filters, they reveal a similar qualitative denoising behavior as nonlinear diffusion: whereas they preserve large scale structures, small scale structures are regarded as noise and are removed.

For achieving a better preservation of small-scale textural patterns, a small but decisive extension of the neighbourhood filters is necessary. Rather than considering only the centre pixel in the similarity of two points, we can regard local balls (patches) around these points. These patches capture the dependencies of neighbouring pixels and thus can distinguish textural patterns. The idea is inspired by works on texture synthesis [12, 13, 14] and has been proposed simultaneously with the nonlocal means filter [15] and the UINTA filter [16]. Both filters use a distance that considers not only the similarity of the central pixel, but also the similarity of its neighbourhood:

$$d^2(x, y) = \int G_\rho(x') \big(I(x - x') - I(y - x')\big)^2 \, dx'. \tag{2}$$

The Gaussian kernel G_ρ, which is not to be confused with the kernel K, acts as a weighted neighbourhood of size ρ. A uniformly weighted box can be chosen as well, which illustrates the basic concept of comparing patches. Since the above similarity measure takes into account complete patches instead of single pixel intensities, only similar textures play a role in the averaging. This removes noise

[1] semi-local due to the spatial distance that plays a role in the similarity

while the fine repetitive structures that are due to the texture are preserved by the filter. A variety of applications exist for this filter. Apart from denoising images, the concept can, for instance, be translated to video enhancement [17] and the smoothing of 3D surfaces [18].

Apart from ρ, the size of the patch, the filter contains another important parameter, namely the width h of the kernel K. It quantifies how fast the weights decay with increasing dissimilarity of respective patches. Statistical reasoning as in [19, 20] allows to determine h automatically via cross-validation or by estimating the noise variance. Ideas to adapt the size of the patch ρ locally to the data have been presented in [21].

Other improvements of the basic filter concern the iteration of the filter. The bilateral filter is an iterative version of the Yaroslavsky filter, and there is no reason why we should not iterate also a neighbourhood filter whose similarities are defined on patches rather than single pixel intensities. The entropy minimisation framework of the UINTA filter [16] leads to such a patch-based variant of the bilateral filter. Other iterative versions have been proposed in [22, 23, 24, 25]. Some of these filters, particularly [23, 24], are designed in a way that brings them closer to local filters by running many iterations with weights $K(x, y)$ computed on the initial image I and emphasising the spatial distance of the pixels rather than the similarity of the patches.

In the present paper, we are concerned with two other important issues in nonlocal filtering: computational complexity and invariance of the patch distance with respect to certain transformations, such as rotation, scaling, and illumination. The next section will deal with the computational complexity and present a novel indexing structure, the cluster forest. Invariant patch comparisons will be the subject of Section 3.

2. FAST NONLOCAL FILTERING

Regarding the computational complexity of nonlocal filters reveals that a price must be payed for the great results. At each pixel, weights to all other pixels have to be computed. This yields a computational complexity of $O(DN^2)$, where N is the number of pixels in the image, and D is the patch size. For larger images, this complexity is quite a burden. Hence, several approximations have been suggested.

The most popular way is to restrict the search to patches in a local neighborhood [15], which turns the initially nonlocal filter into a semi-local one. This reduces the computational complexity to $O(DN)$. Similarly, we can apply random sampling, where samples from the vicinity of the reference patch are preferred [16]. Both strategies assume that the most similar patches are in the vicinity of the reference patch. It has been shown that in many denoising tasks, semi-local filtering not only reduces the computational load, but even improves the denoising quality significantly. However, solving the complexity problem by retreating to a semi-local variant revokes the initial idea and properties of nonlocal filtering. It is definitely not the right strategy in case of applications where true nonlocal filtering is beneficial.

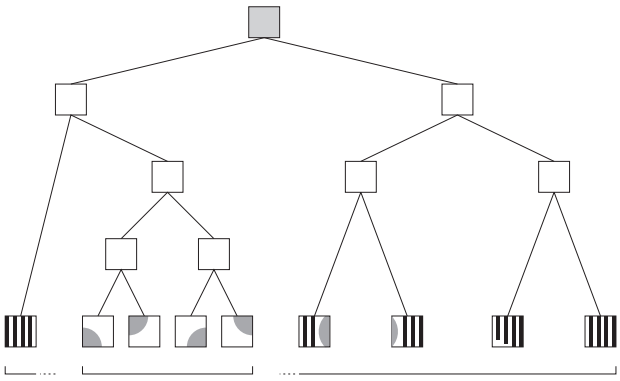

Figure 1. Schematic illustration of a cluster tree. Leafs contain a relatively small set of similar patches.

Speedups without necessarily abandoning the idea of nonlocal filtering have been achieved in [26, 17, 27, 23, 20]. In [26], patch comparison is performed only for a subset of reference patches lying on a coarser grid of the image. The computed weights are then used to restore a whole block of pixels at once. It is obvious that this approach can be used to gain a significant constant factor in the computational complexity, yet for the sake of quality the grid cannot be made arbitrarily coarse. In case of nonlocal filtering, we are hence left with the quadratic time complexity of the original filter.

2.1. Acceleration by preselecting patches

In patch based nonlocal filtering almost all computation time is spent on computing distances between patches. However, only a relatively small part of all patches is sufficiently similar for their kernel weights $K(x, y)$ to play a role in the averaging. Hence, in order to speed up the filter, the basic idea of the approaches in [17, 27] has been to compute distances only for a reduced set of patches. Preselection of patches is performed by some alternative distances, which can be computed very quickly, such as the difference of the patches' means or variances. Indeed, this strategy leads to a significant speedup, particularly in case of large patches. The disadvantage of this approach is that the preselection criterion is hardly related to the distance of patches. Two patches with same means and variances often comprise vastly different textural structures. Although this hardly harms the filtering outcome - for each patch in the preselected set the exact distance is computed - it reduces the efficiency of the method, as the preselected set still contains a large number of dissimilar patches.

2.2. Cluster trees

As a remedy to this problem, we proposed a different way to create sets of potentially similar patches [25]. This method preselects patches by means of the same distance measure that is used in the filtering. In order to accomplish this, the basic idea is to arrange all image patches in

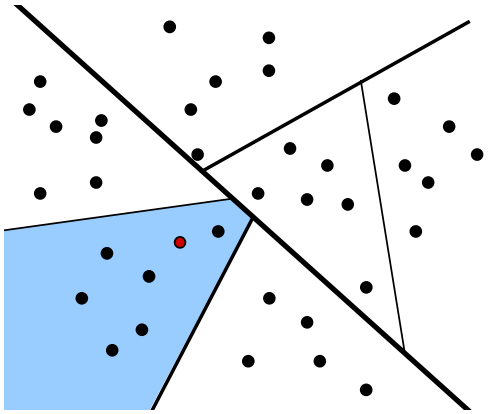

Figure 2. Arrangement of data by a cluster tree allows to quickly access a small set of points whose similarity is measured in patch space.

a binary tree. See Fig. 1 for an illustration. The root node of this tree contains all patches in the image. Performing a k-means clustering with $k = 2$ splits the patches into two clusters represented by the cluster means. These clusters represent the tree nodes at the first level. Each of these nodes can again be separated via k-means. This way, we can recursively build up the cluster tree.

The separation is stopped as soon as the number of patches in a node falls below a certain threshold $N_{min} = 30$. So each leaf node comprises at least N_{min} patches. Finding a local optimum with k-means takes linear time. It has to be applied at each level of the tree, and there are $\log N$ levels. Thus, building the tree runs in $O(DN \log N)$.

Once the tree is built up, we have immediate access to a set of similar patches in constant time. This is achieved by saving for each image patch its corresponding leaf node in an index table. The typical weighted averaging can then be efficiently applied on this subset. In contrast to previous preselection techniques, usage of the same distance for clustering and for filtering ensures preselection and filtering by means of one consistent criterion. Fig. 2 illustrates this preselection in patch space.

The concept of using tree structures to arrange data for efficient access in well known in computer science. The earliest ancestor of the above cluster tree is the so-called kd-tree [28]. The kd-tree always splits the space along coordinate axes, whereas the cluster tree splits the space by means of arbitrary hyperplanes. Moreover, whereas the kd-tree is optimized such that it leads to a balanced tree, the cluster tree seeks to obtain leafs that correspond to natural clusters of the data. The cluster tree concept is also known as tree structured vector quantization (TSVQ) and has been applied in speech processing [29], texture synthesis [30], and image retrieval [31]. The texture synthesis work in [30] is particularly close to the denoising case we consider here.

In a more general context, the cluster tree is an answer to the question how to find approximate nearest neighbors in high-dimensional data with sublinear time complexity. Other methodologies approaching this task can be found in [32, 33, 34], and a recent overview of approximate nearest neighbor algorithms is provided by [35].

2.3. Spill trees

Although the cluster tree arrangement allows to access near neighbors of a query very efficiently, it is obvious that the preselected set of patches is only an approximation of the exact set of nearest neighbors. Neighbors close to the query could be part of a neighboring cluster. Such situations appear especially when the query is close to a decision boundary in the tree. Unfortunately, the high-dimensional nature of the data comes along with a high probability that the query is close to at least one decision boundary.

The exact set of nearest neighbors could be computed by considering additional branches in the tree that may contain nearest neighbor candidates. However, this so-called backtracking decreases the efficiency of the approach considerably. In the case of texture denoising, backtracking effectively results in searching the whole tree.

Fortunately, an approximation of the set of nearest neighbors is sufficient for nonlocal filtering, as we typically deal with sufficiently large sets of similar patches and it is not necessary to have access to the *optimum* set of nearest neighbors to achieve a reasonable averaging of intensities.

Nonetheless, we might be interested in increasing the accuracy of the nearest neighbor sets. This is feasible with so-called spill trees [36] at the cost of some additional memory. Since the problematic areas are along the decision boundaries, we can assign patches close to such a boundary to both subsets. More precisely, a patch x is assigned to both subsets, if $d^2(x, c_1) < d^2(x, c_2) + \tau^2$ or, vice-versa, if $d^2(x, c_2) < d^2(x, c_1) + \tau^2$, where c_1 and c_2 denote the cluster centers of the two subsets; see Fig. 3(a). In case the overlap area τ is as large as the support of the kernel, we could ensure the exact set of relevant nearest neighbors. For the Gaussian kernel with infinite support this is not feasible and having large overlap areas also increases the time for building the tree, which reduces the speedup achievable with the cluster tree. However, already choosing small τ yields an accuracy close to the exact nonlocal means filter and does not demand much more memory than the version with $\tau = 0$.

2.4. Cluster forests

As an alternative strategy to increase the accuracy of nearest neighbor search we propose cluster forests. The key idea consists of combining multiple cluster trees. Having multiple trees with different decision boundaries, we can combine the patches found in each single tree and thereby decrease the probability to miss a nearby patch. This concept is illustrated in Figure 3(b).

The methodology of cluster forests bears similarities with another fast method for approximate near neighbor search called locality sensitive hashing (LSH) [33]. In LSH, randomized hash functions are chosen such that similar points are likely to produce the same hash index. By

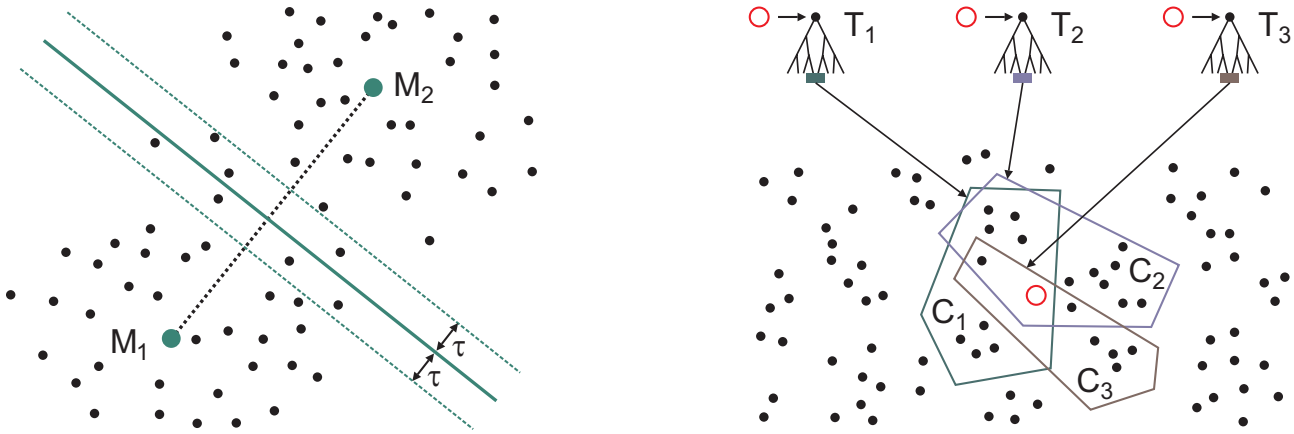

Figure 3. Two possibilities to increase accuracy. **Left:** (a) Patches that are within τ distance of a decision boundary are assigned to both sets. **Right:** (b) The union of near neighbors obtained from multiple trees are considered.

means of multiple randomized hash functions, the probability to find all points close to a query is increased. Upper bounds for the probability to miss a relevant point can be shown and it has been proven that this probability converges to zero in the limit of an infinite amount of hash functions.

Whereas LSH mainly builds upon the randomization of hash functions, cluster forests focus more on capturing the arrangement of the data. The randomization component is provided by the initialization of cluster centers in k-means. Since k-means only converges to the next local minimum of the cost function that aims to minimize the total distance of points to their assigned cluster centers, a randomized initialization leads to different clusters. Especially at nodes close to the root it is very likely that k-means clustering yields different optima, since there is usually no natural separation of the data into two clusters. At nodes close to the leafs, on the other hand, clusters in the data become apparent and are captured by the clustering.

The idea of cluster forests is to exploit both the randomization concept of locality sensitive hashing as well as the cluster arrangement of a single cluster tree. While the first leads to an increased accuracy when using more trees due to the different positioning of decision boundaries at lower levels of the tree, the latter ensures that high accuracies are already achieved for a relatively small number of trees.

These properties are shown in Table 1 and Table 2. Table 1 shows the percentage of correct nearest neighbors returned by a cluster forest. Data points were 9×9 patches from the Barbara test image shown in Figure 4. Also the query points were chosen randomly from this image. For not disturbing the statistics, the query point is always correctly found in a cluster tree, it was removed from the returned set of near neighbors. With a single tree, the nearest neighbor is only found in 65.3% of all cases. The percentage rises dramatically with the number of trees. With only 4 trees already more than 90% of all queries lead to the correct result.

Trees	k	Barbara (no noise)	Barbara ($\sigma = 20$)
1	1	65.3%	26.9%
2	1	81.5%	40.5%
4	1	93.2%	56.9%
8	1	97.4%	70.3%
16	1	97.6%	80.8%
1	10	53.8%	23.5%
2	10	73.9%	34.8%
4	10	87.0%	47.1%
8	10	93.4%	65.0%
16	10	95.7%	76.3%
1	100	42.4%	15.9%
2	100	59.5%	27.5%
4	100	77.2%	38.5%
8	100	86.2%	52.9%
16	100	91.5%	65.6%

Table 1. Average percentage of correct k-nearest neighbors depending of the number of trees in the forest. 1000 samples were drawn from the respective images. The accuracy grows rapidly with the number of trees.

Trees	k	Barbara (no noise)	Barbara ($\sigma = 20$)
1	1	1.095	1.047
2	1	1.034	1.027
4	1	1.012	1.015
8	1	1.004	1.008
16	1	1.004	1.005
1	10	1.170	1.062
2	10	1.071	1.039
4	10	1.030	1.024
8	10	1.016	1.012
16	10	1.010	1.007
1	100	1.191	1.103
2	100	1.097	1.062
4	100	1.049	1.038
8	100	1.027	1.022
16	100	1.012	1.013

Table 2. Average deviation of the k-nearest neighbor distance from the exact distance depending on the number of trees in the forest. 1000 samples were drawn from the respective images. The accuracy grows rapidly with the number of trees.

Figure 4. Standard test image *Barbara* of size 512×512 pixels including Gaussian noise with standard deviation 20 as used in the experiments.

Table 1 also shows that noise in an image complicates the search for the nearest neighbors. This is because noise brings the distribution closer to a uniform distribution, where points are more difficult to be arranged in distinct clusters. Here the randomized component introduced by multiple trees becomes more important and a larger amount of trees is needed to achieve the same percentage of correct neighbors.

For texture denoising, however, we are not mainly interested in finding a high percentage of exact nearest neighbors. We would rather like the returned neighbors to have a small distance to the query. Table 2 shows that the average deviation of the distance compared to that of the exact nearest neighbors is smaller in case of the noisy image. This means, not finding the exact nearest neighbors but only an approximate set of near neighbors is less significant if the noise level is higher. This explains the good performance already achieved with a single cluster tree as reported in [25] and shown in Section 2.6. With a cluster forest, the distance of the approximate nearest neighbors can be brought very close to the exact nearest neighbor distance. This holds true for both the noisy and noise free image.

2.5. Semi-local filtering with cluster trees

Empirical studies have shown that nonlocal filters often perform better when the search for similar patches is restricted to a local subdomain. This corresponds to the prior knowledge that best fitting patches are spatially close. Note that this assumption need not hold true for arbitrary data. Particularly in highly textured images, true nonlocal filtering leads to superior results [25]. However, for most natural images, semi-local filtering increases the denoising quality. For the cluster tree improvement to be practical in a large variety of filtering situations, it should provide the option to restrict the search space locally. This can be achieved by two strategies [25]. The algorithmic approach is by subdividing the image into overlapping blocks. A filtering result using cluster trees is computed independently for each block. The final result is then obtained as the weighted average of the block results.

A more profound approach with the same effect is to shift the locality assumption to the distance measure by having a distance that consists of the patch similarity and the local proximity: $\tilde{d}^2(x,y) = d^2(x,y) + \gamma|x-y|^2$ with $d^2(x,y)$ as defined in (2) and γ steering the importance of the locality constraint, i.e., the size of the neighborhood. This bilateral filter version is more sound and in combination with cluster trees it is usually even faster than the algorithmic way to enforce the locality constraint.

2.6. Experiments

In the experimental evaluation, we quantified the speedup of cluster tree implementations relative to their loss in quality measured by the peak signal-to-noise ratio (PSNR)

$$\mathrm{PSNR} = 10 \log_{10} \left(\frac{255^2}{\frac{1}{|\Omega|} \sum_{i \in \Omega} (u_i - r_i)^2} \right), \quad (3)$$

where u denotes the restored image and r the noise free reference image. In all experiments we used the same Gaussian weighted 9×9 patches ($\rho = 2$). Image intensities were in a range between 0 and 255. Computations times are on a Pentium IV 3.4GHz.

Table 3 shows a comparison of acceleration techniques that have been suggested in the literature. For the comparison we used the 512×512 Barbara test image from Fig.4. Given a fixed local search window size of 21×21 pixels,

	PSNR	Time
Standard nonlocal means	30.31	41940ms
Random sampling, 100 samples	29.55	25410ms
Random sampling, 200 samples	30.16	72480ms
Spatial sub-sampling	29.51	5480ms
Preselection by mean and variance	29.80	17640ms
Cluster tree, $\tau = 0$	29.90	14440ms
Cluster tree, $\tau = 5$	30.08	19580ms
Cluster tree, $\tau = 10$	30.26	31330ms
Cluster forest, 2 trees	30.37	24100ms
Cluster forest, 4 trees	30.53	38144ms

Table 3. Comparison of several fast nonlocal means implementations using the Barbara test image and a 21×21 search window. Regarding both quality and computation time, the cluster tree implementation compares well to existing techniques; see also Fig. 5. Particularly the combination of multiple trees yields very accurate results.

the fastest technique is spatial sub-sampling, i.e., using the distance computed for a point x also for its 8 neighbors [26]. However, the speedup of this method comes at a cost. Fig. 5 shows a zoom into the Barbara image. The spatial sub-sampling clearly blurs the image, which is reflected by the lowest PSNR value. The basic cluster tree implementation is the second fastest method without causing blurring artifacts. It outperforms the preselection of patches by mean and variance [27] as well as random sampling. Comparing the spill tree to cluster forests, it can be observed that the cluster forest yields more accurate results at the same computational load. The results can even outperform the standard nonlocal means approach, which uses the exact weighted sum over all patches within the search window. An explanation for this outcome could be the long tail of the Gaussian kernel, which causes standard nonlocal means to take into account also very dissimilar patches (though with very small weights). This leads to a restrained blurring effect, which is not present in case of the cluster forest that averages only over a restricted set of near neighbors.

Although the computational complexity of cluster forests increases linearly with the number of trees, the empirical numbers show that the increase is in fact smaller. This is due to the duplicates returned by a cluster forest. Removing these duplicates renders the cardinality of the set of near neighbors returned by multiple trees not to increase linearly with the number of trees. Moreover, building the tree takes less time than computing the distances between all queries and their sets of near neighbors as needed for the nonlocal filter, see Figure 6.

In the first experiment we considered semi-local filtering with a restricted search window. Due to the sublinear time complexity for finding a subset of near neighbors, the speedup achievable with the cluster tree or cluster forest implementation becomes much more significant when the search window size increases. This is shown in Table 4. Particularly true nonlocal filtering, where the search window is the whole image domain, becomes very demanding for methods that scale linearly with the size of this window.

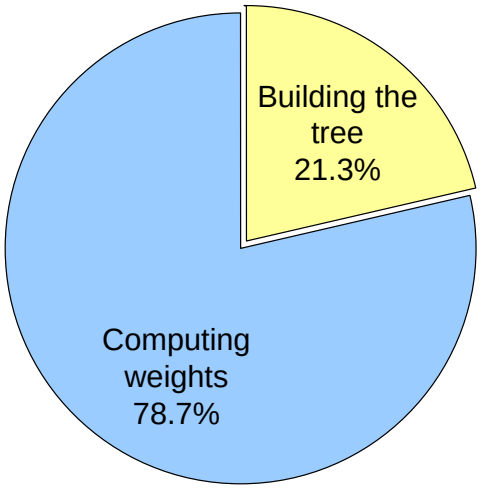

Figure 6. Relative time taken for building a single tree without overlap The image was of size 512×512 pixels. Although building the tree dominates the asymptotic complexity, in practice, most of the time is spent for computing distances of the selected patches.

3. INVARIANT TRANSFORMATIONS

The original nonlocal means filter is based on the patch similarity measure d defined in equation (2). Obviously this measure is not invariant to a number of transformations which may arise in applications to natural images, namely variations in brightness, rotation, and scale.

In the following, we will present methods to make patch comparison and denoising invariant to such variations. As a consequence, denoising can be improved since the algorithm can exploit information from patches which vary in brightness, rotation, or scale.

All subsequent modifications can be efficiently integrated in the cluster tree structure. As a consequence, computations times increase only mildly. We will omit computational details for the sake of simplicity.

3.1. Brightness Invariance

Due to variations in the local lighting conditions common in many natural images the same textural patches may appear in different brightness conditions in two given locations x and y of the image I. Although these locations do correspond to the same underlying pattern, this information is not exploited since the patch d similarity measured by (2) is small.

A straight-forward normalization is obtained by removing the average intensity $\mu = G\rho \star I$ over each patch:

$$d^2(x,y) = \int_\Omega G_\rho(x') \big(I(x-x') - I(y-x') - (\mu(x) - \mu(y))\big)^2 \, dx' \tag{4}$$

Figure 5. Zoom into the Barbara image. **Top row, from left to right:** (a) Noisy input image. (b) Standard nonlocal means. (c) Random sampling with 100 samples. **Middle row:** (d) Spatial sub-sampling. Blurring and block artifacts are visible. (e) Preselection by mean and variance. (f) Cluster tree with no overlap. **Bottom row:** (g) Cluster tree with $\tau = 5$. (h) Cluster forest with 2 trees. (i) Cluster forest with 4 trees. All methods were run with a 21×21 search window. The quality with the cluster tree implementation is at least as good as or better than that of other acceleration techniques; see Table 3 for quantitative results.

Additional invariance to multiplicative brightness changes can be obtained by considering the normalized cross-correlation. However, we did not find substantial improvements in the denoising results.

Of course, the change in average brightness needs to be taken into account in the resulting filtering operation:

$$u(x) = \frac{1}{C_K(x)} \int_\Omega K(x,y)(I(y) + (\mu(x) - \mu(y))\,)\,dy$$
$$C_K(x) = \int_\Omega K(x,y)\,dy \qquad (5)$$

Figure 8 shows that considering brightness-invariant patch distances does indeed lead to improvements in the signal-to-noise ratio of the denoised images.

3.2. Invariance to Rotations

Another common feature of natural textures is that the same textural pattern will appear in other locations of the image in a rotated version. Again, the original patch similarity measure (2) does not explicitly capture this variability. As a consequence, the denoising of a patch located at x does not profit from similar patches located at y if these are rotated with respect to the patch at x.

In contrast to the case of brightness variation considered above, the variation in rotation is substantially more

	17×17	33×33	65×65	129×129	257×257	no window
Standard nonlocal means	27s	106s	410s	1539s	5378s	16107s
Random sampling	13s	50s	209s	902s	4126s	15362s
Spatial sub-sampling	**4s**	15s	65s	262s	957s	3241s
Preselection by mean and variance	13s	34s	88s	221s	591s	1529s
Cluster tree, $\tau = 0$	14s	**14s**	**14s**	**14s**	**14s**	**14s**
Cluster tree, $\tau = 5$	15s	31s	58s	101s	101s	101s
Cluster tree, $\tau = 10$	22s	61s	149s	303s	687s	797s
Cluster forest, 2 trees	20s	22s	22s	22s	22s	22s
Cluster forest, 4 trees	30s	43s	43s	43s	43s	43s
Cluster forest, 8 trees	47s	88s	88s	88s	88s	88s

Table 4. Computation times of several fast nonlocal means implementations depending on the search window size. The methods were run with the 512×512 Barbara test image. For growing window sizes, the cluster tree implementation is increasingly faster than existing techniques. Particularly in case of true nonlocal filtering the speedup is remarkable.

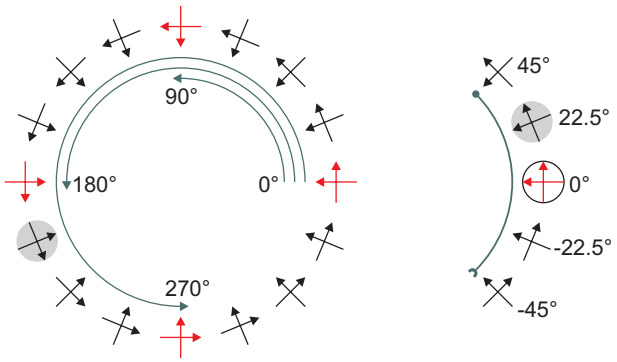

Figure 7. To implement rotation invariance, slightly rotated versions of all patches can be added to the cluster tree. Alternatively or additionally, coarsely rotated versions (by $0°$, $90°$, $180°$, and $270°$) can be considered in place of a single, non-rotated query.

difficult to take into account. Specifically, for realistic patches it is difficult to automatically select a meaningful normalization. We therefore propose a numerical approximation where we merely consider a predefined discrete set $A \subset [0, 2\pi)$ of permissible rotation angles. Subsequently, invariance to these rotation angles can be introduced into the patch similarity measure by simply finding the best rotation to align the two patches centered at x and y:

$$d^2(x,y) = \min_{\phi \in A} \int_\Omega G_\rho(x') \big(I(x-x') - R_\phi I(R_\phi y - x')\big)^2 \, dx', \tag{6}$$

where R_ϕ denotes a rotation (of the image) by an angle ϕ. As in equation (5), the nonlocal filter must be adapted appropriately to take into account the optimal rotation angle.

Figure 9 shows that already considering a very coarse set of 90-rotations provides improvements of signal-to-noise ratios. A more finely sampled set of rotation angles gives rise to even better denoising.

3.3. Invariance to Scale Variation

A common variation of natural textures comes about due to variations in the scale of patterns. In particular due to perspective transformations, more distant textures will be shrunk with respect to the textures closer to the viewer. Again, the classical nonlocal means filter cannot exploit the similarity of structures over different spatial scales: A given patch at location y will not support the denoising of similar patches at location x if the latter one has a different scale.

As in the case of angle variations, a normalization of patches with respect to spatial scale changes is by no means straight-forward. Again, we propose a numerical approximation obtained by considering a discrete set $A \subset (0, \infty)$ of scaling factors. A patch similarity measure which is invariant to scaling in this set of scales is given by:

$$d^2(x,y) = \min_{\sigma \in A} \int_\Omega G_\rho(x') \left(I(x-x') - S_\sigma I(S_\sigma y - x')\right)^2 \, dx', \tag{7}$$

where S_σ denotes a scaling of the image by the factor σ. As for the case of brightness and angle variation, the respective nonlocal means filter needs to be adapted appropriately.

In contrast to rotations which are always in the interval $[0, 2\pi]$, there is no natural bound on the choice of meaningful scale parameters. Therefore the user needs to define a discrete set of meaningful scale parameters. In contrast to the rotation case, where even coarse sets of rotation angles lead to improvements of the signal-to-noise ratio, this is no longer the case for scale changes. Figure 10 shows that a very coarse set of permissible scales $A = \{0.5, 0.71, 1, 1.41, 2\}$ leads to a slight decay of the PSNR from 20.89 to 20.64. A meaningful and sufficiently fine set of scales such as $A = \{0.84, 0.92, 1, 1.09, 1.19\}$ increases the PSNR to 20.95.

4. CONCLUSION

The denoising of textural patterns has recently been addressed by a number of nonlocal filtering approaches such

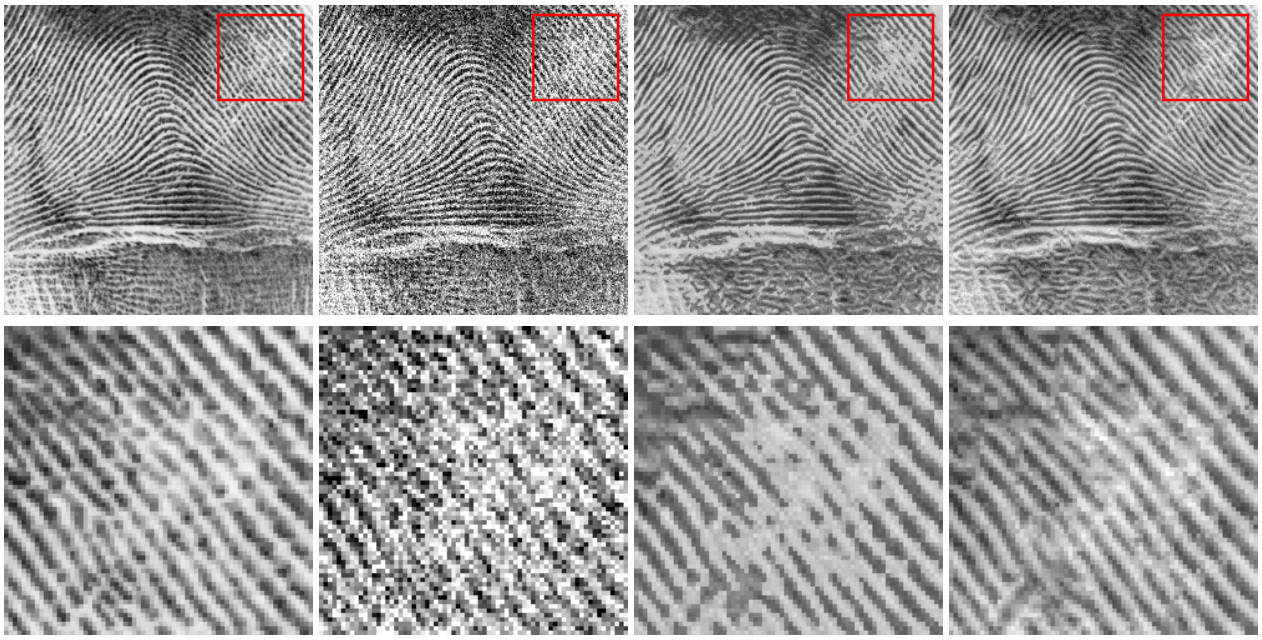

Figure 8. Brightness invariance in texture denoising. **From left to right:** (a) Input image. (b) Gaussian noise with $\sigma = 40$ added. (c) Result with a brightness sensitive distance ($PSNR = 19.28$). (d) Result with a brightness invariant distance ($PSNR = 20.35$). **Bottom:** Zoom into the marked region. Considering brightness transformations leads to significant visual improvements.

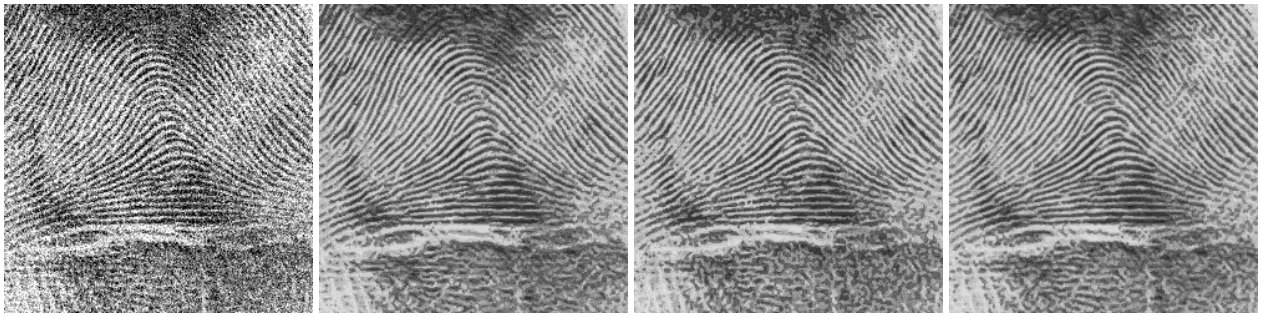

Figure 9. Rotation invariance in texture denoising. **From left to right:** (a) Input image with Gaussian noise added, $\sigma = 40$. (b) Result without considering rotated patches ($PSNR = 19.28$). (c) Patches rotated by $0°$, $90°$, $180°$, and $270°$ considered in distance ($PSNR = 19.32$). (d) Patches rotated by $-20°$, $-10°$, $0°$, $10°$, and $20°$ considered in distance ($PSNR = 19.83$). Considering rotated patches leads to improved PSNR values, though the improvement is smaller than in case of brightness transformations.

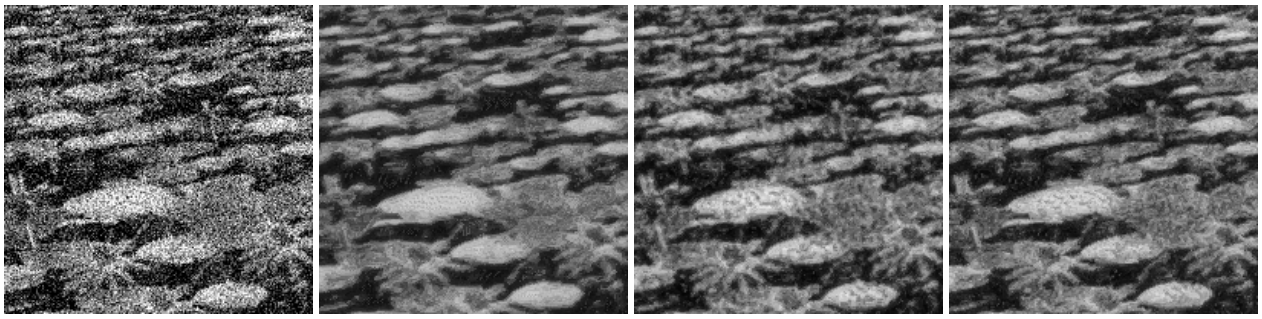

Figure 10. Scale invariance in texture denoising. **From left to right:** (a) Input image with Gaussian noise added, $\sigma = 40$. (b) Result without considering scaled patches ($PSNR = 20.89$). (c) Patches scaled by 0.5, 0.71, 1, 1.41, and 2 ($PSNR = 20.64$). (d) Patches scaled by 0.84, 0.92, 1, 1.09, and 1.19 ($PSNR = 20.95$). Results can even be inferior when considering scaled patches.

as the nonlocal means filter. In this paper, we addressed two important shortcomings of the original nonlocal means filter: Firstly, we proposed and quantitatively compared a number of speed-up techniques based on efficient data structures such as cluster trees, spill trees, and cluster forests indicating that the implementation via cluster forests provides best results. Secondly, we proposed ways to make the nonlocal means filtering robust to variations of the textural patches in brightness, scale, and rotation. Experiments indicate that considering appropriate scales and rotations leads to substantial improvements of the denoising performance.

Acknowledgements

We gratefully acknowledge funding by the German Research Foundation (DFG), grant #DFG-CR-250/1-2.

5. REFERENCES

[1] P. Perona and J. Malik, "Scale space and edge detection using anisotropic diffusion," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 12, pp. 629–639, 1990.

[2] L. I. Rudin, S. Osher, and E. Fatemi, "Nonlinear total variation based noise removal algorithms," *Physica D*, vol. 60, pp. 259–268, 1992.

[3] D. L. Donoho, "De-noising by soft thresholding," *IEEE Transactions on Information Theory*, vol. 41, pp. 613–627, 1995.

[4] L. P. Yaroslavsky, *Digital Picture Processing - An Introduction*, Springer, 1985.

[5] C. K. Chu, I. Glad, F. Godtliebsen, and J. S. Marron, "Edge-preserving smoothers for image processing," *Journal of the American Statistical Association*, vol. 93, no. 442, pp. 526–556, 1998.

[6] G. Winkler, V. Aurich, K. Hahn, and A. Martin, "Noise reduction in images: some recent edge-preserving methods," *Pattern Recognition and Image Analysis*, vol. 9, no. 4, pp. 749–766, 1999.

[7] S. M. Smith and J. M. Brady, "SUSAN: A new approach to low-level image processing," *International Journal of Computer Vision*, vol. 23, no. 1, pp. 45–78, May 1997.

[8] C. Tomasi and R. Manduchi, "Bilateral filtering for gray and color images," in *Proc. Sixth International Conference on Computer Vision*, Bombay, India, Jan. 1998, pp. 839–846, Narosa Publishing House.

[9] D. Barash, "A fundamental relationship between bilateral filtering, adaptive smoothing and the nonlinear diffusion equation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 6, pp. 844–847, 2002.

[10] P. Mrázek, J. Weickert, and A. Bruhn, "On robust estimation and smoothing with spatial and tonal kernels," in *Geometric Properties from Incomplete Data*, R. Klette, R. Kozera, L. Noakes, and J. Weickert, Eds., pp. 335–352. Springer, Dordrecht, 2006.

[11] A. Buades, B. Coll, and J. M. Morel, "The staircasing effect in neighborhood filters and its solution," *IEEE Transactions on Image Processing*, vol. 15, no. 6, pp. 1499–1505, 2006.

[12] K. Popat and R. Picard, "Novel cluster-based probability model for texture synthesis, classification, and compression," in *Proc. SPIE Visual Communications and Image Processing*, 1993.

[13] A. Efros and T. Leung, "Texture synthesis by non-parametric sampling," in *Proc. International Conference on Computer Vision*, Corfu, Greece, Sept. 1999, pp. 1033–1038.

[14] L.-Y. Wei and M. Levoy, "Deterministic texture analysis and synthesis using tree structure vector quantization," in *Proc. Brazilian Symposium on Computer Graphics and Image Processing*, 1999, pp. 207–214.

[15] A. Buades, B. Coll, and J. M. Morel, "A non-local algorithm for image denoising," in *Proc. International Conference on Computer Vision and Pattern Recognition*, 2005, pp. 60–65.

[16] S. Awate and R. Whitaker, "Higher-order image statistics for unsupervised, information-theoretic, adaptive image filtering," in *Proc. International Conference on Computer Vision and Pattern Recognition*, 2005, pp. 44–51.

[17] M. Mahmoudi and G. Sapiro, "Fast image and video denoising via nonlocal means of similar neighborhoods," *Signal Processing Letters*, vol. 12, no. 12, pp. 839–842, 2005.

[18] S. Yoshizawa, A. Belyaev, and H.-P. Seidel, "Smoothing by example: mesh denoising by averaging with similarity-based weights," in *Proc. International Conference on Shape Modeling and Applications*, June 2006, pp. 38–44.

[19] S. Awate and R. Whitaker, "Unsupervised, information-theoretic, adaptive image filtering for image restoration," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 3, pp. 364–376, Mar. 2006.

[20] C. Kervrann, J. Boulanger, and P. Coupé, "Bayesian non-local means filter, image redundancy and adaptive dictionaries for noise removal," in *Scale Space and Variational Methods in Computer Vision*, F. Sgallari, A. Murli, and N. Paragios, Eds. 2007, vol. 4485 of *LNCS*, pp. 520–532, Springer.

[21] C. Kervrann and J. Boulanger, "Local adaptivity to variable smoothness for exemplar-based image denoising and representation," *International Journal of Computer Vision*, To appear.

[22] S. Kindermann, S. Osher, and P. W. Jones, "Deblurring and denoising of images by nonlocal functionals," *SIAM Interdisciplinary Journal*, vol. 4, no. 4, pp. 1091–1115, 2005.

[23] G. Gilboa and S. Osher, "Nonlocal linear image regularization and supervised segmentation," Tech. Rep. CAM-06-47, Department of Mathematics, University of California at Los Angeles, CA, U.S.A., Sept. 2006.

[24] G. Gilboa, J. Darbon, S. Osher, and T. Chan, "Nonlocal convex functionals for image regularization," Tech. Rep. CAM-06-57, Department of Mathematics, University of California at Los Angeles, CA, U.S.A., Oct. 2006.

[25] O. Kleinschmidt T. Brox and D. Cremers, "Efficient nonlocal means for denoising of textural patterns," *IEEE Transactions on Image Processing*, vol. 17, no. 7, July 2008.

[26] A. Buades, B. Coll, and J. M. Morel, "A review of image denoising algorithms, with a new one," *SIAM Interdisciplanary Journal*, vol. 4, no. 2, pp. 490–530, 2005.

[27] P. Coupé, P. Yger, and C. Barillot, "Fast nonlocal means denosing for 3D MR images," in *Proc. International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, R. Larsen, M. Nielsen, and J. Sporring, Eds., 2006, vol. 4191 of *LNCS*, pp. 33–40.

[28] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *Communications of the ACM*, vol. 18, no. 9, pp. 509–517, 1975.

[29] A. Buzo, A. H. Gray, R. M. Gray, and J. D. Markel, "Speech coding based upon vector quantization," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 28, no. 5, pp. 562–574, 1980.

[30] L.-Y. Wei and M. Levoy, "Fast texture synthesis using tree-structured vector quantization," in *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 2000.

[31] D. Nistér and H. Stewénius, "Scalable recognition with a vocabulary tree," in *Proc. International Conference on Computer Vision and Pattern Recognition*, 2006, pp. 2161–2168.

[32] S. Nene and S. Nayar, "A simple algorithm for nearest neighbor search in high dimensions," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 9, pp. 989–1003, 1997.

[33] P. Indyk and R. Motwani, "Approximate nearest neighbors: towards removing the curse of dimensionality," in *Proc. 13th Annual ACM Symposium on Theory of Computing*, 1998, pp. 604–613.

[34] D. Omerčević, O. Drbohlav, and A. Leonardis, "High-dimensional feature matching: employing the concept of meaningful nearest neighbors," in *Proc. International Conference on Computer Vision*, 2007.

[35] T. Darell, P. Indyk, and G. Shakhnarovich, Eds., *Nearest Neighbor Methods in Learning and Vision: Theory and Practice*, MIT Press, 2006.

[36] T. Liu, A. Moore, A. Gray, and K. Yang, "An investigation of practical approximate nearest neighbor algorithms," in *Proc. Neural Information Processing Systems*, 2005, pp. 825–832.