Channel smoothing: Efficient robust smoothing of low-level signal features

Michael Felsberg, P.-E. Forssen and H. Scharr

Michael Felsberg, P.-E. Forssen and H. Scharr, Channel smoothing: Efficient robust smoothing of low-level signal features, 2006, IEEE Transaction on Pattern Analysis and Machine Intelligence, (28), 2, 209-222.
http://dx.doi.org/10.1109/TPAMI.2006.29

Channel Smoothing: Efficient Robust Smoothing of Low-Level Signal Features

Michael Felsberg, Per-Erik Forssén, Hanno Scharr

M. Felsberg and P.-E. Forssén are with the Computer Vision Laboratory, Dept. EE, Linköping University, 581 83 Linköping, Sweden. E-mail: mfe@isy.liu.se, perfo@isy.liu.se.

H. Scharr is with the Forschungszentrum Jülich GmbH, ICG-III, 52425 Jülich, Germany. E-mail: h.scharr@fz-juelich.de

Abstract

In this paper we present a new and efficient method to implement robust smoothing of low-level signal features: B-spline channel smoothing. This method consists of three steps: encoding of the signal features into *channels*, averaging of the channels, and decoding of the channels. We show that linear smoothing of channels is equivalent to robust smoothing of the signal features if we make use of quadratic B-splines to generate the channels. The linear decoding from B-spline channels allows the derivation of a robust error norm, which is very similar to Tukey's biweight error norm. We compare channel smoothing with three other robust smoothing techniques: non-linear diffusion, bilateral filtering, and mean-shift filtering, both theoretically, and on a 2D orientation-data smoothing task. Channel smoothing is found to be superior in four respects: it has a lower computational complexity, it is easy to implement, it chooses the global minimum error instead of the nearest local minimum, and it can also be used on non-linear spaces, such as orientation space.

Index Terms

robust smoothing, channel representation, diffusion filtering, bilateral filtering, mean-shift, B-spline, orientation smoothingrobust smoothing, channel representation, diffusion filtering, bilateral filtering, mean-shift, B-spline, orientation smoothing

I. INTRODUCTION

The aim of this paper is to derive and to investigate *channel smoothing* based on B-splines as a new way to implement robust smoothing of low-level signal features with the focus on image processing and computer vision applications. By signal features we mean dense estimates of model parameters based on measured signals, for instance the gray level of an image, the local orientation field of a multi-dimensional signal, the disparity map between two images, or the flow field of an image sequence. The method is, however, much more general and can be applied to any kind of regularly sampled parameter space.

Robust smoothing is based on a robust error norm and is often preferable compared to linear smoothing, i.e., optimization based on the L_2-norm. Least-squares optimization is much more

sensitive to outliers and cannot deal with stochastic processes containing non-stationary parts. Features of real images and image sequences typically contain outliers and discontinuities. Applying least-squares optimization to these features therefore suffers from two effects: First, outliers result in an offset of the estimate. Second, the data on both sides of a discontinuity (e.g. an edge) are treated as if they belong to the same stochastic process, yielding a blurring of the discontinuity. Robust smoothing, however, is well suited for processing noisy features, since it neglects outliers and preserves discontinuities for an appropriate choice of the error norm [1].

The main drawback of robust smoothing is the high computational complexity of its evaluation. It is usually implemented as an iterative algorithm to find local solutions (e.g. diffusion filtering [2], bilateral filtering [3], [4][1], mean-shift [9], [10], [11]), since the underlying optimization problem is non-convex. Concerning computational complexity, least-squares methods are preferable, since they are equivalent to solving a linear problem. Hence, the simplicity and effectiveness of the least-squares solution often misleads developers of signal processing algorithms to use least-squares optimization although robust techniques are more appropriate to the problem.

In this paper we present a method to implement robust smoothing which avoids the high computational complexity of the methods known in the literature: *Channel smoothing* [12], [13]. Let us introduce the idea of channel smoothing by a simple example, channel smoothing of a 1D scalar signal $f(x)$. The method consists of three steps:

1) encoding of a signal into N channels,
2) smoothing of each channel separately, and
3) decoding of the channels into a 1D signal.

The encoding step then assigns channel weights $c_n(x)$ to signal values $f(x)$ (compare Fig. 1, where dot sizes correspond to sample values of $c_n(x)$):

$$c_n(x) = F_n(f(x)) \qquad n = 1, \ldots, N \ ,$$

where the lower index n indicates the channel number and F_n is an encoding function. This function F_n has its single maximum at $f = n$, decays when $|f - n|$ becomes larger, and $F_n = 0$, if $|f - n| > b$, where b is some upper bound given by the encoding function. The decoding step

[1]Bilateral filtering is equivalent to the non-linear Gaussian filter in [5], [6], [7] and the SUSAN noise filtering technique [8].

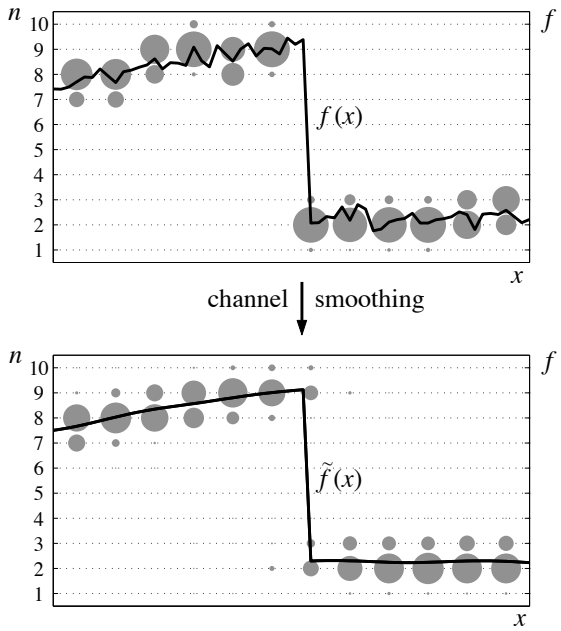

Fig. 1. Channel smoothing of a 1D signal. Top: the signal (solid line) and its channel representation (dots). Bottom: the averaged channels (dots) and the decoded signal (solid line). The dots are located at (x_s, n), where x_s are samples of x. The sizes of the dots indicate the channel values, i.e., the sample values $c_n(x_s)$.

is essentially the inverse of the encoding:

$$\tilde{f}(x) = \tilde{F}(c_1(x), \ldots, c_N(x)) \quad .$$

Thus, encoding followed by decoding yields identity $\tilde{f}(x) = f(x)$. If the signal is altered (e.g. smoothed) in-between encoding and decoding, things are less simple. In Fig. 1 top, for a given sample position x_s there is one value $f(x_s)$ encoded in three neighboring channels. Averaging each channel yields the result in Fig. 1 bottom, where close to the discontinuity, six or more channels are non-zero. In those points the channel representation contains multiple modes[2] (compare Fig. 2, where the situation for a fixed location x_s is depicted). As we see there, we have to decide if we decode the left or right portion of the signal via a decoding window. This corresponds to robust statistics, where the non-used data is rejected as outliers. Using the full data without a decoding window simply returns an average value, i.e. no robustness is obtained.

[2]Multiple modes also occur when multiple signals (e.g. from multiple measurements) are encoded in the same channels.

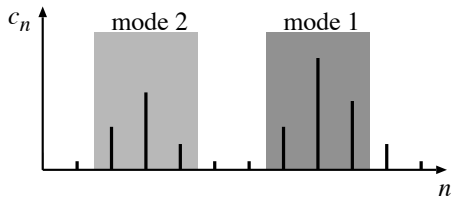

Fig. 2. Two possible modes obtained by local decoding.

The proposed implementation of robust smoothing simplifies and speeds up the computation significantly, without introducing relevant inaccuracies. Evidently, this approach is not restricted to 1D signals, nor is it restricted to scalar-valued features, but vector-valued features and new specific properties arising for nD signals (e.g. anisotropic averaging of the channels [14]) are beyond the scope of this paper.

In what follows, we consider nD feature functions as nD surfaces in an $(n+1)$D space,[3] e.g., the local orientation of an image forms a surface in 3D space. Therefore, we implicitly assume *labeled-line coding* [16] of the data. In order to generate a tractable data volume, the $(n+1)$D space is filtered and subsampled along the feature dimension. As a result, a stack of nD signals, the *channels*, are obtained (the first step, *encoding*). The channel set is called the *channel representation* [17], [18], [16] of the feature data, which resembles *population coding* [19] from computational neurobiology.

In the second step (*smoothing*), each channel is convolved separately with an averaging kernel, e.g. a Gaussian kernel. In the third step (*decoding*), the resulting channel representation is reduced to intensity coding again, i.e., the smoothed feature map. If the decoding was a linear method, all three steps could be combined in a single linear operator. However, since we are aiming at a robust non-linear technique, we apply local decoding.

Previous decoding methods [20], [21] do not lead to a result that is equivalent to robust smoothing in general. Their output shows systematic distortions depending on the absolute positions of the channel centers. This *quantization effect of the channel representation* can be avoided by an appropriate decoding which is also a topic of this paper. As a consequence, the

[3]This is basically the same idea as in the Beltrami framework [15].

channel smoothing method presented in this paper is equivalent to robust smoothing.
The paper is organized as follows:

In Sect. II we motivate a spline approximation of the robust error norm and its influence function through constraints on locality, linearity, simplicity, and stability. Requiring minimal curvature of the error norm singles out cubic B-splines for the error norm and quadratic B-splines for the influence function. A brief introduction to spline approximations is given.

In Sect. III we introduce the quadratic B-spline channel representation, its encoding and local decoding. Averaging the channels leads to properties known from robust smoothing. However, the output suffers from a quantization effect which is avoided by the virtual shift decoding scheme.

In Sect. IV we relate channel smoothing to three other approaches which implement robust smoothing: non-linear diffusion [22], [23], bilateral filtering [5], [24], and the mean-shift approach [11]. In the comparisons, we address outlier suppression and computational complexity on a theoretical level and the accuracy of the robust estimates on an experimental level.

The paper ends with a concluding section, acknowledgments, and references.

II. B-Spline Approximation of a Robust Error Norm

In the first part of this section, we motivate our choice of a B-spline robust error norm approximation by formulating several practical constraints. In the second part, we give a brief introduction to B-spline interpolation and explain how to approximate general robust error norms. Finally, the approximation of the corresponding influence functions is derived.

A. Why Using B-Splines?

The choice of approximation method for robust smoothing is not uniquely determined and to some extent heuristic. However, there are several practical considerations that clearly favor B-spline interpolation. In particular:

1) The approximation should be linear in order to allow the derivation of general theorems.
2) The approximation should be local, since we want to minimize the number of memory accesses.
3) The approximation should be simple, i.e., it should contain few coefficients, since all subsequent computations must be applied to the coefficients.

4) The approximation should be smooth and non-oscillating, in order to result in a stable method.

The linearity constraint implies that we have to use a linear transform (e.g. Fourier transform, power series, etc.). Taking into account the second constraint, all global transformations are excluded. The remaining possible methods include local linear transforms, in particular (certain) wavelet transforms and all kinds of piecewise defined power series. The local Fourier transform is excluded, since we require locality in a strict sense, i.e., compact support.

The constraint of a minimum number of coefficients implies that the piecewise defined power series is reduced to a piecewise defined polynomial approximation, i.e., a spline approximation. Requiring smoothness for a wavelet transform with finite support is best fulfilled in case of Daubechies wavelets ([25], Sect. 2.4). However, their order of smoothness grows much slower than that of splines (factor 5, [25], pg. 173), that means more coefficients are required for an approximation than in the case of splines. Hence, minimizing the number of coefficients excludes wavelets as a choice.

Finally, in order to have the least possible oscillations in the approximation, we use cubic splines, since these have minimum curvature in curvilinear coordinates [26]. In what follows, we therefore make use of cubic B-splines in order to approximate robust error norms. For obtaining compact support, the robust error norms have to converge to a constant value for large arguments. Furthermore, they must be zero for argument zero. These properties have to be fulfilled by the approximation as well, i.e., for certain points the approximation must be exact. Therefore, we do not use the B-spline *approximation* method [27] but the B-spline *interpolation* method to approximate the error norm.

B. B-Spline Interpolation

Function interpolation with B-splines[4] is a well-known technique [29], [27]. Therefore, we just summarize the required definitions of centralized[5] B-spline interpolation with unit distance henceforth.

[4]In the literature the term 'B-spline' is used in different contexts. Sometimes the whole linear combination of coefficients and basis functions is called a B-spline ('B' like Bézier?) [28] and sometimes B-spline indicates only the basis function ('B' for basis) [27]. Throughout this paper we make use of the second alternative.

[5]Centralized B-splines are symmetric about zero.

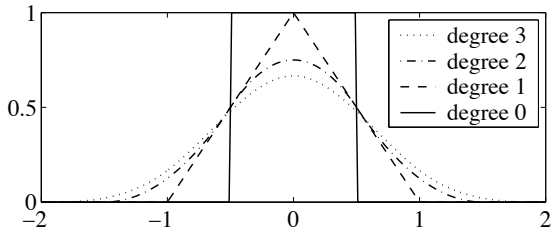

Fig. 3. B-spline basis functions of degree zero to three.

The B-spline of degree $k \geq 0$, cf. also Fig. 3, is defined by[6]

$$B_0(f) = \Pi(f) \tag{1}$$

$$B_k(f) = (B_0 * B_{k-1})(f) = \frac{k+2f+1}{2k} B_{k-1}(f+\frac{1}{2}) + \frac{k-2f+1}{2k} B_{k-1}(f-\frac{1}{2}) \tag{2}$$

where $\Pi(f)$ is the rectangle function [30], pg. 52, and $*$ denotes the convolution operator. The B-spline interpolation of a function $P(f)$ is obtained by a linear combination of integer shifted B-splines:

$$P(f) \approx \sum_{n \in \mathbb{Z}} \alpha_n B_k(f-n) \ . \tag{3}$$

In practice this sum is finite and without loss of generality we can assume that n is an integer in $[1, N]$, such that $(k+1)/2$ and $N-(k-1)/2$ are the bounds of the approximation interval. We obtain other approximation intervals by a simple scaling and shift of the origin.

The linear coefficients α_n are obtained by recursive filtering [27] or equivalently by solving the $N-k+1$ interpolation conditions

$$P(m) = \sum_{n=1}^{N} \alpha_n B_k(m-n) \tag{4}$$

for all $m \in (k+1)/2 + [0, N-k]$. The number of unknowns is N, and hence, we need $k-1$ boundary conditions in order to obtain a unique solution.

In what follows, we consider optimization problems where the error functional

$$E(f_0) = \int \rho(f-f_0)\mathrm{pdf}(f)\, df = (\rho * \mathrm{pdf})(f_0) \tag{5}$$

[6]Since we apply the B-splines to (stochastic) signals $f(\mathbf{x})$ below, we use f as the argument in this section. Furthermore, the spatial argument $\mathbf{x}$ is omitted if it is not relevant.

shall be minimized. In this functional, $\rho(\cdot)$ denotes the robust error norm and $\mathrm{pdf}(f)$ is the probability density function of f. Since we want to approximate the robust error norm $\rho(\cdot)$ with minimum curvature, we choose $k = 3$, i.e., cubic splines, and therefore need two boundary conditions. Assuming that the robust error norm is non-constant only on a finite interval, which is the case for many norms (e.g. truncated quadratic, Tukey's biweight, GNC function, Hampel's function, and Andrew's sine) [31], the approximation interval is equal to that non-constant part of the norm. Furthermore, the approximation should have zero slope at the boundary, which results in the condition $\rho' = 0$ at both boundaries.

For error norms that only converge toward a constant value, without reaching it, one has to truncate the error norm at a certain point. The boundary conditions are then given by the derivative of the norm at the boundaries.[7]

C. Approximation of the Influence Function

At a minimum (*mode*) of the error functional (5), its derivative must vanish:

$$0 = \left.\frac{\partial E(f_0)}{\partial f_0}\right|_{f_0=f_m} = -\int \rho'(f - f_m)\mathrm{pdf}(f)\, df = -(\psi * \mathrm{pdf})(f_m) \ . \tag{6}$$

The derivative of the error norm $\psi(\cdot) = \rho'(\cdot)$ is called the *influence function* of the robust optimization problem [31]. Knowing the cubic approximation of the robust error norm, this influence function is obtained by exploiting the derivative theorem of B-splines [27]:

$$B_k'(f) = B_{k-1}(f + \frac{1}{2}) - B_{k-1}(f - \frac{1}{2}) \ . \tag{7}$$

The derivative can be computed from splines with degree $k-1$ shifted by $\pm\frac{1}{2}$. It is therefore straightforward to extract a *quadratic approximation of the influence function* from the approximated error norm:

$$\psi(f) = \rho'(f) \approx \left(\sum_{n=1}^{N} \alpha_n B_3(f-n)\right)' = \sum_{n=1}^{N} \alpha_n (B_2(f-n+\frac{1}{2}) - B_2(f-n-\frac{1}{2}))$$

$$= \alpha_1 B_2(f-\frac{1}{2}) - \alpha_N B_2(f-N-\frac{1}{2}) + \sum_{n'=\frac{3}{2}}^{N-\frac{1}{2}} (\alpha_{n'+\frac{1}{2}} - \alpha_{n'-\frac{1}{2}}) B_2(f-n') \ .$$

[7]To obtain the minimum curvature property, it is necessary to have the same derivative at both boundaries [32].

Since the first two terms do not influence the approximation result in the approximation interval $[2, N-1]$, we obtain an ordinary quadratic B-spline with coefficients

$$\alpha'_n = \alpha_{n+\frac{1}{2}} - \alpha_{n-\frac{1}{2}} \qquad n \in [1.5, N-0.5] \ . \tag{8}$$

For least-squares estimates, the influence function is simply identity ($\psi(f) = \rho'(f) = f$), which means that we obtain a unique solution to minimizing the error (5) (convex problem). However in general, robust error norms lead to several local minima (i.e., zeros from (6)). These must be compared, and the smallest one must be chosen to obtain the global minimum error. This procedure is straightforward in the context of B-spline influence functions, and is the subject of the following section.

III. Quadratic B-Spline Channels

In this section we introduce the quadratic B-spline channel representation. The encoding is formulated in Sec. III-A, the locally linear decoding in Sec. III-B. The latter leads to a particular robust error norm and influence function (see Sec. III-C), which are approximated by the decoding scheme. The difference can be explained in terms of a quantization effect (see Sec. III-D) and can be compensated for by the virtual shift decoding scheme (see Sec. III-E).

A. Encoding B-Spline Channels

The quadratic B-spline approximation of the influence function from Sec. II-C will be used to compute robust means (solutions to (5)). Notice that the knots of the B-splines (i.e., samples in the feature domain) are independent of the spatial position such that the B-spline values can be efficiently computed knot-wise for all positions. This computational scheme leads to (sparse) vectors of B-spline values at every spatial position. These can be considered as a *channel representation* [17] of the feature map, since the quadratic B-splines fulfill the requirements for a channel basis:

1) they have compact support (locality)
2) they are smooth (continuously differentiable)
3) they are non-negative (monopolarity)
4) they add to one (constant L_1-norm).

Consequently, the *quadratic B-spline channel representation* of a bounded signal $f(\mathbf{x}) \in [1.5, N-0.5]$ is given by the encoding into N *channels*

$$c_n(f;\mathbf{x}) = B_2(f(\mathbf{x}) - n) \qquad n = 1 \ldots N \ . \tag{9}$$

At each position $\mathbf{x}$ the N *channel values* $c_n(f)$ form a *channel vector* $\mathbf{c}(f) = (c_1(f), \ldots, c_N(f))^T$. A signal $f(\mathbf{x})$ that is bounded to the range $[A, B]$ is transformed as

$$\tilde{f}(\mathrm{x}) = \frac{N-2}{B-A}(f(\mathrm{x}) - A) + 1.5 \ ,$$

such that its range lies in $[1.5, N-0.5]$ before it can be encoded into channels.

One advantage of the channel representation is its flexibility with respect to the topology of the feature space to be encoded. The channel representation is not restricted to Euclidean spaces. It can be applied to any regularly sampled space. Hence, periodic domains like the unit circle, e.g., 2D orientation data, are encoded in basically the same way as described above. The only difference is that the two basis functions close to the lower bound are actually the same as those close to the upper bound. As such, we have to add them in two single channels:

$$\mathbf{c} \doteq [c_1 + c_{N-1} \ c_2 + c_N \ c_3 \ c_4 \ \ldots \ c_{N-2}]^T \ . \tag{10}$$

Although the channel representation of a deterministic signal is just a matter of coding strategy, the situation changes for stochastic signals. In this case, the channel vector at a position $\mathbf{x}$ is related to the probability density function (pdf) of the generating stochastic process at position $\mathbf{x}$ (see also [33] for the case of population codes). *Averaging the elements of the channel vector is equivalent to sampling a kernel density estimate with kernel* B_2. The expectation value of the kernel density estimate is $(B_2 * \mathrm{pdf})(f)$ [34], and hence, we obtain

$$E\left\{\sum_{\mathbf{x}} c_n(f;\mathbf{x})\right\} = (B_2 * \mathrm{pdf})(f)\Big|_{f=n} \ . \tag{11}$$

Decoding the channel representation means extracting the modes of this kernel density estimate (11). In general, this is done using the coefficients α'_n from (8). However, in practical applications, it is useful to have as few non-zero coefficients as possible in order to reduce the computational complexity. A plausible strategy to reduce the number of coefficients while maintaining a correct decoding is to use a similar robust error norm that requires fewer coefficients. Changing the error norm does not effect the results of the optimization significantly unless the penalty functions are too different, see Sect. III-C, page 16.

B. Decoding via a Decoding Window

It can easily be verified that a possible B-spline channel decoding is obtained by the following linear interpolation [21]:

$$f = \sum_{n=1}^{N} n c_n(f) \ . \tag{12}$$

This decoding is global, i.e., the corresponding error norm is quadratic, not robust. In order to introduce robustness, we may reduce the sum to a reasonable range; in other words, we apply a decoding window to the coefficients. Unperturbed B-spline channel vectors, corresponding to a single value, contain only three adjacent non-zero values. Consequently, a decoding window of width three, placed around a suitable channel n_0, is the minimum choice of width.

After processing (spatial averaging), a channel vector often contains more than three non-zero values. This is either due to noise, or due to multiple valued signals and larger decoding windows might be considered. The choice of the reconstruction window is inherently heuristic, since all sizes ≥ 3 are correct for unperturbed channels and all finite windows lead to robust behavior. The appropriate window width depends on the respective application, the data to be processed, and the number of channels.

For *minimum computational effort*, the reconstruction window has to be as small as possible (width 3). Changing the *degree of robustness* of the method, i.e. changing the width of the error norm, is therefore not done by changing the reconstruction window, but by changing the number of channels. For noisy data, this means that the *appropriate number of channels depends on the (Gaussian) noise variance* σ_N of the data. For simplicity assume that a window of width three results in an outlier rejection starting from ± 1 channels around the channel closest to the estimated mean. In order to reject no more than 5% of the inlier samples, the distance between two channels must be greater than $4\sigma_n$.

For general, perturbed channel vectors, the channel values inside the decoding window do not sum up to one and therefore the coefficients in the window must be normalized before decoding:

$$f_{n_0} = \frac{\sum_{n=n_0-1}^{n_0+1} n c_n(f)}{\sum_{n=n_0-1}^{n_0+1} c_n(f)} = \frac{c_{n_0+1}(f) - c_{n_0-1}(f)}{c_{n_0-1}(f) + c_{n_0}(f) + c_{n_0+1}(f)} + n_0 \ . \tag{13}$$

For periodic domains, the channel vector has to be extended at both ends as $[c_N \ \mathbf{c} \ c_1]^T$ before (13) is applied. After the decoding, the result has to be mapped to the correct interval by a modulo operation (and possibly a scaling).

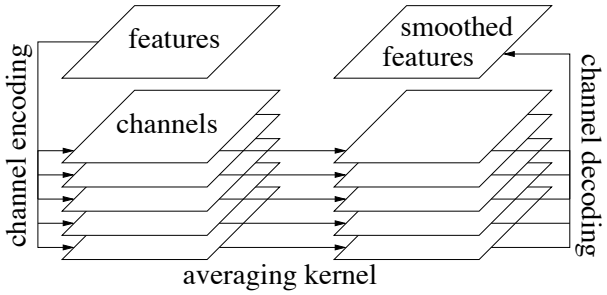

Fig. 4. Schematic overview of channel smoothing. The signal is encoded according to (9) and decoded according to (13) or (30).

The window center n_0 can be chosen in various ways, most of them being quite ad hoc. A common method is to search for the maximum of the denominator in (13), i.e., the largest windowed sum of channel values [20].[8]

C. Channel Smoothing and Robust Estimation

As pointed out in the introduction, we consider *channel smoothing* in this paper, i.e., the channels that are obtained from the encoding are linearly averaged before being decoded, see Fig. 4. Channel smoothing applied to the gray values of an image results in a method for image enhancement or denoising, see Fig. 5 c). Note that this is an experiment on the simplest type of signal feature, i.e., its intensity. In our main experiment in Sect. IV-C, we present the results for a more complex signal feature, namely the orientation field.

Smoothing the channels with a linear filter is equivalent to averaging the original signal feature (here: the signal itself) if, and only if, the signal is smooth enough that no channel values are outside the decoding window (see Sect. III-D). If the signal is sufficiently smooth, the decoding formulae (12) and (13) are equivalent since no non-zero channel values are neglected in (13) and the denominator is one. Hence, we obtain by linearity (where $h(\mathbf{x})$ is the smoothing kernel):

$$\sum_{n=n_0(\mathbf{x})-1}^{n_0(\mathbf{x})+1} n(h * c_n(f))(\mathbf{x}) = (h * f)(\mathbf{x}) \quad .$$

[8]In this paper we only address the decoding of one encoded value. Indeed, the channel representation can be used to represent and to process multiple values. In this case, the signal must be decoded at all local maxima of the windowed sum [20].

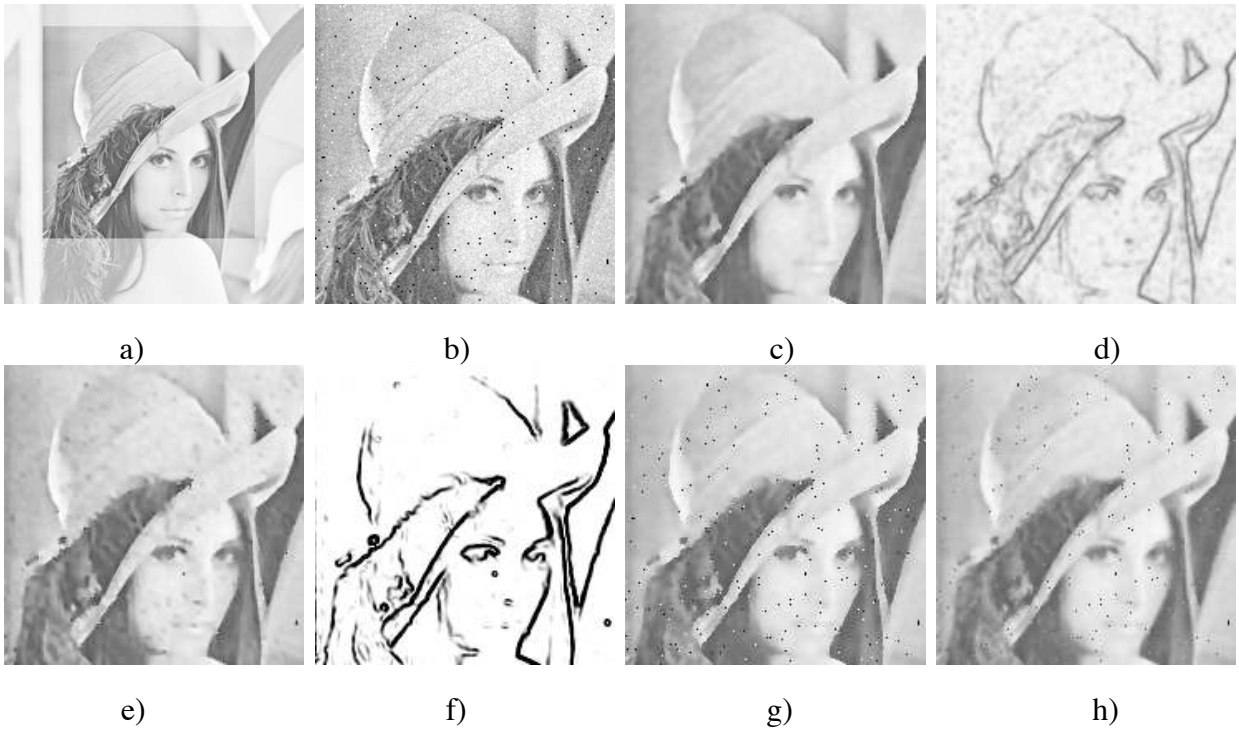

Fig. 5. Experiment on gray value smoothing. a) Original image, downsampled to 256x256, crop area 181x181. b) Cropped image with added Gaussian noise $\sigma = 2\%$ and 1% Salt & Pepper noise, SNR 8dB. c) Result from channel smoothing, 9 channels and binomial filter of width 7x7. d) Robust error (5), white=0, black=$\frac{23}{24}$. e) Diffusion result, scale for structure estimation: 1, scaling of gray levels (see [35], (2)): 20, timestep: 0.24, iterations: 8. f) Diffusivity of result in the range from 0 (black) to 1 (white). g) Result from mean-shift filtering, parameters $h_s = 3.09$, $h_r = 0.29$. h) Result from bilateral filtering, parameters $\sigma_s = 1.5$ (with a support of width 11) and $\sigma_r = 0.3$.

If the signal is not sufficiently smooth, the decoding from the smoothed channels is no longer identical to the averaged signal. However, if the deviation from the smoothness constraint is small, the difference between the decoding and the averaged signal is also small. This difference increases with the deviation from the smoothness condition until the signal contains a discontinuity. In this case, the channel smoothing results in a totally different result to that of averaging the signal. The channel smoothing does not smooth discontinuities, i.e., non-stationary signal parts. This is a correct behavior since it is only sensible (from a stochastic point of view) to smooth stationary signals.

This behavior can be formalized by means of robust statistics. Assume that decoding the sum of the channels of m signal samples results in the value n_0. Further assume that n_0 is a channel

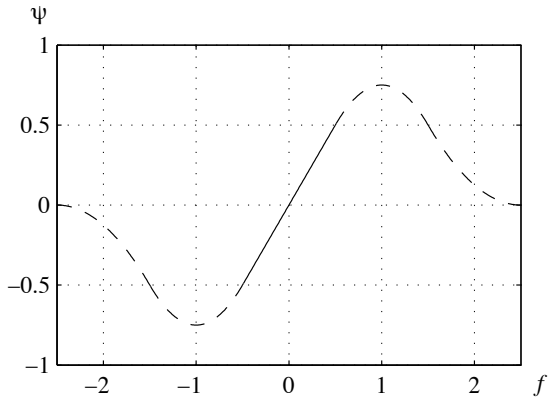

Fig. 6. Influence function of channel smoothing according to (14). The solid part is linear.

center. If we add the channels of another data sample, the reconstruction value will change according to a certain function. The latter is obtained from the decoding formula (13), which results in a linear combination of B-splines, see Fig. 6. This function is linear at the origin and becomes zero for large arguments, i.e., it can be considered as a robust influence function.

This *effective influence function of channel smoothing* can be computed analytically from the formulae (9) and (13) under the assumption of an infinite number of samples ($m \to \infty$).[9] Shifting the origin by n_0, we obtain the following result (see also Fig. 6):

$$\psi(f) = B_2(f-1) - B_2(f+1) = \begin{cases} -B_2(f+1) & f \in (-2.5, -0.5] \\ f & f \in (-0.5, 0.5] \\ B_2(f-1) & f \in (0.5, 2.5] \\ 0 & \text{otherwise.} \end{cases} \tag{14}$$

Integrating the influence function yields the robust error norm [31], where we have to add an offset such that the error norm is zero for $f = 0$:

$$\rho(f) = \int_{-\infty}^{f} \psi(f')\, df' - \int_{-\infty}^{0} \psi(f')\, df' \ . \tag{15}$$

Substituting (14) and applying the integration theorem on splines [27] yields

[9]More formally, this computation is done in expectation-value sense, as it is common for approximation schemes of robust estimation.

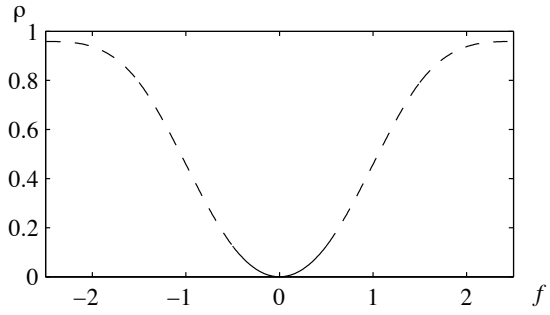

Fig. 7. Error norm of channel smoothing according to (16). The solid part is quadratic.

$$\rho(f) = -B_3(f + \frac{1}{2}) - B_3(f - \frac{1}{2}) + 2B_3(\frac{1}{2})$$

and substituting (2) results in

$$\rho(f) = \begin{cases} \frac{f^2}{2} & |f| \leq 0.5 \\ \frac{23}{24} - \frac{5-2|f|}{6} B_2(|f| - 1) - B_2(f) & 0.5 < |f| < 2.5 \\ \frac{23}{24} & \text{otherwise.} \end{cases} \tag{16}$$

The error norm is plotted in Fig. 7. Channel smoothing corresponds to a minimization of (5) with error norm (16) if f_0 lies at a discrete channel position n_0. The restriction $f_0 = n_0$ is avoided by applying the virtual shift decoding (see below).

In [31], the authors present a method for converting a robust estimator into the penalty function of a line process. The penalty function is an important tool to compare different robust estimators and similar penalty functions imply a similar performance in application. In Fig. 8 we compare the penalty functions corresponding to the error norm (16) (for calculation see [35]) and Tukey's biweight (see [31]). As can be seen, the penalty functions do not differ qualitatively and in [31] it is claimed that the exact formulation of the influence function is not critical as long as the penalty function has a similar shape. Hence, the influence function (14) can be used instead of other, similar influence functions (e.g. Tukey, MFT, Leclerc, Geman & McClure).

D. Quantization Effects in Decoding

The decoding method introduced in III-B ensures that unperturbed channel vectors are decoded correctly. However, if the channel vector corresponds to a smoothed pdf estimate, the decoding

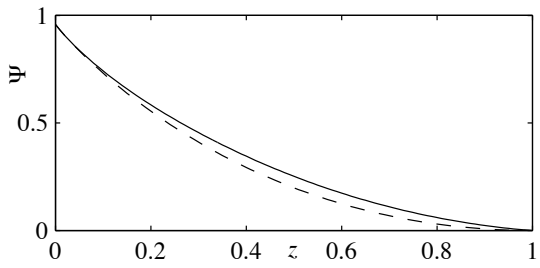

Fig. 8. Penalty functions of channel smoothing (solid) and of Tukey's biweight (dashed, rescaled).

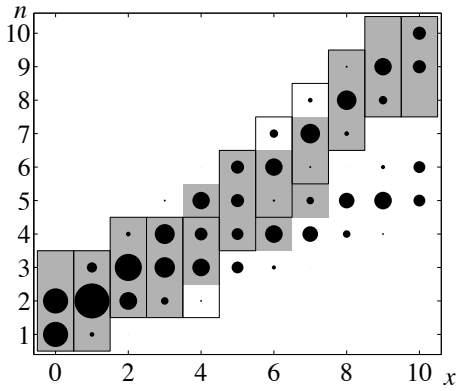

Fig. 9. Influence of the finite window on the decoding (see Fig. 10). We have generated the depicted channel representation by adding the channel representations of two linear functions. The channel values are indicated by the size of the dots. The gray boxes indicate the positions of the decoding window selected by the maximum denominator of (13). $x = 0, 1$: all values are inside the window; the decoded signal is the linear average. $x = 2, 3, 4$: small values are outside the window; the reconstructed signal is close to the linear average. $x = 5, 6, 7$: large values are outside the window and the window is not centered to one maximum; the reconstructed signal is somewhere in between a linear and a robust average. $x = 8, 9, 10$: the window covers only the channel values of one of the original functions; the reconstructed signal is equal to the stronger original function. Additionally requiring the center value of the window being a local maximum results in the black rectangles. Three positions change: The signal deviates earlier from the average, once in the wrong direction (4) and twice in the right direction (6,7).

only results in a correct result (according to (6)) if the mode is located at a channel center. In the general case, it might lead to some non-appropriate averaging, cf. Fig. 9 and Fig. 10. The exact behavior depends on the pdf and the channel distance.

As can be seen in Fig. 10, the decoding (13) results in values that depend on the absolute position of the channel centers. In the following, we refer to this effect as the *quantization effect* of channel smoothing. Whereas in Fig. 10 we considered the results from adding the channel

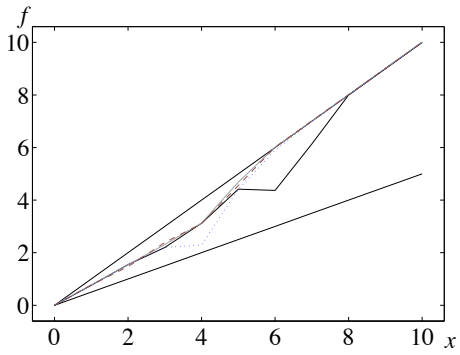

Fig. 10. Functions corresponding to Fig. 9. The two linear functions correspond to the considered channel representation, where the steeper one has slightly stronger channel values (10%). The solid non-linear function is the decoded signal obtained from (13) with the maximum denominator choice for n_0. The dotted curve is obtained by choosing n_0 being a local maximum. The dashed curve is the decoded signal from the virtual shift decoding. The gray curve is the true mode of the kernel density estimate.

representations of two functions, it is more realistic to look at the decoding of channels which are averaged over the spatial domain, see Fig. 11.

The quantization effect occurs, since according to (13), the influence function is always centered at a certain channel n_0 instead of being centered at the robust mean f_0. By choosing the optimal decoding window, we can ensure that n_0 is the integer closest to f_0, but the remaining deviation $f_0 - n_0$ causes the quantization effect.

E. Suppressing Quantization Effects: Virtual Shift Decoding

The basic approach to removing the quantization effect is to apply a virtual shift to the decoding window such that it is centered at the true robust mean. The proper shift, i.e., $f_0 - n_0$, is obtained by virtually shifting the error norm according to the channel data. In order to compute the shifted error, we interpolate the channel vector using quadratic B-splines. The interpolation is obtained by computing new B-spline coefficients using recursive filters [27]. The required filter is obtained by inverting the z-transform of the sampled quadratic B-spline:

$$B_2(n) = \frac{1}{8}\begin{cases} 6 & \text{if } n = 0 \\ 1 & \text{if } |n| = 1 \\ 0 & \text{otherwise} \end{cases} \qquad \stackrel{z}{\leftrightarrow} \frac{1}{8}(z + 6 + z^{-1}) \ .$$

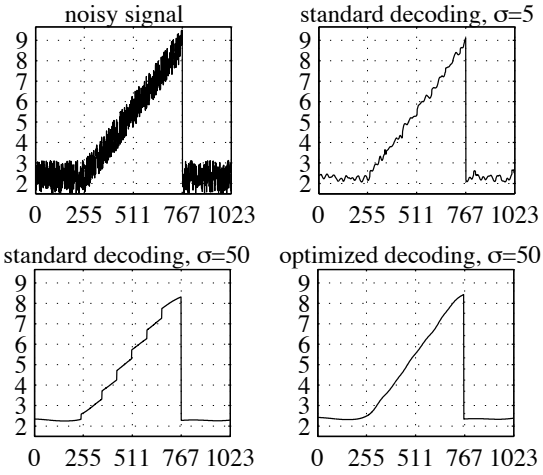

Fig. 11. Channel smoothing of a noisy 1D signal (top left) using ten channels. The channel positions are given by the indexes on the vertical axis. The horizontal axis is the time axis. Averaging each channel with a Gaussian kernel ($\sigma = 5$) results in a reduced amount of noise (top right). Further increasing the kernel width to $\sigma = 50$ results in a noise free signal, which however suffers from the quantization effect (lower left). The quantization effect is less visible here than in Fig. 10, the function is just slightly over-attracted by the channel centers. This results in sudden changes from one sample to the next if the true mode lies close to the middle between two channels. Due to the higher sampling rate in this example, these changes appear as vertical jumps. The virtual shift decoding results in a noise free signal without showing the quantization effect (lower right).

Taking the inverse results in

$$B_2^{-1}(n) \overset{z}{\leftrightarrow} \frac{8}{z + 6 + z^{-1}} = 8 \left(\frac{1}{1 - z_1 z^{-1}} \right) \left(\frac{-z_1}{1 - z_1 z} \right) \tag{17}$$

where $z_1 = 2\sqrt{2} - 3$. The new channel coefficients c' are hence obtained by two recursive filters in both directions:

$$c_n^+ = c_n + z_1 c_{n-1}^+, \qquad (n = 2, \ldots, N) \tag{18}$$

$$c_n^- = z_1 (c_{n+1}^- - c_n^+), \qquad (n = N-1, \ldots, 1) \tag{19}$$

$$c'_n = 8 c_n^- \ . \tag{20}$$

It is sensible to assume that all channel values with $n \notin 1, \ldots, N$ are zero. Therefore, the initial conditions are set such that

$$c_1^+ = c_1 \tag{21}$$

$$c_N^- = \frac{z_1}{z_1^2 - 1} c_N^+ \ . \tag{22}$$

For periodic domains the initial conditions are different. Instead of recursive filtering we apply the DFT (implemented as a FFT) to obtain the interpolation coefficients. This is possible since the coefficients in (4) form a circulant matrix for periodic channels and they are obtained as

$$\mathrm{DFT}_N(\mathbf{c}'') = 8(\mathrm{DFT}_N([6\ 1\ 0 \ldots 0\ 1]_N))^{-1}\mathrm{DFT}_N(\mathbf{c}) \ , \tag{23}$$

where $\mathrm{D}FT_N$ denotes the N-point DFT along the channel index. The new channel vector $\mathbf{c}''$ must be extended at both ends according to

$$\mathbf{c}' = [c''_{N-1}\ c''_N\ \mathbf{c}''^T\ c''_1\ c''_2]^T$$

in order to process it further as if it was non-periodic.

Sampling the continuous function

$$p(f) = \sum_{n=1}^{N} c'_n B_2(f-n) \tag{24}$$

results in the original channel vector $\mathbf{c}$ again, i.e., $p(f)$ is a proper interpolation of the channel vector. However, the interpolation coefficients c'_n might become negative and are therefore not a valid channel representation and cannot be considered as samples of a pdf. As a consequence, the interpolated function $p(f)$ might also contain (slightly) negative regions.[10] The potentially negative coefficients are due to the missing band-limitation of the empirical density that is used to build-up the channel vector. The B-splines are not ideal low-pass filters, and hence, high frequency components in the empirical density cause aliasing. In the case of aliasing, however, the samples of the B_2-filtered pdf are not identical to the discrete convolution of the sampled pdf and the sampled B-spline. Inverse filtering of the channel vector therefore leads to a discrete signal different from the sampled pdf and in particular potentially negative.

From (11) we know that the channel vector corresponds to the sampled, smoothed pdf, and therefore, we get the approximation $p(f) \approx (B_2 * \mathrm{pdf})(f)$. Using this function $p(f)$ and the influence function of a virtually shifted decoding window, we are hence looking for the zeros of

$$(\psi * \mathrm{pdf})(f_0) = ((B_2(\cdot - 1) - B_2(\cdot + 1)) * \mathrm{pdf})(f_0) \approx p(f_0 - 1) - p(f_0 + 1) \tag{25}$$

[10]Non-negative coefficients c'_n imply a non-negative function $p(f)$.

and therefore, assuming that $|f_0 - n_0| \leq 0.5$ (i.e., the robust mean lies closest to channel n_0), we obtain

$$\begin{aligned} 0 &= p(\beta + n_0 - 1) - p(\beta + n_0 + 1) = \sum_{n=1}^{N} c'_n (B_2(\beta + n_0 - n - 1) - B_2(\beta + n_0 - n + 1)) \\ &= \sum_{n=n_0-1}^{n_0+1} (c'_{n-1} - c'_{n+1}) B_2(\beta + n_0 - n) \ , \end{aligned}$$

where $f_0 - n_0 = \beta$ indicates the virtual shift. This expression[11] boils down to a quadratic equation

$$0 = \lambda\beta^2 + \mu\beta + \nu \qquad \text{with} \tag{26}$$

$$\lambda = (c'_{n_0-2} - 2c'_{n_0-1} + 2c'_{n_0+1} - c'_{n_0+2})/2 \tag{27}$$

$$\mu = (-c'_{n_0-2} + 2c'_{n_0} - c'_{n_0+2})/2 \tag{28}$$

$$\nu = (c'_{n_0-2} + 6c'_{n_0-1} - 6c'_{n_0+1} - c'_{n_0+2})/8 \tag{29}$$

such that the minimum of the error corresponds to

$$\beta = \frac{-\mu/2 + \sqrt{\mu^2/4 - \nu\lambda}}{\lambda} \ . \tag{30}$$

Solutions where $|\beta| > 1/2$ must be excluded, since they are located outside the unit interval around n_0. For valid solutions of β, the decoded robust mean is given by $f_0 = n_0 + \beta$. For periodic domains, the correct values are obtained by an additional modulo operation.

The question arises, "what is the proper choice of n_0 in this context?" There surely exist several heuristics to reduce the number of candidates to a few or even one, but from a general point of view we have to compute $\beta_0 = \beta(n_0)$ for all n_0. After excluding invalid solutions, we obtain several local minima of the error function. For each of these we have to compute the robust error (5):

$$\begin{aligned} E(n_0) =& (\rho * \mathrm{pdf})(n_0 + \beta_0) = \left(\left[\frac{23}{24} - B_3(\cdot - \tfrac{1}{2}) - B_3(\cdot + \tfrac{1}{2}) \right] * \mathrm{pdf} \right)(n_0 + \beta_0) \\ &\overset{(2)}{\approx} \frac{23}{24} - ((B_0(\cdot - \tfrac{1}{2}) + B_0(\cdot + \tfrac{1}{2})) * p)(n_0 + \beta_0) \\ &\overset{(24)}{=} \frac{23}{24} - \sum_{n=-2}^{2} c'_{n_0+n} (B_3(\beta_0 - n - \tfrac{1}{2}) + B_3(\beta_0 - n + \tfrac{1}{2})). \end{aligned}$$

[11]Note that according to the last line the coefficients c'_0 or c'_{N+1} might be required. From the initial conditions it follows that $c'_0 = z_1 c'_1$ and $c'_{N+1} = z_1 c'_N$.

This leads to the expression

$$E(n_0) = \frac{23}{24} + \beta_0\nu + \beta_0^2\mu/2 + \beta_0^3\lambda/3 - \frac{c'_{n_0-2} + 24c'_{n_0-1} + 46c'_{n_0} + 24c'_{n_0-1} + c'_{n_0-2}}{48} \quad . \quad (31)$$

By comparing the local minima and choosing that n_0 for which (31) is minimal, we obtain the global minimum and its corresponding robust mean $f_0 = n_0 + \beta_0$.

Note that the virtual shift decoding requires a larger computational effort compared to local linear decoding. However, the effort scales with the number of channels and the compensation for the quantization effect is less relevant for large numbers of channels, e.g., if 256 intensities are encoded in 32 channels. Hence, we recommend the use of virtual shift decoding for channel smoothing with only a few channels (high noise variance) and to use local linear decoding for channel smoothing with many channels (low noise variance). The whole algorithm for channel smoothing is summarized in Fig. 12.

IV. Comparison to other Approaches

In this section we compare channel smoothing to well-established methods for robust smoothing.

A. Other Approaches to Robust Smoothing

We chose the following methods for comparison: diffusion filtering, bilateral filtering, and the mean-shift method. All three methods are briefly explained in the supplemental material to this paper [35]. For initial comparison, we depict results obtained by all methods in Fig. 5.[12]

Besides the afore mentioned methods, there exist several other approaches for non-linear smoothing which are more or less related to robust statistics. One of these approaches is wavelet shrinkage [36] and it has been shown for 1D signals that one-scale Haar wavelet shrinkage is equivalent to one diffusion step with a suitable diffusivity [37]. Apparently, there is a connection between wavelet shrinkage and robust techniques, see for example [38], where a wavelet decomposition according to a hybrid loss function (L_2 near the origin and L_1 in the periphery, the Huber-norm) is applied. However, the exact relation between multi-scale wavelet shrinkage (as a whole) and robust smoothing still remains unclear. In addition, wavelet shrinkage

[12]The width of the error norm (edge-stopping function) applied in diffusion filtering can be widened or longer diffusion times can be used, to make the outliers 'vanish'. Actually, the outliers never vanish – at best they are averaged with the background.

1) Choose a number N of channels according to the noise variance
2) Transform the feature value range to $[1.5, N-0.5]$ or, if periodic, to $[0, N)$
3) Encode the feature into channels using (9) (and if periodic (10)); in practice:
   i. $m = \mathrm{round}(f)$, $f_m = f - m$
   ii. $c_{m-1} = (f_m - 0.5)^2/2$
   iii. $c_m = 0.75 - (f_m)^2$
   iv. $c_{m+1} = (f_m + 0.5)^2/2$
   v. $c_n = 0$ for $n \notin \{m-1, m, m+1\}$
4) Average each channel with a suitable averaging filter

5a) If N is large, apply (13) to decode the result

5b) Otherwise start virtual shift decoding for each channel vector:
   i. Apply recursive filtering (18–22) resp. DFT filtering (23) to the channel vector
   ii. Calculate $\beta(n_0)$ according to (27–30) and $E(n_0)$ according to (31)
   iii. Select $n_m = \mathrm{argmin} E(n_0)$ and return $n_m + \beta(n_m)$ (if periodic modulo N) as decoded the value

6) Transform the decoded result to original interval

Fig. 12. Algorithm of channel smoothing

on 2D orientation data has, to the best of our knowledge, not appeared in the literature so far. Therefore, we exclude wavelet shrinkage from the subsequent discussions and experiments.

In our experiment below (Sec. IV-C), we will apply the four methods in the case of robust orientation smoothing, which is non-trivial due to the periodic topology of orientation space [39], [40], [41]. For diffusion filtering, a standard technique is to diffuse the two components of an orientation vector field separately and to renormalize the vectors afterwards. Although this approach theoretically violates the semi-group property and might destroy the topology of orientation data [42], the results are sufficiently accurate if the normalization is applied repeatedly between sufficiently short diffusion times [39]. More theoretically sound methods make use of the Levi-Civita connection [43], [44] and the metric imposed by the manifold [45], but these methods are slower than component-wise diffusion and the accuracy gain is neglectable in most

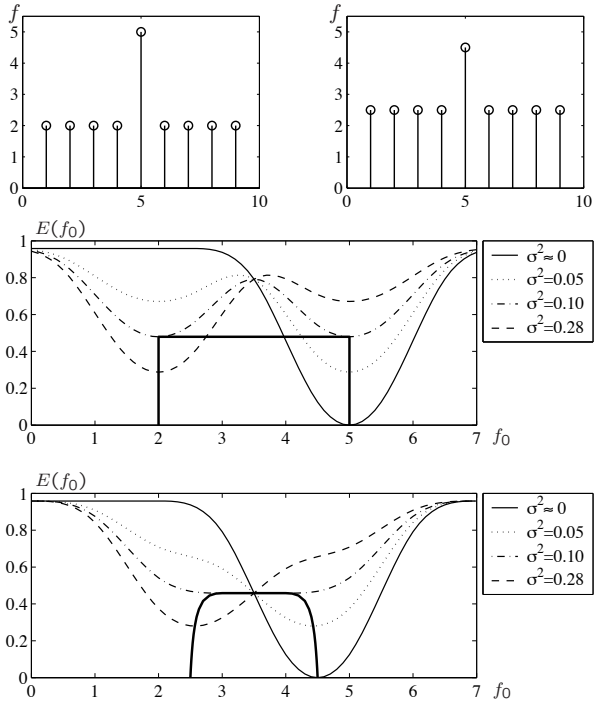

Fig. 13. Outlier treatment by channel smoothing. The curves in the plots at the middle and at the bottom show the error (5) of the corresponding estimates. If the outlier is sufficiently different from the surrounding values (top left), channel smoothing picks either the outlier value (solid line and dotted line in the middle) or the surrounding values (dashed line in the middle). The boundary case is obtained for a filter coefficient of $\frac{1}{2}$ at the origin (dash-dotted line). The bold line shows the trajectory of the global minimum error depending on the smoothing kernel. The ordinate corresponds to the decoded value and the abscissa to the residual error. If the outlier is too similar to the surrounding values (top right), channel smoothing corresponds to a weighted averaging and the global minimum can be placed anywhere between 2.5 and 4.5 (plot at the bottom). The minimum error trajectory is a smooth curve.

cases. In addition, we want to be able to weight the orientation vectors. Therefore, we stick to the standard technique of smoothing the two components.

B. Theoretic Comparisons on Outlier Treatment and Computational Complexity

Although all four afore mentioned methods implement robust smoothing, they behave differently concerning outliers. Assuming a signal like the one in Fig. 13, top left, and considering the value at the position 5, i.e., the outlier position, we observe different options for the result of robust smoothing:

case **r**:outlier remains unchanged

case **l**: value is moved to the closest local minimum after averaging

case **g**:value is moved to the global minimum after averaging

case **s**: outlier is replaced with average of surrounding values

The different methods lead to the results indicated in Tab. I.

TABLE I

DIFFERENT TREATMENT OF OUTLIERS FOR THE FOUR METHODS UNDER COMPARISON. ABBREVIATIONS: DIFFUSIVITY FUNCTION (DF), BACKGROUND (BG), FEATURE DOMAIN WINDOW (FW), EPANECHNIKOV KERNEL (EK), ERROR FUNCTION (EF), AVERAGING KERNEL VALUE AT ORIGIN (AK).

method	condition	case
diffusion	gradient outside DF	r
diffusion	gradient inside DF	l
bilateral	BG outside FW	r
bilateral	BG inside FW	l
mean-shift	BG outside EK	r
mean-shift	BG inside EK	l
channel	BG outside EF & AK > 0.5	r
channel	BG outside EF & AK < 0.5	s
channel	BG inside EF & AK > 0.5	l
channel	BG inside EF & AK < 0.5	g

The different behaviors concerning outliers also influence the behavior in case of noise cut-off due to the finite range of the image data. Modeling Gaussian noise with cut-off as a combination of Gaussian noise and (less than 50%) outliers allows the assumption that the visual impression of the image contrast is hardly affected by channel smoothing due to correct outlier removal. In the case of the other three methods, changes of contrast are visible. However, a thorough investigation of this behavior is out of the scope of this paper.

The aim of approximation methods is to obtain an accurate result with *low computational complexity*. Hence, a complexity analysis (for the case of periodic features as applied in Sect. IV-C) of all four methods is essential in the comparison of the methods. The computational complexities are summarized in Tab. II.

In all cases, except for extremely narrow error functions (i.e., many channels), channel smoothing is the fastest method. Moreover, channel smoothing can easily be implemented on a parallel

TABLE II

COMPUTATIONAL COMPLEXITY OF THE FOUR METHODS UNDER COMPARISON IN FLOPS, WHERE N_s IS THE SPATIAL SIZE OF THE KERNEL, K IS THE NUMBER OF CHANNELS, $E\{N_i\}$ IS THE AVERAGE NUMBER OF ITERATIONS FOR MEAN-SHIFT [46] (TYPICALLY BETWEEN 2 AND 20), AND N_i IS THE NUMBER OF ITERATIONS FOR DIFFUSION. GCPP IS THE ABBREVIATION FOR 'GENERAL COMPLEXITY PER PIXEL', TCPP FOR 'TYPICAL COMPLEXITY PER PIXEL' IN THOUSAND OPERATIONS.

method	GCPP	parameters	TCPP
mean-shift	$27N_s^2E\{N_i\}$	$N_s \in [7, 13]$	2.5 – 100
bilateral	$18N_s^2$	$N_s \in [15, 36]$	4 – 25
diffusion[a]	$90N_i$	$N_i \in [20, 200]$	2 – 20
channel[b]	$54K + 23$	$K \in [5, 20]$	0.3 – 1.2

[a]anisotropic, non-linear diffusion, separable Sobel filters

[b]separable smoothing kernel built from four-fold cascaded box filters implemented as moving average

computer, since no data dependencies occur. On parallel systems with parallel, shared memory access[13], the time complexity of channel smoothing scales down with the number of parallel processors. However, channel smoothing is more memory consuming than the other three methods.

C. Robust Orientation Smoothing Experiment

In this experiment, we compared the four methods for the application of orientation smoothing. The orientation itself is extracted from the test images in two ways: By central differences[14] and by the Riesz transform [47]. The orientation θ is represented in *double angle representation* [48], [49] $o(\mathbf{x}) = \exp(i2\theta(\mathbf{x}))$ to avoid problems with the directional sense.[15] The double angle representation of the gradient orientation is weighted by setting $|o|$ to the gradient magnitude

$$o_g(\mathbf{x}) = \frac{(\partial_x f(\mathbf{x}) + i\partial_y f(\mathbf{x}))^2}{|\partial_x f(\mathbf{x}) + i\partial_y f(\mathbf{x})|} \tag{32}$$

[13]The simultaneous memory access is guaranteed to be conflict-free for channel smoothing.

[14]We made use of the gradient function of Matlab.

[15]Averaging the double angle representation is equivalent to tensor averaging [39].

and the Riesz orientation is weighted by setting $|o|$ to the sine magnitude of the local phase ($|\sin\varphi|$) [50]

$$o_r(\mathbf{x}) = \frac{q(\mathbf{x})^2}{|q(\mathbf{x})|\sqrt{p(\mathbf{x})^2 + |q(\mathbf{x})|^2}} \ , \tag{33}$$

where p is the DC-free original signal and q is the Riesz transform [47] of the signal. Note that in contrast to the other three methods, which directly smooth the complex functions $o_g(\mathbf{x})$ and $o_r(\mathbf{x})$, channel smoothing is applied to the double angle directly. The orientation angle is channel encoded and the weighting is used to pre-weight the channel vectors, cf. [51] for a similar experiment on multiple orientation estimation.

Our test signals are constructed to contain (orientation) discontinuities and outliers. We chose synthetic patterns in order to have access to ground truth. The patterns contain orientation steps at different angles, different curvatures, different intensities, different local frequencies, and different phase-differences, see Fig. 14, upper left, for one instance.[16] The ground truth of all images is the same and it is depicted in Fig. 14, upper right.

In order to test the methods under various noise conditions, we added Gaussian noise of different variance and different amounts of Salt & Pepper noise to the test patterns. One of these instances can be found in Fig. 14. In total, we used 256 different instances of the test pattern.

For each test pattern, the orientation was extracted according to (32) and (33). We then applied the four different methods with varying parameters, minimizing the mean orientation error according to [52]:

$$\Delta\theta = \cos^{-1}\left(\sqrt{\frac{\sum w(\mathbf{x})\cos^2(\theta_m(\mathbf{x}) - \theta_t(\mathbf{x}))}{\sum w(\mathbf{x})}}\right) \ , \tag{34}$$

where θ_m and θ_t indicate the estimated orientation and the ground truth, respectively, and $w(\mathbf{x})$ is the index function in Fig. 14 (middle left). The orientation errors are documented in Fig. 15 and Fig. 16.

The parameter optimization has been performed by a multi-grid brute-force search. For bilateral filtering and mean-shift filtering, we optimized with respect to spatial and range-domain width; in the case of diffusion filtering with respect to the number of iterations and width of the edge-stopping function. The minimum error of each method and each test image was used for

[16] A high-dimensional random vector determines intensity, frequency, and phase.

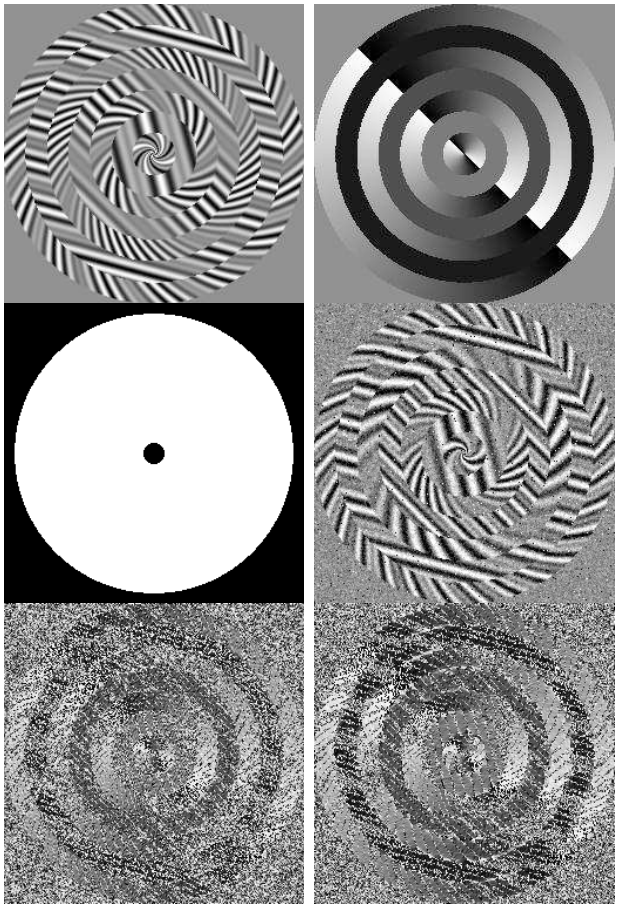

Fig. 14. Upper left: one instance of the test pattern. Upper right: ground truth orientation, encoded in gray levels. Middle left: index where the estimate and the ground truth are compared. Middle right: one instance of a test pattern with 1% Salt & Pepper noise and Gaussian noise of variance 0.01. Bottom left: raw orientation estimate from the gradient of the noisy image. Bottom right: raw orientation estimate from the Riesz transform of the noisy image.

the comparison. For channel smoothing we optimized *a linear model* for the number of channels and the width of the smoothing kernel, i.e., our comparison favors the other methods.

From the experiments we draw the following conclusions:

- Methods based on the Riesz transform are better than those based on the gradient. In the Riesz transform all frequencies are weighted in the same way, which means that it is the optimal choice for white noise.
- For the gradient data, channel smoothing and diffusion filtering result in comparably good estimates, the difference of 0.4° is hardly significant for 256 test images. One exception is the case of very low Gaussian noise, cf. separate discussion further below.

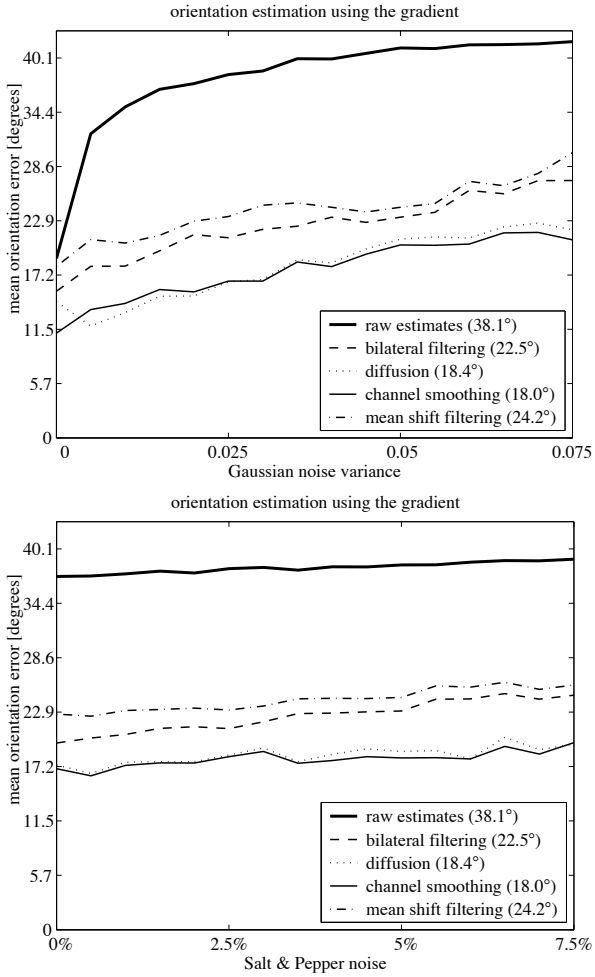

Fig. 15. Result of the comparison using gradient data. Top: Error depending on the amount of Gaussian noise. Bottom: Error depending on the amount of Salt & Pepper noise.

- For gradient data, bilateral filtering and mean-shift filtering perform significantly poorer than diffusion filtering and channel smoothing.
- For the Riesz transform data, the situation changes. All four methods lie closer together. Channel smoothing performs clearly the best, whereas diffusion filtering and bilateral filtering result in equivalently good estimates. Again, mean-shift filtering produces the worst estimate.

As pointed out above, diffusion filtering behaves differently for very low Gaussian noise and high Gaussian noise. The reason for this different behavior is the treatment of outliers. For low

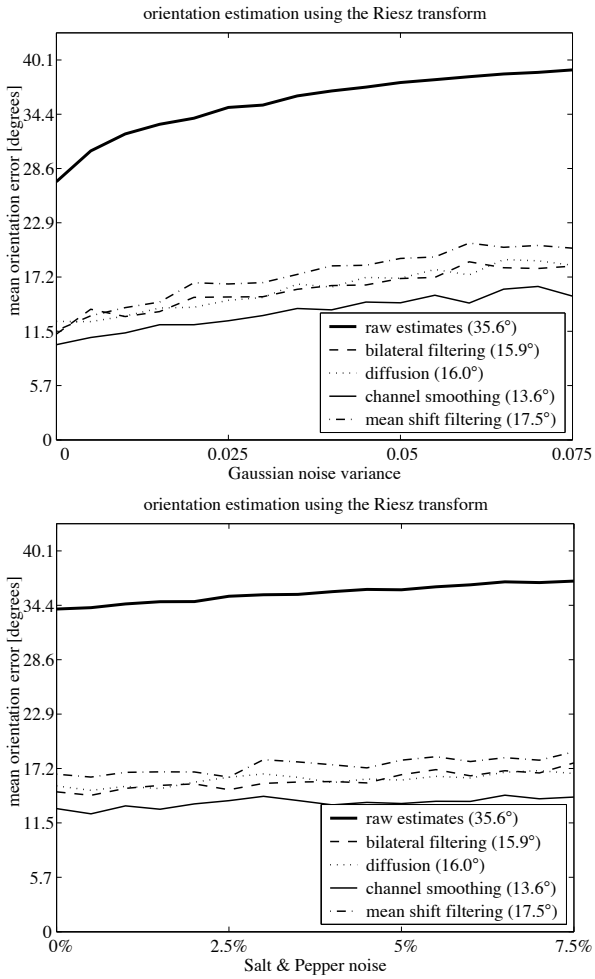

Fig. 16. Result of the comparison using Riesz transform data. Top: Error depending on the amount of Gaussian noise. Bottom: Error depending on the amount of Salt & Pepper noise.

Gaussian noise diffusion, filtering does not affect the Salt & Pepper noise, see Sect. IV-B. For larger amounts of Gaussian noise, the diffusion process 'catches' the outlier and averages it with the rest of the data in the neighborhood. The effective diffusivity function is virtually convolved with the Gaussian distribution and the outlier falls into the resulting broader support.

Nevertheless, our consideration of outlier treatment from Sect. IV-B was confirmed by the low Gaussian noise case. We extracted the orientation error of the smoothed gradient orientation for the cases of Gaussian noise with variance 0 and 0.005 as a function of the amount of Salt & Pepper noise. We fitted a line to the data of each method and compared the slopes. In the ideal

data	dependency
gradient data	0.85
bilateral filtering	0.90
diffusion filtering	0.97
channel smoothing	0.33
mean-shift filtering	0.98

TABLE III

DEPENDENCY ON SALT & PEPPER NOISE. THE NUMBERS INDICATE THE SLOPE OF THE FITTED LINE IN DEGREES PER PERCENTAGE SALT & PEPPER NOISE. WHEREAS CHANNEL SMOOTHING REDUCES THE DEPENDENCY SIGNIFICANTLY, THE OTHER THREE METHODS DO NOT IMPROVE THE INDEPENDENCE. DUE TO THE RELATIVELY SMALL AMOUNT OF SAMPLES (32 IMAGES), THE DIFFERENCES OF THE OTHER THREE METHODS ARE NOT SIGNIFICANT.

case, the slope should be zero, meaning that all outliers had been removed without affecting the other values. However in practice, none of the methods except for channel smoothing reduced the dependency of the error on the outliers, see Tab. III.

V. CONCLUSION

In this paper we have established an equivalence between robust smoothing and channel smoothing. We have derived the robust error norm that corresponds to the simplest case of channel decoding, showing that it is very similar to Tukey's function. We have presented an exhaustive experiment on orientation smoothing which confirms the theoretical considerations.

A main benefit of this equivalence is the decrease of computational complexity by one to two orders compared to three well-known methods of robust smoothing: diffusion filtering, bilateral filtering, and mean-shift filtering. Furthermore, the results from channel smoothing are comparable and in many cases better than those of the other three approaches, since channel smoothing searches for the global minimum of the robust error norm and not for the closest local minimum. A further fundamental advantage is the capability of the channel representation to encode features in a non-linear space, e.g., periodic spaces. Finally, channel smoothing is extremely simple to implement, even on a parallel architecture.

We come to the conclusion that linear averaging of B-spline channels is a good, efficient, and simple way to implement robust smoothing which is superior to diffusion filtering, bilateral

filtering, and mean-shift filtering in many cases. Nevertheless there may be applications or computer architectures where one of the other methods is better suited.

ACKNOWLEDGMENTS

We appreciate the discussions with Rudolf Mester on the spline sampling theory and with Ullrich Köthe on error functionals and influence functions. We thank Richard Bowden for proof-reading the manuscript. This work has been supported by DFG Grant FE 583/1-2 (Felsberg), EC Grant IST-2003-004176 COSPAL (Felsberg & Forssén)[17], and DFG Grant Scha 927/1-2 (Scharr).

REFERENCES

[1] M. J. Black, G. Sapiro, D. H. Marimont, and D. Heeger, "Robust anisotropic diffusion," *IEEE Transactions on Image Processing*, vol. 7, no. 3, pp. 421–432, 1998.

[2] J. Weickert, "Theoretical foundations of anisotropic diffusion in image processing," in *Computing, Suppl. 11*, pp. 221–236. 1996.

[3] C. K. Chu, I. K. Glad, F. Godtliebsen, and J. S. Marron, "Edge-preserving smoothers for image processing," *J. American Statistical Assoc.*, vol. 93, no. 442, pp. 526–541, 1998.

[4] C. Tomasi and R. Manduchi, "Bilateral filtering for gray and color images," in *Proceedings of the 6th ICCV*, 1998.

[5] J. Weule, *Iteration nichtlinearer Gauss-Filter in der Bildverarbeitung*, Ph.D. thesis, Heinrich-Heine-Universität Düsseldorf, 1994.

[6] V. Aurich and J. Weule, "Non-linear gaussian filters performing edge preserving diffusion," in *Proc. 17. DAGM-Symposium*. 1995, pp. 538–545, Springer.

[7] F. Godtliebsen, E. Spjøtvoll, and J.S. Marron, "A nonlinear gaussian filter applied to images with discontinuities," *J. Nonpar. Statist.*, vol. 8, pp. 21–43, 1997.

[8] 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, 1997.

[9] K. Fukunaga and L. D. Hostetler, "The estimation of the gradient of a density function, with applications in pattern recognition," *IEEE Trans. Information Theory*, vol. 21, no. 1, pp. 32–40, 1975.

[17]However, this paper does not necessarily represent the opinion of the European Community, and the European Community is not responsible for any use which may be made of its contents.

[10] Yizong Cheng, "Mean shift, mode seeking, and clustering," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 17, no. 8, pp. 790–799, August 1995.

[11] D. Comaniciu and P. Meer, "Mean shift: A robust approach toward feature space analysis," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 24, no. 5, pp. 603–619, 2002.

[12] P.-E. Forssén, G. H. Granlund, and J. Wiklund, "Channel representation of colour images," Tech. Rep. LiTH-ISY-R-2418, Dept. EE, Linköping University, 2002.

[13] P.-E. Forssén, *Low and Medium Level Vision using Channel Representations*, Ph.D. thesis, Linköping University, Sweden, 2004.

[14] M. Felsberg and G. Granlund, "Anisotropic channel filtering," in *Proc. 13th Scandinavian Conference on Image Analysis*, Gothenburg, Sweden, 2003, LNCS 2749, pp. 755–762.

[15] N. Sochen, R. Kimmel, and R. Malladi, "A geometrical framework for low level vision," *IEEE Trans. on Image Processing, Special Issue on PDE based Image Processing*, vol. 7, no. 3, pp. 310–318, 1998.

[16] H. P. Snippe and J. J. Koenderink, "Discrimination thresholds for channel-coded systems," *Biological Cybernetics*, vol. 66, pp. 543–551, 1992.

[17] G. H. Granlund, "An associative perception-action structure using a localized space variant information representation," in *Proc. Int. Workshop on Algebraic Frames for the Perception-Action Cycle*. 2000, Springer, Heidelberg.

[18] K. Nordberg, G. H. Granlund, and H. Knutsson, "Representation and Learning of Invariance," in *Proceedings of IEEE International Conference on Image Processing*, 1994.

[19] N. L. Lüdtke, R. C. Wilson, and E. R. Hancock, "Probabilistic population coding of multiple edge orientation," in *Proceedings of IEEE ICIP*, 2002, vol. II, pp. 865–868.

[20] P.-E. Forssén, "Sparse representations for medium level vision," Lic. Thesis LiU-Tek-Lic-2001:06, Dept. EE, Linköping University, February 2001.

[21] M. Felsberg, H. Scharr, and P.-E. Forssén, "The B-spline channel representation: Channel algebra and channel based diffusion filtering," Tech. Rep. LiTH-ISY-R-2461, Dept. EE, Linköping University, September 2002.

[22] J. Weickert, "A review of nonlinear diffusion filtering," in *Scale-Space Theory in Computer Vision*, 1997, vol. 1252 of *LNCS*, pp. 260–271, Springer, Berlin.

[23] P. Perona and J. Malik, "Scale-space and edge detection using anisotropic diffusion," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 12, no. 7, pp. 629–639, 1990.

[24] G. Winkler and V. Liebscher, "Smoothers for discontinuous signals," *J. Nonpar. Statistics*, vol. 14, no. 1-2, pp. 203–222, 2002.

[25] A. K. Louis, P. Maaß, and A. Rieder, *Wavelets*, Teubner, Stuttgart, 1994.

[26] M. Jacob, T. Blu, and M. Unser, "A unifying approach and interface for spline-based snakes," in *Proc. SPIE Medical Imaging: lmage Processing*, 2001, vol. 4322, pp. 340–347.

[27] M. Unser, "Splines – a perfect fit for signal and image processing," *IEEE Signal Processing Magazine*, vol. 16, pp. 22–38, November 1999.

[28] E. Weisstein, "World of mathematics," `http://mathworld.wolfram.com`, July 2003, (Accessed 21 July 2003).

[29] C. de Boor, *Splinefunktionen*, Birkhäuser, 1990.

[30] R. N. Bracewell, *The Fourier transform and its applications*, McGraw Hill, 1986.

[31] M. J. Black and A. Rangarajan, "On the unification of line processes, outlier rejection, and robust statistics with applications in early vision," *International Journal of Computer Vision*, vol. 19, no. 1, pp. 57–91, 1996.

[32] G. Dahlquist and Å. Björck, *Numerical Mathematics and Scientific Computation*, SIAM, Philadelphia, 2005, to appear.

[33] R. S. Zemel, P. Dayan, and A. Pouget, "Probabilistic interpretation of population codes," *Neural Computation*, vol. 10, no. 2, pp. 403–430, 1998.

[34] C. M. Bishop, *Neural Networks for Pattern Recognition*, Oxford University Press, New York, 1995.

[35] M. Felsberg, P.-E. Forssén, and H. Scharr, "Supplemental material for: Channel smoothing," `www.somewhere.net`, 2005.

[36] A. Chambolle and B. J. Lucier, "Interpreting translation-invariant wavelet shrinkage as a new image smoothing scale space," *IEEE Trans. Image Processing*, vol. 10, no. 7, pp. 993–1000, 2001.

[37] P. Mrázek, J. Weickert, and G. Steidl, "Correspondences between wavelet shrinkage and nonlinear diffusion," in *Scale Space Methods in Computer Vision*, L. D. Griffin and M. Lillholm, Eds. 2003, vol. 2695 of *LNCS*, pp. 101–116, Springer, Heidelberg.

[38] A. G. Bruce, I. M. Donoho, and R. D. Martin, "Denoising and robust nonlinear wavelet analysis," *Wavelet Applicat.*, vol. 2242, April 1994.

[39] P. Perona, "Orientation diffusions," *IEEE Transactions on Image Processing*, vol. 7, no. 3, pp. 457–467, 1998.

[40] B. Tang, G. Sapiro, and V. Caselles, "Diffusion of general data on non-flat manifolds via harmonic maps theory: The direction diffusion case," *International Journal of Computer Vision*, vol. 36, no. 2, pp. 149–161, 2000.

[41] D. Tschumperlé and R. Deriche, "Orthonormal vector sets regularization with pdes and applications," *International Journal of Computer Vision*, vol. 50, no. 3, pp. 237–252, 2002.

[42] X. Pennec and N. Ayache, "Uniform distribution, distance and expectation problems for geometric features processing," *J. Math. Imaging Vis.*, vol. 9, no. 1, pp. 49–67, 1998.

[43] T. Chan and J. Shen, "Variational restoration of nonflat image features: Models and algorithms," *SIAM J. Appl. Math.*, vol. 61, no. 4, pp. 1338–1361, 2000.

[44] L. A. Vese and S. J. Osher, "Numerical methods for p-harmonic flows and applications to image processing," *SIAM J. Numer. Anal.*, vol. 40, no. 6, pp. 2085–2104, 2002.

[45] R. Kimmel and N. Sochen, "Orientation diffusion or how to comb a porcupine," *Journal of Visual Communication and Image Representation*, pp. 238–248, 2001.

[46] D. Comaniciu and P. Meer, "Mean shift analysis and applications," in *Proceedings of ICCV'99*, Corfu, Greece, Sept 1999, pp. 1197–1203.

[47] M. Felsberg and G. Sommer, "The monogenic signal," *IEEE Transactions on Signal Processing*, vol. 49, no. 12, pp. 3136–3144, December 2001.

[48] G. H. Granlund and H. Knutsson, *Signal Processing for Computer Vision*, Kluwer Academic Publishers, Dordrecht, 1995.

[49] G. H. Granlund, "In search of a general picture processing operator," *Computer Graphics and Image Processing*, vol. 8, pp. 155–173, 1978.

[50] M. Felsberg and G. Sommer, "A new extension of linear signal processing for estimating local properties and detecting features," in *22. DAGM Symposium Mustererkennung, Kiel*, G. Sommer, N. Krüger, and C. Perwass, Eds. 2000, pp. 195–202, Springer, Heidelberg.

[51] B. Johansson, "Representing multiple orientations in 2D with orientation channel histograms," Tech. Rep. LiTH-ISY-R-2475, Dept. EE, Linköping University, November 2002.

[52] H. Knutsson and M. Andersson, "Robust N-dimensional orientation estimation using quadrature filters and tensor whitening," in *Proceedings of IEEE International Conference on Acoustics, Speech, & Signal Processing*, Adelaide, Australia, April 1994, IEEE.