Chapter 2
Adaptive Filtering Using Channel Representations

Michael Felsberg

Abstract This review article gives an overview on adaptive filtering methods based on channel representations. The framework of channel representations and its relation to density estimation is introduced. The fast and accurate scheme of virtual shift decoding is introduced and applied in several variants of channel smoothing:

- channel smoothing with alpha-synthesis for improving stability of edge-enhancing filtering
- orientation adaptive channel smoothing with applications to coherence-enhancing filtering
- channel smoothing using graph-cuts for improving filtering results at corners
- channel-coded feature maps (CCFMs) which lead to a significant speed-up of channel averaging
- CCFM-based smoothing based on optimal parameters derived from a novel uncertainty relation

For each method, an algorithmic description and some examples of results are provided, together with discussions and references of the original papers. Cross connections to other articles in this volume are given where appropriate.

2.1 Introduction

Adaptive filtering is an important field of image processing that has been considered by many researchers during the past three decades. Many different variants of adaptivity have been discussed in the literature, but the most relevant instances are probably orientation and scale.

The concept of scale was first introduced systematically in terms of the concept of *linear scale space* [232, 261, 450], establishing a 3D space of spatial coordinates and a scale coordinate. Often identified with Gaussian low-pass filtering, a rigorous

M. Felsberg (✉)
Computer Vision Laboratory, Department of Electrical Engineering, Linköping University, 58183 Linköping, Sweden
e-mail: michael.felsberg@liu.se

L. Florack et al. (eds.), *Mathematical Methods for Signal and Image Analysis and Representation*, Computational Imaging and Vision 41,
DOI 10.1007/978-1-4471-2353-8_2,

analysis of underlying scale space axioms [442] has led to the discovery of the Poisson scale space [138] and more general α scale spaces [116].

In practice, discrete scale spaces are mostly sub-sampled with increasing scale parameter, leading to the concept of scale-pyramids [63, 188], multi-scale analysis and wavelet theory [92, 298], cf. Chap. 7. While pyramids and wavelets speedup the computation of linear operators and transforms, non-linear scale space methods are widely used, e.g. for image enhancement. Non-linear scale space is based on a non-stationary or anisotropic diffusivity function [334, 432], cf. also Chaps. 5 and 9.

More recently, non-linear methods have been introduced which are less directly connected to linear scale space and diffusion, but allow for faster processing and partially superior results [139, 346]. The former method is based on wavelets, whereas the latter one is based on the channel representation [189] and is called channel smoothing. Combining the channel representation with a systematic decimation of spatial resolution, similar to the pyramid approach, has been applied in blob-detection [160] and in channel-coded feature maps (CCFM) [241, 244], a density representation in spatio-featural domain, see also [137].

Non-linear filtering based on anisotropic diffusivity is closely related to orientation adaptive filtering [134]. Orientation adaptive filtering, often referred to steerable filters [171], goes actually back on the work [259]. In more recent work, it has been extended to orientation selective channel smoothing [136] and orientation scores and their processing [117].

This review paper focusses on the channel-based variants of adaptive filtering and it is structured as follows. The framework of channel representations and its relation to density estimation is introduced in Sect. 2.2. The fast and accurate scheme of virtual shift decoding is introduced in Sect. 2.3. Sections 2.4–2.8 summarize several variants of channel smoothing:

1. channel smoothing with alpha-synthesis for improving stability of edge-enhancing filtering
2. orientation adaptive channel smoothing with applications to coherence-enhancing filtering
3. channel smoothing using graph-cuts for improving filtering results at corners
4. channel-coded feature maps (CCFMs) which lead to a significant speed-up of channel averaging
5. CCFM-based smoothing based on optimal parameters derived from a novel uncertainty relation

For each method, an algorithmic description and some examples of results are provided, together with discussions of the original papers.

2.2 The Channel Representation

Channel coding, also called population coding [347, 460], is a biologically inspired data representation, where features are represented by weights assigned to ranges of feature values [189, 230], see Fig. 2.1. Similar feature representations can also be found in the visual cortex of the human brain, e.g. in the cortical columns.

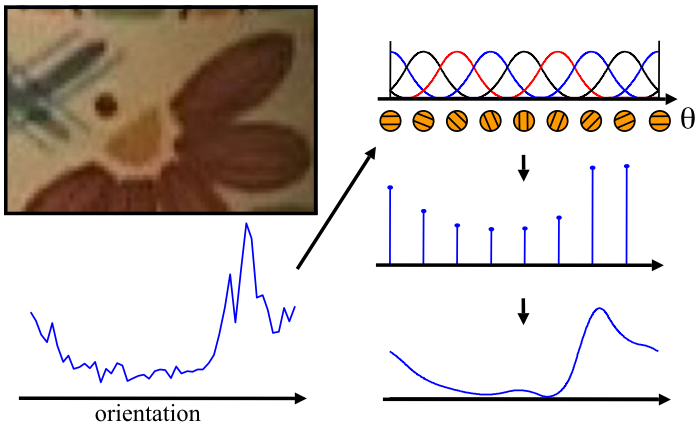

Fig. 2.1 Orientation distribution is encoded into channels, resulting in a (low-pass filtered) reconstruction using maximum entropy [242], see also Chap. 16. Figure courtesy by Erik Jonsson

The closer the current feature value f to the respective feature interval center n, the higher the channel weight c_n:

$$c_n(f) = k(f-n), \quad n \in \mathbb{N}, \tag{2.1}$$

where $k(\cdot)$ is a suitable kernel function and where f has been scaled such that it has a suitable range (note that we chose to place the channel centers at integers). By introducing z as a continuous feature coordinate, $k_n(z) = k(z-n)$, and $\delta_f(z) = \delta(z-f)$ denoting the Dirac-delta at f, the encoding can be written as a scalar product

$$c_n(f) = \langle \delta_f | k_n \rangle = \int \delta_f(z) k_n(z)\, dz \tag{2.2}$$

or as a sampled correlation in the feature-domain:

$$c_n = (\delta_f \star k)(n) = \int \delta_f(z') k(z'-z)\, dz' \Big|_{z=n}. \tag{2.3}$$

From the weights of all channels the feature value can be decoded unambiguously by finding the mode, where the decoding depends on the kernel function. In what follows, we have been using quadratic B-splines:

$$B_2(z) = \begin{cases} (z+3/2)^2/2 & -3/2 < z \le -1/2, \\ 3/4 - z^2 & -1/2 < z \le 1/2, \\ (z-3/2)^2/2 & 1/2 < z < 3/2, \\ 0 & \text{otherwise.} \end{cases} \tag{2.4}$$

This results in the encoding scheme Algorithm 2.1, where N denotes the number of channels.

Algorithm 2.1 Channel Encoding Algorithm

Require: $f \in [1.5; N - 0.5]$

1: $\mathbf{c} \Leftarrow 0$
2: **for all** samples f **do**
3: $\quad i \Leftarrow \text{round}(f)$
4: $\quad g \Leftarrow f - i$
5: $\quad c_{i-1} \Leftarrow c_{i-1} + (g - 1/2)^2/2$
6: $\quad c_i \Leftarrow c_i + 3/4 - g^2$
7: $\quad c_{i+1} \Leftarrow c_{i+1} + (g + 1/2)^2/2$
8: **end for**

The features can be scalar valued or vector valued, e.g. grey-scales, color vectors, or orientations. In the case of scalar features the decoding from quadratic B-splines has been considered in detail in [139], see next section. For the case of non-interfering channel weights, a simplified scheme based on the quotient of linear combinations can be used:

$$M_n = c_{n-1} + c_n + c_{n+1}, \qquad n_0 = \arg\max M_n, \qquad \hat{f} = \frac{c_{n_0+1} - c_{n_0-1}}{M_{n_0}} + n_0, \tag{2.5}$$

where $\hat{f}$ is our estimate of the feature f that had been encoded in c_n.

2.3 Virtual Shift Decoding

The decoding scheme (2.5) is not sufficient in all applications due to its quantization effects, i.e., the decoding result is biased toward the channel centers. If avoiding quantization effects in the decoding is essential, the *virtual shift decoding* [139] can be applied. For its derivation the reader should refer to the original paper. The algorithm works as follows. Let $h = 2\sqrt{2} - 3$. In a first processing step, the channels are forward and backward filtered according to

$$c_n^+ = c_n + h c_{n-1}^+, \quad (n = 2, \ldots, N), \tag{2.6}$$

$$c_n^- = h(c_{n+1}^- - c_n^+), \quad (n = N - 1, \ldots, 1), \tag{2.7}$$

$$c_n' = 8 c_n^-. \tag{2.8}$$

For finite domains, the boundary conditions are given as

$$c_1^+ = c_1, \tag{2.9}$$

$$c_N^- = \frac{h}{h^2 - 1} c_N^+. \tag{2.10}$$

For periodic domains, a DFT-based method should be applied instead of (2.6–2.10)

$$\text{DFT}_N(\mathbf{c}'') = 8(\text{DFT}_N([6\ 1\ 0\ \ldots\ 0\ 1]_N))^{-1}\text{DFT}_N(\mathbf{c}), \tag{2.11}$$

where DFT_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

$$p(f) = \sum_{n=1}^{N} c'_n B_2(f-n) \tag{2.12}$$

results in the original channel vector $\mathbf{c}$ again, i.e., $p(f)$ is a proper interpolation of the channel vector.

In order to extract the modes f_n at channels n, define $\beta_n = f_n - n$ and solve

$$0 = \lambda\beta_n^2 + \mu\beta_n + \nu \quad \text{with} \tag{2.13}$$

$$\lambda = (c'_{n_0-2} - 2c'_{n_0-1} + 2c'_{n_0+1} - c'_{n_0+2})/2, \tag{2.14}$$

$$\mu = (-c'_{n_0-2} + 2c'_{n_0} - c'_{n_0+2})/2, \tag{2.15}$$

$$\nu = (c'_{n_0-2} + 6c'_{n_0-1} - 6c'_{n_0+1} - c'_{n_0+2})/8 \tag{2.16}$$

such that the minimum of the error corresponds to

$$\beta_n = \frac{-\mu/2 + \sqrt{\mu^2/4 - \nu\lambda}}{\lambda}. \tag{2.17}$$

Solutions where $|\beta_n| > 1/2$ must be excluded, since they are located outside the unit interval around n. For valid solutions of β_n, the decoded mode is given by $f_n = n + \beta_n$. For periodic domains, the correct values are obtained after a modulo operation.

In order to extract the maximum likelihood estimate or to order the modes, the robust error needs to be known. It is given at channel n as

$$E(n) = \frac{23}{24} + \beta_n\nu + \beta_n^2\mu/2 + \beta_n^3\lambda/3 - \frac{c'_{n_0-2} + 24c'_{n_0-1} + 46c'_{n_0} + 24c'_{n_0-1} + c'_{n_0-2}}{48}. \tag{2.18}$$

The virtual shift decoding algorithm is summarized in Algorithm 2.2.

Channel representations obviously need more memory than directly storing features, but this investment pays off in several ways which we will show in the subsequent sections.

Algorithm 2.2 Virtual Shift Decoding Algorithm

Require: $\mathbf{c}$ is non-negative and normalized

if periodic domain **then**
  $\mathbf{c} \Leftarrow \mathrm{IDFT}_N(8(\mathrm{DFT}_N([6\ 1\ 0\ \ldots\ 0\ 1]_N))^{-1}\mathrm{DFT}_N(\mathbf{c}))$
  $\mathbf{c} \Leftarrow [c_{N-1}\ c_N\ \mathbf{c}^T\ c_1\ c_2]^T$
else
  $h \Leftarrow 2\sqrt{2}-3$
  for $n = 2$ to N **do**
    $c_n \Leftarrow c_n + hc_{n-1}$
  end for
  $c_N \Leftarrow 8\frac{h}{h^2-1}c_N$
  for $n = N-1$ to 1 **do**
    $c_n \Leftarrow h(c_{n+1} - 8c_n)$
  end for
end if
$\boldsymbol{\lambda} \Leftarrow \mathrm{conv}(\mathbf{c}, [-\frac{1}{2}\ 1\ 0\ -1\ \frac{1}{2}])$
$\boldsymbol{\mu} \Leftarrow \mathrm{conv}(\mathbf{c}, [-\frac{1}{2}\ 0\ 1\ 0\ -\frac{1}{2}])$
$\boldsymbol{\nu} \Leftarrow \mathrm{conv}(\mathbf{c}, [-\frac{1}{8}\ -\frac{3}{4}\ 0\ \frac{3}{4}\ \frac{1}{8}])$
$\boldsymbol{\beta} \Leftarrow (-\boldsymbol{\mu}/2 + \sqrt{\boldsymbol{\mu}.^2/4 - \boldsymbol{\nu}.\boldsymbol{\lambda}})./\boldsymbol{\lambda}$
$\boldsymbol{\gamma} \Leftarrow \mathrm{conv}(\mathbf{c}, [\frac{1}{48}\ \frac{1}{2}\ \frac{23}{24}\ \frac{1}{2}\ \frac{1}{48}])$
$\mathbf{f} \Leftarrow \boldsymbol{\beta} + [1\ 2\ \ldots\ N]$
$\mathbf{E} \Leftarrow \frac{23}{24} + (-1 < 2\boldsymbol{\beta} < 1).(\boldsymbol{\beta}.\boldsymbol{\nu} + \boldsymbol{\beta}.^2.\boldsymbol{\mu}/2 + \boldsymbol{\beta}.^3.\boldsymbol{\lambda}/3 - \boldsymbol{\gamma})$

2.4 Channel Smoothing

The idea of channel smoothing is based on considering the feature f in the encoding (2.1) as a stochastic variable. It has been shown in [139] that the distribution p_f is approximated by c_n in expectation sense (see also Fig. 2.2):

$$E\{c_n(f)\} = (p_f \star k)(n) \tag{2.19}$$

such that $\hat{f}$ becomes a maximum-likelihood estimate of f.

If we assume that p_f is locally ergodic, we can estimate $\hat{f}$ from a local image region, which corresponds to a local averaging of the channel weights within a spatial neighborhood. The algorithm consisting of the three steps channel encoding, channel averaging, and channel decoding is called channel smoothing (see Algorithm 2.3 and Fig. 2.3) and has been shown to be superior to many other robust smoothing methods [139], cf. Chap. 1. Due to its structure, Algorithm 2.3 is very well suited for parallel implementations. A nearly identical scheme has been proposed independently, but later, in [329].

Due to the point-wise decoding in Algorithm 2.3, the positioning of region boundaries might violate the sampling theorem, resulting in unstable edge-pixels. To avoid this effect, a modification to the channel decoding has been proposed

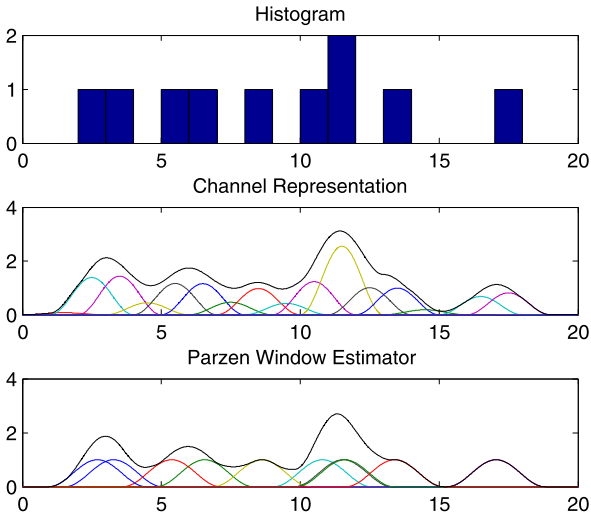

Fig. 2.2 Schematic comparison of density estimation using a histogram (*top*), a channel representation (*middle*), and a Parzen window estimator (*bottom*)

Algorithm 2.3 Channel Smoothing Algorithm

Require: $f \in [1.5; N - 0.5]$

1: **for all x do**
2: $\quad \mathbf{c}(\mathbf{x}) \Leftarrow \text{encode}(f(\mathbf{x}))$
3: **end for**
4: **for** $n = 1$ to N **do**
5: $\quad c_n \Leftarrow \text{conv2}(c_n, g_\sigma)$
6: **end for**
7: **for all x do**
8: $\quad [\mathbf{f}(\mathbf{x})\ \mathbf{E}(\mathbf{x})] \Leftarrow \text{decode}(\mathbf{c}(\mathbf{x}))$
9: $\quad i(\mathbf{x}) \Leftarrow \arg\max_n E_n(\mathbf{x})$
10: $\quad [\hat{f}(\mathbf{x})\ \hat{E}(\mathbf{x})] \Leftarrow [f_{i(\mathbf{x})}(\mathbf{x})\ E_{i(\mathbf{x})}(\mathbf{x})]$
11: **end for**

in [159], called α-synthesis, which creates smooth transitions between neighborhoods with different feature levels. Instead of extracting only the maximum in (2.5) or Algorithm 2.3, line 9, all local decodings are combined, weighted by the respective robust errors

$$\hat{f} = \frac{\sum_n f_n (\frac{23}{24} - E_n)^\alpha}{\sum_n (\frac{23}{24} - E_n)^\alpha}. \tag{2.20}$$

For the choice of α see [159]; we used $\alpha = 2$ throughout this paper. This method avoids aliasing artifacts at edges, but it does not avoid the rounding of corners, see Fig. 2.4.

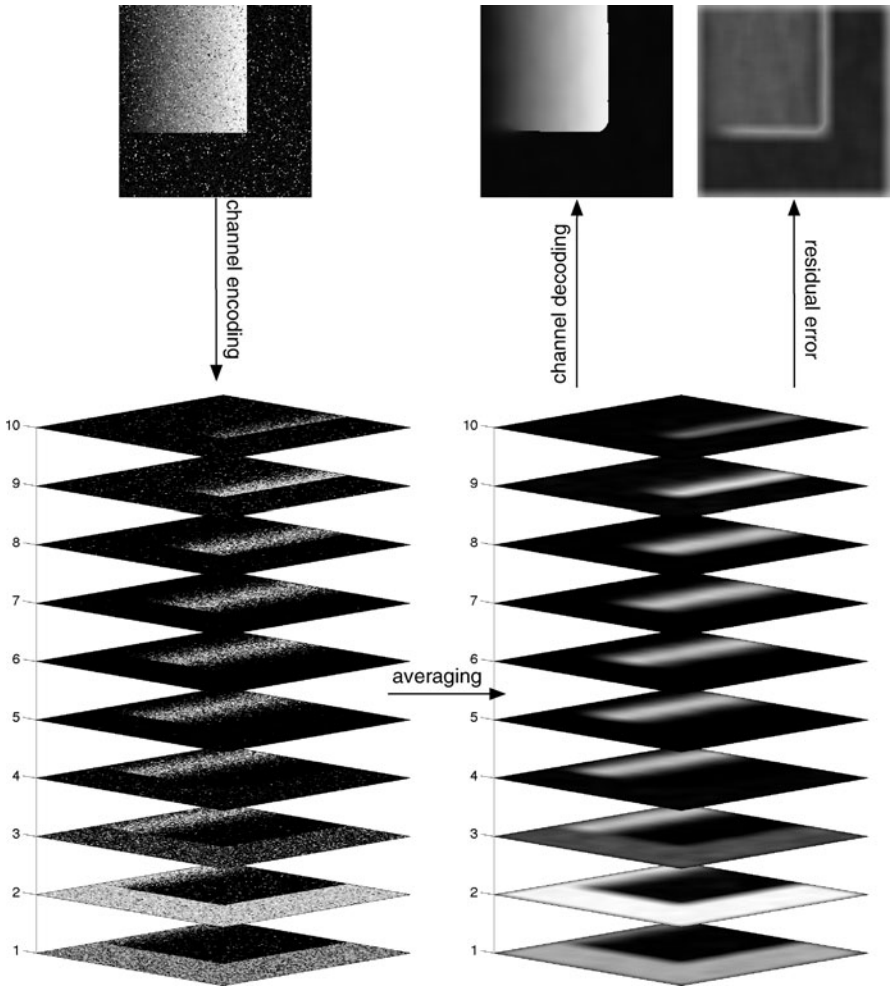

Fig. 2.3 The simple synthetic example (*top left*) smoothed by channel smoothing, using a Gaussian filter with $\sigma = 10$. No quantization effects are visible but note the rounding of the corner. *On the top right*: Robust error E of the decoding

Fig. 2.4 *From left to right*: Original noisy test image, result without alpha-synthesis (note the flip-overs at the edges), and result with alpha-synthesis

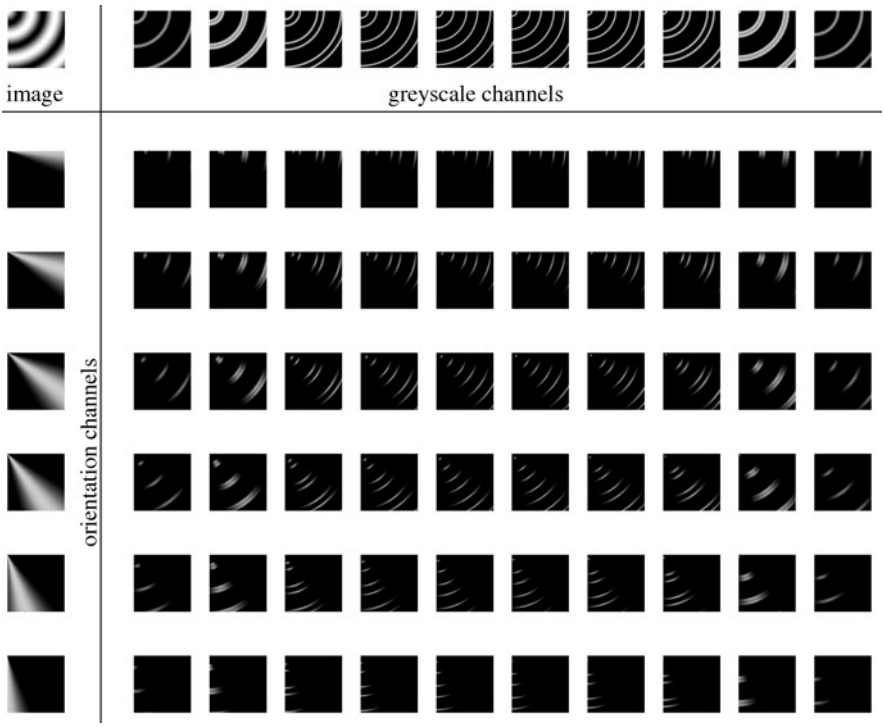

Fig. 2.5 Channel matrix using ten grey-scale channels and six orientation channels

2.5 Orientation Adaptive Channel Smoothing

The contents of this section is based on the publication [136], but the work is also closely related to Chaps. 3 and 10, as well as [295].

Channel smoothing as introduced in the previous section preserves edges, similar to edge-enhancing diffusion. Close to an edge, the filter support of a comparable linear filter is a semi-disc lying entirely on one side of the edge, thus giving a stable response everywhere except for points lying exactly on the edge. The latter problem has been solved by α-synthesis. For elongated, line-like structures, for instance in images of finger prints, the ordinary channel smoothing becomes however instable, as the majority of points are edge points. As a consequence, the resulting image suffers from 'flip-over' effects and contours become fragmented or displaced.

This problem is avoided if the channel smoothing is only performed along a 1D subspace, given by the local signal orientation θ_f, similar to the principle of adaptive filters [259] or coherence enhancing diffusion [439]. Hence, the channel smoothing operation must depend on the local signal orientation, which is itself represented using channels. The orientation information is typically of much slower variation than the intensity information and thus ordinary channel smoothing gives good results [139].

Algorithm 2.4 Orientation Adaptive Channel Smoothing Algorithm

Require: $f \in [1.5; N - 0.5]$
Require: $\theta_f \in [1.5; M - 0.5]$
1: **for all x do**
2: $\mathbf{c}_f(\mathbf{x}) \Leftarrow \text{encode}(f(\mathbf{x}))$
3: $\mathbf{c}_\theta(\mathbf{x}) \Leftarrow \text{encode}(\theta_f(\mathbf{x}))$
4: **end for**
5: **for** $m = 1$ to M **do**
6: $c_{\theta,m} \Leftarrow \text{conv2}(c_{\theta,m}, g_\sigma)$
7: **for** $n = 1$ to N **do**
8: $c_{n,m} \Leftarrow \text{conv2}(c_{\theta,m} c_{f,n}, g_m)$
9: **end for**
10: **end for**
11: **for all x do**
12: **for** $n = 1$ to N **do**
13: $[c_{n,m}(\mathbf{x})]_m \Leftarrow \text{normalize}([c_{n,m}(\mathbf{x})]_m)$
14: $[\mathbf{f}(\mathbf{x})\ \mathbf{E}(\mathbf{x})] \Leftarrow \text{decode}([c_{n,m}(\mathbf{x})]_m)$
15: $i(\mathbf{x}) \Leftarrow \arg\max_m E_m(\mathbf{x})$
16: $c_{f,n}(\mathbf{x}) \Leftarrow \frac{23}{24} - E_{i(\mathbf{x})}(\mathbf{x})$
17: **end for**
18: $\mathbf{c}_f(\mathbf{x}) \Leftarrow \text{normalize}(\mathbf{c}_f(\mathbf{x}))$
19: $[\mathbf{f}(\mathbf{x})\ \mathbf{E}(\mathbf{x})] \Leftarrow \text{decode}(\mathbf{c}_f(\mathbf{x}))$
20: $i(\mathbf{x}) \Leftarrow \arg\max_n E_n(\mathbf{x})$
21: $[\hat{f}(\mathbf{x})\ \hat{E}(\mathbf{x})] \Leftarrow [f_{i(\mathbf{x})}(\mathbf{x})\ E_{i(\mathbf{x})}(\mathbf{x})]$
22: **end for**

The smoothed orientation channels and the grey-scale channels are then combined in an outer product, see Fig. 2.5. The channel matrix encodes explicitly which grey-scale is present at which orientation. Hence, it is straightforward to adapt the smoothing kernel to the local orientation: we simply use *differently oriented* anisotropic smoothing kernel g_m for each orientation channel.

This smoothed, extended channel representation is then decoded in three steps using the standard decoding: Decode the orientation channel for each grey-scale, use the obtained error estimates to build a new grey-scale channel vector, and decode the new channel vector. The complete algorithm is summarized in Algorithm 2.4.

The ratio behind the sketched method is as follows. For oriented structures the channel matrix has a clear 2D maximum. Generating the grey-scale channel vector in line 16 yields a vector with a maximum at the appropriate grey-scale. The effective filter kernel is dominated by the anisotropic kernel corresponding to the strongest orientation channel, i.e., the structure is smoothed along its orientation and maintained perpendicular to its orientation, see Fig. 2.6.

For unoriented structures, the orientation decoding is random, but the resulting grey-scale channel will still be correct as the grey-scales are identical for all orientations. The effective filter kernel is a combination of anisotropic kernels with random

Fig. 2.6 Fingerprint experiment from [136]. *From left to right*: Original images, results from coherence enhancing diffusion [420], and results from channel smoothing. *Top*: Fingerprint at 300 dpi. *Bottom*: Zoomed detail. For further details and parameters, refer to [136]

orientation, thus resulting in an isotropic kernel. Hence, the filter output corresponds to isotropic smoothing.

A further aspect of adaptive filtering is the choice of the smoothing kernels, depending on the noise level [132] (for related work on noise level estimation, see also [140, 363]) and the noise distribution, e.g., multiplicative noise [372]. The selection of filter kernels is however out of the scope of this review and the interested reader is referred to the original publications.

2.6 Channel Smoothing Without Corner Rounding

The method described in this section is based on the publication [133], which proposes a method to avoid rounding of corners by restricting the smoothing to a generic domain where the respective channels are active. This means in practice: channel values should be averaged on bounded domains. Before looking into the issue of determining the active region of a channel, we have to modify the averaging step in channel smoothing in order to apply it to a bounded domain. Filtering of (uncertain) data f from a bounded domain is well modeled in terms of normalized convolution [258] of 0th order (normalized averaging):

$$\hat{f} = \frac{(a * (bf))}{(a * b)}, \tag{2.21}$$

where a denotes the applicability function, i.e., a suitable averaging kernel (typically a Gaussian function), b is the certainty function, which describes the bounded

domain Ω:

$$b(\mathbf{x}) = \begin{cases} 1 & \mathbf{x} \in \Omega, \\ 0 & \mathbf{x} \notin \Omega, \end{cases} \tag{2.22}$$

and $*$ denotes the convolution operation.

The co-domain of normalized convolution is however unbounded, and therefore, we cannot apply it directly to the case of channels c_n that are active in a bounded region. To remain within the active region of each channel, we mask the result from normalized convolution to the *same bounded domain* of each channel by means of the certainty function b_n:

$$\hat{c}_n = b_n \frac{(a * (b_n c_n))}{(a * b_n)}. \tag{2.23}$$

What remains to be considered is the estimation of the active region, or equivalently, the certainty function for each channel. We have to find those regions where the image was sufficiently stationary in order to produce *similar* channel vectors $\mathbf{c}(\mathbf{x})$. Similar channel vectors have the same active components, where we classify the activity by a simple threshold θ. As stationarity does not make sense without spatial context, we require the active region to be as connected as possible.

For each channel n we formulate the following objective function:

$$E(b_n) = \sum_{\mathbf{x}} b_n(\mathbf{x})(\theta - c_n(\mathbf{x})) + \lambda \sum_{\{\mathbf{x},\mathbf{y}\} \in \mathcal{N}} |b_n(\mathbf{x}) - b_n(\mathbf{y})|, \tag{2.24}$$

where we use the following parameters throughout the remainder of this paper:

- $\mathcal{N}$ is the four-neighborhood
- $\lambda = 0.3$ is the penalty for discontinuities in b_n
- $\theta = \frac{1}{16}$ is the threshold for active channels

All parameters were chosen according to [133] (as is the width of the Gaussian filter $\sigma = 10$ and the number of channels $N = 10$). The interested reader is referred to [132, 161] for estimation of the channel averaging filter and the number of channels. The threshold θ can be derived from classical decision theory (see e.g. [406], Chap. 3) and depends also on the number of channels. The meta parameter λ depends on the neighborhood structure and the signal statistics. It should be at least one third of the maximum channel value (minus θ) to fill in one-pixel gaps in a contour with four-neighborhood. Too large values will remove structural details from the active region. For quadratic B-spline channels and $\theta = \frac{1}{16}$ this happens for $\lambda \geq \frac{11}{32}$.

A binary labeling problem as formulated in (2.24) (see also Chap. 14) is efficiently solved by graph-cut algorithms [52]. Using graph-cut for determining the activation of channels, we obtain the graph-cut channel smoothing algorithm as given in Algorithm 2.5. The synthetic example from Fig. 2.3 shows that graph-cut channel smoothing does not suffer from the drawback of multi-label graph-cut (coarse quantization), nor does it suffer from rounding of corners as pure channel smoothing does, see Fig. 2.7. The computational complexity of the proposed method is

Algorithm 2.5 Graph-Cut Channel Smoothing Algorithm

Require: $f \in [1.5; N - 0.5]$

for all x do
  $\mathbf{c}(\mathbf{x}) \Leftarrow \text{encode}(f(\mathbf{x}))$
end for
for $n = 1$ to N **do**
  $b_n \Leftarrow \text{binary\_graph\_cut}(c_n, \mathcal{N}, \lambda, \theta)$
  $c_n \Leftarrow b_n \text{conv2}(b_n c_n, g_\sigma)/\text{conv2}(b_n, g_\sigma)$
end for
for all x do
  $[\mathbf{f}(\mathbf{x})\ \mathbf{E}(\mathbf{x})] \Leftarrow \text{decode}(\mathbf{c}(\mathbf{x}))$
  $i(\mathbf{x}) \Leftarrow \arg\max_n E_n(\mathbf{x})$
  $[\hat{f}(\mathbf{x})\ \hat{E}(\mathbf{x})] \Leftarrow [f_{i(\mathbf{x})}(\mathbf{x})\ E_{i(\mathbf{x})}(\mathbf{x})]$
end for

somewhat higher than that of pure channel smoothing and it is dominated by the N binary graph-cut computations.

2.7 Channel-Coded Feature Maps

Channel-coded feature maps have been suggested in [243] and are related to the theory in Chap. 4 in the sense that spatio-featural densities are estimated using channel representations.

One major drawback of channel smoothing is the extensive use of memory if many feature channels are required. A high density of channels is only reasonable if the spatial support is large, which implies that the individual feature channels are heavily low-pass filtered along the spatial dimension. Therefore, the feature channels have a lower band limit and can be sub-sampled in the spatial domain without losing information. If the three steps of channel encoding, channel averaging, and sub-sampling are integrated into a single step, channel-coded feature maps (CCFMs) are generated. The advantage of CCFMs is a much higher number of channels, e.g. by combining several features as in Fig. 2.8, without increasing the memory requirements significantly. The CCFM encoding of a single feature point can be written as (cf. (2.1)):

$$c_{l,m,n}(f(x,y),x,y) = k_f(f(x,y) - n)k_x(x - l)k_y(y - m), \tag{2.25}$$

where k_f, k_x, k_y are the 1D kernels in feature domain and spatial domain. Note that x and y are scaled such that they suit the integer spatial channel centers l, m. Similar to (2.1), the encoding (2.25) of a set of feature points can be written as a scalar product in 3D function space or as a 3D correlation, where we use

$$\delta_f(x,y,z) = \delta(z - f(x,y)) \tag{2.26}$$

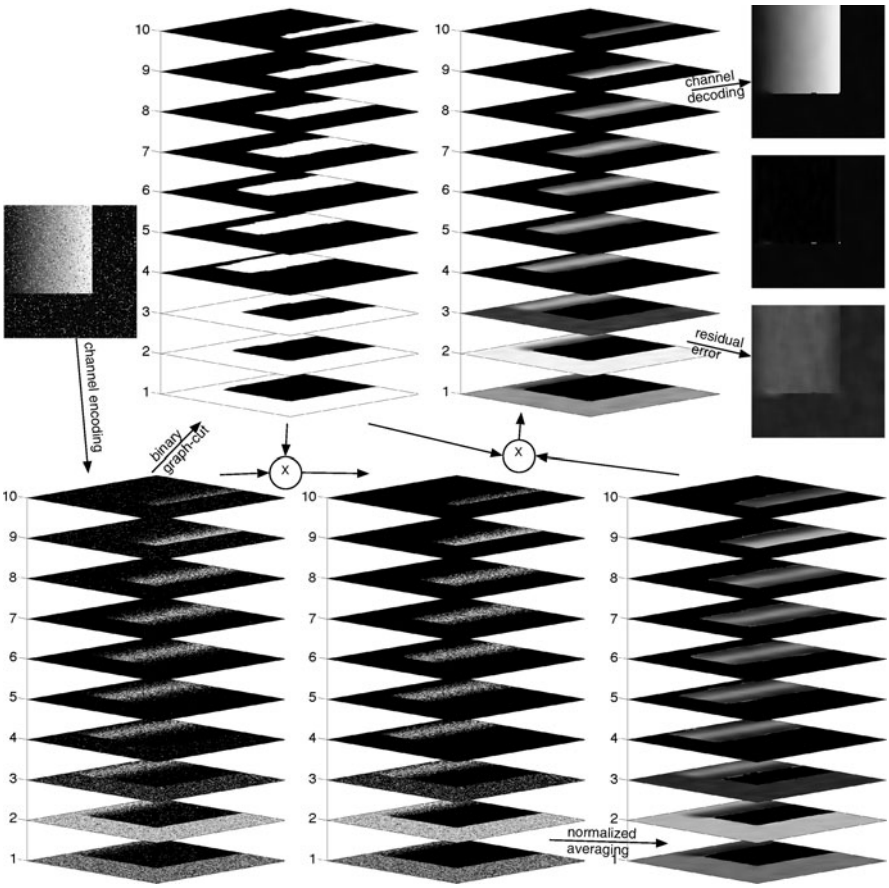

Fig. 2.7 Simple synthetic example from Fig. 2.3 smoothed by graph-cut channel smoothing, using a Gaussian filter with $\sigma = 10$. *Top right*: Result from graph-cut channel smoothing. *Below*: Actual absolute error compared to noise-free gradient image and reconstruction error. Note the absence of rounding of the corner

and $k_{f,n}(z) = k_f(z-n)$, $k_{x,l}(x) = k_x(x-l)$, $k_{y,m}(y) = k_y(y-m)$:

$$c_{l,m,n}(f) = \langle \delta_f | k_{f,n} k_{x,l} k_{y,m} \rangle = \iiint \delta_f(x,y,z) k_{f,n}(z) k_{x,l}(x) k_{y,m}(y)\, dz\, dy\, dx$$
$$= (\delta_f \star (k_f k_x k_y))(n,m,l). \tag{2.27}$$

The final formulation is the starting point of the CCFM scale space, see next section. CCFMs can be computed very efficiently using monopieces [241, 244], but here we restrict ourselves to the more basic Algorithm 2.6, where $\otimes$ denotes the outer (Kronecker) product. CCFMs and their derivatives can be used efficiently for robust visual tracking [243].

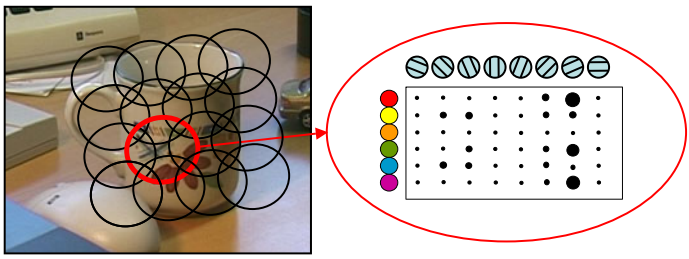

Fig. 2.8 Simultaneous encoding of orientation and color in a local image region. Figure taken from [241] courtesy Erik Jonsson

Algorithm 2.6 CCFM Algorithm

Require: $f \in [1.5; N - 0.5]$
Require: $\mathbf{x} = (x, y)^T \in [1.5; X - 0.5] \times [1.5; Y - 0.5]$
1: $\mathbf{C} \Leftarrow 0$
2: **for all x do**
3: $\quad \mathbf{c}_f \Leftarrow \text{encode}(f(\mathbf{x}))$
4: $\quad \mathbf{c}_x \Leftarrow \text{encode}(x)$
5: $\quad \mathbf{c}_y \Leftarrow \text{encode}(y)$
6: $\quad \mathbf{C} \Leftarrow \mathbf{C} + \mathbf{c}_f \otimes \mathbf{c}_x \otimes \mathbf{c}_y$
7: **end for**

2.8 CCFM Scale Space

This section summarizes some recent results [135] on the concept of CCFM scale space and spatial-featural uncertainties.

The starting point is to embed the image $f(x, y)$ as a 3D surface according to (2.26). One might try to generate a 3D α scale space [116] (Gaussian as a special case $\alpha = 1$ and all α-kernels are symmetric, i.e., correlation and convolution are the same):

$$F_s(x, y, z) = (k_s^{(\alpha)} \star \delta_f)(x, y, z). \tag{2.28}$$

However, the semi-group property of scale space implies that all dimensions (spatial dimensions and the feature dimension) become increasingly blurred. Despite the fact that this implies a rapidly growing loss of information with increasing scale and a singular zero scale, this procedure is insensible from a statistical perspective and does not comply with the notion of scale selection [126, 290].

Since the latter argument is not straightforward, we explain our rationale in some more detail. From the requirement that the dimensionless derivative attains its maximum at a position proportional to the wavelength of the signal [290] (Sect. 13.1), we conclude that the scale of a structure is proportional to its spatial scale (a trivial fact) and anti-proportional to its feature scale. The latter can be shown by looking at the Taylor expansion of a harmonic oscillation $A \sin(\omega x)$ in the origin: $A\omega x$. The

steepness of a sinusoid $A\omega$ in the origin grows linearly with the amplitude and the frequency, i.e., it is anti-proportional to the wavelength $\lambda = \frac{2\pi}{\omega}$.

Alternatively, one can consider the energy of a harmonic oscillation. The energy is proportional to the square of the amplitude times the square of the frequency: $E \propto A^2\omega^2 \propto \frac{A^2}{\lambda^2}$. That means, if we apply a 3D lowpass filter to the spatio-featural domain, the energy decays with a power of four. Hence, scale selection would favor the highest possible frequencies in nearly all cases. If we scale the amplitude anti-proportionally to the spatial domain, the change of energy is balanced and will reflect intrinsic properties of the signal.

This relation is formalized in terms of a spatio-featural uncertainty relation, which is derived based on the group structure of spatio-featural transformations. We choose a methodology which is based on the isotropic model used in [264], although restricted to the 1D case. The higher-dimensional case generalizes straightforwardly. The group that we consider contains the shearing group and the translation group given as

$$x' = x + t_x, \tag{2.29}$$

$$f' = f + \tan(\phi)x + t_f. \tag{2.30}$$

The shearing transformation corresponds to the rotation of a Euclidean space and is obtained since the f-coordinate is a null-vector [264], i.e., $f \cdot f = 0$. The parametrization is chosen such that it reflects the fact that points move along the surface/curve with angle ϕ towards the ground plane. Using this definition we state the following

Theorem 2.1 *Let the spatio-featural domain be described by the isotropic model. The uncertainty product in the spatio-featural domain has a lower bound*

$$\exists k > 0: \quad (\Delta x)(\Delta f) \geq k \tag{2.31}$$

and the lower bound is given as

$$k = \frac{1}{2}\sigma_f \sigma_x, \tag{2.32}$$

where σ_f^2 is the variance of the feature domain marginal distribution and σ_x^2 is the variance of the spatial domain distribution.

The proof of this theorem is given in [135]. As a consequence of this theorem, optimal parameters for CCFM computation are derived. As an experimental validation, images have been reconstructed from these CCFMs showing that a good perceptual quality is maintained for a wide range of channels, see Fig. 2.9. The CCFM-smoothing algorithm is summarized in Algorithm 2.7, where the numbers of channels X, Y, and N are the optimized parameters mentioned before and the interpolation in line 3 generates a new channel vector from adjacent channel vectors by linear interpolation.

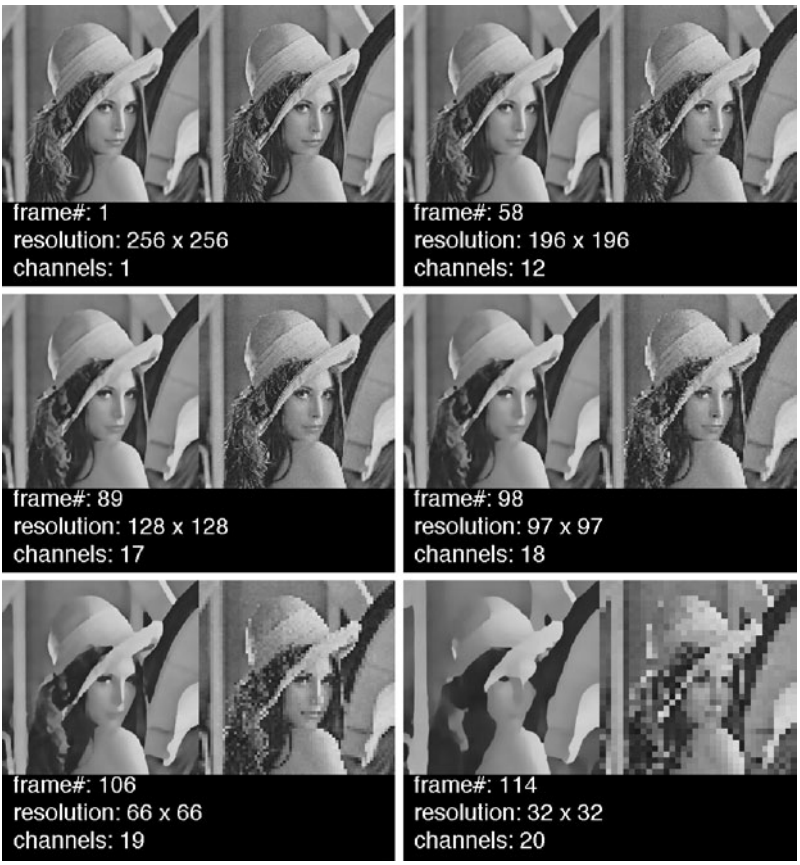

Fig. 2.9 Examples for CCFM-smoothing at different scales. The spatial and featural (denoted as *channels*) resolutions are given according to the spatio-featural uncertainty. The feature considered here is the greyscale

Algorithm 2.7 CCFM Smoothing Algorithm

Require: $f \in [1.5; N - 0.5]$
Require: $\mathbf{x} = (x, y)^T \in [1.5; X - 0.5] \times [1.5; Y - 0.5]$
$\mathbf{C} \Leftarrow \text{CCFM}(x, y, f)$
for all x do
  $\mathbf{c}_f \Leftarrow \text{interpolate}(\mathbf{C}, \mathbf{x})$
  $[\mathbf{f}(\mathbf{x})\ \mathbf{E}(\mathbf{x})] \Leftarrow \text{decode}(\mathbf{c}_f)$
  $i(\mathbf{x}) \Leftarrow \arg\max_n E_n(\mathbf{x})$
  $[\hat{f}(\mathbf{x})\ \hat{E}(\mathbf{x})] \Leftarrow [f_{i(\mathbf{x})}(\mathbf{x})\ E_{i(\mathbf{x})}(\mathbf{x})]$
end for

2.9 Conclusion

In this review paper, we have given a compact and concise overview over the field of channel-based filtering. Research on this topic is still in progress and more results on the efficient computation and other types of features are to be expected in the near future. Code for most of the presented work is available at the author's website.

Acknowledgements The author would like to thank P.-E. Forssén for various discussions about the paper, in particular on alpha-synthesis. The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreements no 215078 (DIPLECS) and 247947 (GARNICS) as well as the VR project 2009-4282.

http://www.springer.com/978-1-4471-2352-1

Mathematical Methods for Signal and Image Analysis and Representation

Florack, L.; Duits, R.; Jongbloed, G.; van Lieshout, M.C.; Davies, L. (Eds.)

2012, XII, 320 p., Hardcover
ISBN: 978-1-4471-2352-1