Tutorial on Channel Representations

Michael Felsberg

Computer Vision Laboratory
Linköping University, Sweden
What are channels?
Channel Representation

- *distribution functions* (Granlund, 1973)
- *channel coding* (Snippe/Koenderink, 1992)
- *bandpass channels* (Howard/Rogers, 1995)
- *population coding* (Zemel et al., 1998)
- *channel representation* (Granlund, 2000)
- *channel filtering* (Felsberg/Granlund, 2003)
- *channel smoothing* (Felsberg et al., 2006)
- “*bilateral filtering*” (Paris/Durand, 2006)
- *orientation scores* (Duits et al., 2007)
- *channel coded feature maps* (Jonsson/Felsberg, 2007)
- *distribution fields* (Sevilla-Lara/Learned-Miller, 2012)
Channel Representation

- distribution functions (Granlund, 1973)
- channel coding (Snippe/Koenderink, 1992)
- bandpass channels (Howard/Rogers, 1995)
- population coding (Zemel et al., 1998)
- channel representation (Granlund, 2000)
- channel filtering (Felsberg/Granlund, 2003)
- channel smoothing (Felsberg et al., 2006)
- “bilateral filtering” (Paris/Durand, 2006)
- orientation scores (Duits et al., 2007)
- channel coded feature maps (Jonsson/Felsberg, 2007)
- distribution fields (Sevilla-Lara/Learned-Miller, 2012)
Channel Representation

The channel representation is a biologically inspired method for data representation where features are represented by weights assigned to ranges of feature values. This approach is similar to feature maps and can be seen as channel coding or population coding.

In Channel Representation, a feature value is encoded as a weighted sum of channel responses. The encoding can be written as:

\[ c_n(f) = \langle \delta_f | k_n \rangle = \int \delta_f(z) k_n(z) \, dz \]

where \( k_n(z) = k(z - n) \) and \( \delta_f(z) = \delta(z - f) \).

The closer the current feature value is to the feature value associated with a channel, the higher the channel weight.

By introducing a suitable kernel function and where the channel centers are placed at integers, the channel representation can be used to enhance images. This approach is widely used for image enhancement.

More recently, non-linear methods have been introduced which are less computationally intensive than linear methods and partially superior results. The former method is based on iterative methods, and the latter approach uses channel smoothing.

In Section II, we introduce lesser known relevant techniques: channel representation, channel smoothing, CCFMs. In Section III, we propose the novel iterative methods. We show some examples and propose a scale selection scheme.

The CCFM scale space is generated by applying the principles of linear scale space smoothing to CCFMs and simultaneously increasing the resolution of linear operators and transforms. Non-linear scale space methods are applied based on a new uncertainty relation: the spatiotemporal uncertainty relation.

In Section IV, we develop a scale selection scheme based on the spatiotemporal uncertainty relation and an image reconstruction algorithm which generates images from CCFMs. By subsampling this space and subsequent reconstruction, we can generate images very similar to those generated by the pyramid approach.

The latter method has been applied in blob detection and in channel-coded feature maps (CCFM). More recently, non-linear methods have been introduced which are less computationally intensive than linear methods and partially superior results. The former method is based on iterative methods, and the latter approach uses channel smoothing.

We show some examples and propose a scale selection scheme. Further details can be found in [9].
Channel Representation

\[ c_n(f) = \langle \delta_f | k_n \rangle = \int \delta_f(z) k_n(z) \, dz \]

\[ k_n(z) = k(z - n) \]

\[ \delta_f(z) = \delta(z - f) \]
Encode values in $K$-D channel vector

$$x_k = F_k(f) \quad k = 1, \ldots, K$$

Motivated from population coding, sparse coding
Channel Representation

\[ x_k = F_k(f) \quad k = 1, \ldots, K \]

Motivated from population coding, sparse coding
Channel Representation

Motivated from population coding, sparse coding
Channel Representation

\[
x = [\begin{array}{ccccccccccc}
0 & 0.25 & 1.0 & 0.25 & 0 & 0 & 0 & 0.25 & 1.0 & 0.25 & 0 & 0 & 0
\end{array}]^T
\]
B-Spline Encoding

- The value of the $k$-th channel is obtained by
  \[ x_k(f) = B_2(f - k) \quad k = 1 \ldots K \]
  ($f$ is shifted and rescaled such that the channels are at integer positions)

\[ -2 \quad -1 \quad 0 \quad 1 \quad 2 \]

\[ 0 \quad 0.2 \quad 0.4 \quad 0.6 \quad 0.8 \quad 1 \]

- degree 3
- degree 2
- degree 1
- degree 0
B-Spline Encoding

- The value of the $k$-th channel is obtained by

$$x_k(f) = B_2(f - k) \quad k = 1 \ldots K$$

($f$ is shifted and rescaled such that the channels are at integer positions)

$$k = \text{round}(f)$$

$$x[k-1] = (f-k-0.5)^2/2$$

$$x[k] = 0.75 - (f-k)^2$$

$$x[k+1] = (k-f-0.5)^2/2$$

$$x[1 \ldots k-2] = x[k+2 \ldots K] = 0$$
Channels are ...

- soft histograms
- frame vector projections
- different from Parzen window/kernel density estimators (not located at samples)
Kernel Density Estimation

• Estimate pdf from samples by convolving with a kernel function

\[ \tilde{p}(f) = \frac{1}{N} \sum_{n=1}^{N} k(f - f_n) \]

• Expectation of estimate:

\[ \mathbb{E}\{\tilde{p}(f)\} = \int k(f - f') p(f') \, df' = (k * p)(f) \]
Relation to Channels

• Adding channel representation of samples = sampled kernel density estimation

\[
E \left\{ \frac{1}{N} \sum_{n=1}^{N} u_k(f_n) \right\} = E \{ \tilde{p}(f) \}\bigg|_{f=k} = \left( B_2 \ast p \right)(f)\bigg|_{f=k} \quad k = 1 \ldots K
\]
Problem: Image Denoising

- Real data is noisy and discontinuous
  - Inlier noise
  - Salt&Pepper noise
  - Image discontinuities
Channel Smoothing
Decoding

• Normalized convolution of the channel vector

\[ f_{k_0} = \frac{u_{k_0+1}(f) - u_{k_0-1}(f)}{u_{k_0-1}(f) + u_{k_0}(f) + u_{k_0+1}(f)} + k_0 \]

• Choice of \( k_0 \):
  – Largest denominator (3-box filter)
  – Additional: local maximum
• Minimize error functional:

\[ E(f_0) = \int (f - f_0)^2 p(f) \, df \]

\[ f_0 = \arg \min E(f_0) \]

• Idea of robust error norm:
  – saturated for outliers
  – quadratic near the origin

• in Bayesian sense
LS & Robust Optimization

• Minimize error functional:

\[ E(f_0) = \int \rho(f - f_0) p(f) \, df \]

\[ f_0 = \arg \min E(f_0) \]

• Idea of robust error norm:
  – saturated for outliers
  – quadratic near the origin

• in Bayesian sense
LS & Robust Optimization

- **Necessary condition:**
  \[ 0 = \int (f - f_0) \ p(f) \ df \]
  \[ f_0 = \int f \ p(f) \ df \]

- **Robust influence:**
  - zero for outliers
  - no direct solution
LS & Robust Optimization

• Necessary condition:

\[ 0 = \int \psi(f - f_0) p(f) \, df \]

\[ \psi = \rho' \]

• Robust influence:
  – zero for outliers
  – no direct solution

Efficient methods required!
Influence Function of C.R.

Obtained from linear decoding:

\[ \psi(f) = B_2(f - 1) - B_2(f + 1) \]

\[ f_{k_0} = \frac{u_{k_0+1}(f) - u_{k_0-1}(f)}{u_{k_0-1}(f) + u_{k_0}(f) + u_{k_0+1}(f)} + k_0 \]
Error Norm of C.R.

Obtained by integrating the influence function:

$$\rho(f) = 2B_3\left(\frac{1}{2}\right) - B_3\left(f + \frac{1}{2}\right) - B_3\left(f - \frac{1}{2}\right)$$
Quantization Effect

- Noisy signal
- Standard decoding, $\sigma=5$
- Standard decoding, $\sigma=50$
- Optimized decoding, $\sigma=50$
Quantization Effect

Not available in multiple dimensions!
Algorithm 1 Channel smoothing algorithm.

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

1: for $n = 1$ to $N$ do
2: \hspace{1em} $c_n \leftarrow B_2(f - n)$
3: \hspace{1em} $c_n \leftarrow g_\sigma \ast c_n$
4: end for
5: $[\hat{f}, \hat{\rho}] \leftarrow \text{decode}(\mathbf{c})$
Algorithm 1 Channel smoothing algorithm.

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

1: for $n = 1$ to $N$ do
2: \hspace{1cm} $c_n \leftarrow B_2(f - n)$
3: \hspace{1cm} $c_n \leftarrow g_\sigma \ast c_n$
4: end for
5: $[\hat{f}, \hat{\rho}] \leftarrow \text{decode}(c)$
Algorithm 1 Channel smoothing algorithm.

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

1: for \( n = 1 \) to \( N \) do
2: \( c_n \leftarrow B_2(f - n) \)
3: \( c_n \leftarrow g_\sigma * c_n \)
4: end for
5: \([\hat{f}, \hat{\rho}] \leftarrow \text{decode}(c)\)
Algorithm 1 Channel smoothing algorithm.

Require: $f \in [1.5; N - 0.5]$  
1: for $n = 1$ to $N$ do  
2: $c_n \leftarrow B_2(f - n)$  
3: $c_n \leftarrow g_\sigma \ast c_n$  
4: end for  
5: $[\hat{f}, \hat{\rho}] \leftarrow \text{decode}(c)$
Image Denoising
Random Sample

- Real data is incomplete
Orientation Estimation

orientation estimation using the Riesz transform

- raw estimates (35.6°)
- bilateral filtering (15.9°)
- diffusion (16.0°)
- channel smoothing (13.6°)
- mean shift filtering (17.5°)

Gaussian noise variance vs. mean orientation error

orientation estimation using the Riesz transform

- raw estimates (35.6°)
- bilateral filtering (15.9°)
- diffusion (16.0°)
- channel smoothing (13.6°)
- mean shift filtering (17.5°)

Salt & Pepper noise vs. mean orientation error
Disparity Estimation
Disparity Estimation
Drawback

• no coherence enhancing filtering possible
Channel Matrix
Channel Matrix
Experiment

original image  coherence enhancing diffusion  anisotropic channel smoothing
Experiment

original image  coherence enhancing diffusion  anisotropic channel smoothing
Corner Detection
Motivation CCFM

frame#: 103
resolution: 78 x 78
channels: 19
The entire CCFM algorithm can be summarized into a single encoding equation according to $x_{op}$:

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

where $k_f$, $k_x$, $k_y$ are the xD kernels in feature domain and spatial domain respectively.

Note that $x$ and $y$ are scaled such that they suit the integer spatial channel centers.

Note further that the previous definition of CCFMs assumes separable kernels but we could easily use non-separable kernels, e.g., in the case of orientation data.

Similar to $x_{op}$, the encoding $x_{zp}$ can be written as a scalar product in zD function space or as a zD convolution:

$$c_{l,m,n}(f(x, y)) = \langle f(x, y), \psi_{l,m,n} \rangle$$

The final formulation is the starting point of our new method.

3.1 Channel-Coded Feature Maps and Linear Scale-Space Theory

The starting point is to embed the image $b_{ox, y}$ as a zD surface according to $B_{ox, y, z}; w_p = \langle b_{ox, y}, \psi_{l,m,n} \rangle$.

Generate a zD scale-space by a Gaussian as a special case.

$B_{ox, y, z}; s_p = k_{ox, y, z}; s_p \ast B_{ox, y, z}; w_p$.
Object Recognition
COIL-100 Objects

- All 100 objects
- 12 / 60 view for training / evaluation

<table>
<thead>
<tr>
<th>Method</th>
<th>ROC int</th>
</tr>
</thead>
<tbody>
<tr>
<td>KLD, $\theta$</td>
<td>0.9817</td>
</tr>
<tr>
<td>SVD, $\theta$</td>
<td>0.9840</td>
</tr>
<tr>
<td>KLD, RGB</td>
<td>0.9983</td>
</tr>
<tr>
<td>SVD, RGB</td>
<td>0.9998</td>
</tr>
<tr>
<td>KLD, $hs\theta$</td>
<td>0.9939</td>
</tr>
<tr>
<td>SVD, $hs\theta$</td>
<td>1.0000</td>
</tr>
</tbody>
</table>
New Linear Scale-Space

• simultaneously increasing scale in spatial domain and feature domain is obviously wrong

• from a statistical point of view it makes sense to increase feature resolution with decreasing spatial resolution
Algorithm 7 CCFM smoothing algorithm.

Require: \( f \in [1.5; N^{-0.5}] \)

Require: \( x = (x, y)^T \in [1.5; X^{-0.5}] \times [1.5; Y^{-0.5}] \)

1: \( C \leftarrow \text{CCFM}(x, y, f) \)
2: for all \( x \) do
3: \( c_f \leftarrow \text{interpolate}(C, x) \)
4: \([f(x) \ E(x)] \leftarrow \text{decode}(c_f)\)
5: \( i(x) \leftarrow \arg \max_n E_n(x) \)
6: \([\hat{f}(x) \ \hat{E}(x)] \leftarrow [f_i(x)(x) \ E_i(x)(x)]\)
7: end for
Algorithm 7 CCFM smoothing algorithm.

Require: \( f \in [1.5; N \rightarrow 0.5] \)

Require: \( x = (x, y)^T \in [1.5; X \rightarrow 0.5] \times [1.5; Y \rightarrow 0.5] \)

1: \( C \leftarrow \text{CCFM}(x, y, f) \)
2: for all \( x \) do
3: \( c_f \leftarrow \text{interpolate}(C, x) \)
4: \( [f(x) \ E(x)] \leftarrow \text{decode}(c_f) \)
5: \( i(x) \leftarrow \arg \max_n E_n(x) \)
6: \( [\hat{f}(x) \ \hat{E}(x)] \leftarrow [f_{i(x)}(x) \ E_{i(x)}(x)] \)
7: end for

frame#: 103
resolution: 78 x 78
channels: 19