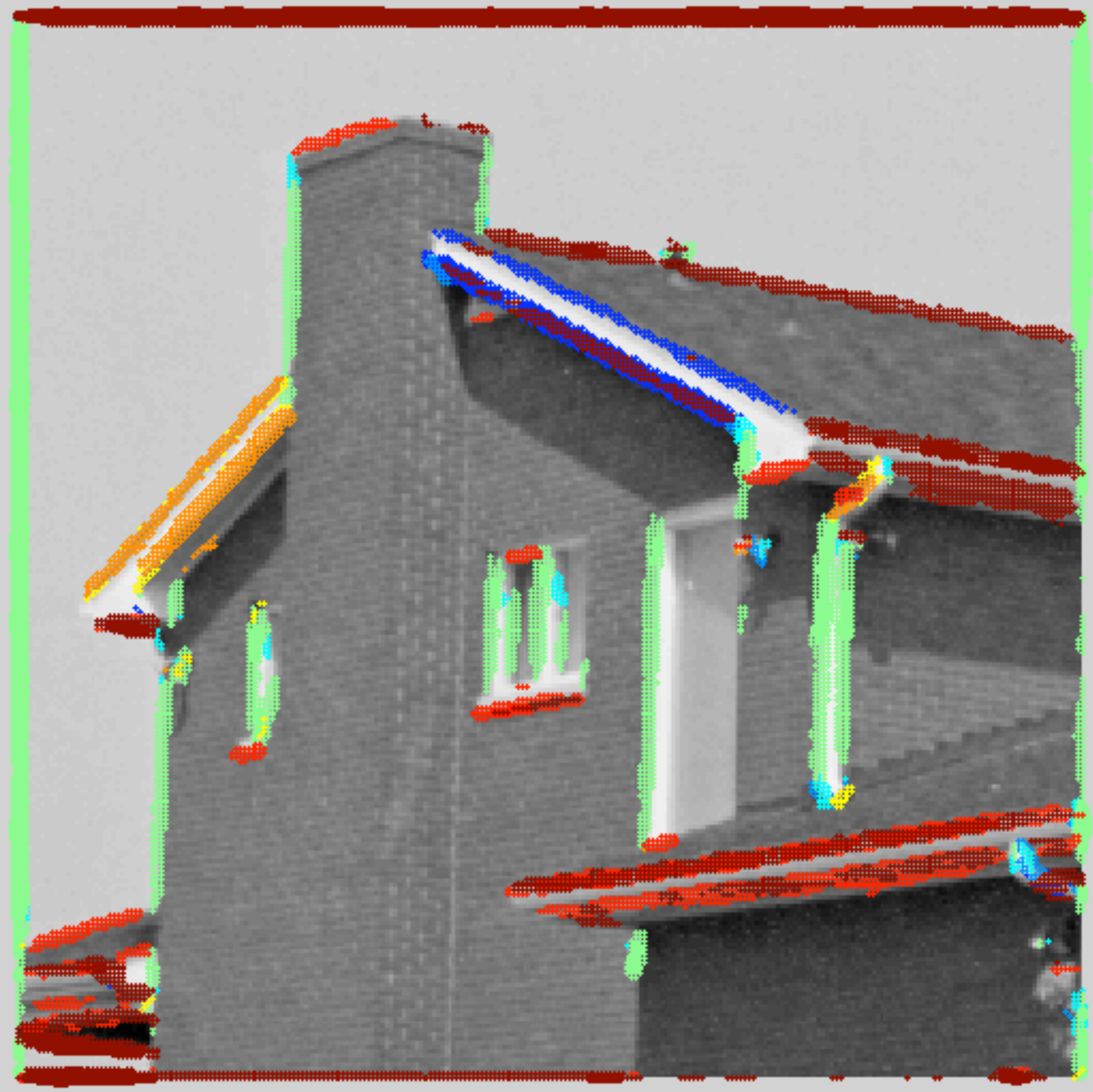
Tutorial on Channel Representations

Michael Felsberg

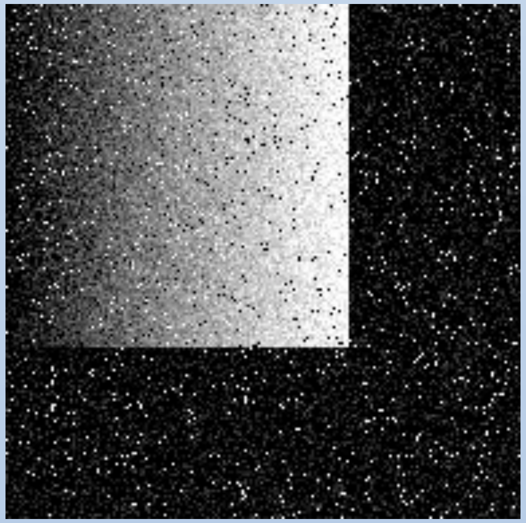
Computer Vision Laboratory
Linköping University, Sweden

What are channels?



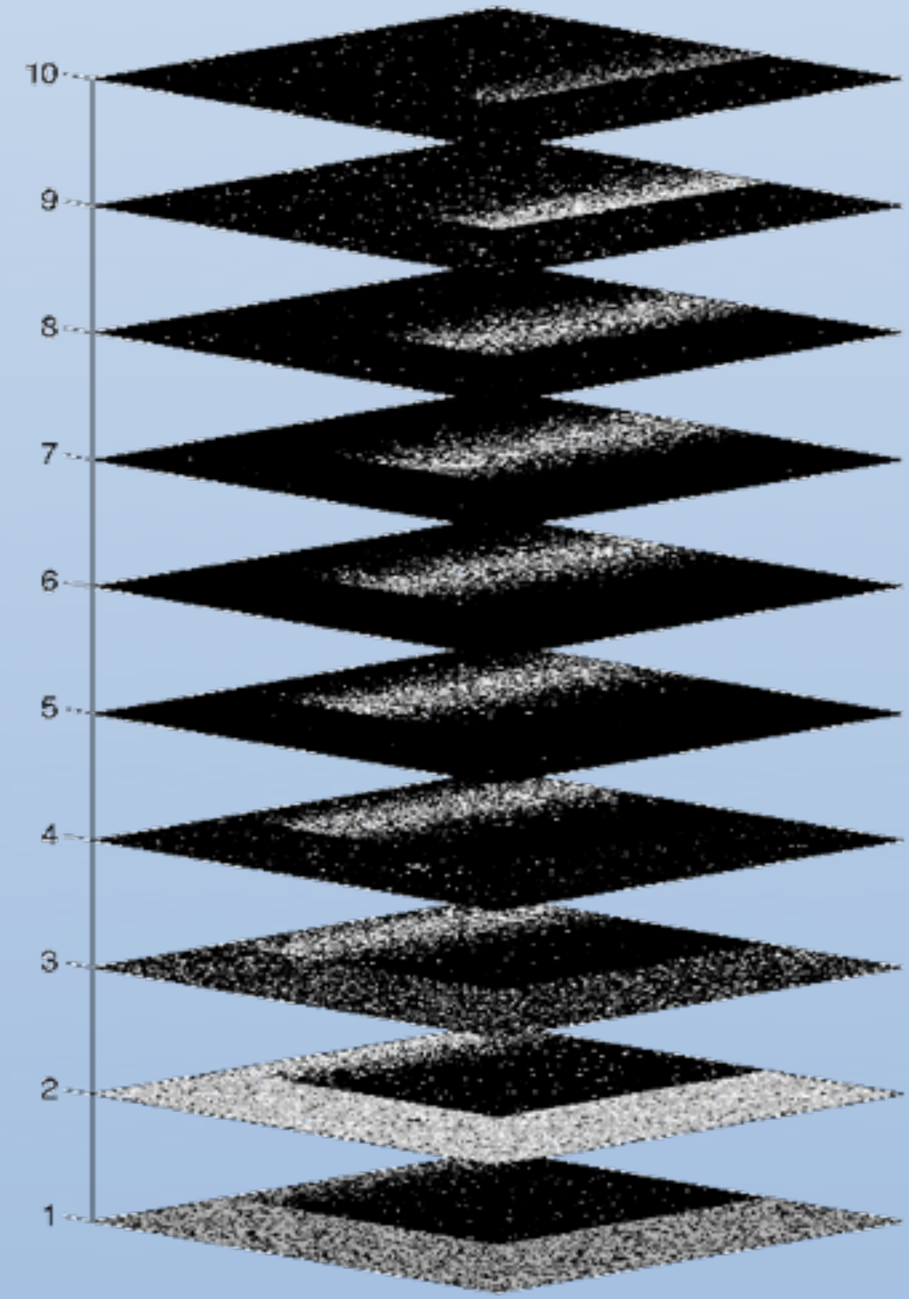
Channel Representation

- *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)

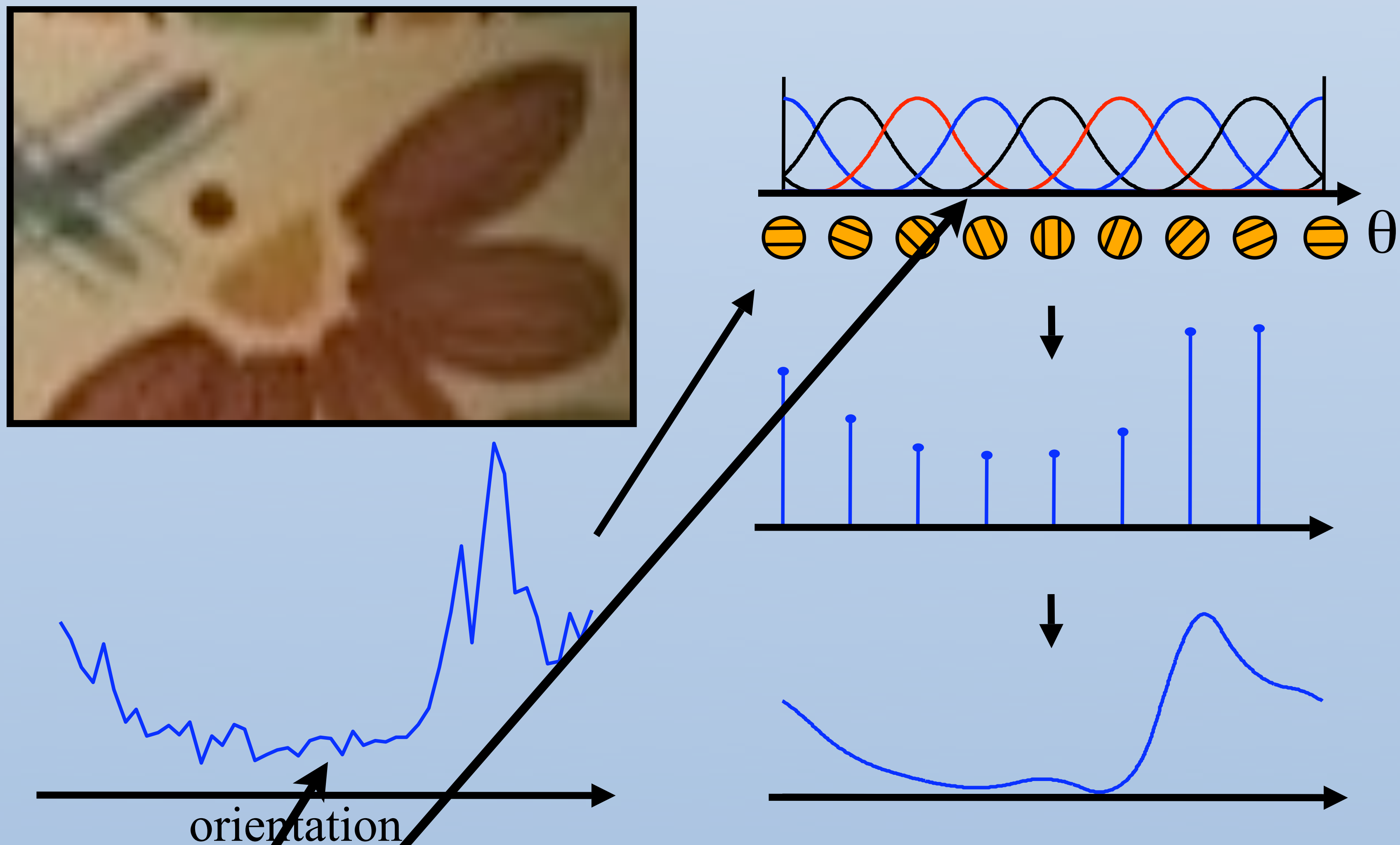


Channel Representation

- *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

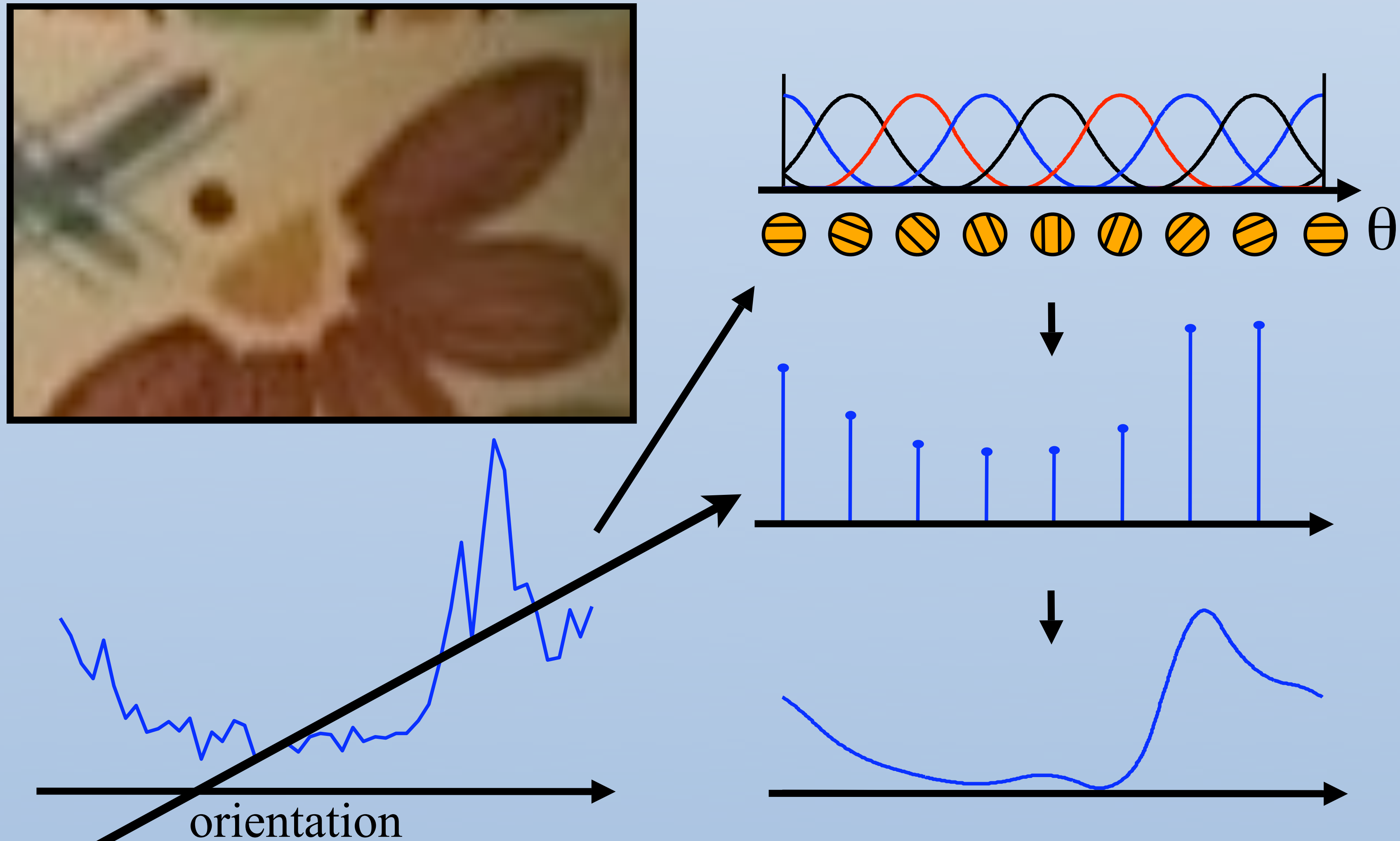


$$c_n(f) = \langle \delta_f | k_n \rangle = \int \delta_f(z) k_n(z)\, dz$$

$$k_n(z) = k(z-n)$$

$$\delta_f(z) = \delta(z-f)$$

Channel Representation

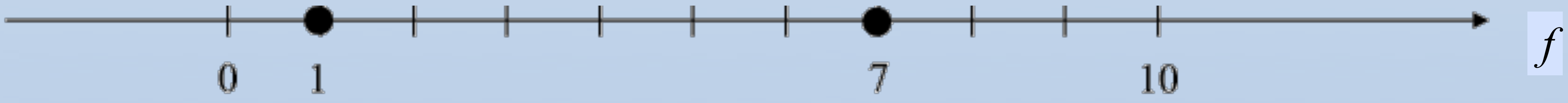


$$k_n(z) = k(z-n)$$

$$\delta_f(z) = \delta(z-f)$$

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

Channel Representation

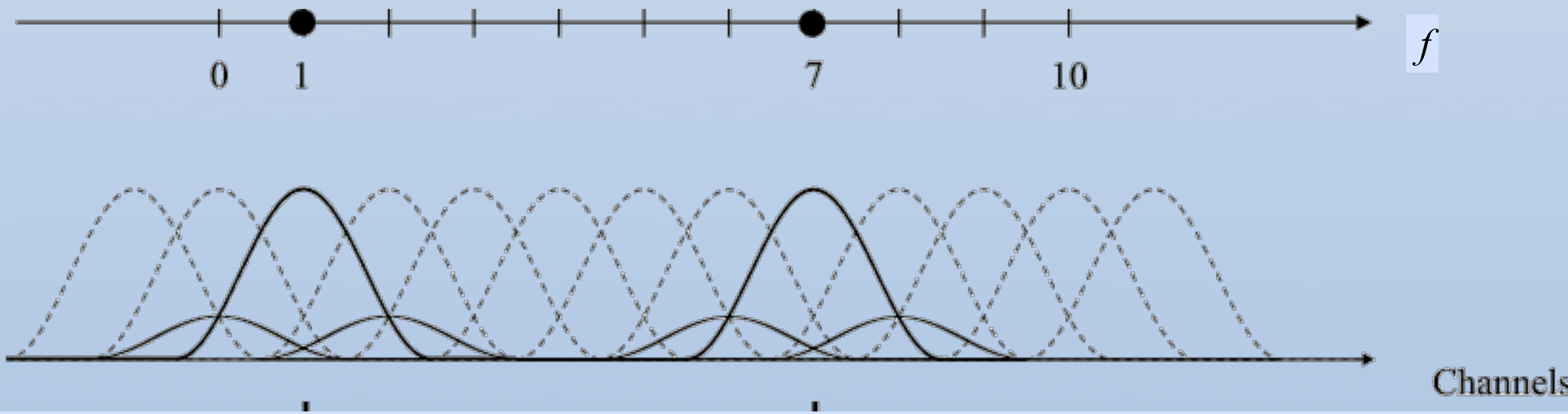


Encode values in K-D channel vector

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

Motivated from population coding, sparse coding

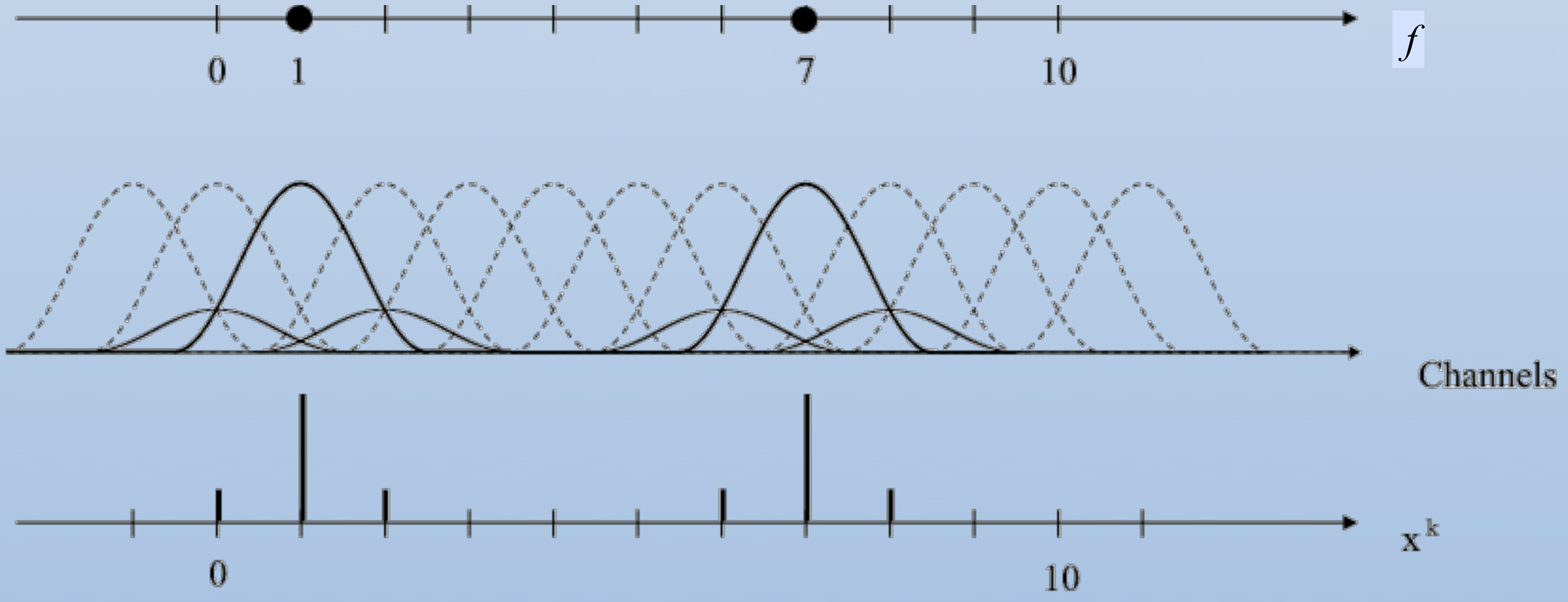
Channel Representation



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

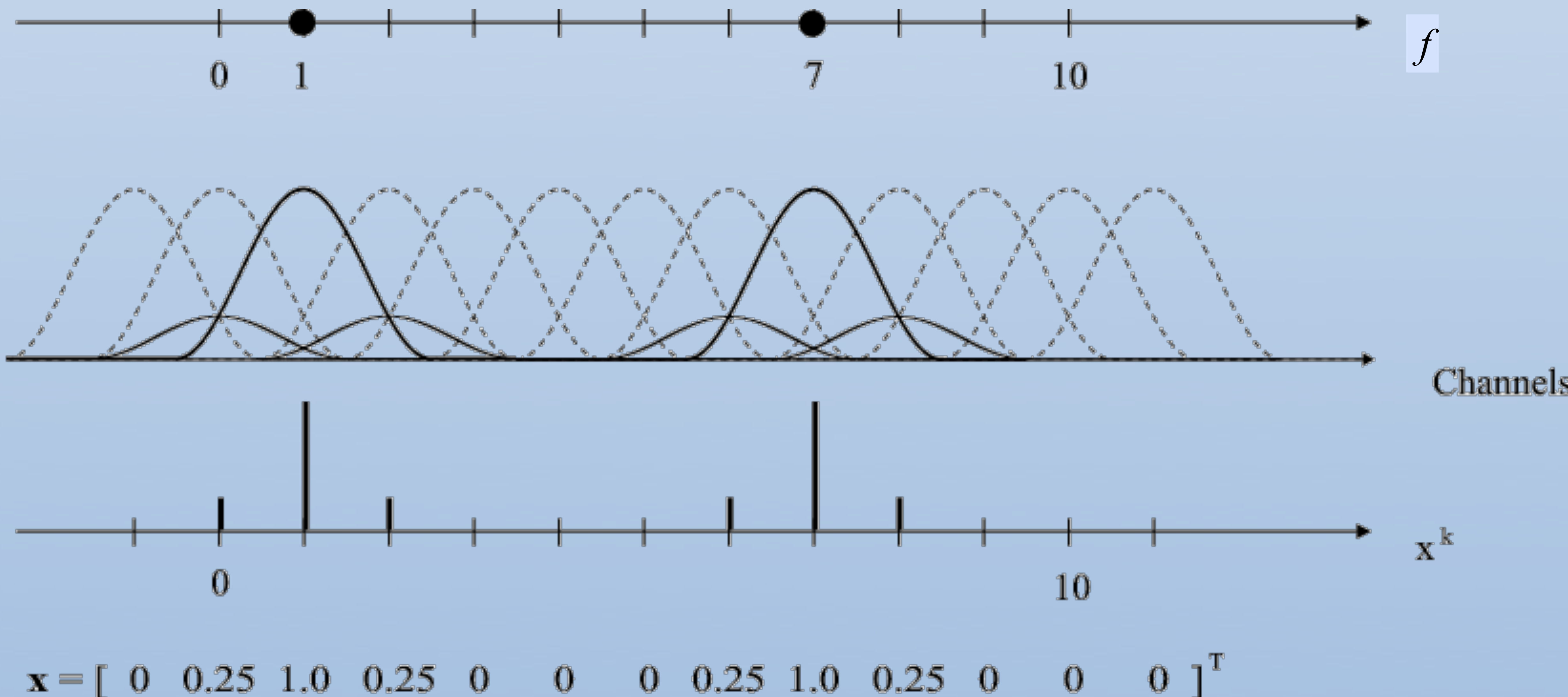
Motivated from population coding, sparse coding

Channel Representation



Motivated from population coding, sparse coding

Channel Representation

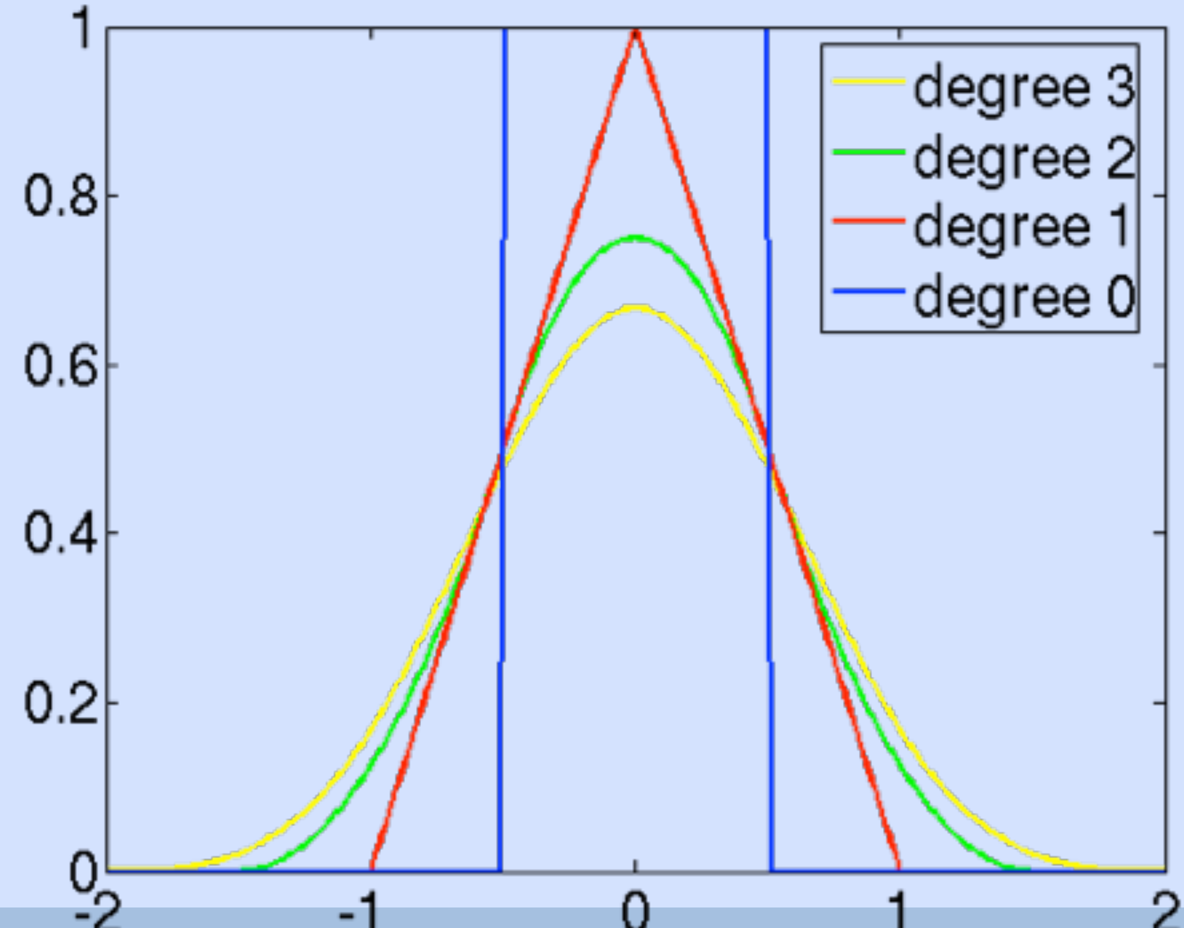


B-Spline Encoding

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

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

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



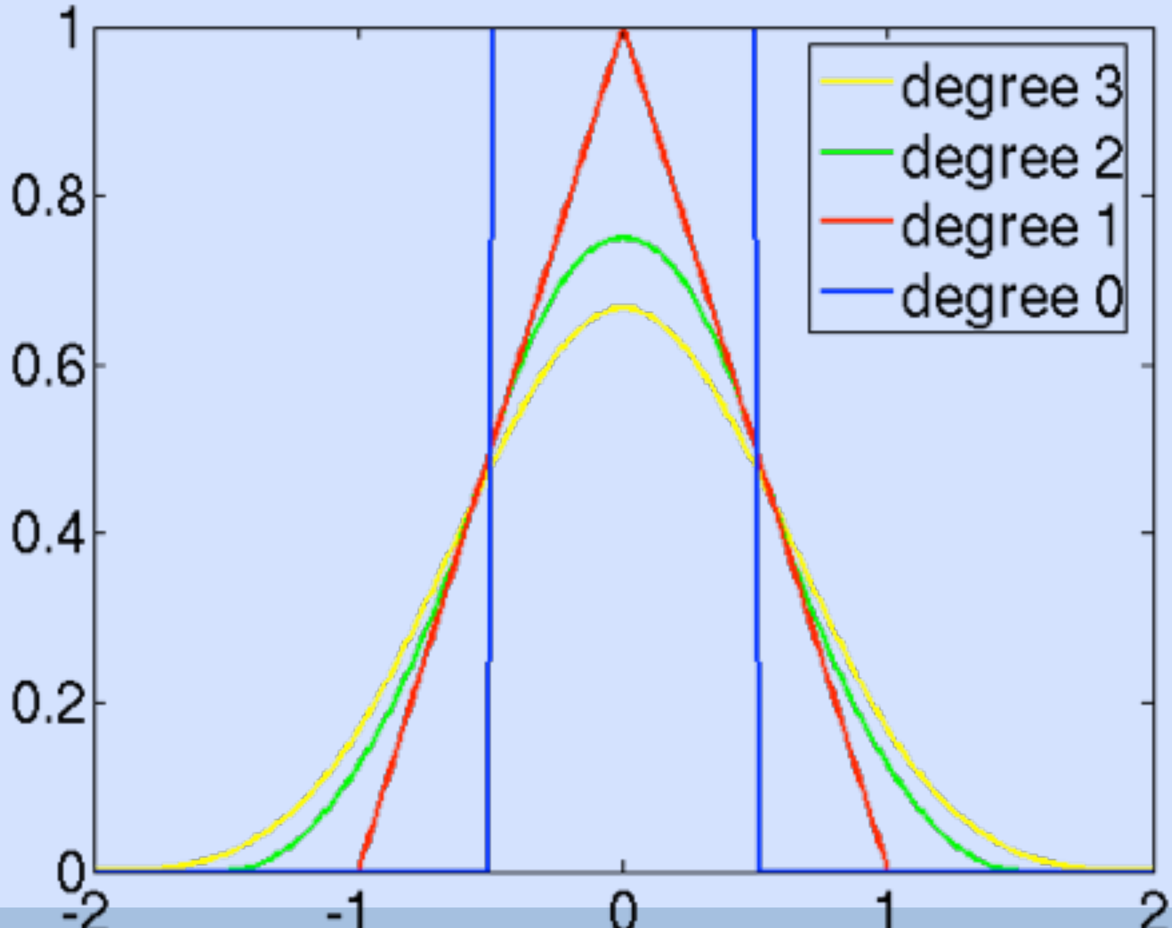
B-Spline Encoding

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

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

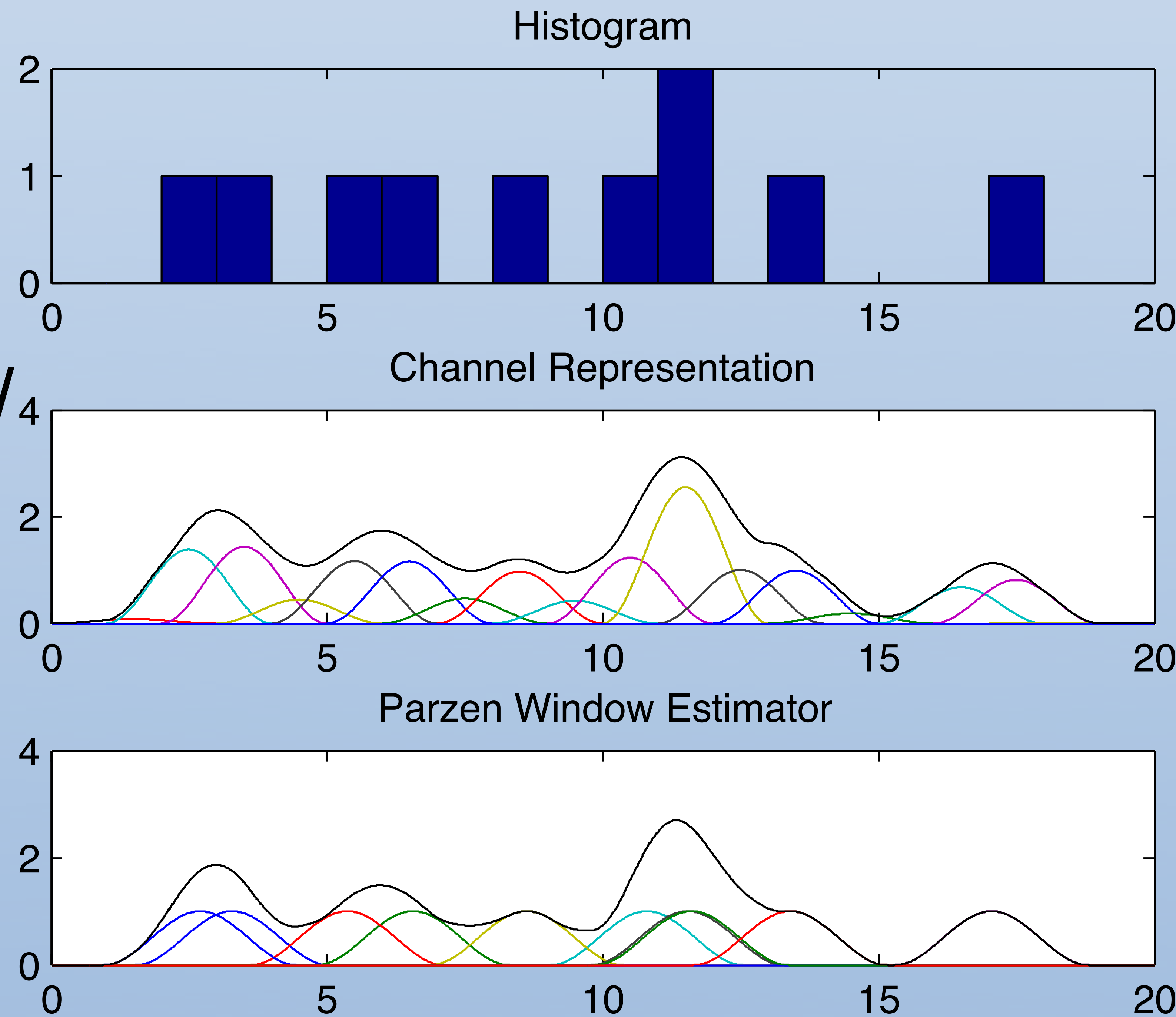
(f is shifted and rescaled such that the channels are at integer positions)

$k = \mathrm{round}(f)$
$x[k\text{-}1] = (f\text{-}k\text{-}0.5)^2/2$
$x[k] = 0.75\text{-}(f\text{-}k)^2$
$x[k\text{+}1] = (k\text{-}f\text{-}0.5)^2/2$
$x[1\ldots k\text{-}2] = x[k\text{+}2\ldots K] = 0$



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:

$$\mathrm{E}\{\tilde{p}(f)\} = \int k(f - f')p(f')\,df' = (k * p)(f)$$

Relation to Channels

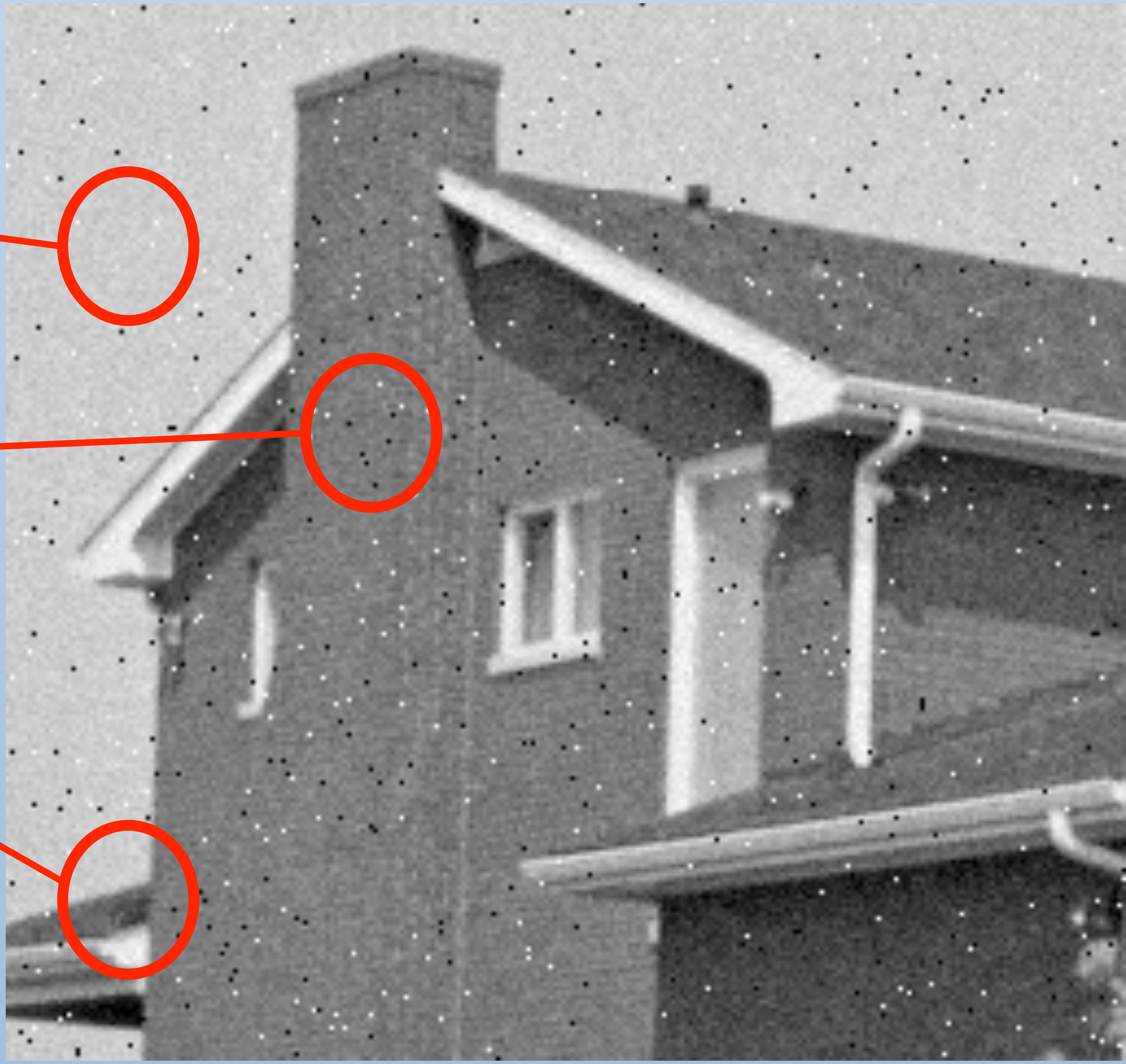
- Adding channel representation of samples = sampled kernel density estimation

$$\mathrm{E}\left\{\frac{1}{N}\sum_{n=1}^{N} u_k\left(f_n\right)\right\} = \mathrm{E}\left\{\tilde{p}\left(f\right)\right\}\Big|_{f=k}$$

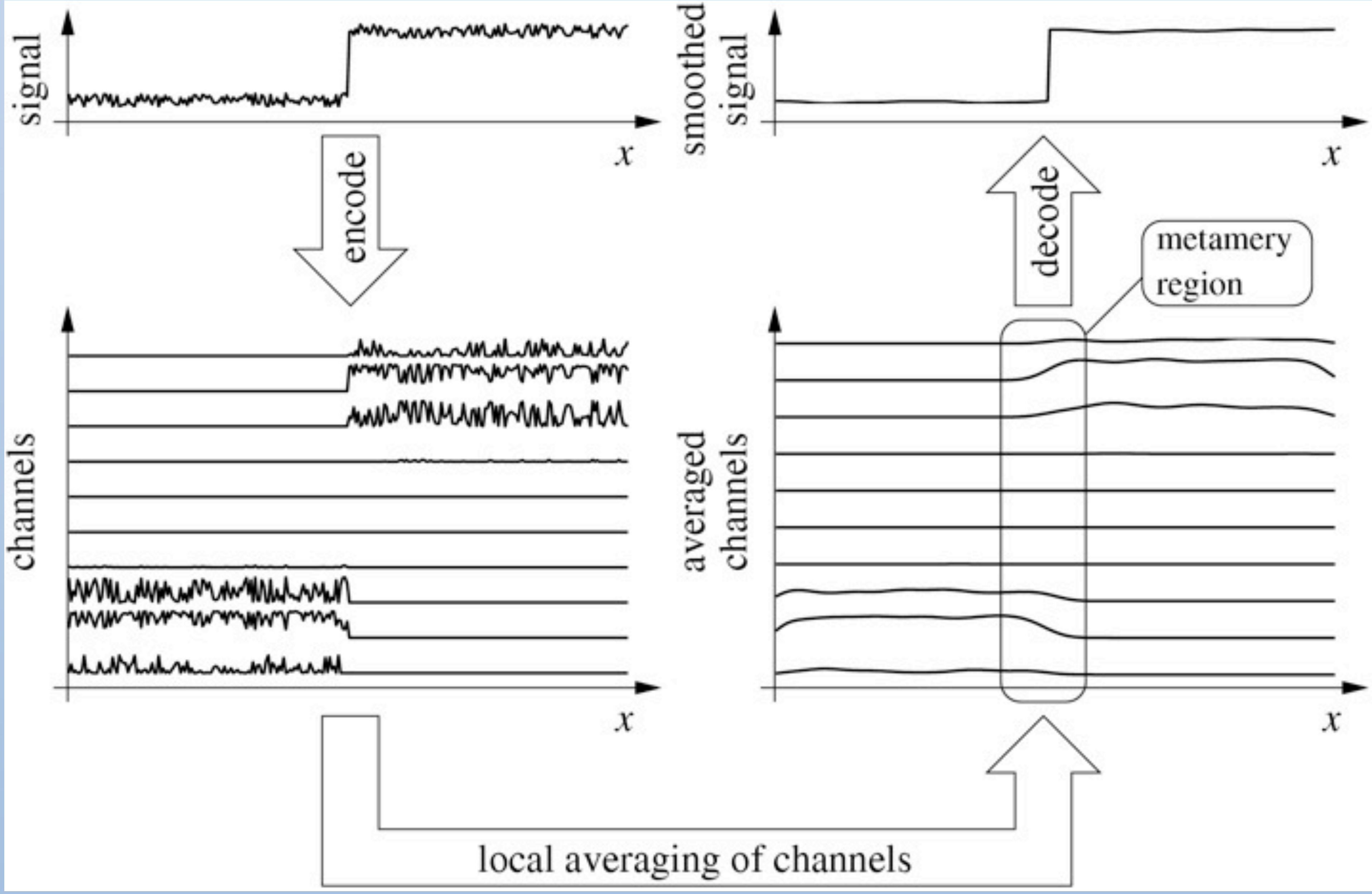
$$= \left(B_2 * p\right)\left(f\right)\Big|_{f=k} \qquad k = 1 \ldots K$$

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

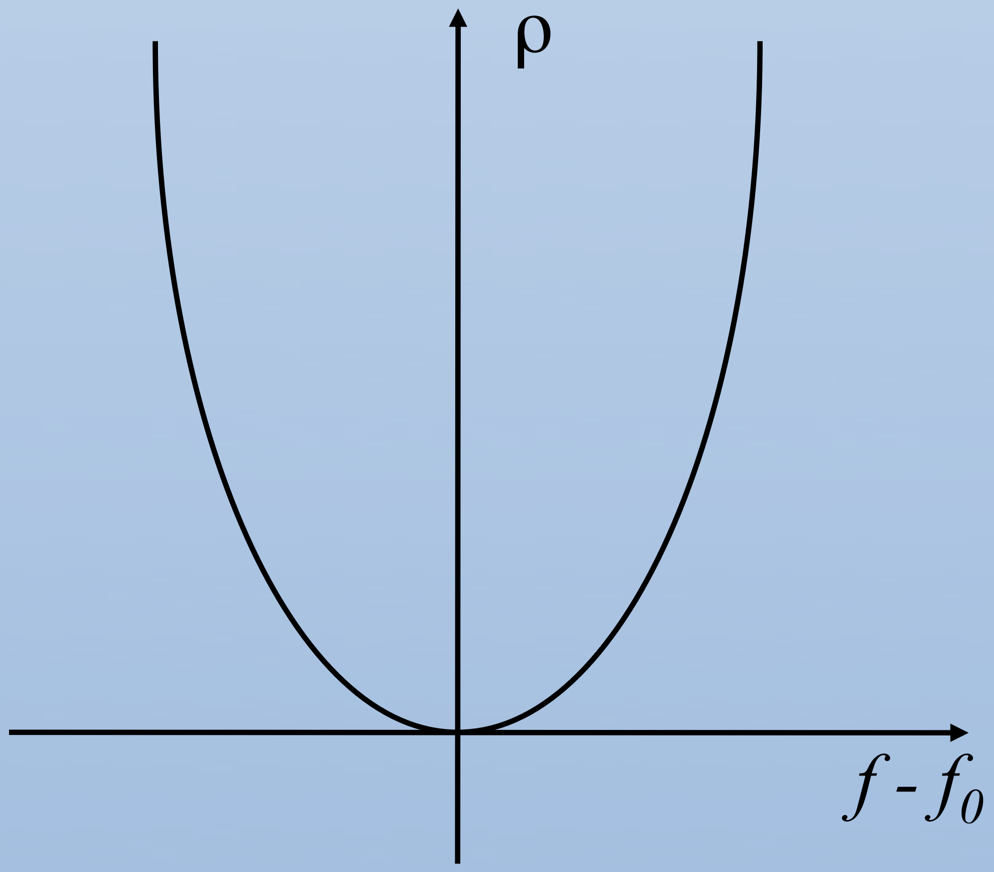
LS & Robust Optimization

- 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

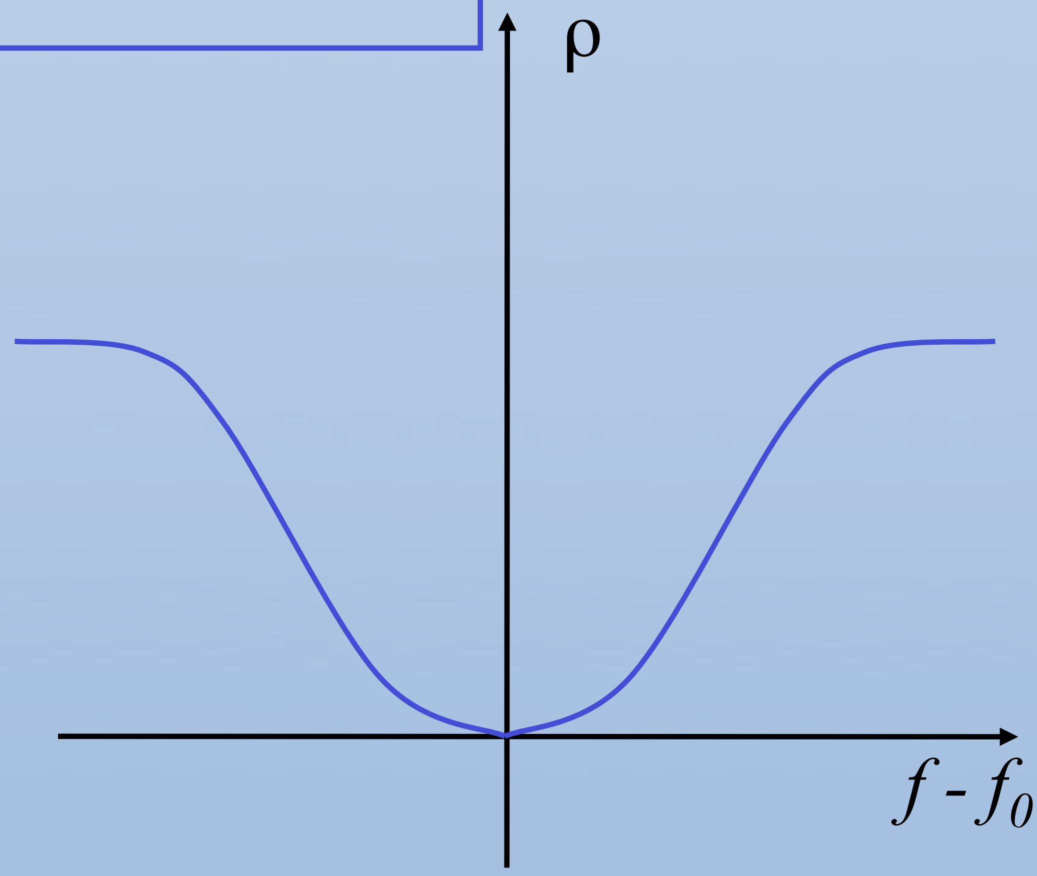
- Minimize error functional:

$$E(f_0) = \int \boxed{\rho(f - f_0)} p(f)\, df$$

$$f_0 = \arg\min E(f_0)$$

Robust error norm

- Idea of robust error norm:
  - –saturated for outliers
  - –quadratic near the origin
- in Bayesian sense



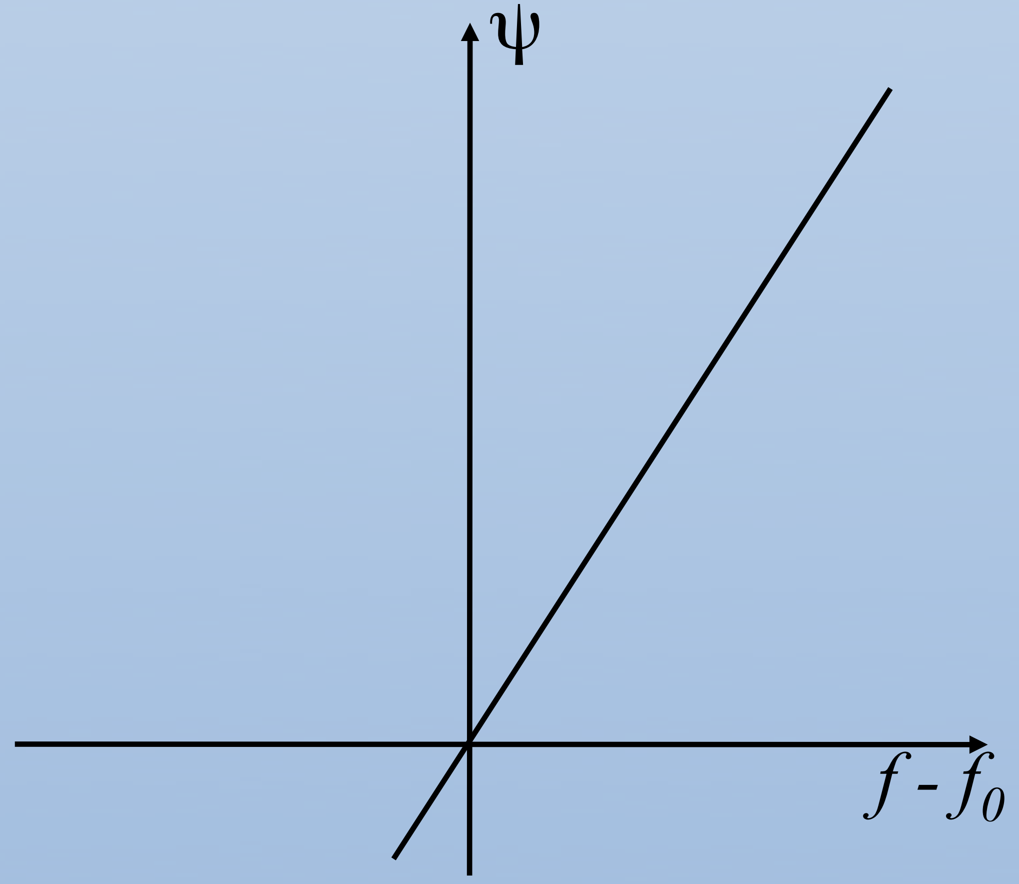
LS & Robust Optimization

- Necessary condition:

$$0 = \int \left(f - f_0\right) \; p\left(f\right) df$$

$$f_0 = \int f \; p\left(f\right) df$$

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



LS & Robust Optimization

- Necessary condition:

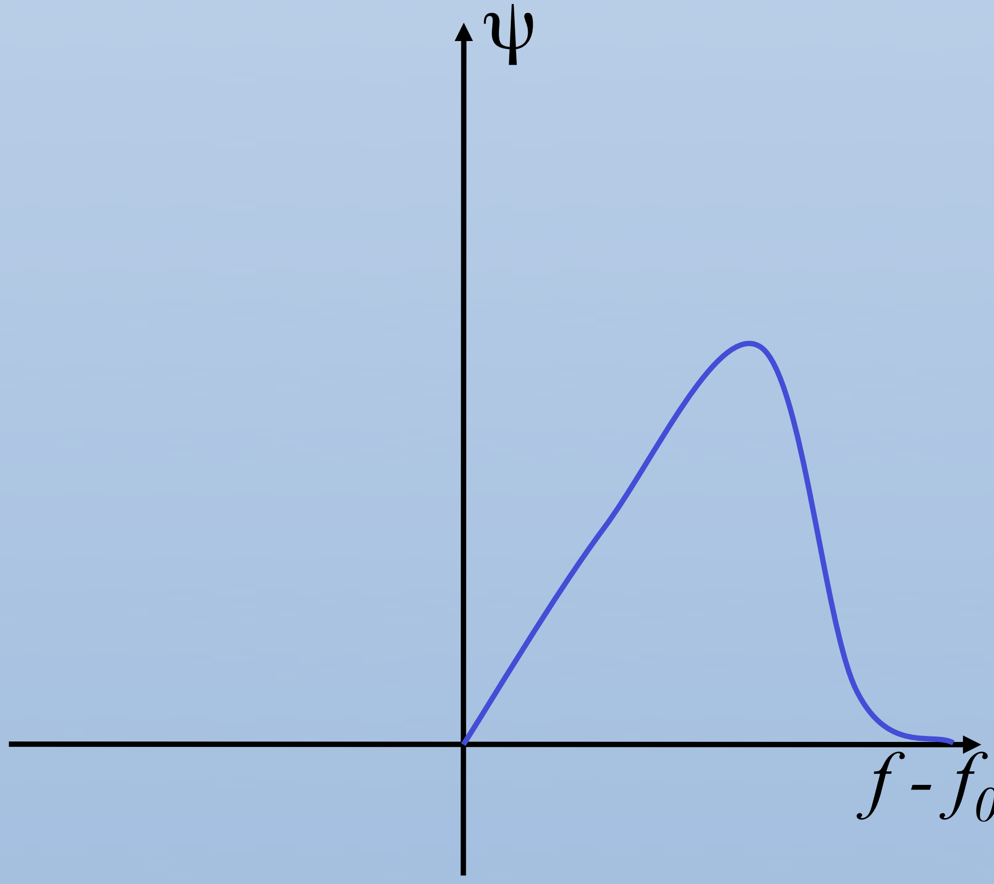
$$0 = \int \psi(f - f_0) p(f)\, df$$

$$\boxed{\psi = \rho'}$$

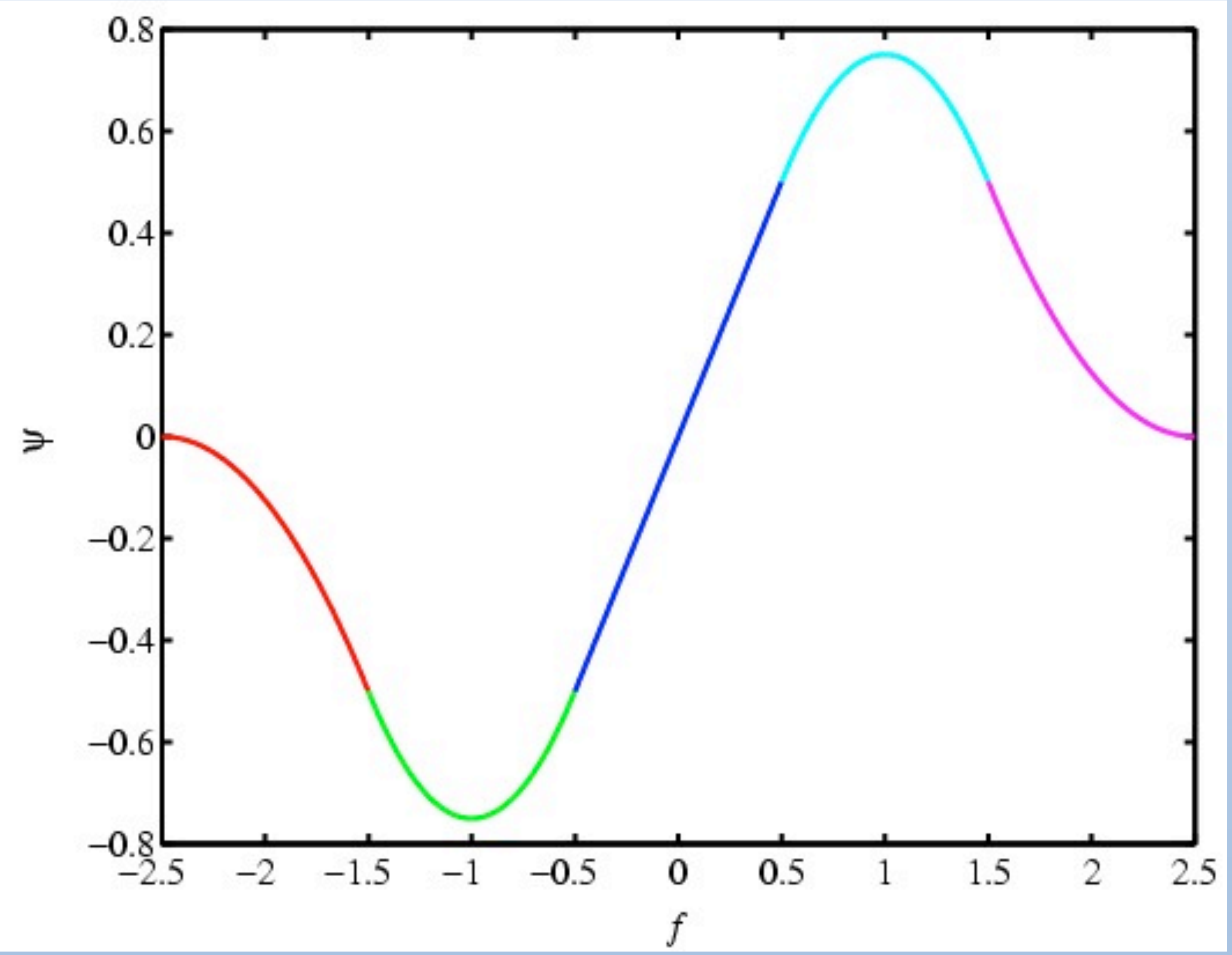
Influence function

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

Efficient methods required!



Influence Function of C.R.



Obtained from linear decoding:

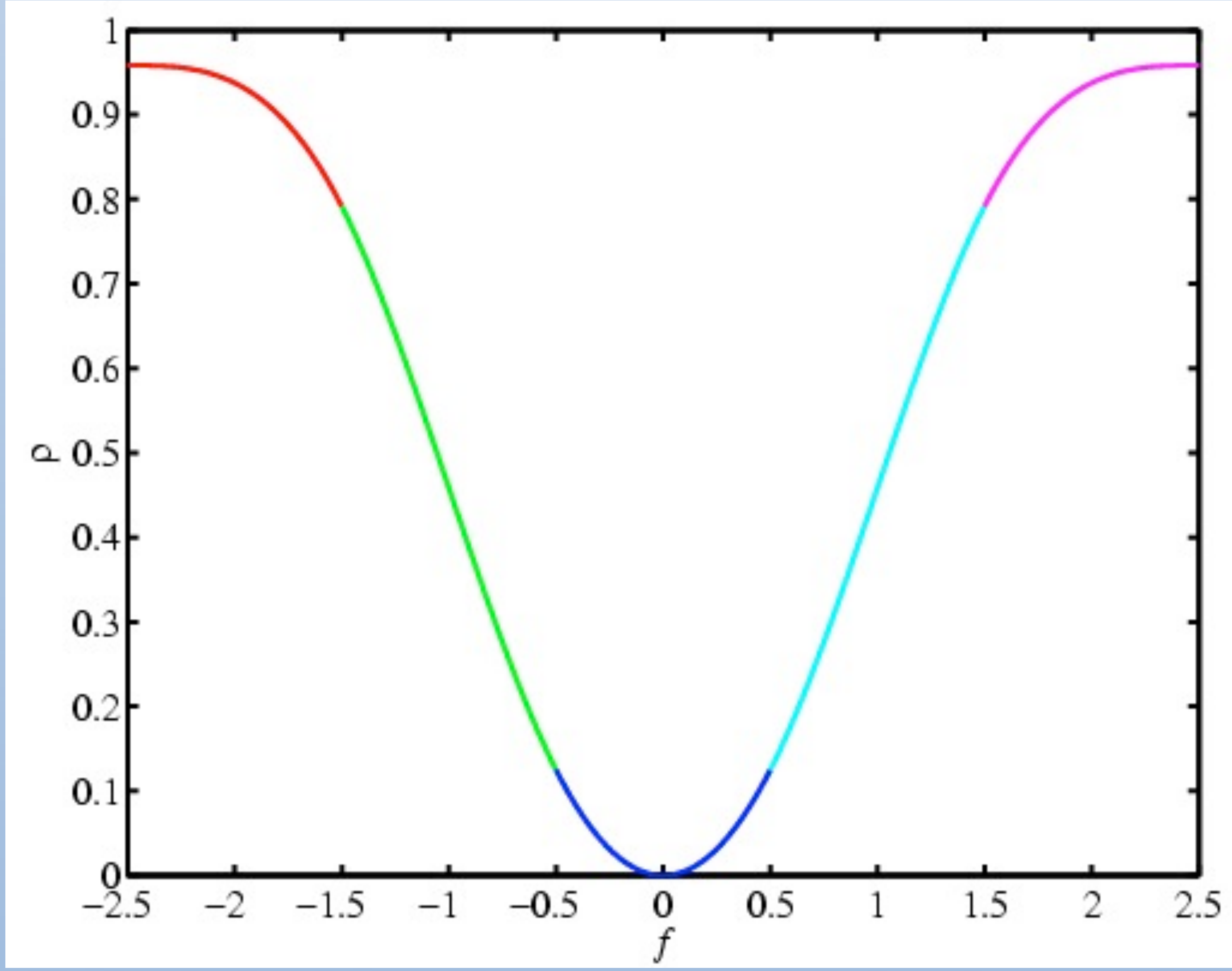
$$\psi(f) = B_2(f-1) - B_2(f+1)$$

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

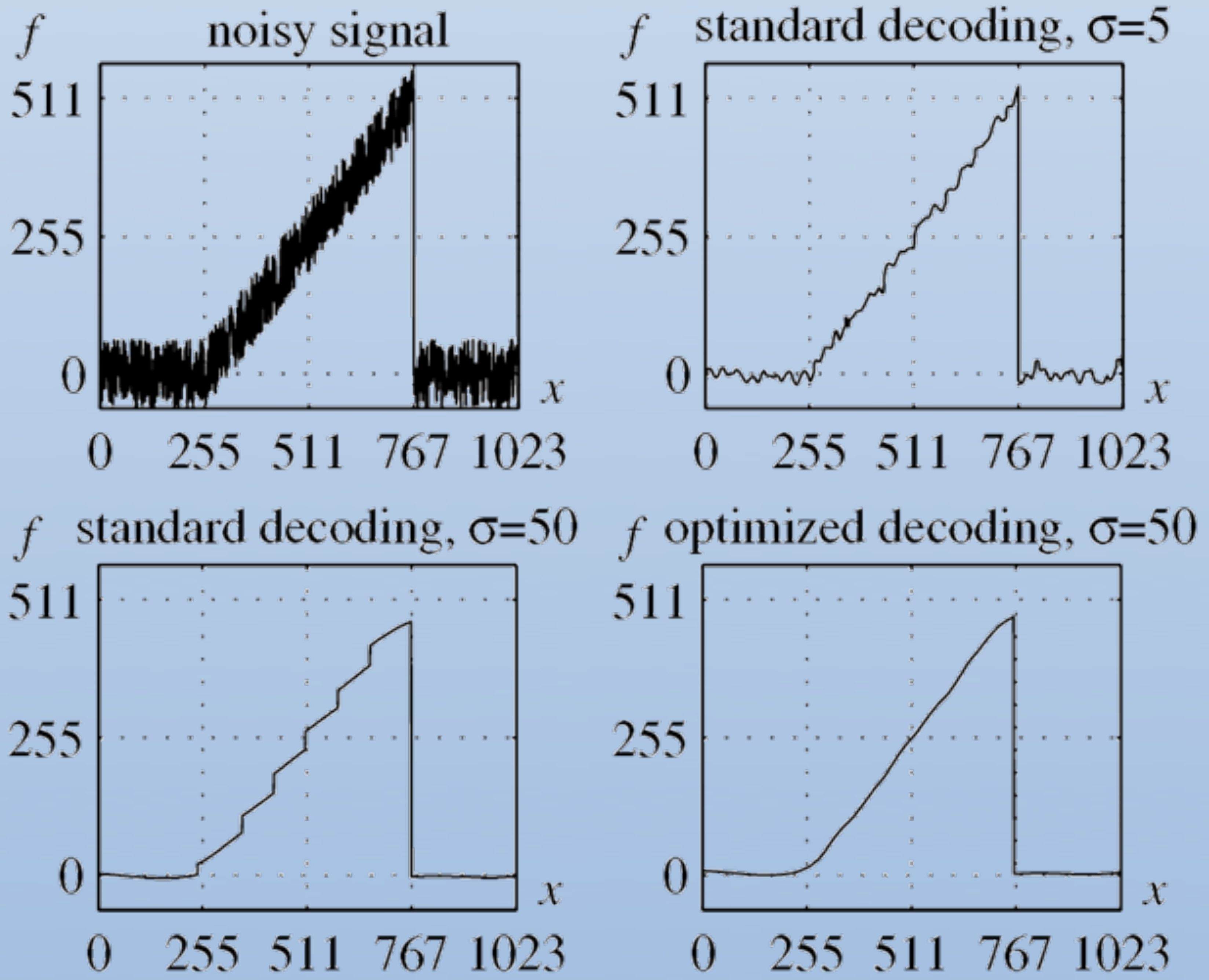
Error Norm of C.R.

Obtained by integrating the influence function:

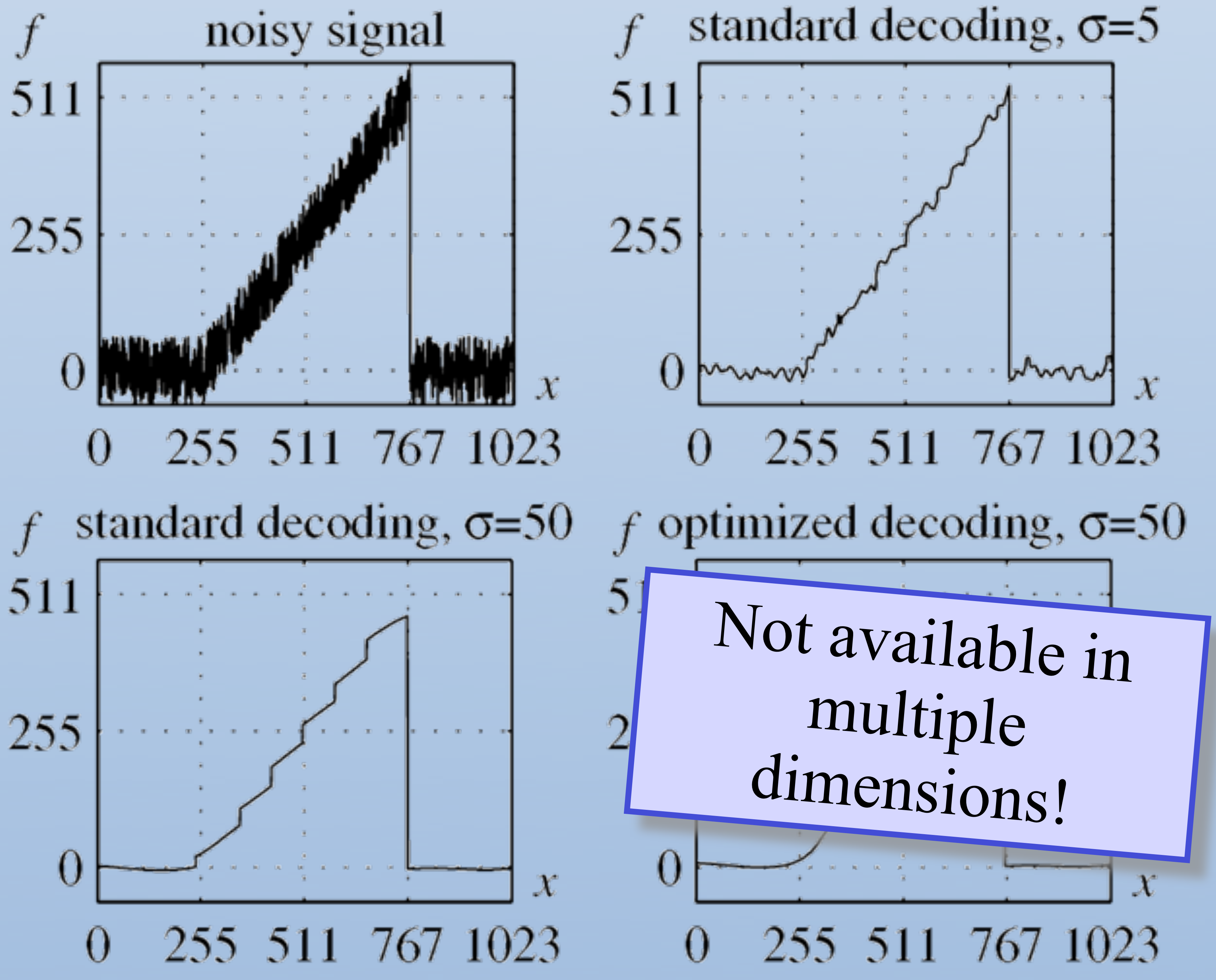
$$\rho(f) = 2B_3\left(\frac{1}{2}\right) - B_3\left(f + \frac{1}{2}\right) - B_3\left(f - \frac{1}{2}\right)$$



Quantization Effect



Quantization Effect



Channel Smoothing

Algorithm 1 Channel smoothing algorithm.

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

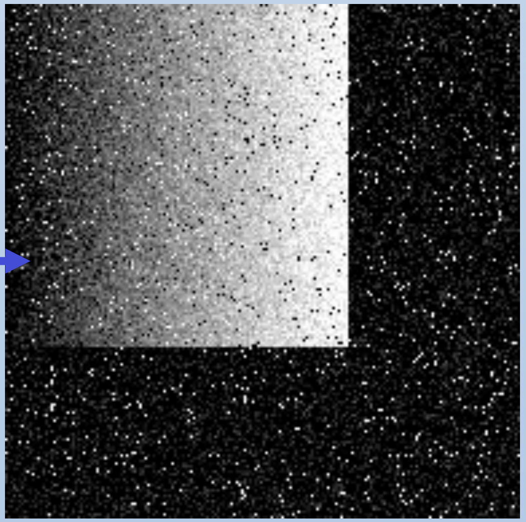
for $n = 1$ to N **do**

$\quad c_n \Leftarrow B_2(f - n)$

$\quad c_n \Leftarrow g_\sigma * c_n$

end for

$[\hat{f}, \hat{\rho}] \Leftarrow \mathrm{decode}(\mathbf{c})$



Channel Smoothing

Algorithm 1 Channel smoothing algorithm.

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

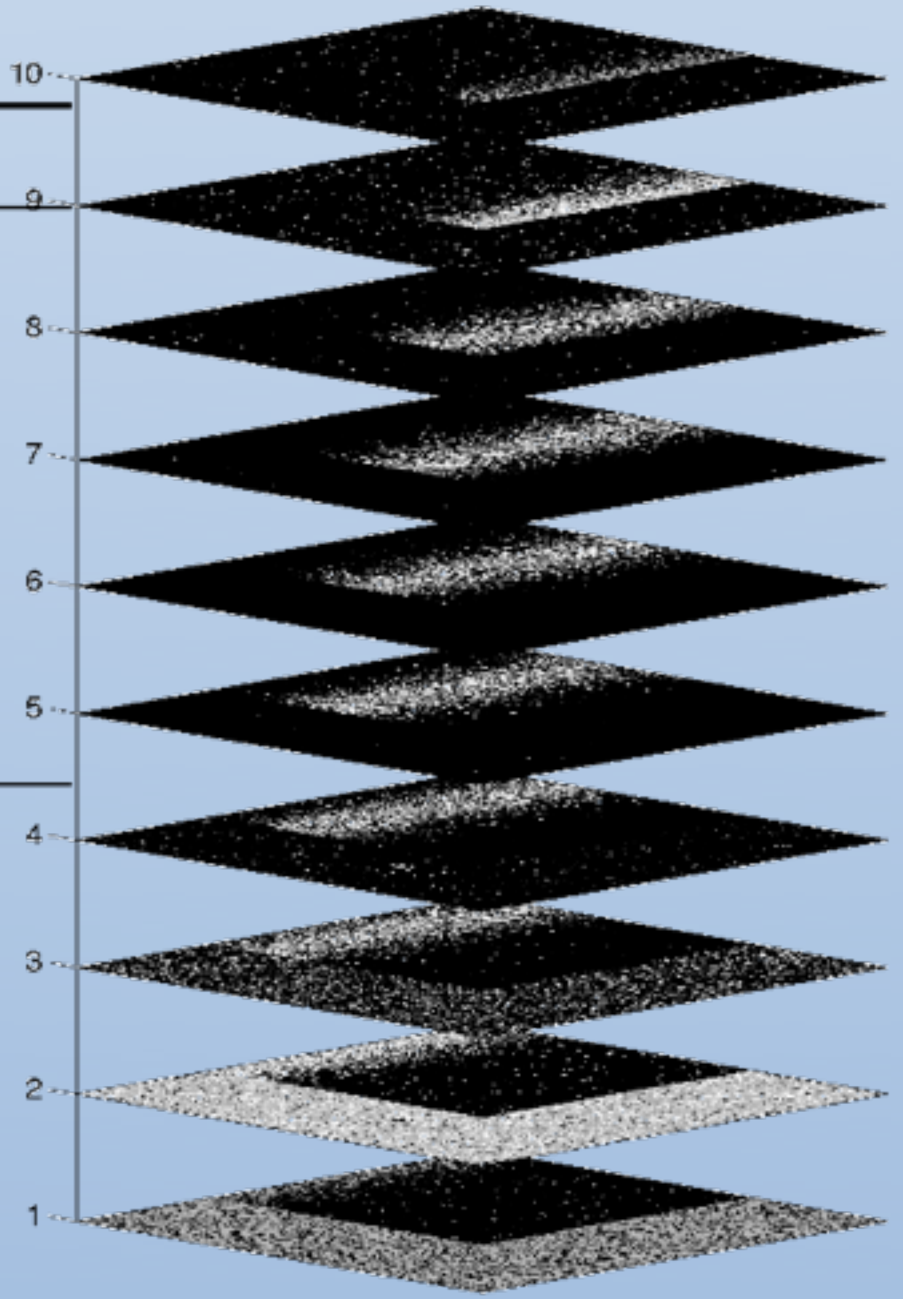
for $n = 1$ to N **do**

$\quad \boxed{c_n \Leftarrow B_2(f - n)}$

$\quad c_n \Leftarrow g_\sigma * c_n$

end for

$[\hat{f}, \hat{\rho}] \Leftarrow \mathrm{decode}(\mathbf{c})$



Channel Smoothing

Algorithm 1 Channel smoothing algorithm.

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

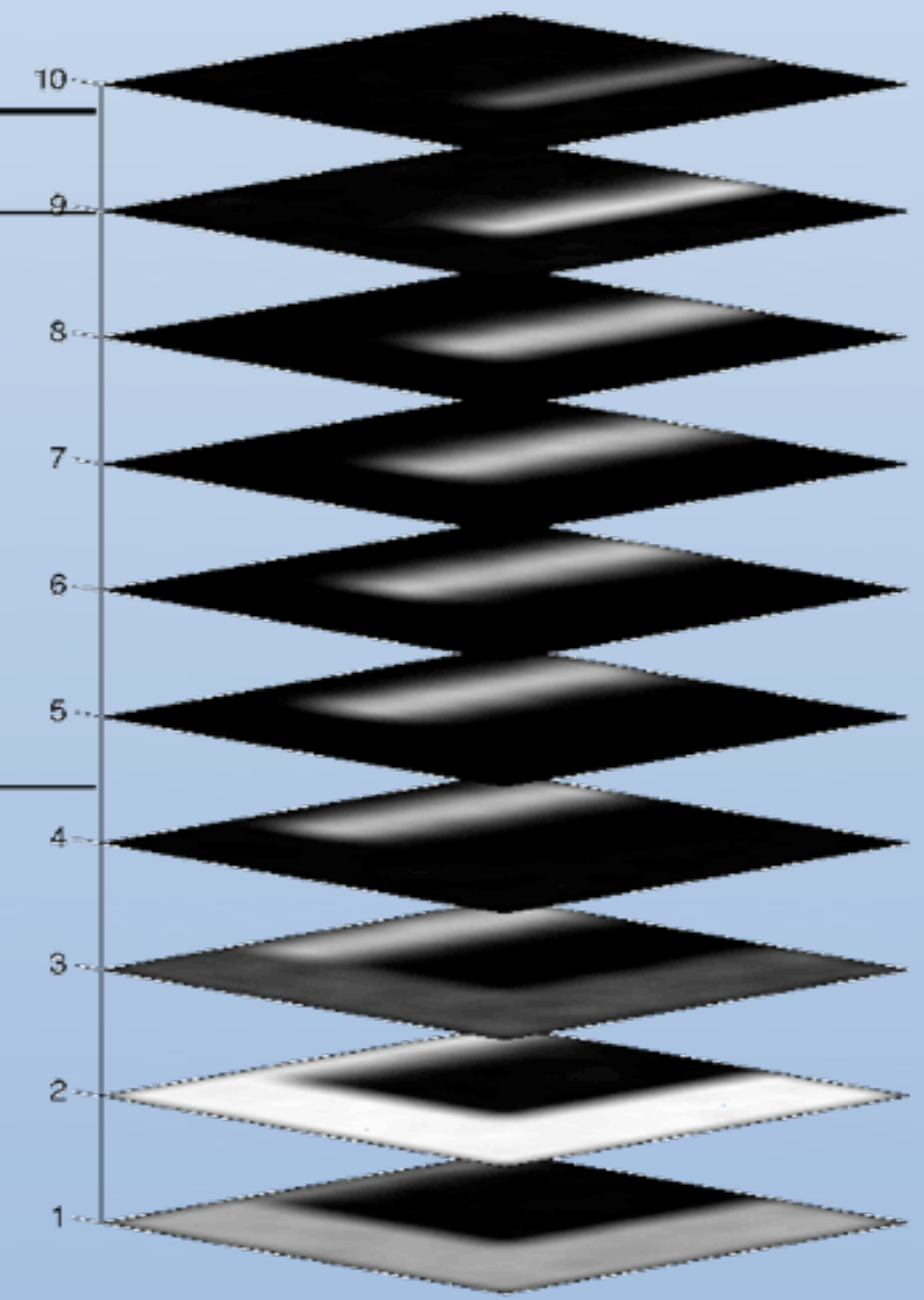
for $n = 1$ to N **do**

  $c_n \Leftarrow B_2(f - n)$

  $\boxed{c_n \Leftarrow g_\sigma * c_n}$

end for

$[\hat{f}, \hat{\rho}] \Leftarrow \text{decode}(\mathbf{c})$



Channel Smoothing

Algorithm 1 Channel smoothing algorithm.

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

for $n = 1$ to N **do**

  $c_n \Leftarrow B_2(f - n)$

  $c_n \Leftarrow g_\sigma * c_n$

end for

$\boxed{[\hat{f}, \hat{\rho}] \Leftarrow \text{decode}(\mathbf{c})}$

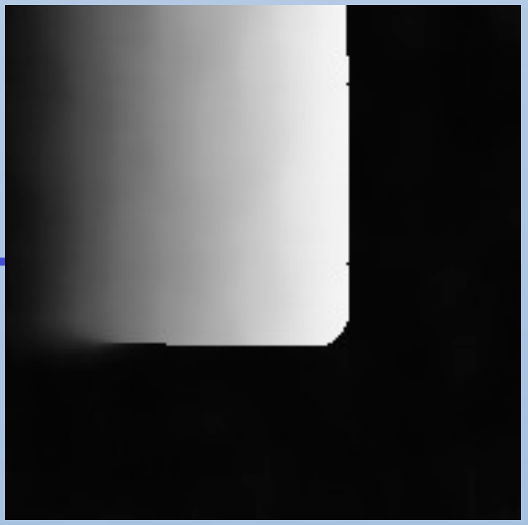
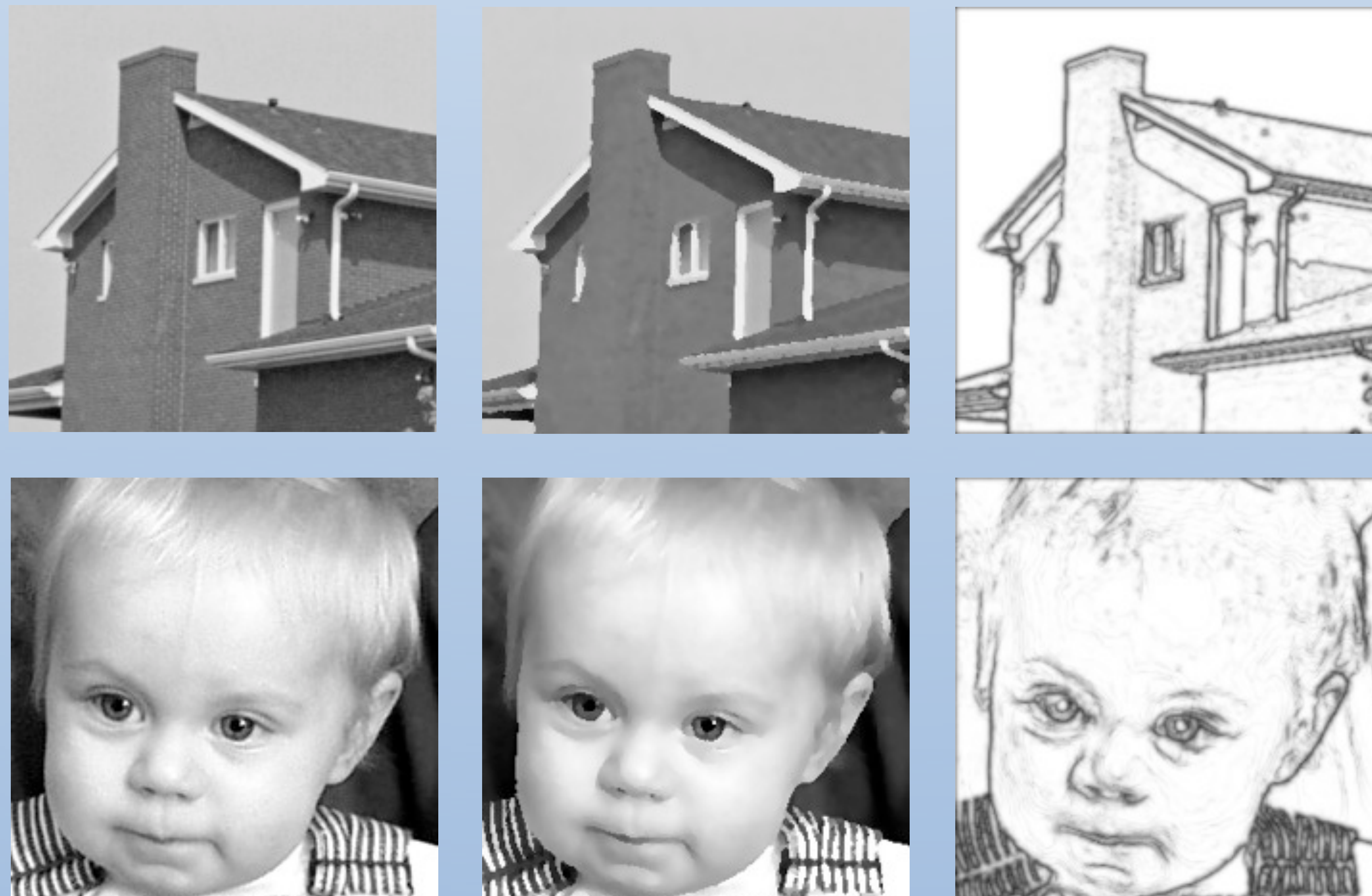
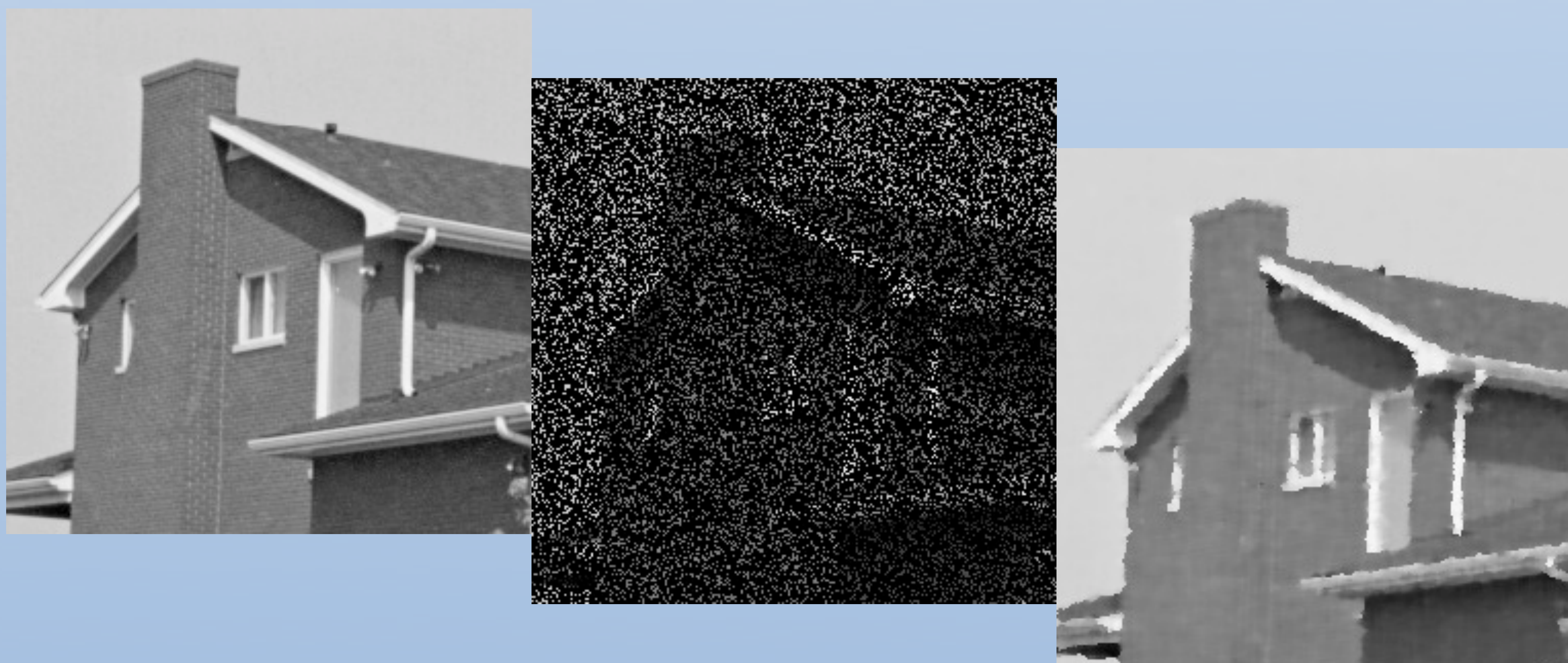


Image Denoising

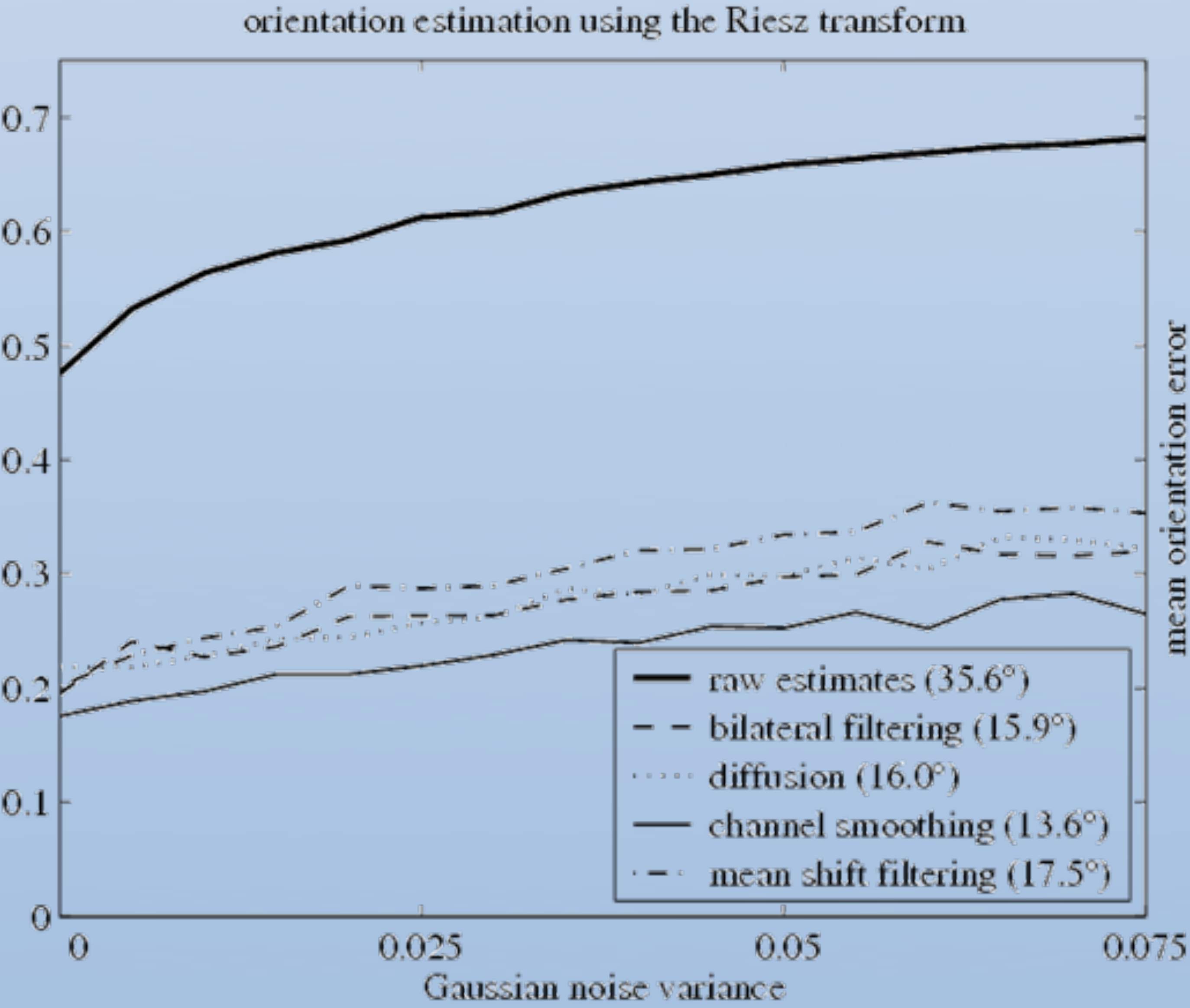


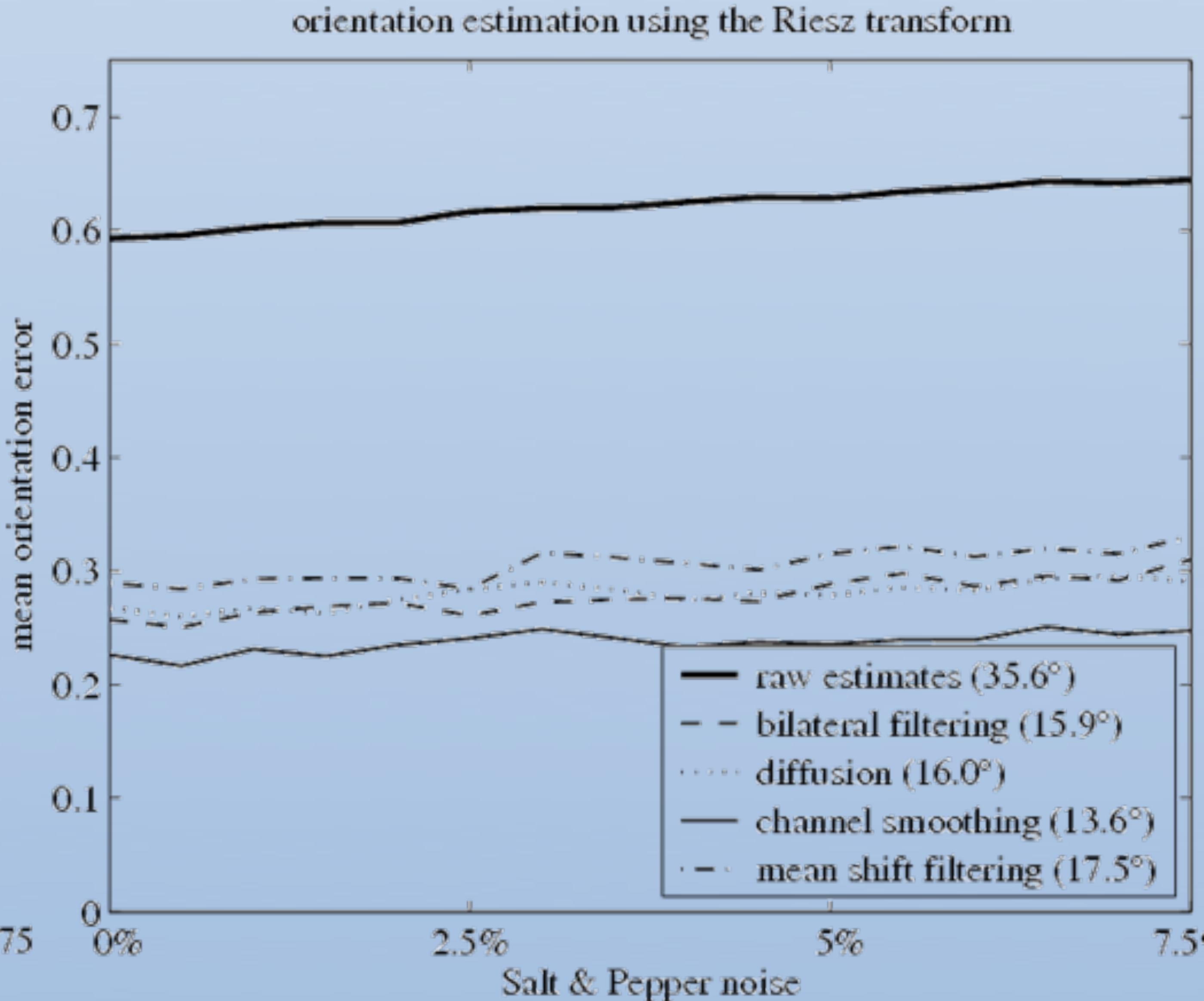
Random Sample

- Real data is incomplete



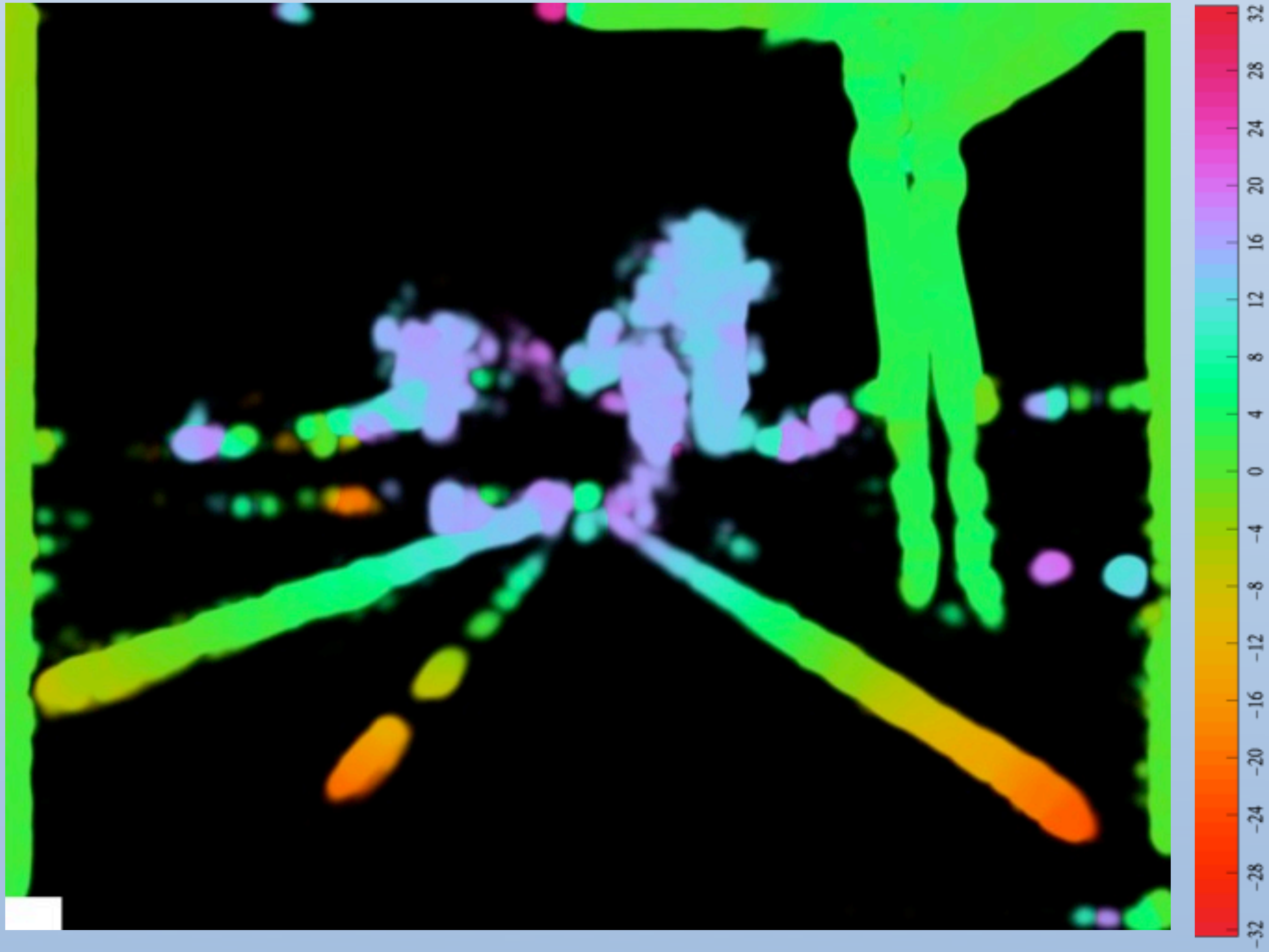
Orientation Estimation





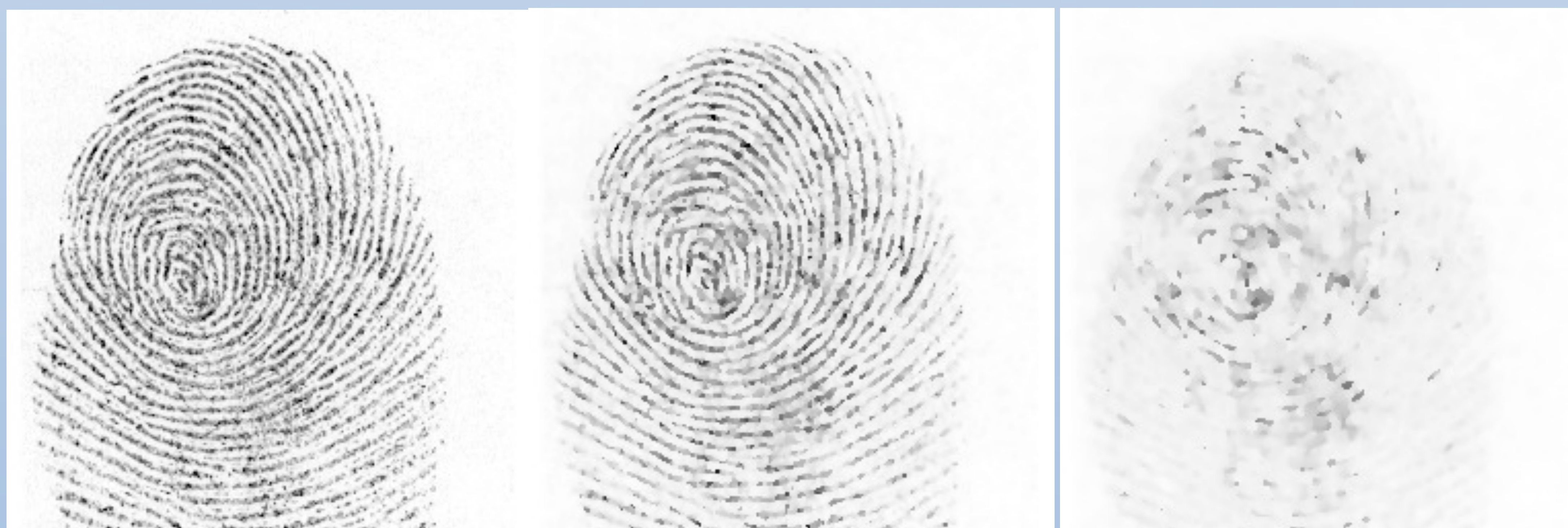
Disparity Estimation

Disparity Estimation

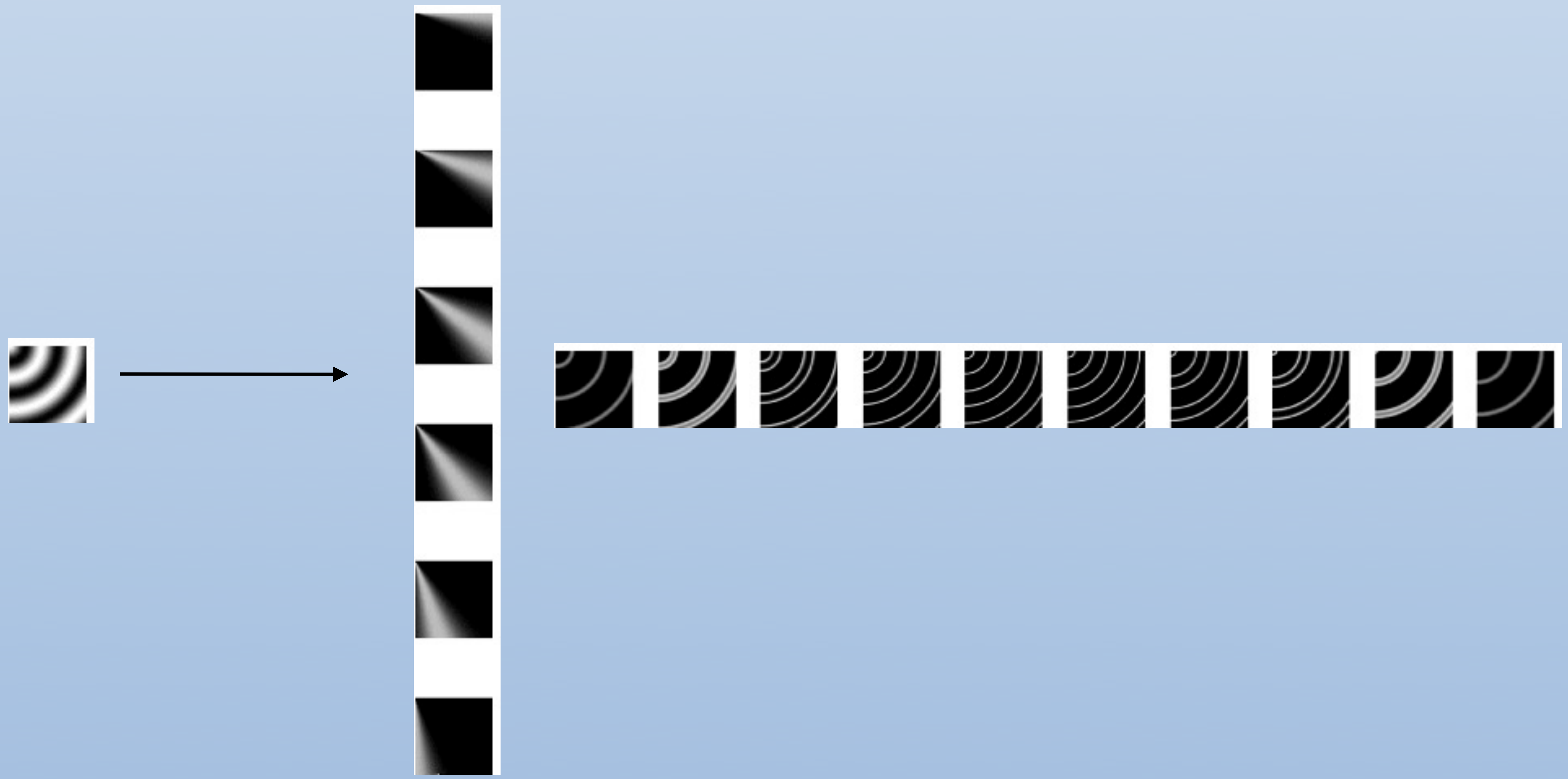


Drawback

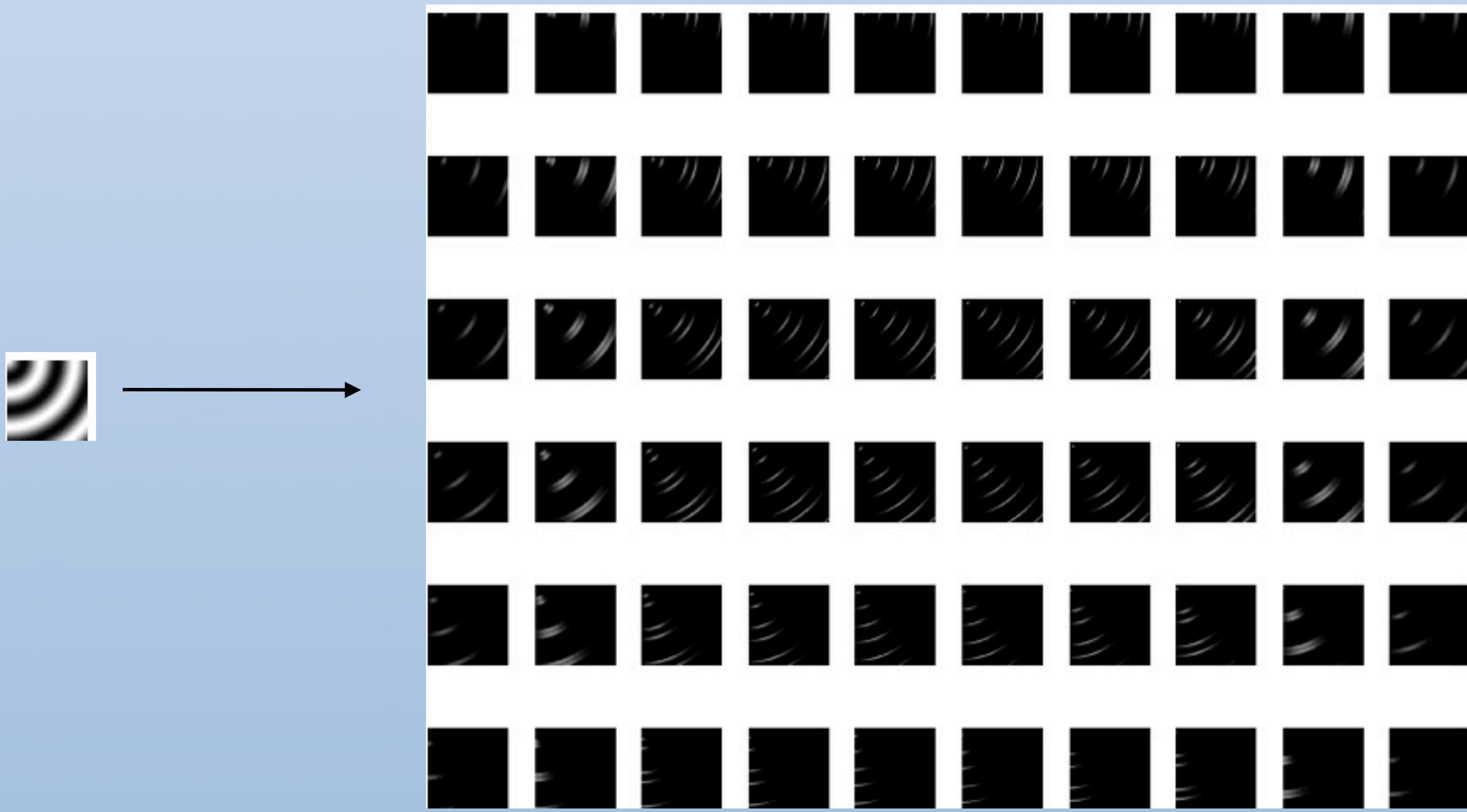
- no coherence enhancing filtering possible



Channel Matrix



Channel Matrix



Experiment

original image

coherence enhancing diffusion

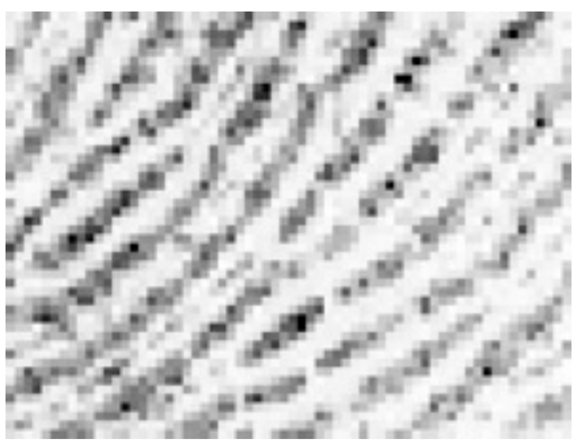
anisotropic channel smoothing

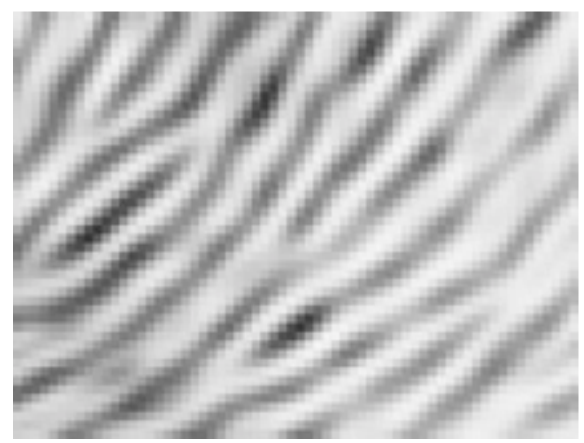
Experiment

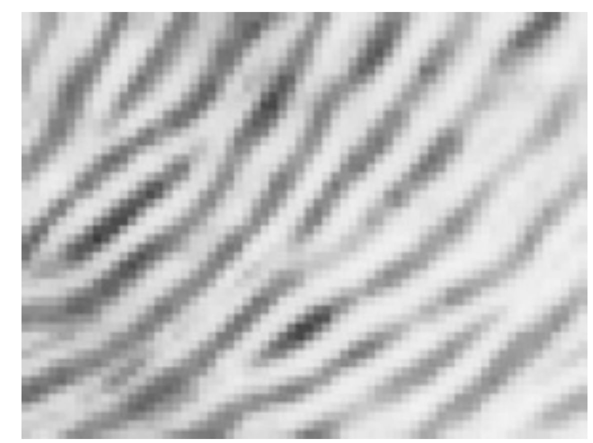
original image

coherence enhancing diffusion

anisotropic channel smoothing



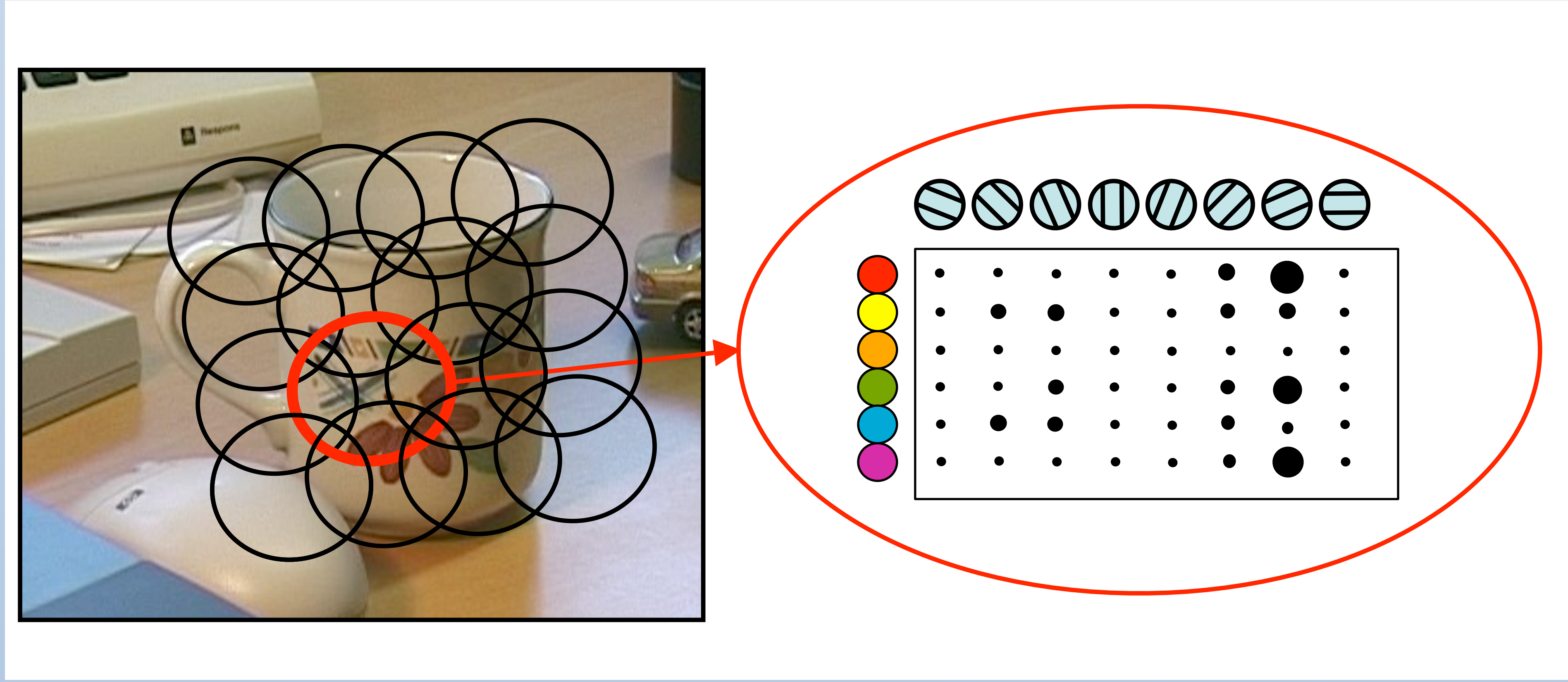




Corner Detection

Motivation CCFM

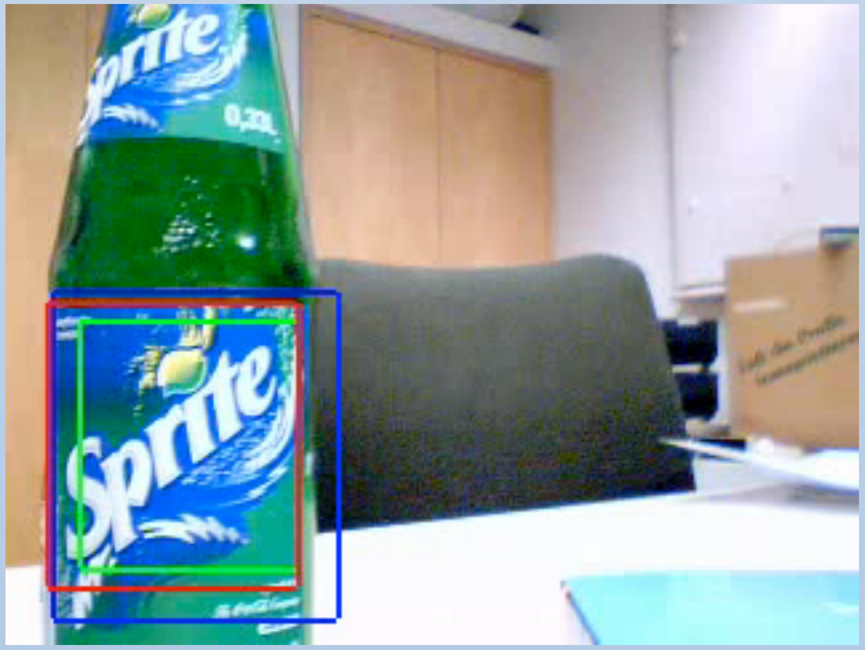
CCFMs



- point-wise encoding

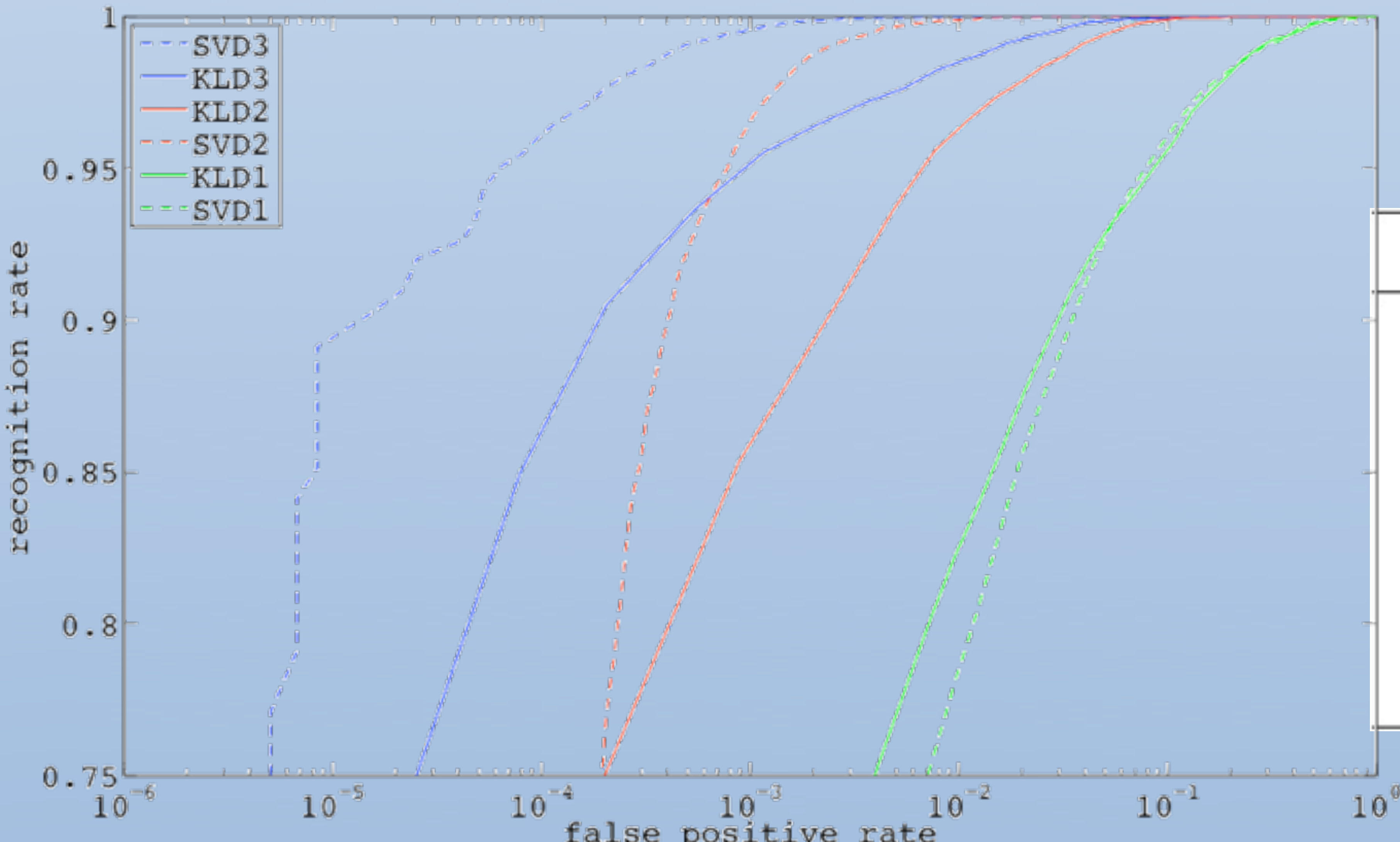
$$c_{l,m,n}(f(x,y)) = k_f(f(x,y)-n)k_x(x-l)k_y(y-m)$$

Object Recognition



COIL-100 Objects

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

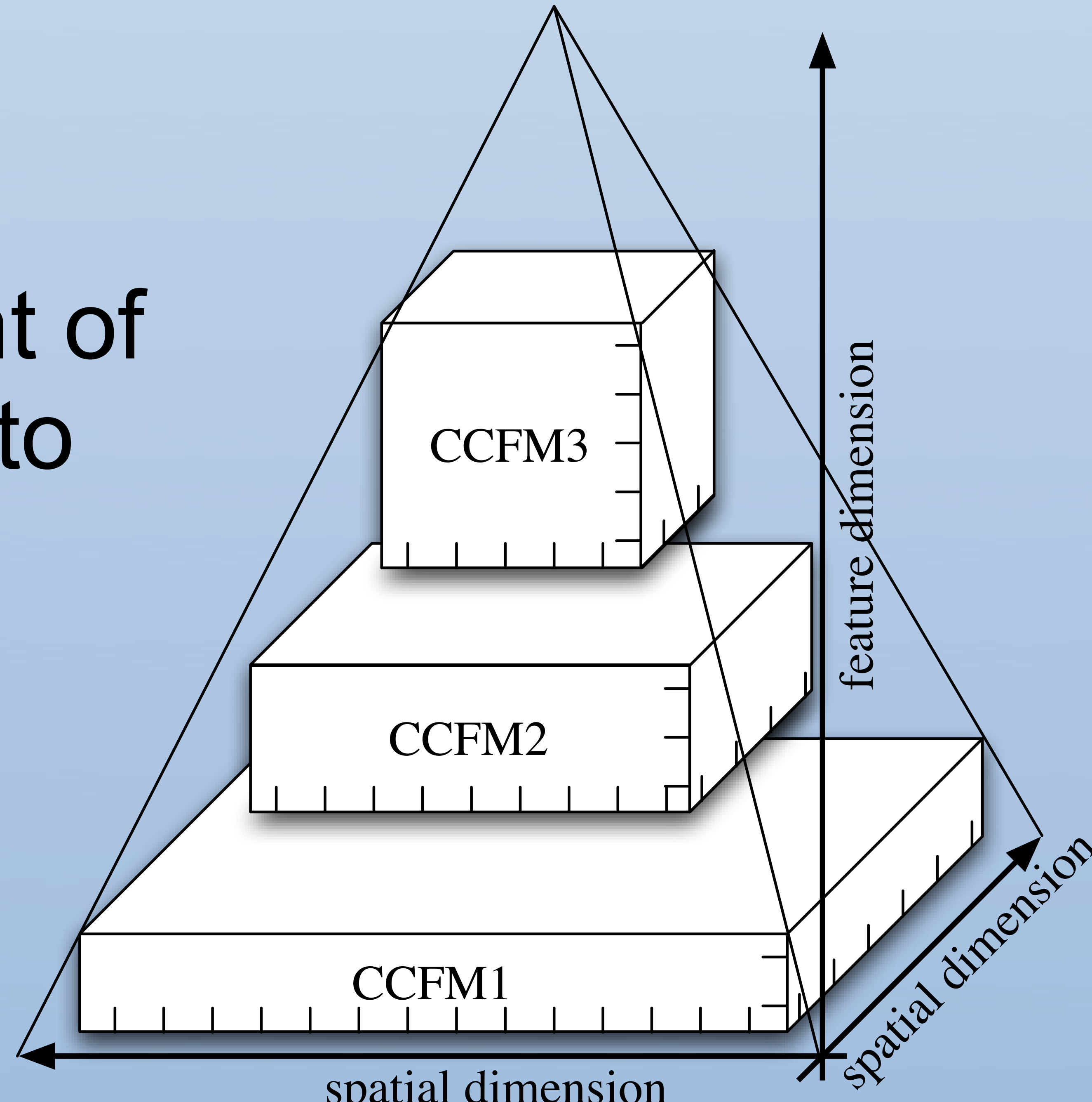


Method	ROC int
KLD, θ	0.9817
SVD, θ	0.9840
KLD, RGB	0.9983
SVD, RGB	0.9998
KLD, hsθ	0.9939
SVD, hsθ	1.0000

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



CCFM Smoothing

Algorithm 7 CCFM smoothing algorithm.

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

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

$\mathbf{C} \Leftarrow \mathrm{CCFM}(x,y,f)$

for all x do

$\quad \mathbf{c}_f \Leftarrow \mathrm{interpolate}(\mathbf{C},\mathbf{x})$

$\quad [\mathbf{f}(\mathbf{x})\ \mathbf{E}(\mathbf{x})] \Leftarrow \mathrm{decode}(\mathbf{c}_f)$

$\quad i(\mathbf{x}) \Leftarrow \arg\max_n E_n(\mathbf{x})$

$\quad [\hat{f}(\mathbf{x})\ \hat{E}(\mathbf{x})] \Leftarrow [f_{i(\mathbf{x})}(\mathbf{x})\ E_{i(\mathbf{x})}(\mathbf{x})]$

end for

CCFM Smoothing

Algorithm 7 CCFM smoothing algorithm.

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