Adaptive Filtering using Channel Representations

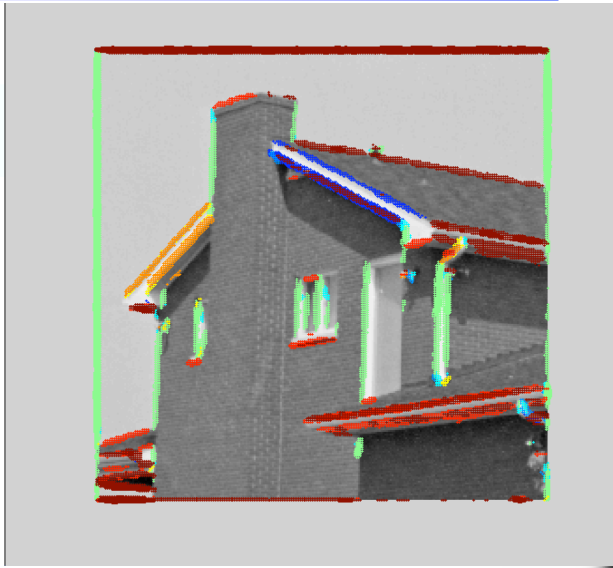
Michael Felsberg

Computer Vision Laboratory
Linköping University, Sweden

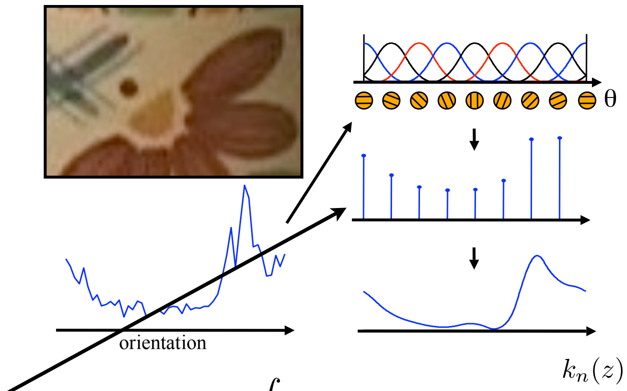
Collaborators

- Gösta Granlund
- Per-Erik Forssen
- Björn Johansson
- Erik Jonsson
- Johan Hedborg
- Martin Berg
- Norbert Krüger, et al. (Stirling)
- Hanno Scharr (Intel)
- Remco Duits, Bart ter Haar Romeny (Eindhoven)
- Didier Stricker, Alain Pagani (Fraunhofer Darmst.)

What are channels ?



Channel Representation



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

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

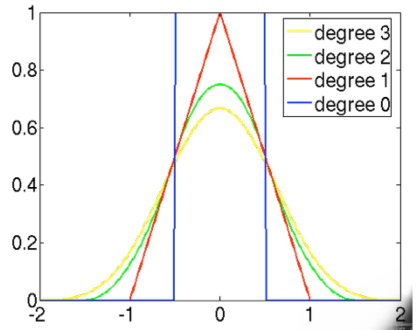
$$\delta_f(z) = \delta(z-f)$$

B-Spline Encoding

- The value of the n-th channel is obtained by
  $c_n(f) = B_2(f - n) \qquad n = 1 \ldots N$
  (f is shifted and rescaled such that the channels are at integer positions)

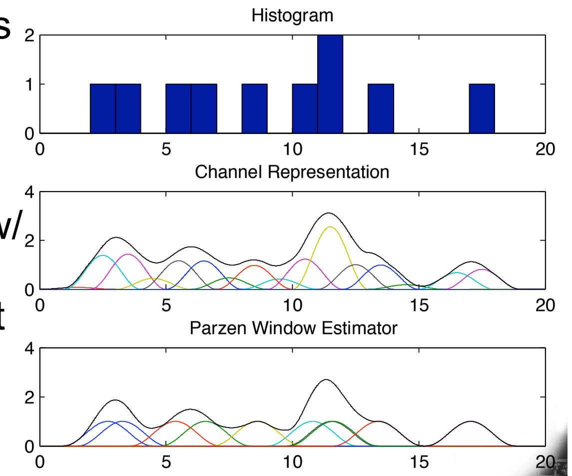
Algorithm 1 Channel encoding algorithm.

Require: $f \in [1.5; N - 0.5]$
1: $\mathbf{c} \Leftarrow 0$
2: **for all** samples f **do**
3: $i \Leftarrow \mathrm{round}(f)$
4: $g \Leftarrow f - i$
5: $c_{i-1} \Leftarrow c_{i-1} + (g - 1/2)^2/2$
6: $c_i \Leftarrow c_i + 3/4 - g^2$
7: $c_{i+1} \Leftarrow c_{i+1} + (g + 1/2)^2/2$
8: **end for**



Channels are ...

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



Relation to Density Estimation

- Adding channel representation of samples = sampled kernel density estimation

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

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

First Application

- Line/edge detection by approximated entropy

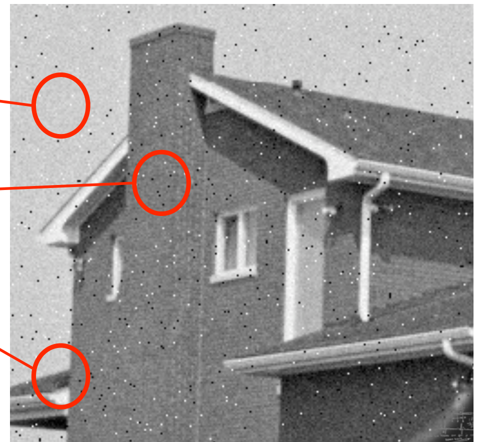
$$\tilde{H}(\mathbf{x}) = -\sum_{n=1, c_n(\mathbf{x}) \neq 0}^{N} c_n(\mathbf{x}) \log c_n(\mathbf{x})$$

$$E\{\tilde{H}\} = H_{B_2 * p}.$$

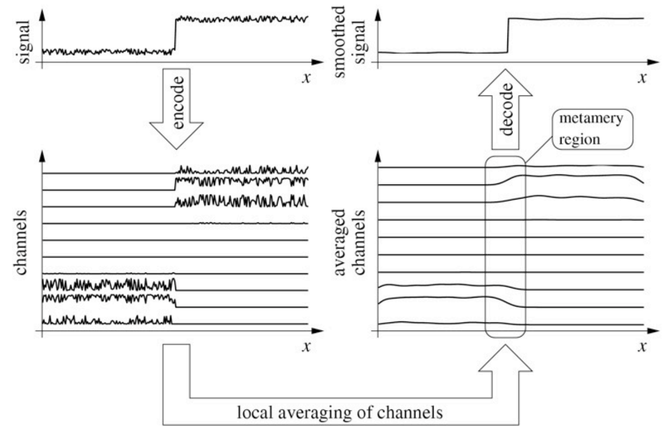
Edge-Energy

Problem: Image Denoising

- Real data is noisy and discontinuous
  - Inlier noise
  - Outlier noise
  - Image discontinuities



Channel Smoothing



Decoding

- Choose n_0:
  - –Largest response of 3-box filter

$$M_n = c_{n-1} + c_n + c_{n+1} \qquad n_0 = \arg\max M_n$$

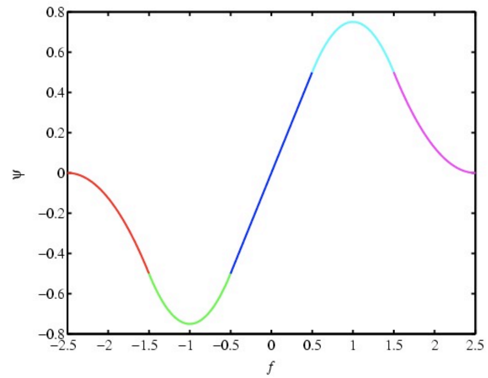
  - –Additional: local maximum
- Normalized convolution of the channel vector

$$\hat{f} = \frac{c_{n_0+1} - c_{n_0-1}}{M_{n_0}} + n_0$$

Influence Function of C.R.

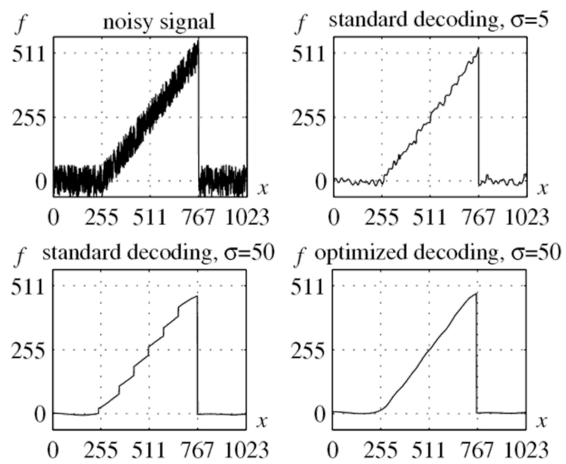
Obtained from linear decoding:

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



$$\hat{f} = \frac{c_{n_0+1} - c_{n_0-1}}{M_{n_0}} + n_0$$

Quantization Effect



Quadratic Decoding I

- Idea: detect local maximum of B-spline interpolated channel vector
  - Step 1: recursive filtering to obtain interpolation coefficients:

$$c_n^+ = c_n + hc_{n-1}^+ , \qquad c_1^+ = c_1$$

$$c_n^- = h(c_{n+1}^- - c_n^+), \quad c_N^- = \frac{h}{h^2-1}c_N^+$$

$$c_n' = 8c_n^- \quad . \qquad h = 2\sqrt{2} - 3$$

Quadratic Decoding II

- Step 2: detect zeros

$$0 = \lambda\beta_n^2 + \mu\beta_n + \nu \qquad \text{with}$$

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

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

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

Quadratic Decoding II

–Step 2: detect zeros

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

Quadratic Decoding III

–Step 3: compute error

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

–Step 4: sort $n + \beta_n$ according to their error ($n + \beta_n$ with least error is most probable)

–Step 5: shift and rescale to original interval

Quadratic Decoding III

–Step 3: compute error

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

Rule of thumb:

- Few channels (e.g. 8): quadratic decoding
- Many channels (e.g. 32): linear decoding

–Step 4: sort $n + \beta_n$ according to their error ($n + \beta_n$ with least error is most probable)

–Step 5: shift and rescale to original interval

Algorithm 2 Virtual shift decoding algorithm.

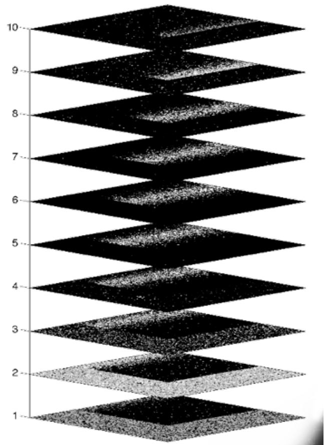
Require: $\mathbf{c}$ is non-negative and normalized

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**
    $c_n \Leftarrow c_n + h c_{n-1}$
  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
$\boldsymbol{\lambda} \Leftarrow \mathrm{conv}(\mathbf{c}, [-\frac{1}{2}\ 1\ 0\ -1\ \frac{1}{2}])$
$\boldsymbol{\mu} \Leftarrow \mathrm{conv}(\mathbf{c}, [-\frac{1}{2}\ 0\ 1\ 0\ -\frac{1}{2}])$
$\boldsymbol{\nu} \Leftarrow \mathrm{conv}(\mathbf{c}, [-\frac{1}{8}\ -\frac{3}{4}\ 0\ \frac{3}{4}\ \frac{1}{8}])$
$\boldsymbol{\beta} \Leftarrow (-\boldsymbol{\mu}/2 + \sqrt{\boldsymbol{\mu}.^2/4 - \boldsymbol{\nu}.\boldsymbol{\lambda}})./\boldsymbol{\lambda}$
$\boldsymbol{\gamma} \Leftarrow \mathrm{conv}(\mathbf{c}, [\frac{1}{48}\ \frac{1}{2}\ \frac{23}{24}\ \frac{1}{2}\ \frac{1}{48}])$
$\mathbf{f} \Leftarrow \boldsymbol{\beta} + [1\ 2 \ldots N]$
$\mathbf{E} \Leftarrow \frac{23}{24} + (-1 < 2\boldsymbol{\beta} < 1).(\boldsymbol{\beta}.\boldsymbol{\nu} + \boldsymbol{\beta}.^2.\boldsymbol{\mu}/2 + \boldsymbol{\beta}.^3.\boldsymbol{\lambda}/3 - \boldsymbol{\gamma})$

Channel Smoothing

Algorithm 3 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**
  $c_n \Leftarrow \text{conv2}(c_n, g_\sigma)$
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



Channel Smoothing

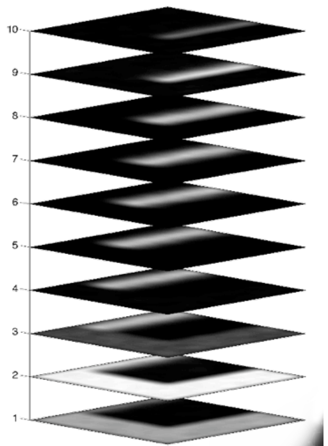
Algorithm 3 Channel smoothing algorithm.

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**
  $c_n \Leftarrow \text{conv2}(c_n, g_\sigma)$
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

Channel Smoothing

Algorithm 3 Channel smoothing algorithm.

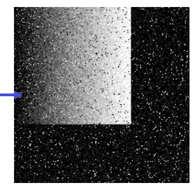
Require: $f \in [1.5; N-0.5]$
for all x do
  $\mathbf{c}(\mathbf{x}) \Leftarrow \text{encode}(f(\mathbf{x}))$
end for
for $n = 1$ to N **do**
  $c_n \Leftarrow \text{conv2}(c_n, g_\sigma)$
end for
for all x do
  $[\mathbf{f}(\mathbf{x})\ \mathbf{E}(\mathbf{x})] \Leftarrow \text{decode}(\mathbf{c}(\mathbf{x}))$
  $i(\mathbf{x}) \Leftarrow \arg\max_n E_n(\mathbf{x})$
  $[\hat{f}(\mathbf{x})\ \hat{E}(\mathbf{x})] \Leftarrow [f_{i(\mathbf{x})}(\mathbf{x})\ E_{i(\mathbf{x})}(\mathbf{x})]$
end for



Channel Smoothing

Algorithm 3 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**
  $c_n \Leftarrow \text{conv2}(c_n, g_\sigma)$
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



Computer Vision Laboratory

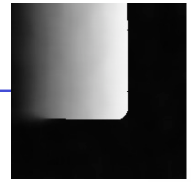
Image Denoising

Computer Vision Laboratory

Channel Smoothing

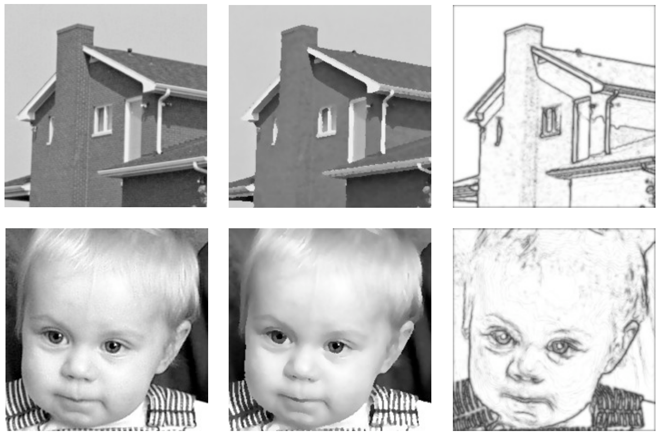
Algorithm 3 Channel smoothing algorithm.

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**
  $c_n \Leftarrow \text{conv2}(c_n, g_\sigma)$
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



Computer Vision Laboratory

Image Denoising

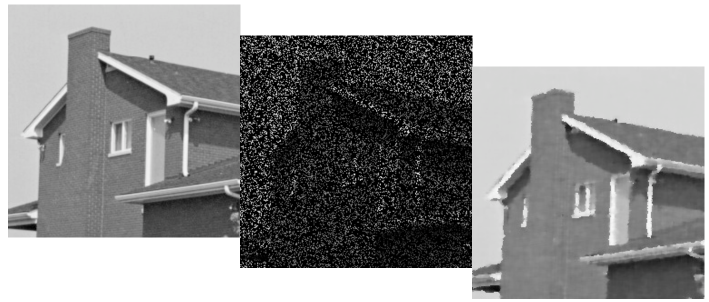


Computer Vision Laboratory

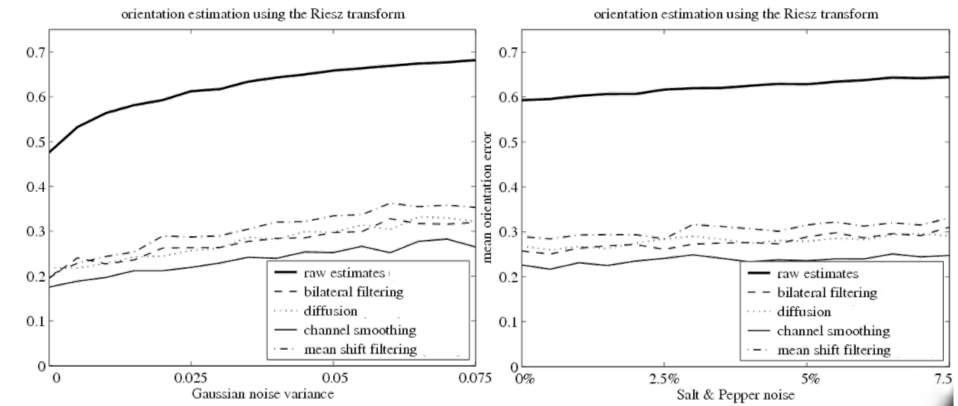
Image Denoising

Random Sample Reconstruction

- Real data is incomplete
- Strong regularization required

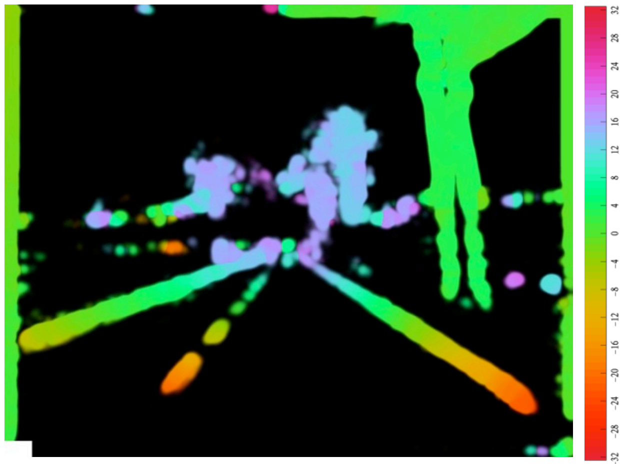


Orientation Estimation



Disparity Estimation

Disparity Estimation



alpha-Synthesis

- linear B-spline decoding

$$M_n = c_{n-1} + c_n + c_{n+1} \qquad \hat{f}_r = \frac{c_{n_r+1} - c_{n_r-1}}{M_{n_r}} + n_r$$

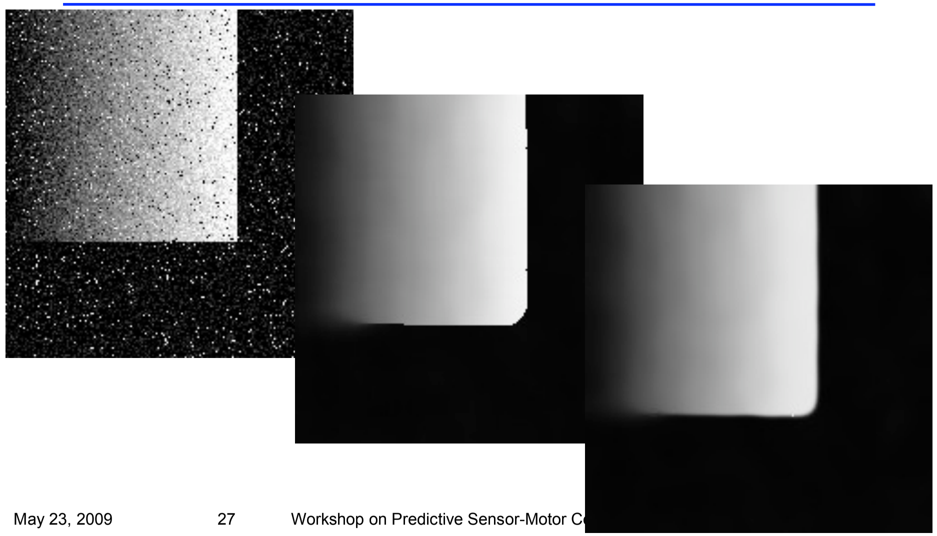
$$n_0 = \arg\max M_n$$

$$\hat{f} = \frac{c_{n_0+1} - c_{n_0-1}}{M_{n_0}} + n_0 \qquad \hat{f} = \frac{\sum_r \hat{f}_r M_{n_r}^\alpha}{\sum_l M_{n_l}^\alpha} .$$

- virtual shift decoding

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

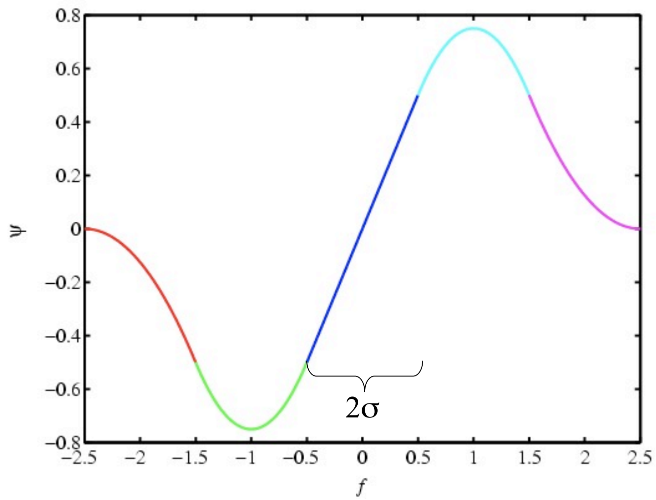
alpha-Synthesis Result



Channel Smoothing Parameters

- At least two parameters: channel resolution and averaging kernel
- Channel resolution estimated from noise level
- Kernel a la Wiener
- Possibly different parameter for each channel

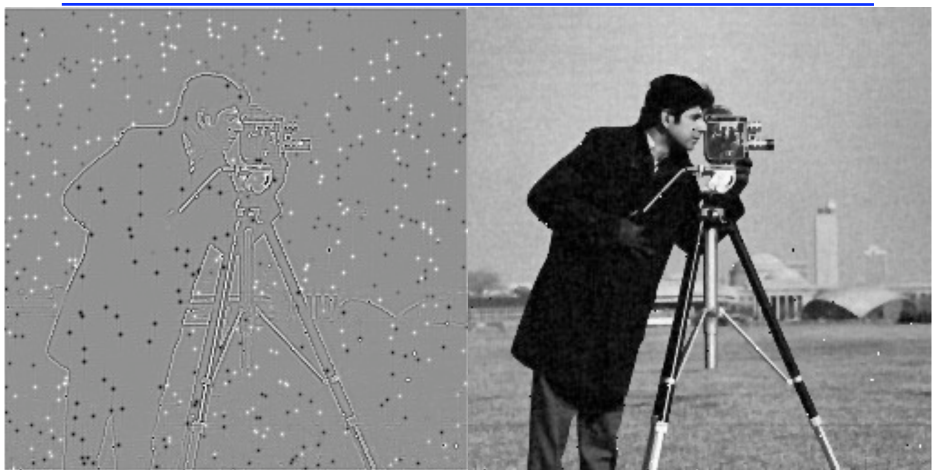
Channel Resolution



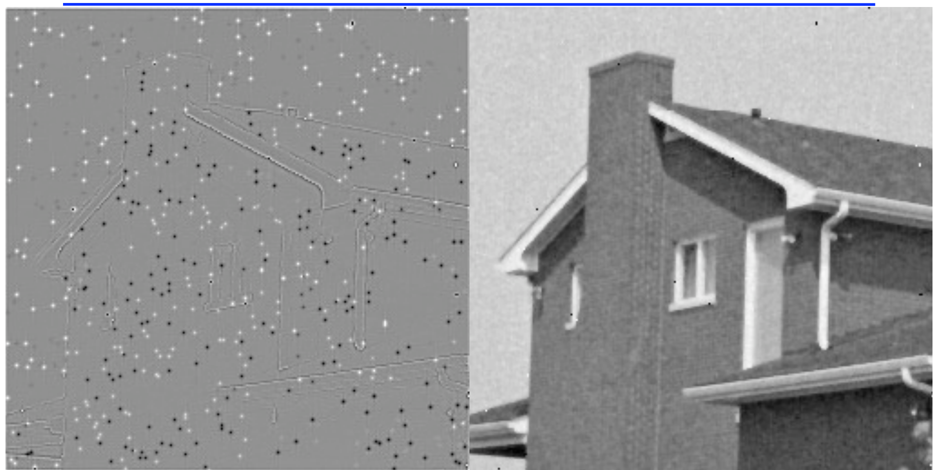
Experiment 2

Experiment 1

Experiment 2



Experiment 1



Masked Normalized Averaging

- Normalized averaging:

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

- Averaging filter / applicability: a
- Certainty:

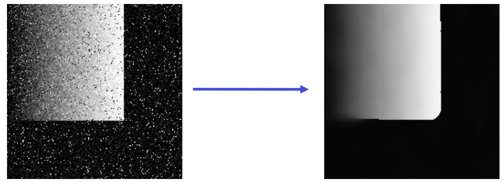
$$b(\mathbf{x}) = \begin{cases} 1 & \mathbf{x} \in \Omega \\ 0 & \mathbf{x} \notin \Omega \end{cases}$$

- Mask result:

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

Channel Smoothing

- Task: Reduce noise without blurring edges
  - locally linear
- Problem: No consideration of the multidimensional neighborhood structure



Graph-Cut Channel Smoothing

Algorithm 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

Binary Graph-Cut

- Solves globally this labeling problem:

$$\hat{f} = \arg\min E(f) = \arg\min E_{\text{smooth}}(f) + E_{\text{data}}(f)$$

- where

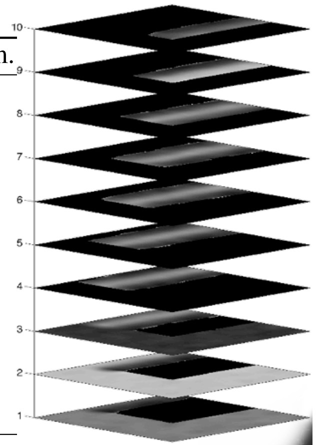
$$E_{\text{data}}(f) = \sum_{p \in \mathcal{P}} D_p(f_p)$$

$$E_{\text{smooth}}(f) = \sum_{\{p,q\} \in \mathcal{N}} V_{p,q}(f_p, f_q)$$

Graph-Cut Channel Smoothing

Algorithm 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

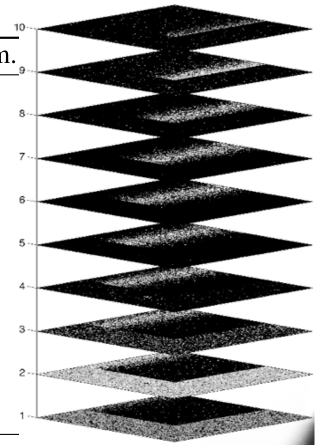


Computer Vision Laboratory

Graph-Cut Channel Smoothing

Algorithm 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



May 23, 2009 Workshop on Predictive Sensor-Motor Control LiU

Computer Vision Laboratory

Graph-Cut Channel Smoothing

Algorithm 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

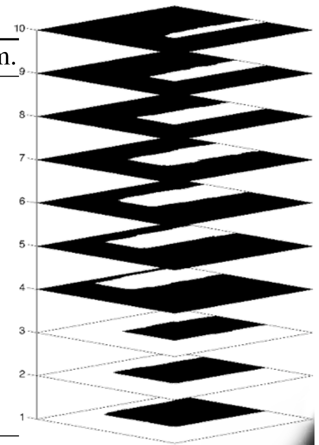
May 23, 2009 Workshop on Predictive Sensor-Motor Control LiU

Computer Vision Laboratory

Graph-Cut Channel Smoothing

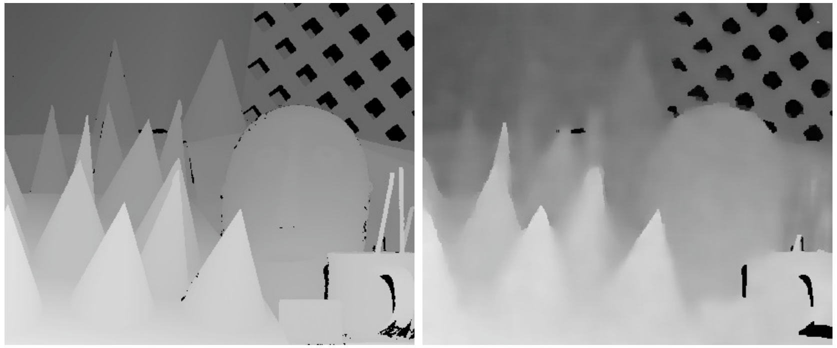
Algorithm 5 Graph-cut channel smoothing algorithm.

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



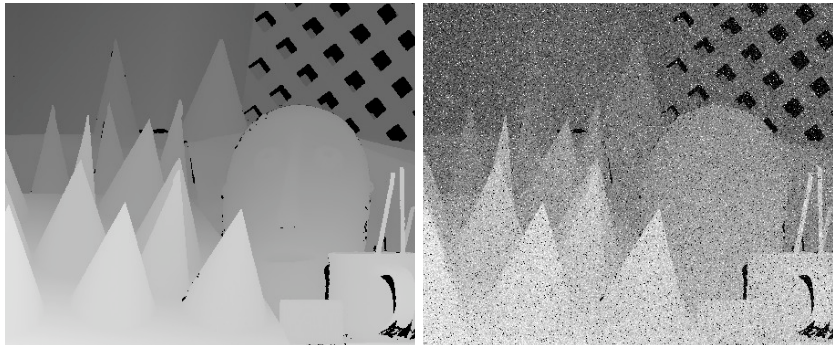
May 23, 2009 Workshop on Predictive Sensor-Motor Control LiU

Evaluation (Middlebury)



Ground truth GC channel smoothing

Evaluation (Middlebury)

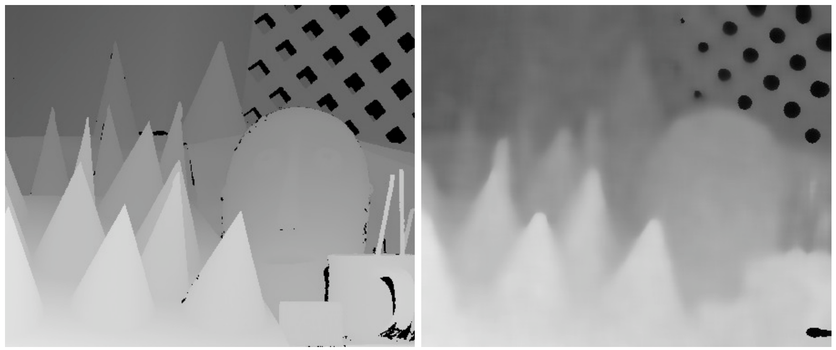


Ground truth Gaussian noise (10% std) and S&P (5%)

Drawback Channel Smoothing

- no coherence enhancing filtering

Evaluation (Middlebury)

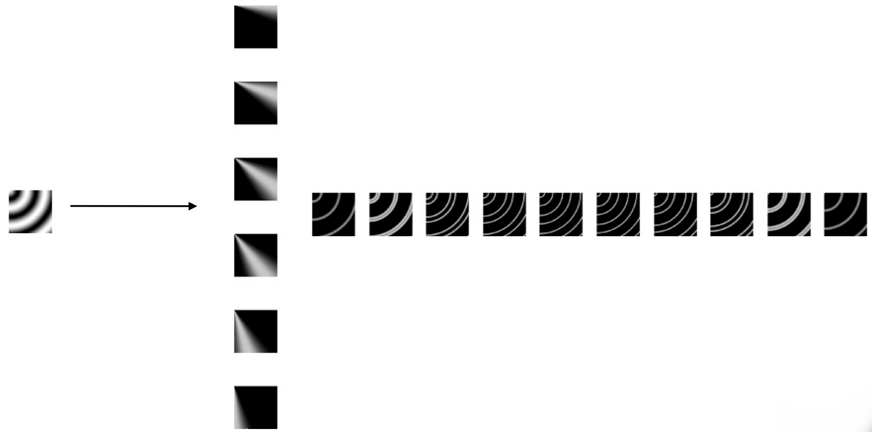


Ground truth Channel smoothing

Experiment

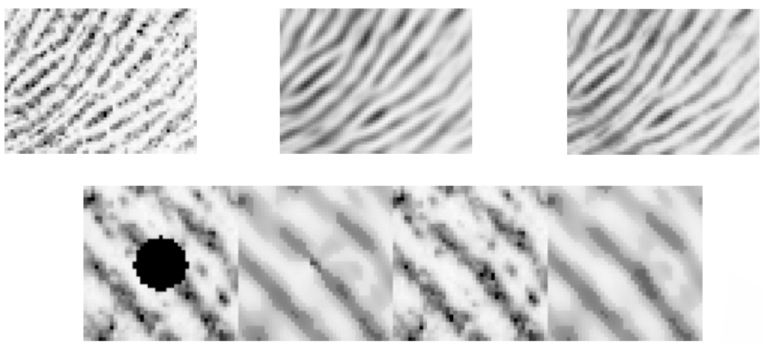
original image | coherence enhancing diffusion | anisotropic channel smoothing

Orientation Adaptive CS

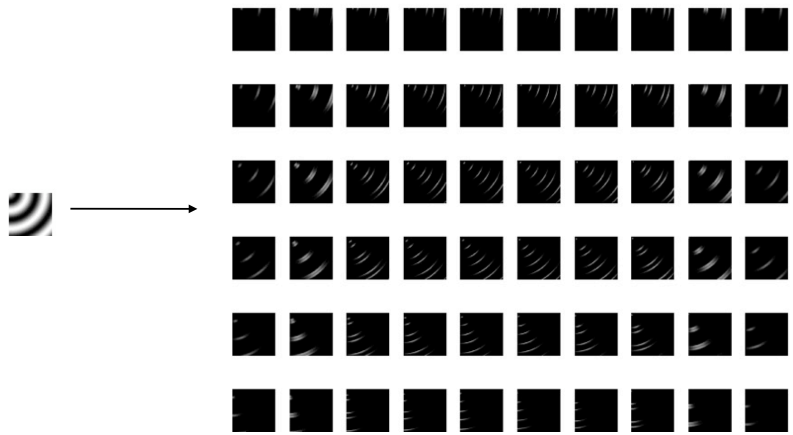


Experiment

original image | coherence enhancing diffusion | anisotropic channel smoothing

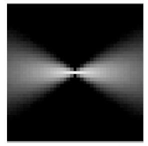


Orientation Adaptive CS



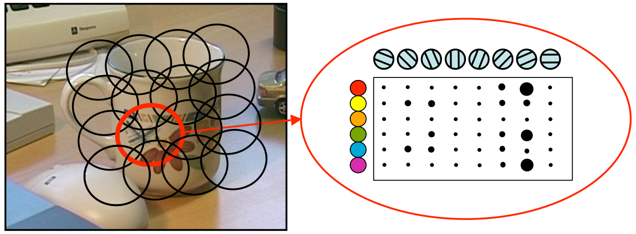
Filter Adaptation

- Significant improvement by bringing in model knowledge:
  - –Perpendicular orientation should not be influenced
  - –Polar filter design is appropriate for orientation data
- Model fitting of “hourglass” filter



Corner Detection

CCFMs



- point-wise encoding

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

Different Formulations

- Scalar product / correlation

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

- where

$$\delta_f(x,y,z) = \delta(z - f(x,y))$$

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

Algorithm

- direct implementation

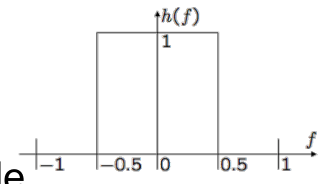
Algorithm 6 CCFM algorithm.

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

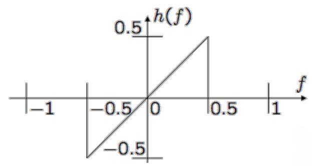
- other alternatives: "mono-pieces"

P-Channel Representation

- Simplest channel representation: Histogram
- Basis function:



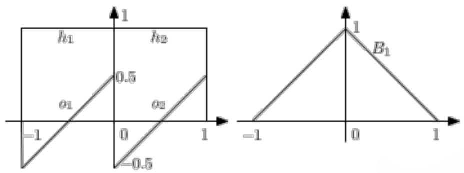
- No reconstruction possible
- Second basis function:



P-Channel Conversion

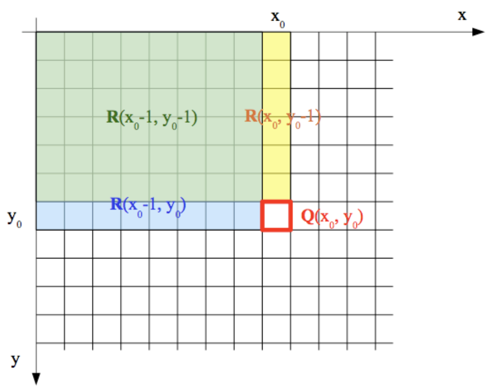
- Probabilistic analysis with overlapping kernels
- Avoid high computational cost using P-channels (mono-pieces)
- Convert P-channels to linear splines

$$B_1 = \frac{h_1 + h_2}{2} + o_1 - o_2$$



Integral Image

- any statistical moment can be computed by integral images



COIL-100 Objects

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

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

Videos

New Linear Scale-Space

- 3D linear scale-space

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

$$\delta_f(x,y,z) = \delta(z - f(x,y))$$

- Parabolic PDE?
- Just 3D Gaussian / alpha Kernel?

Comparing CCFMs

- Several alternatives tested:
  - –Euclidean distance
  - –relative information / Kullback-Leibler divergence
  - –least-squares mapping to index
  - –chi-squared distance
  - –square-root distance / Bhattacharyya coefficient
  - –quadratic form distance
  - –earth mover's distance

Considerations

- simultaneously increasing scale in spatial domain and feature domain is obviously wrong (consider e.g. decreasing scale)
- from a statistical point of view it makes sense to increase feature resolution with decreasing spatial resolution

Theorem

- f-x uncertainty relation

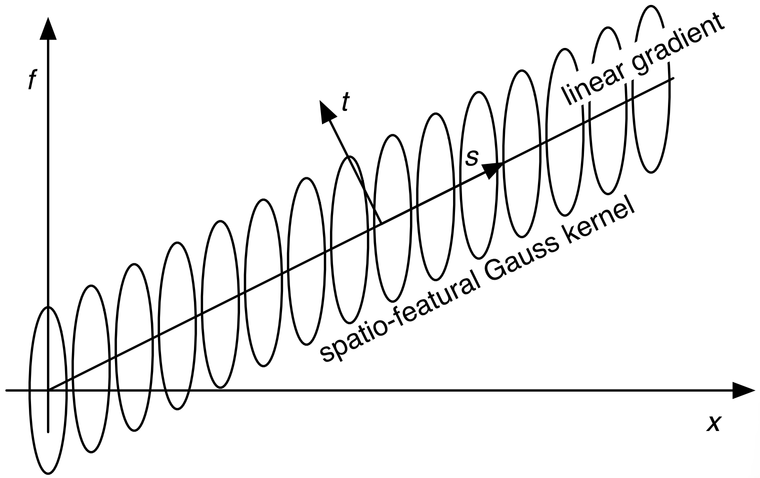
$$\exists k > 0 : \qquad (\Delta x)(\Delta f) \geq k$$

$$k = \frac{1}{2}\sigma_f \sigma_x$$

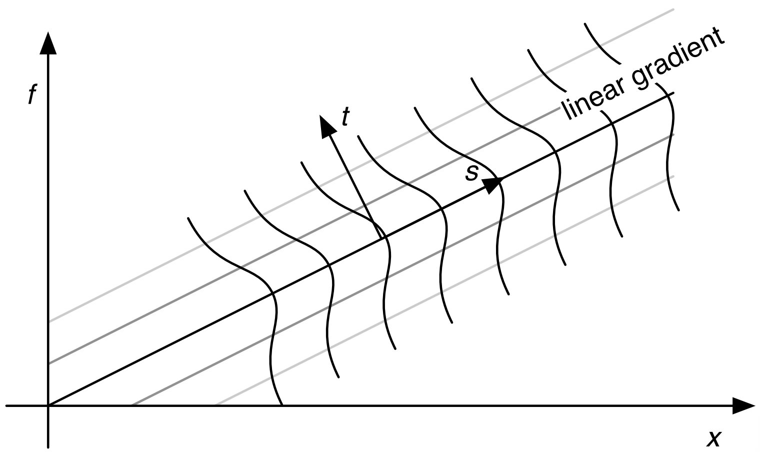
- proof based on isotropic geometry

$$x' = x + t_x$$
$$f' = f + \tan(\phi)x + t_f$$

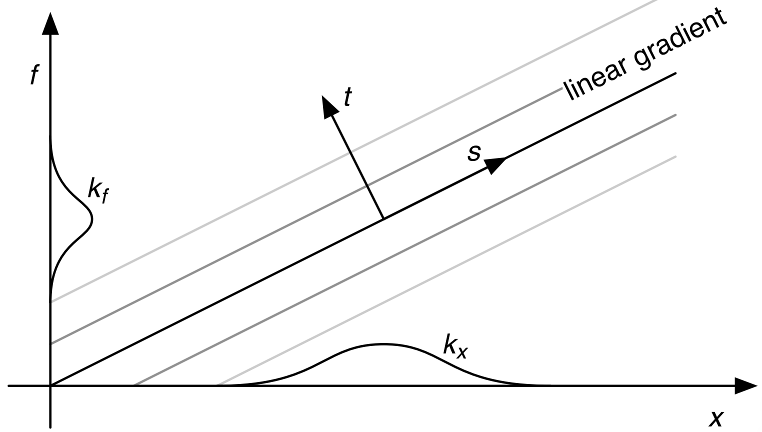
Example



Example



Example



Channel Pyramid

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
  $\mathbf{c}_f \Leftarrow \mathrm{interpolate}(\mathbf{C},\mathbf{x})$
  $[\mathbf{f}(\mathbf{x})\ \mathbf{E}(\mathbf{x})] \Leftarrow \mathrm{decode}(\mathbf{c}_f)$
  $i(\mathbf{x}) \Leftarrow \arg\max_n E_n(\mathbf{x})$
  $[\hat{f}(\mathbf{x})\ \hat{E}(\mathbf{x})] \Leftarrow [f_{i(\mathbf{x})}(\mathbf{x})\ E_{i(\mathbf{x})}(\mathbf{x})]$
end for

- Questions?
- Comments?
- Contact mfe@isy.liu.se
- Further info: www.cvl.isy.liu.se