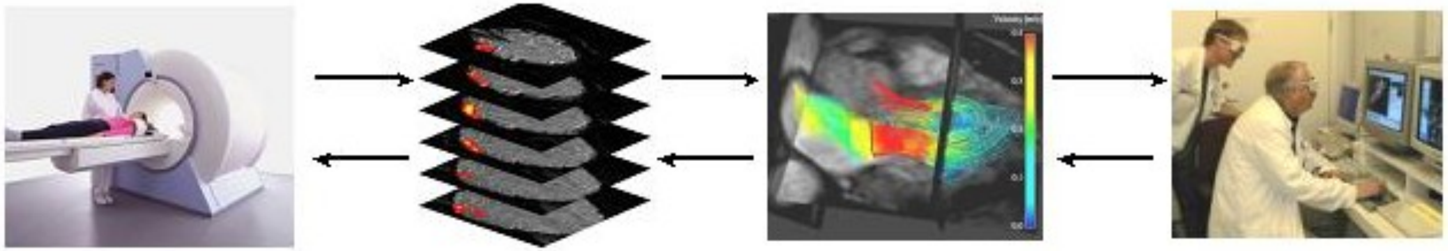
ACCOUNTING FOR UNCERTAINTY IN MEDICAL DATA: A CUDA IMPLEMENTATION OF NORMALIZED CONVOLUTION

Stefan Lindholm and Joel Kronander

Scientific Visualization Group
Linköping University

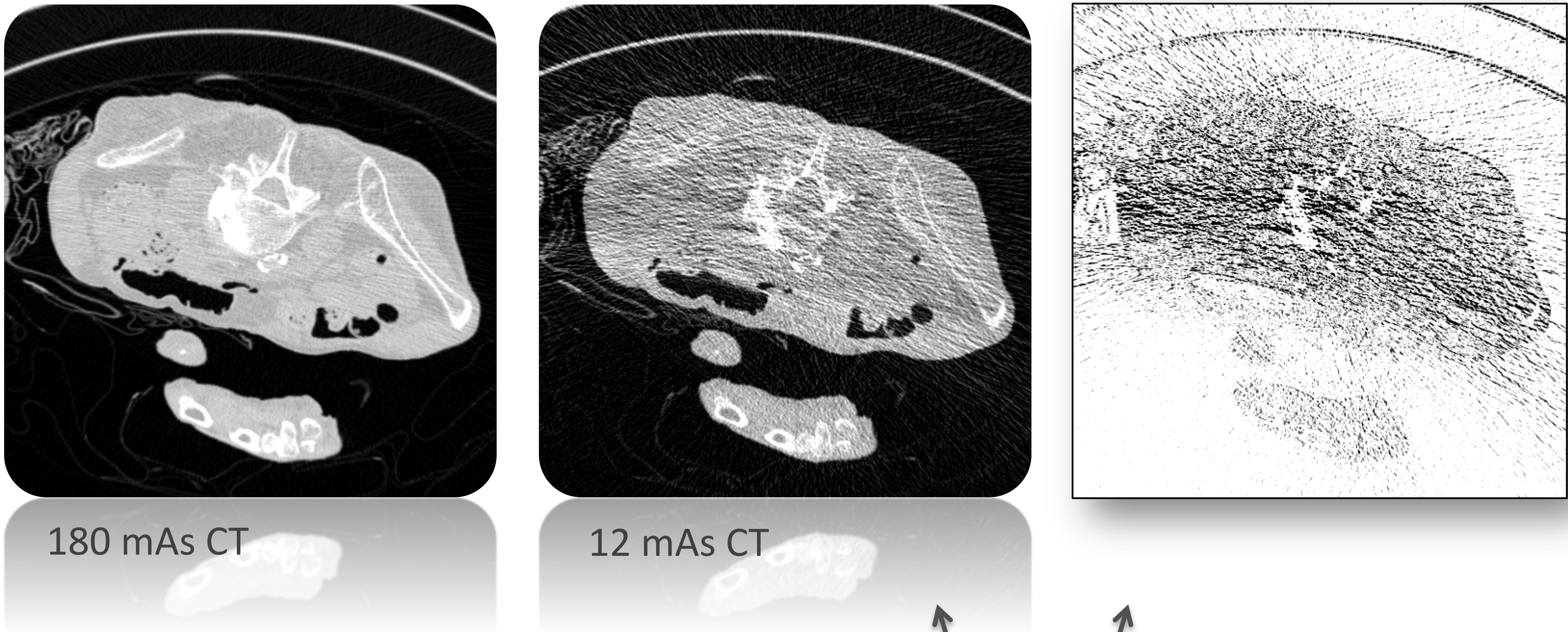
Edge Detection
Bilateral Filtering
Transfer Functions
Gradient Estimation

FILTERING MEDICAL IMAGES



UNCERTAINTY

KNOWN UNCERTAINTY

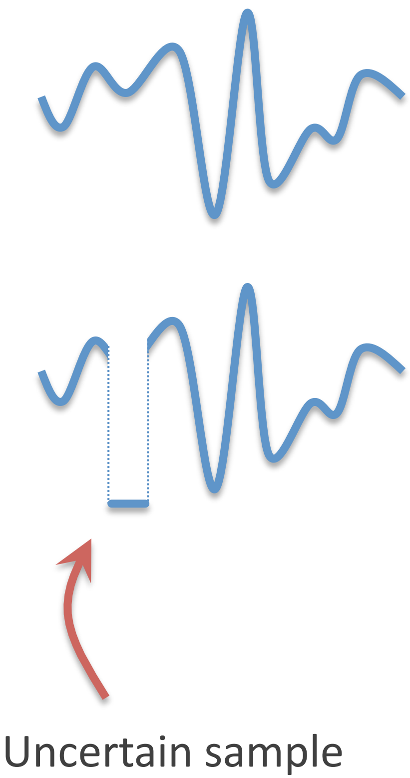


180 mAs CT

12 mAs CT

Notation: $s(k)$ $r(k)$

Known Uncertainty



Filtering a known signal is straightforward

..but..

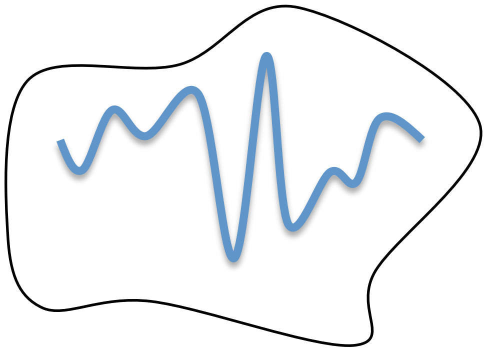
How do we treat uncertainty?

Local filter kernels, adaptive to the local signal uncertainty!

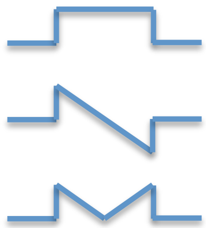
THEORY

Convolution

Polynomial expansion



$h = s * f$



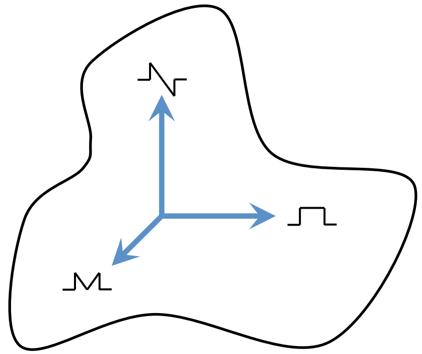
$f_1(l)$

$\vdots$

$f_M(l)$

$$h(k) = \sum_l s(k+l)\overline{f(-l)}$$

G



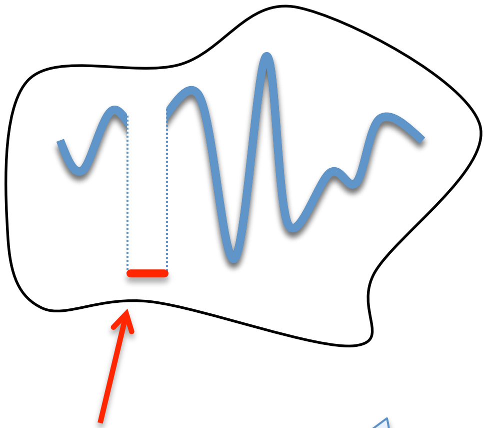
$$h_m(k) = \langle s(k+l) | \overline{f_m}(-l) \rangle$$

$$h_m(k) = \sum_l s(k+l) G_l \overline{f_m(-l)}$$

Interpret convolution as a projection

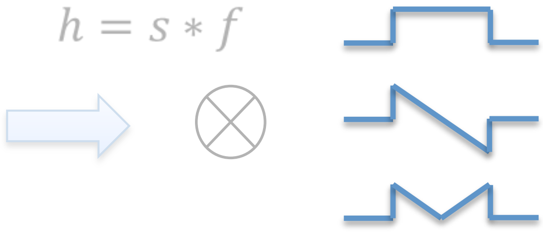
Normalized Convolution

Polynomial expansion



Uncertainty!

$h = s * f$

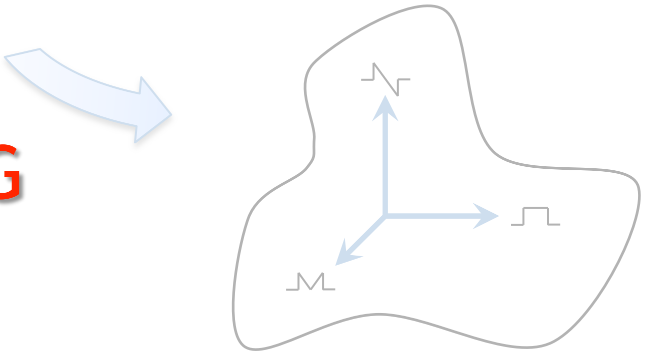


$f_1(l)$

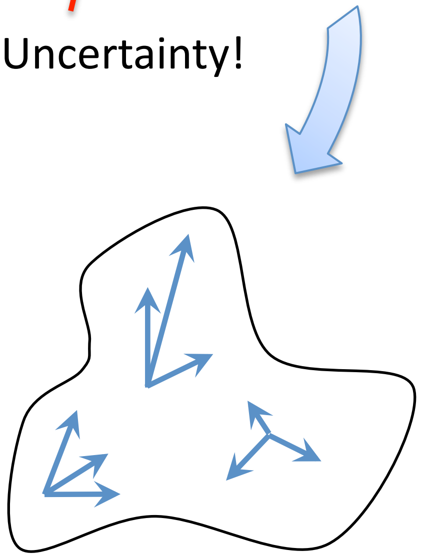
$\vdots$

$f_M(l)$

$$h(k) = \sum_l s(k+l)\overline{f(-l)}$$

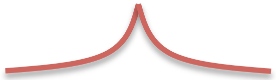


G



$$h_m(k) = \langle s(k+l) | \overline{f_m}(-l) \rangle$$

$$h_m(k) = \sum_l s(k+l) G_{kl} \overline{f_m(-l)}$$



Adaptive metric!

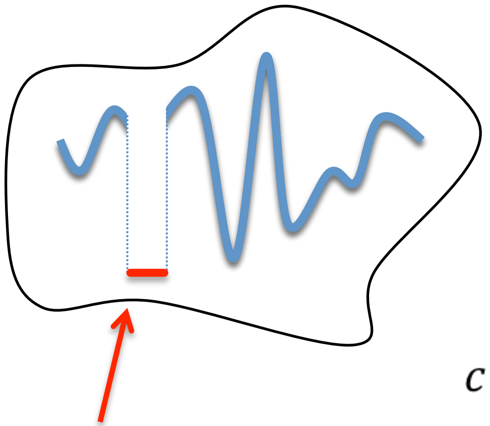
Spatially varying!

Applicability: $a(l)$

Signal certainty: $r(k)$

$$h_m(k) = \sum_l s(k+l) r(k+l) a(l) f_m(l)$$

Normalized Convolution

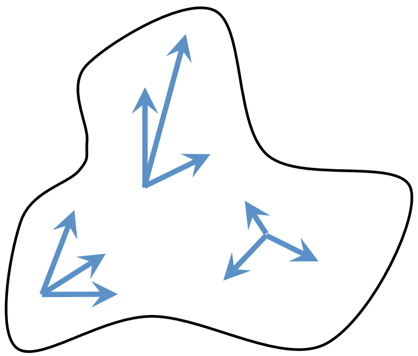


Uncertainty!

Polynomial Expansion for Orientation
and Motion Estimation. [Far02] Farnebäck

$$c = \begin{pmatrix} \langle a * b_1 * \overline{b_1} | r \rangle & \cdots & \langle a * b_1 * \overline{b_M} | r \rangle \\ \vdots & \ddots & \vdots \\ \langle a * b_M * \overline{b_1} | r \rangle & \cdots & \langle a * b_M * \overline{b_M} | r \rangle \end{pmatrix}^{-1} \begin{pmatrix} \langle a * b_1 | r * s \rangle \\ \vdots \\ \langle a * b_M | r * s \rangle \end{pmatrix}$$

One matrix inversion per pixel in the image!



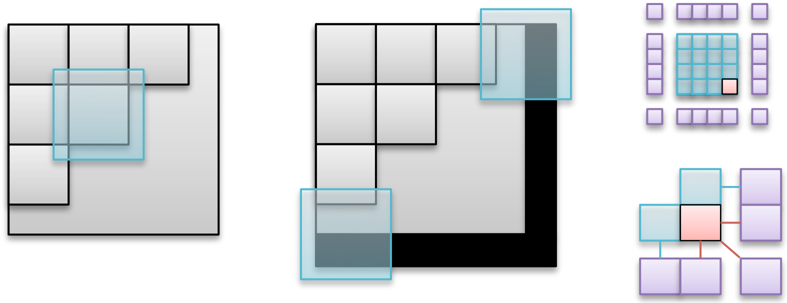
Adaptive metric!
Spatially varying!

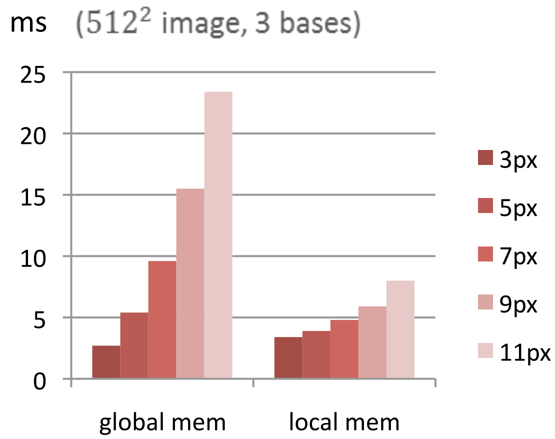
Pre-computation!
G is only *M x M*!

CUDA

CUDA

- Standard optimizations
  - 512 cores
  - Coalesced memory reads
  - Latency hiding
  - Loop unrolling
- Optimizations for Normalized Convolution
  - Templates for recursive matrix inversion
  - Pre-processor defines



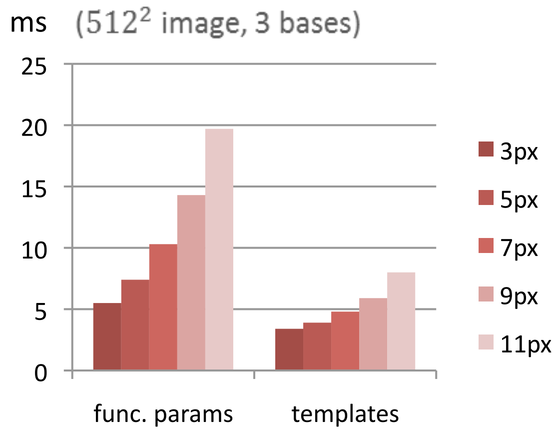


CUDA

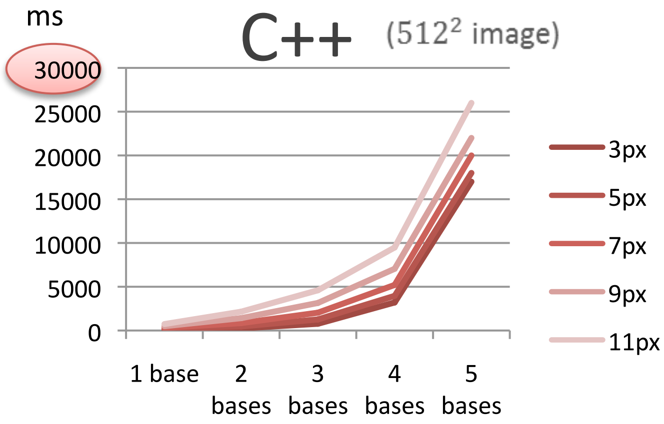
$$A^{-1} = \frac{1}{\det(A)} * Adj(A)$$

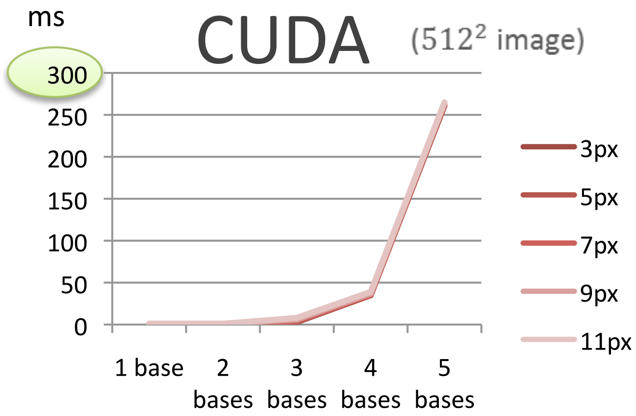
- Standard optimizations
  - 512 cores
  - Coalesced memory reads
  - Latency hiding
  - Loop unrolling
- Optimizations for Normalized Convolution
  - Templates for recursive matrix inversion
  - Pre-processor defines

```
template<int M> __device__ float Determinant()
                {... Determinant<M-1>() ...}
template<> __device__ float Determinant<3>();
template<> __device__ float Determinant<2>();
template<> __device__ float Determinant<1>();
```



CUDA





Conclusions

G is only $M \times M$

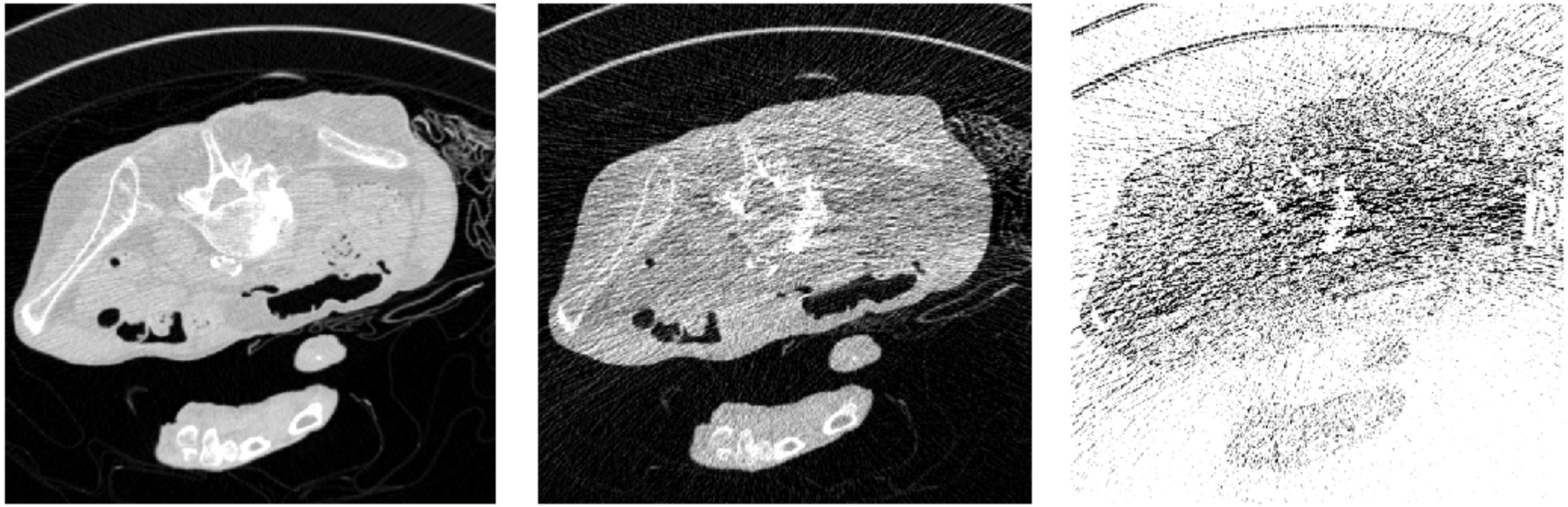
…but even $M \times M$ is costly

…and we need precision

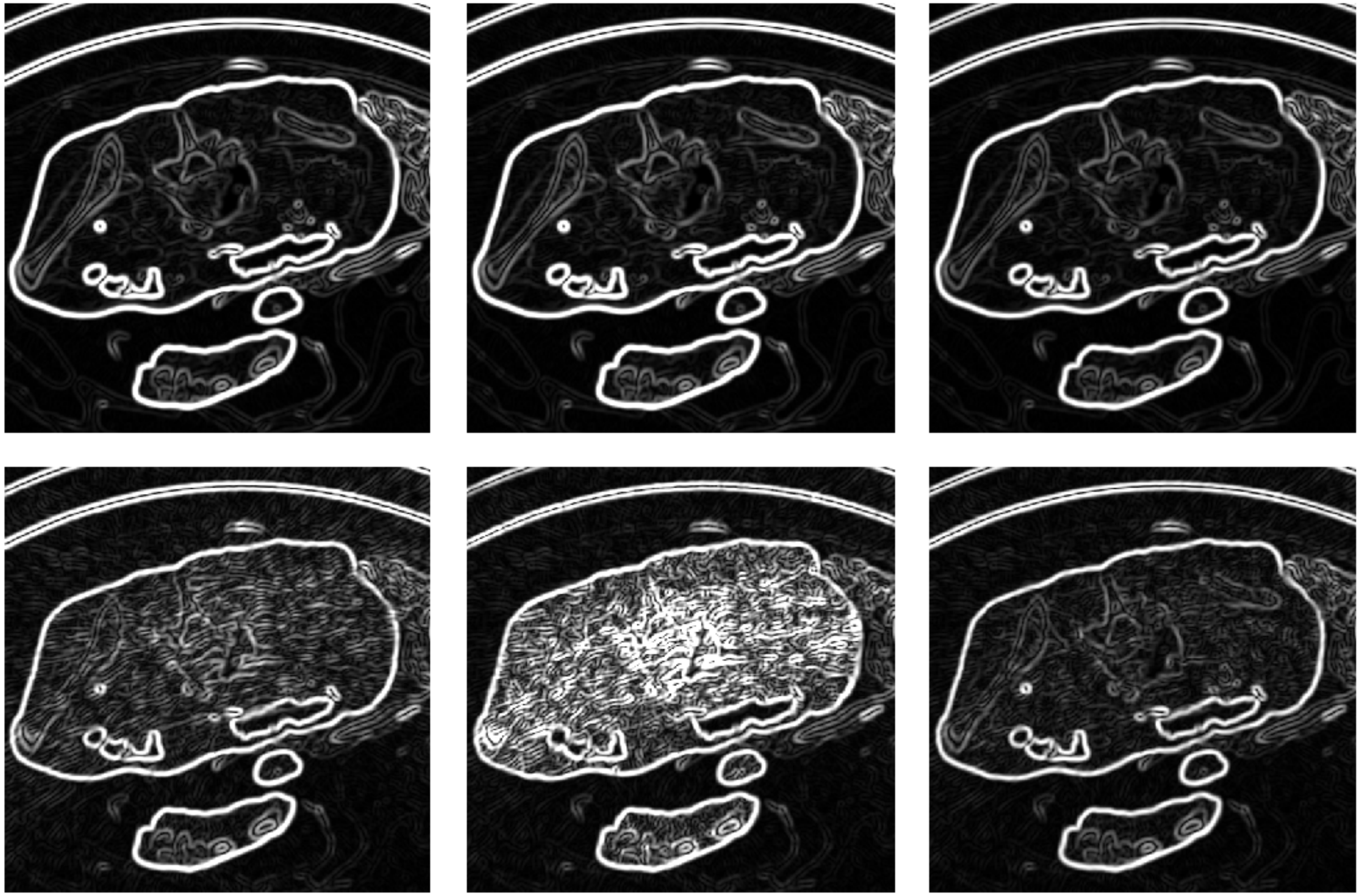
We can achieve real-time performance!

RESULTS

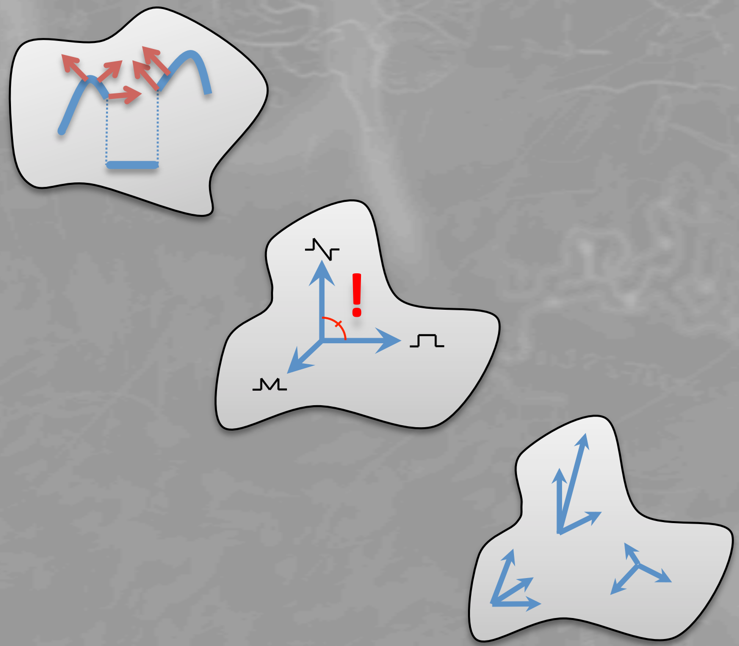
RESULTS



RESULTS



Thank You!



Linköping University
expanding reality

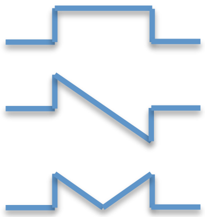
NORRKÖPING VISUALIZATION CENTER

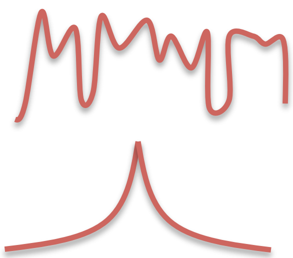
CRESEARCH

$$h_m(k) = \sum_l s(k+l)\overline{f_m(-l)}$$

$$h_m(k) = \sum_l s(k+l)G_l\overline{f_m(-l)}$$

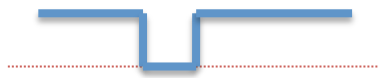
$$h_m(k) = \sum_l s(k+l)r(k+l)a(l)f_m(l)$$

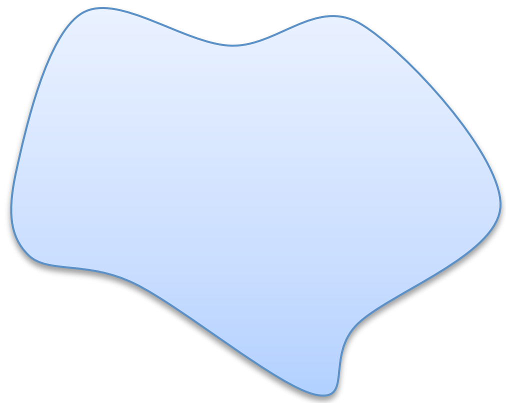




$$h(k) = \langle s(k+l) | f^*(-l) \rangle$$

$$c = G^{-1}$$
$$c = (B^* G_0 B)^{-1}$$



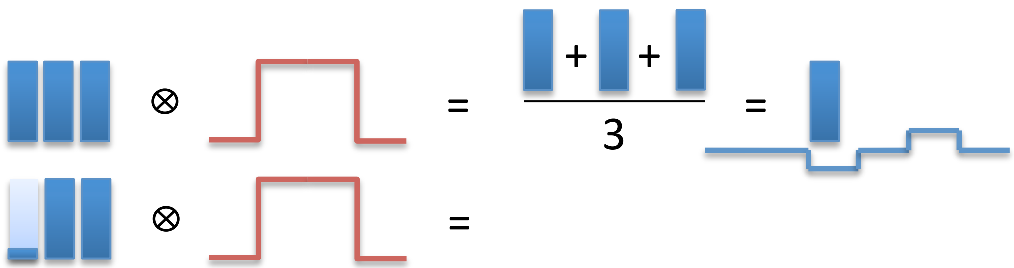


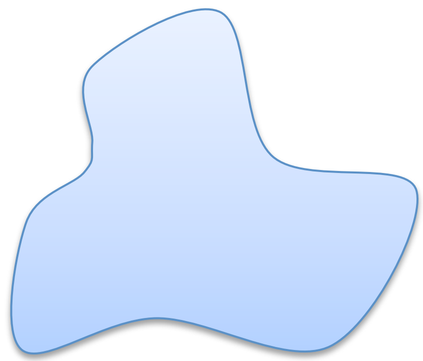
`[111] * [111] = (1+1+1)/3 = 1`

`[?11]`$_{[011]}$`* [111] =`

`(0+1+1)/3 = 0.67`

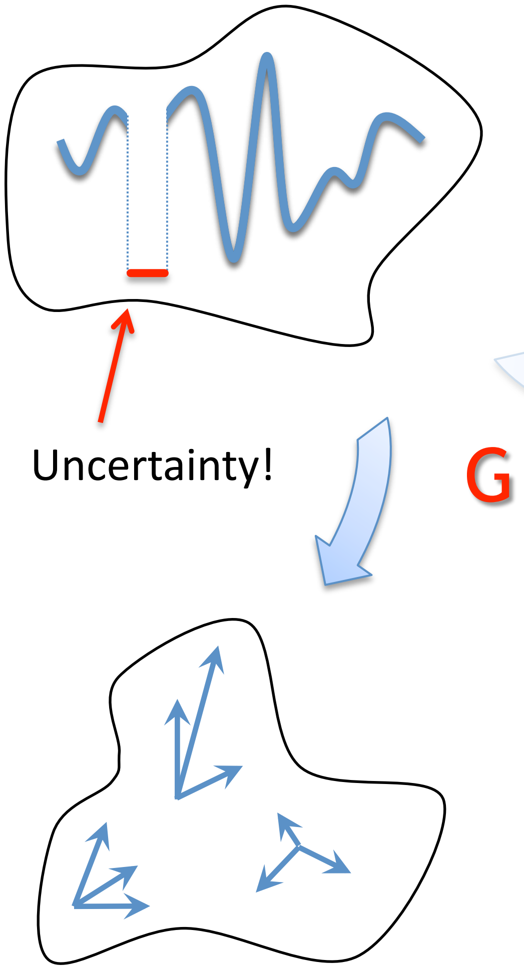
`(1+1)/2 = 1`





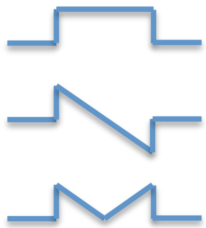
CONVOLUTION

Polynomial expansion



$h = s * f$

$\otimes$

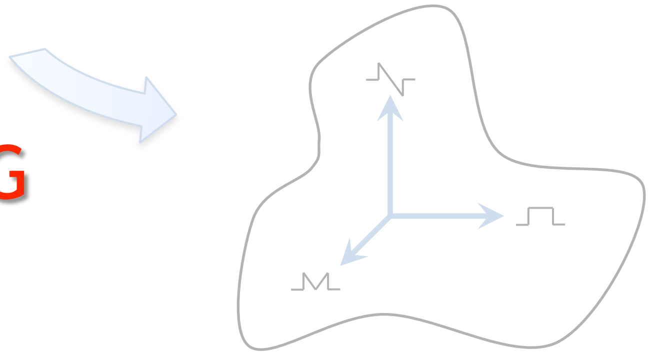


$f_1(l)$

$\vdots$

$f_M(l)$

$$h(k) = \sum_l s(k+l)\overline{f(-l)}$$

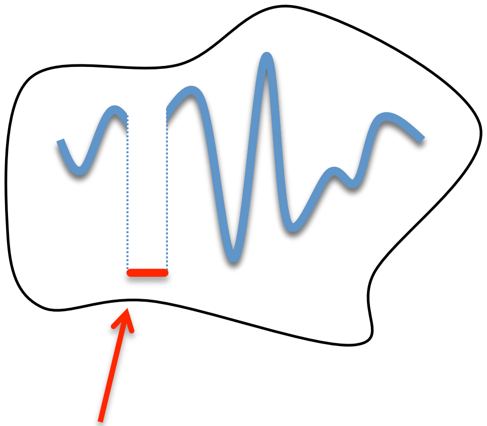


$$h_m(k) = \langle s(k+l) | \overline{f_m}(-l) \rangle$$

$$h_m(k) = \sum_l s(k+l) G_{kl} \overline{f_m(-l)}$$

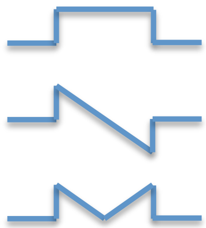
$$h_m(k) = \sum_l s(k+l) r(k+l) a(l) f_m(l)$$

Normalized Convolution



Uncertainty!

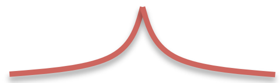
Polynomial expansion



$f_1(l)$

$\vdots$

$f_M(l)$



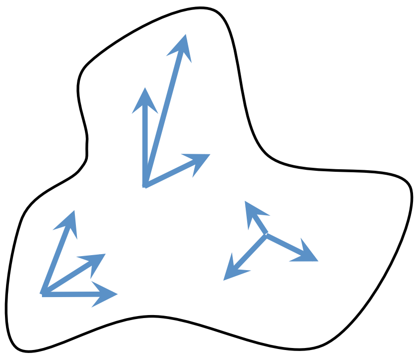
Applicability: a(l)

Signal certainty: $r(k)$

$$h_m(k) = \sum_l s(k+l)r(k+l)a(l)f_m(l)$$

$$c = G^{-1}h_m$$

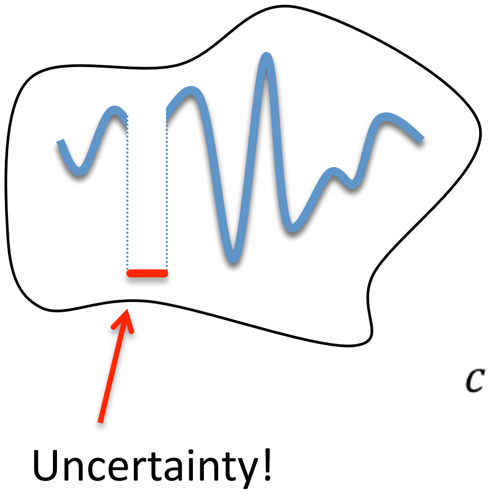
Weighted least squares



Different metric!

Spatially varying!

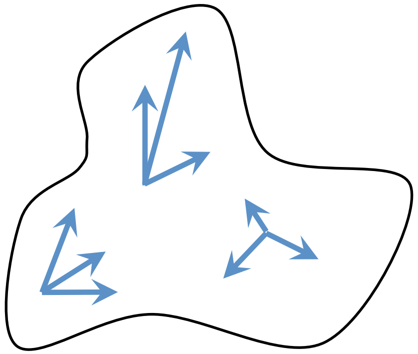
Normalized Convolution



Polynomial Expansion for Orientation and Motion Estimation. [Far02] Farnebäck

$$c = \begin{pmatrix} \langle a * b_1 * \overline{b_1} | r \rangle & \cdots & \langle a * b_1 * \overline{b_M} | r \rangle \\ \vdots & \ddots & \vdots \\ \langle a * b_M * \overline{b_1} | r \rangle & \cdots & \langle a * b_M * \overline{b_M} | r \rangle \end{pmatrix}^{-1} \begin{pmatrix} \langle a * b_1 | r * s \rangle \\ \vdots \\ \langle a * b_M | r * s \rangle \end{pmatrix}$$

$$c = G^{-1} h_m$$



Different metric! ☹
Spatially varying! ☹

G is only M x M! ☺
Pre-computation! ☺