GigaNumerics in Deep Structure Analysis of Images

Employing MathLink and the Parallel Computing Toolkit

Bart Janssen

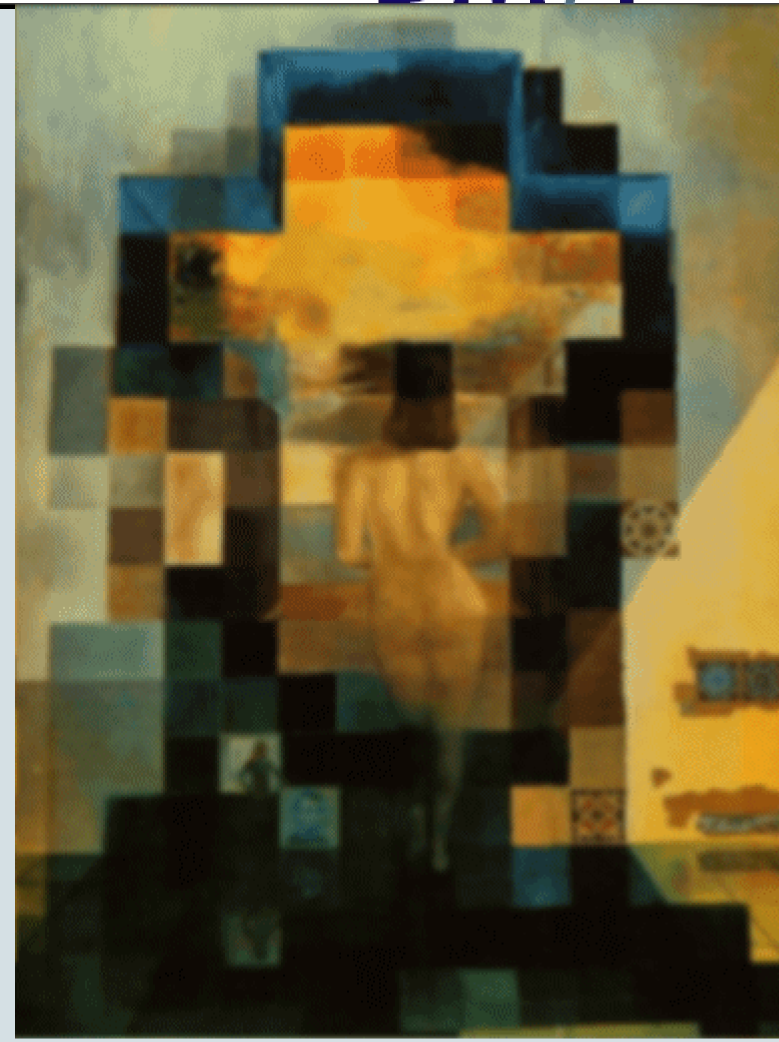
Luc Florack and Bart ter Haar Romeny

Outline

- Scale Space
- Toppoints
- Image Reconstruction
- Mathematica Implementation

The problem of scale:

Objects live at different scales



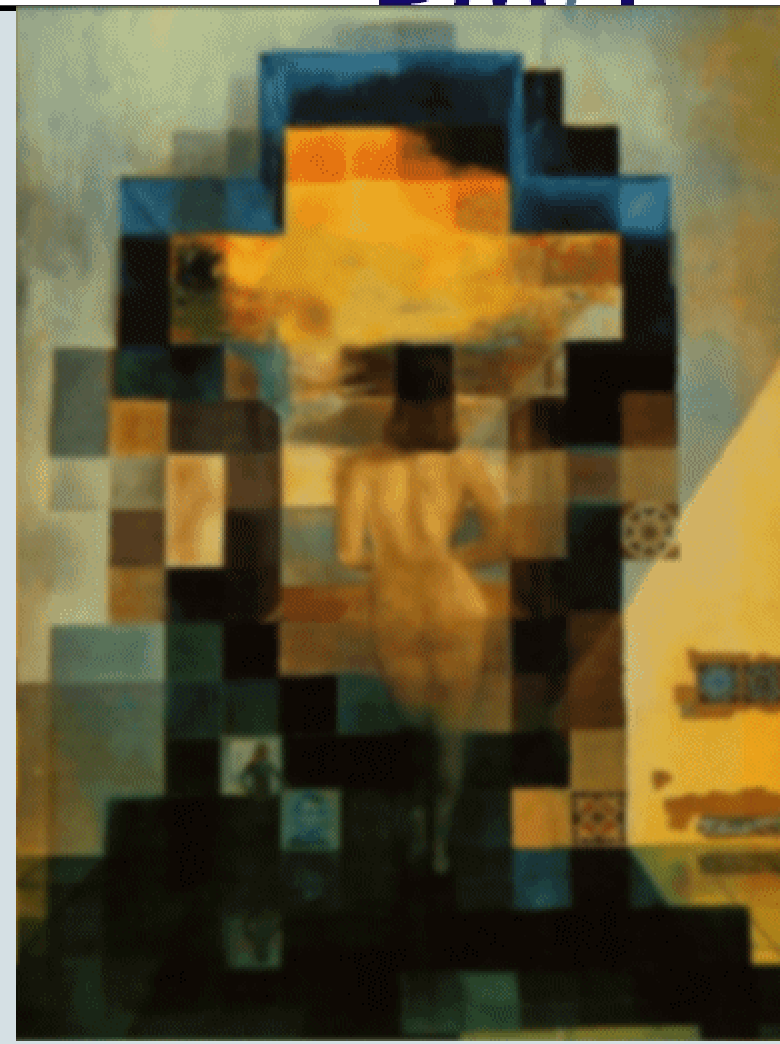
Gala looking into the Mediterranean Sea
Salvador Dali

The problem of scale:

Objects live at different scales

Solution?

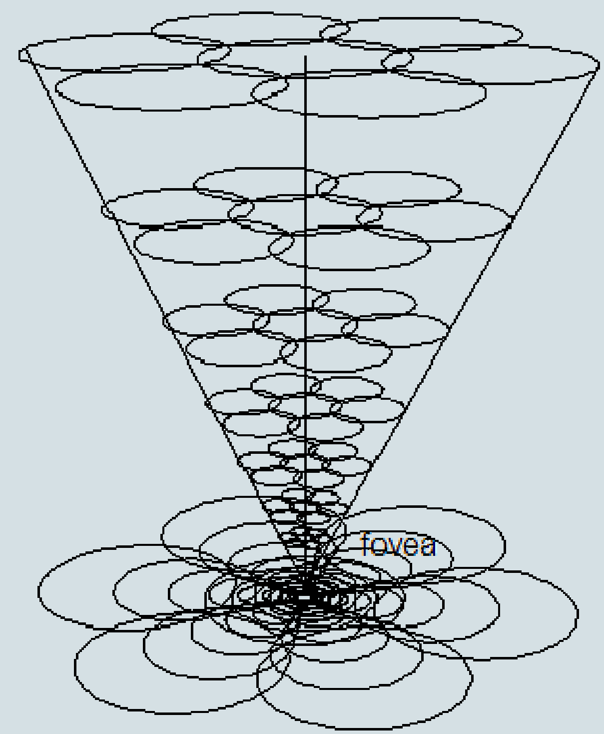
Look at all scales simultaneously



Gala looking into the Mediterranean Sea
Salvador Dali

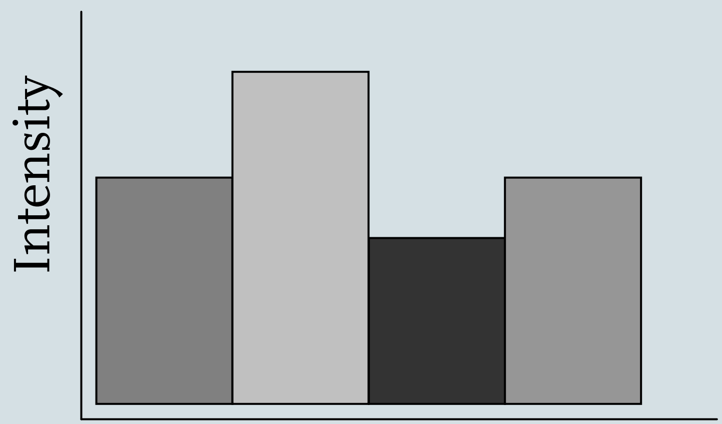
Scale Space in Human Vision

- The human visual system is a *multi-scale sampling device*
- The retina contains *receptive fields* of varying size.

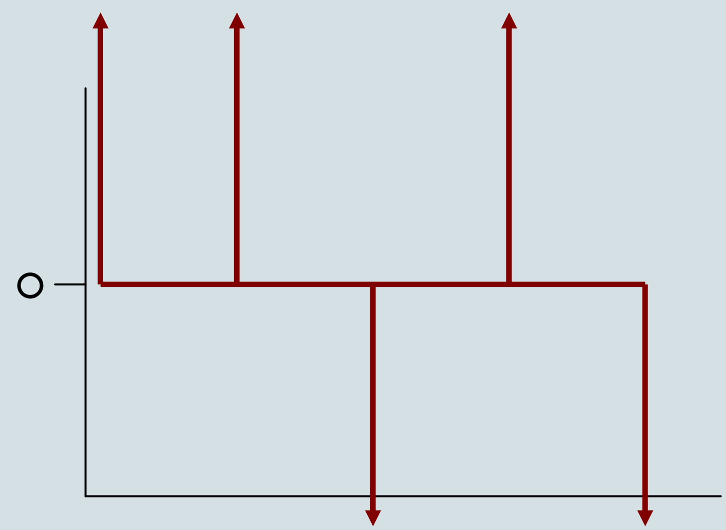


Scale for Calculating Derivatives

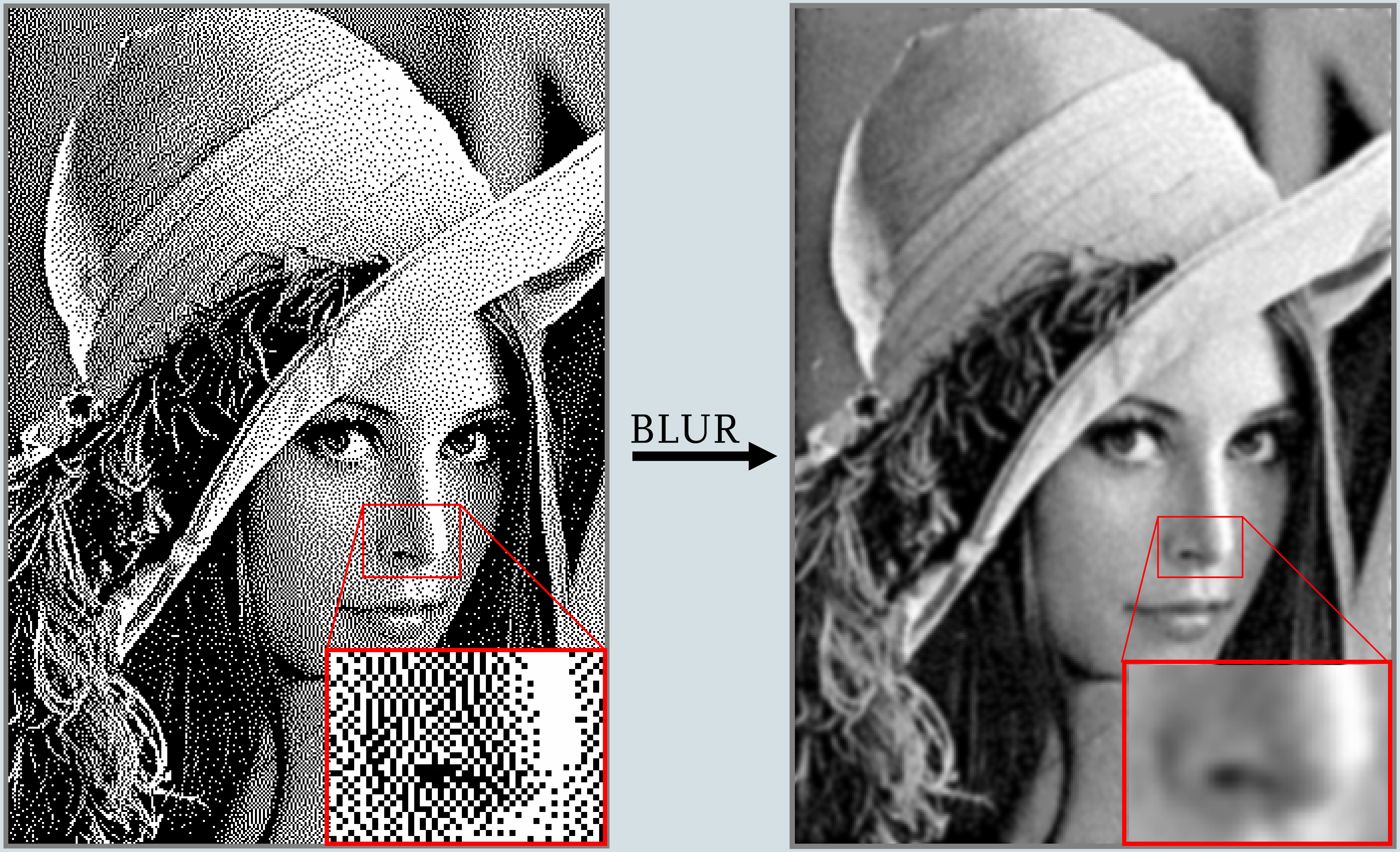
- To calculate derivatives we need smooth (continuous) data.
- Image data is not smooth.



Pixel data



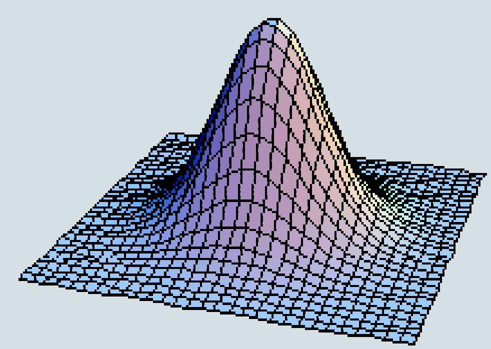
1st Order Derivative


BLUR

Practical Implementation

- Scale-Space Axioms
  - Linearity
  - Spatial shift invariance
  - Isotropy
  - Causality
  - *Separability*
- Lead to the Gaussian Kernel

$$G(x,\sigma)=\frac{1}{(2\pi\sigma^2)^{D/2}}e^{-\frac{x_1^2+\ldots+x_D^2}{2\sigma^2}}$$



Calculating Derivatives with Gaussians

- The derivative of the data at a scale σ is defined as

$$\frac{\partial}{\partial x_1}\{L_0(x) \otimes G(x;\sigma)\}$$

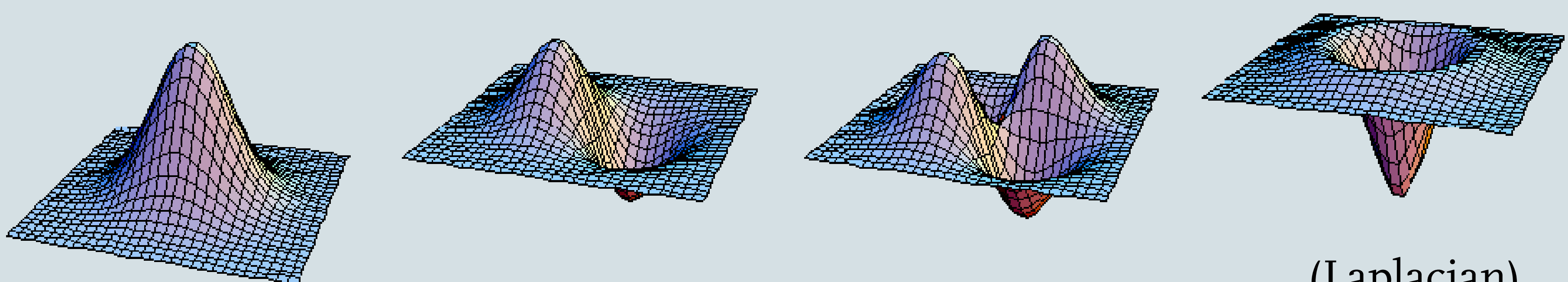
- Due to nice properties of the Gaussian this can be rewritten as

$$L_0(x) \otimes \frac{\partial}{\partial x_1} G(x;\sigma)$$

Calculating Derivatives of Images

- Differentiation becomes Integration!

...ListConvolve / FFT-method

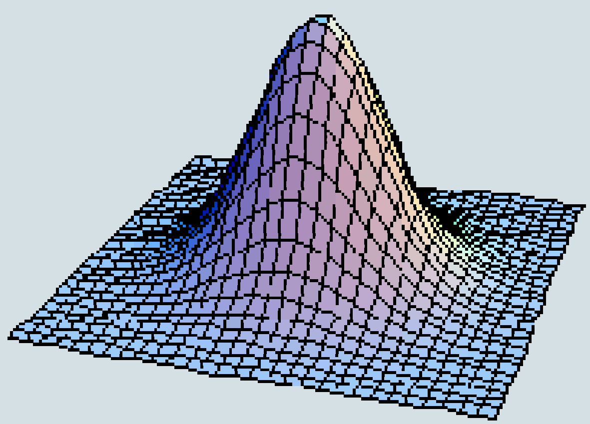


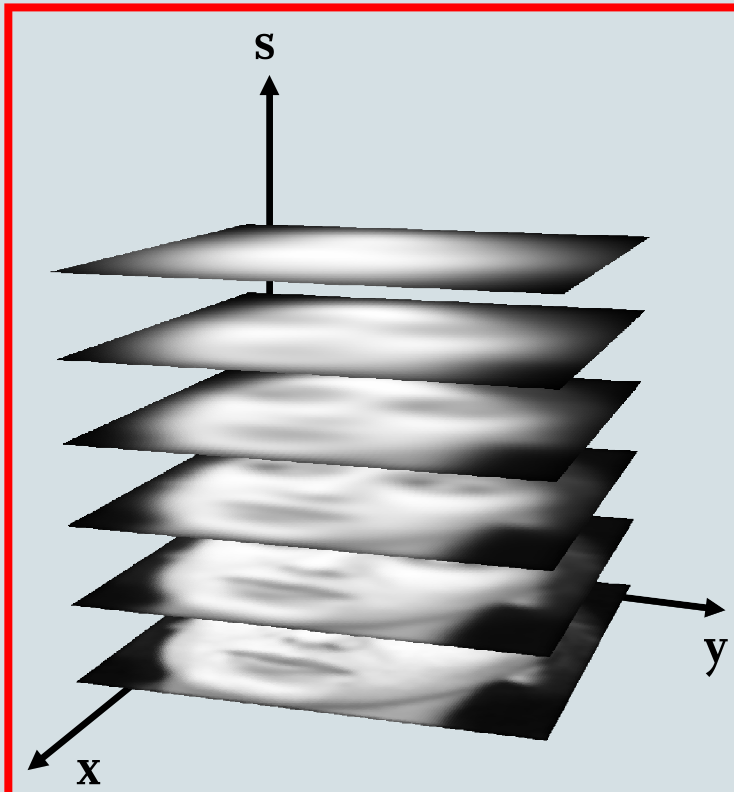
(Laplacian)

Gaussian Scale Space

$$u(\mathbf{x}; s) = (f * \varphi_s)(\mathbf{x})$$

$$\varphi_s(\mathbf{x}) = \frac{1}{\sqrt{4\pi s}^{\,n}} e^{-\frac{||\mathbf{x}||^2}{4s}}$$





Outline

- Scale Space
- Toppoints
- Image Reconstruction
- Mathematica Implementation

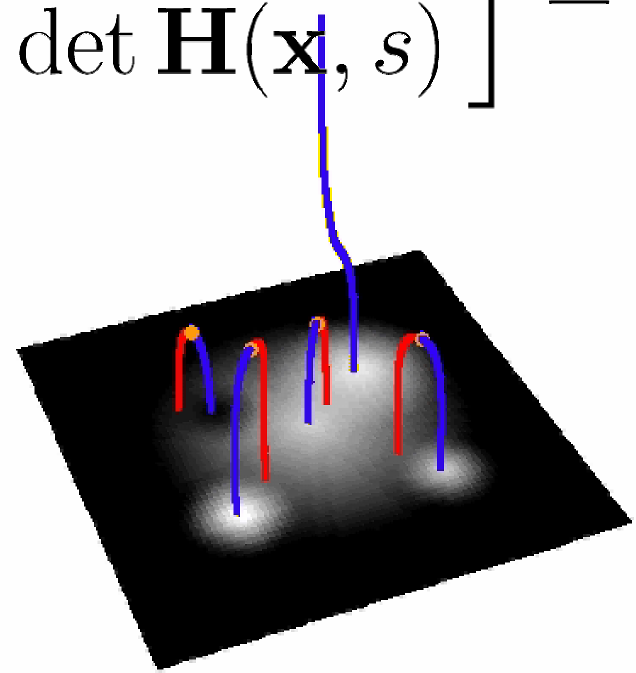
Singular points of a Gaussian scale space image

$$f(x,s) = \frac{x^3}{3} - sx$$

$s < 0$

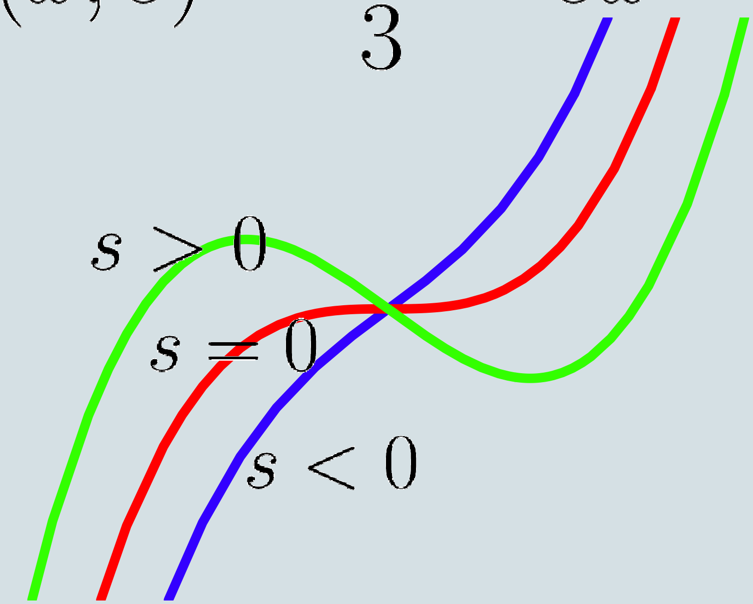
Fold Catastrophe

$$\begin{bmatrix} \nabla u(\mathbf{x}, s) \\ \det \mathbf{H}(\mathbf{x}, s) \end{bmatrix} = \mathbf{0}$$

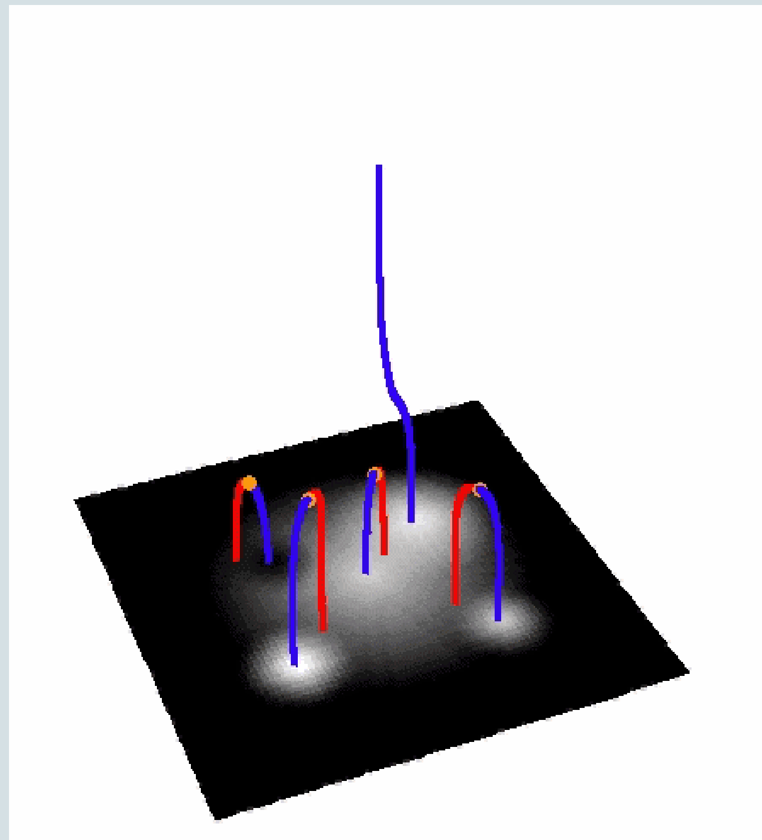


Singular points of a Gaussian scale space image

$$f(x,s) = \frac{x^3}{3} - sx$$



Fold Catastrophe



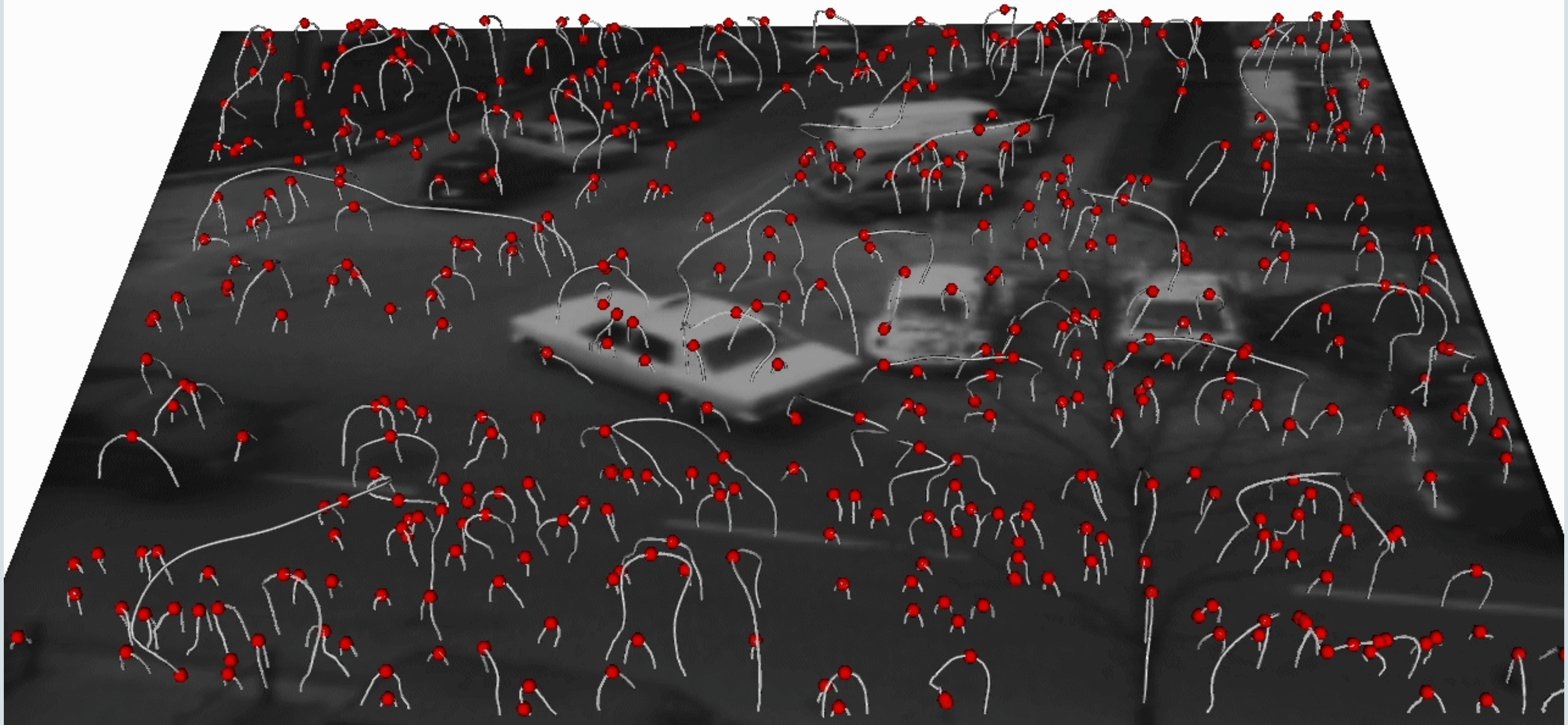
Stability of Top Points

- We can calculate the variance of the displacement of top points under noise.
- We need 4^{th} order derivatives in the top-points for that.

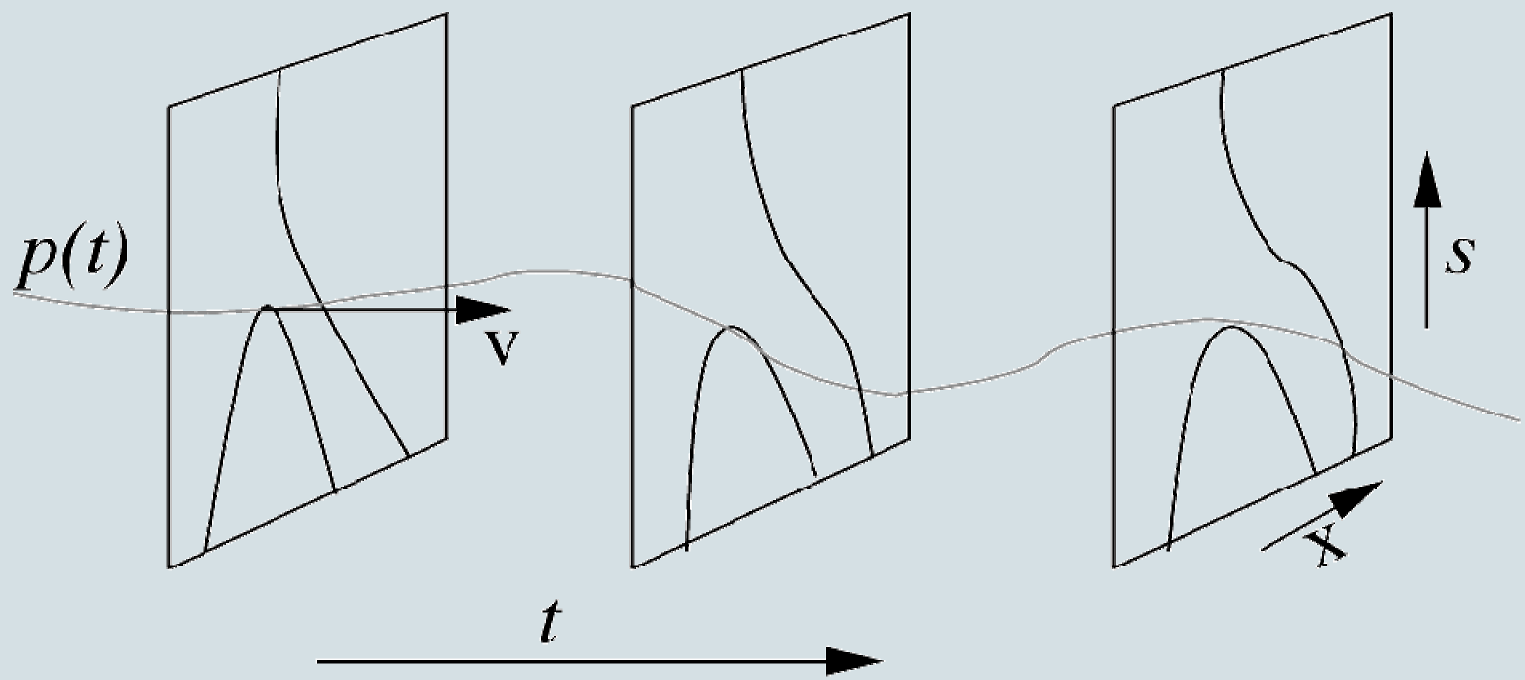
Applications

- Optic flow estimation
- Matching
- Image Editing / Segmentation?

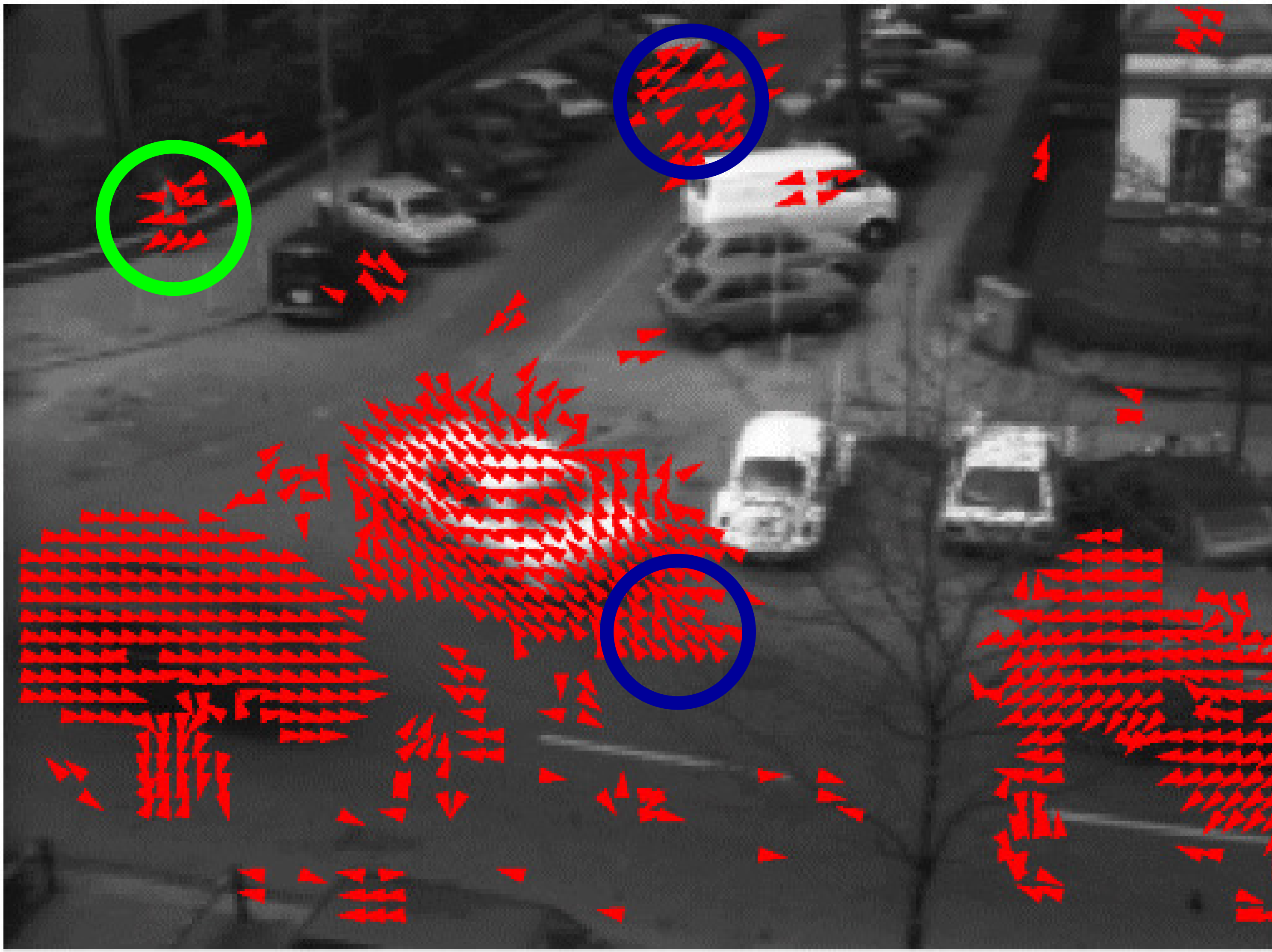
Optic Flow Estimation



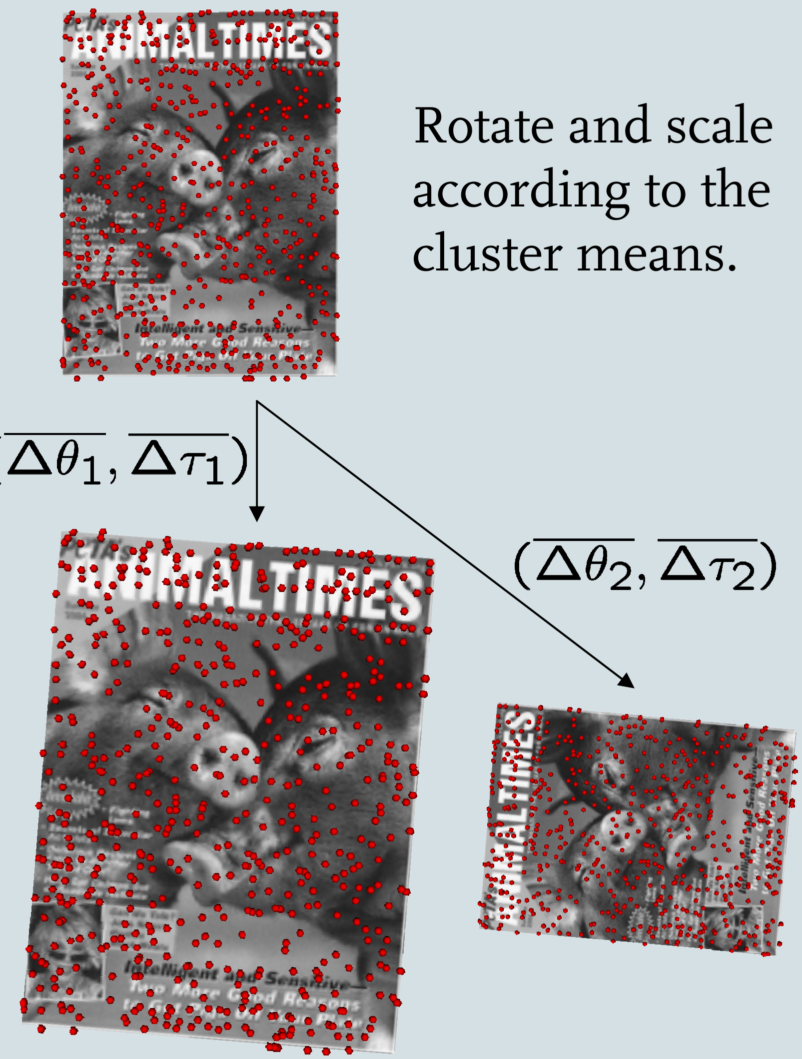
Optic Flow Estimation



$$\frac{d}{dt}\begin{bmatrix} \nabla u(t;\mathbf{x},s) \\ \det \mathbf{H}(t;\mathbf{x},s) \end{bmatrix} = \mathbf{0}$$



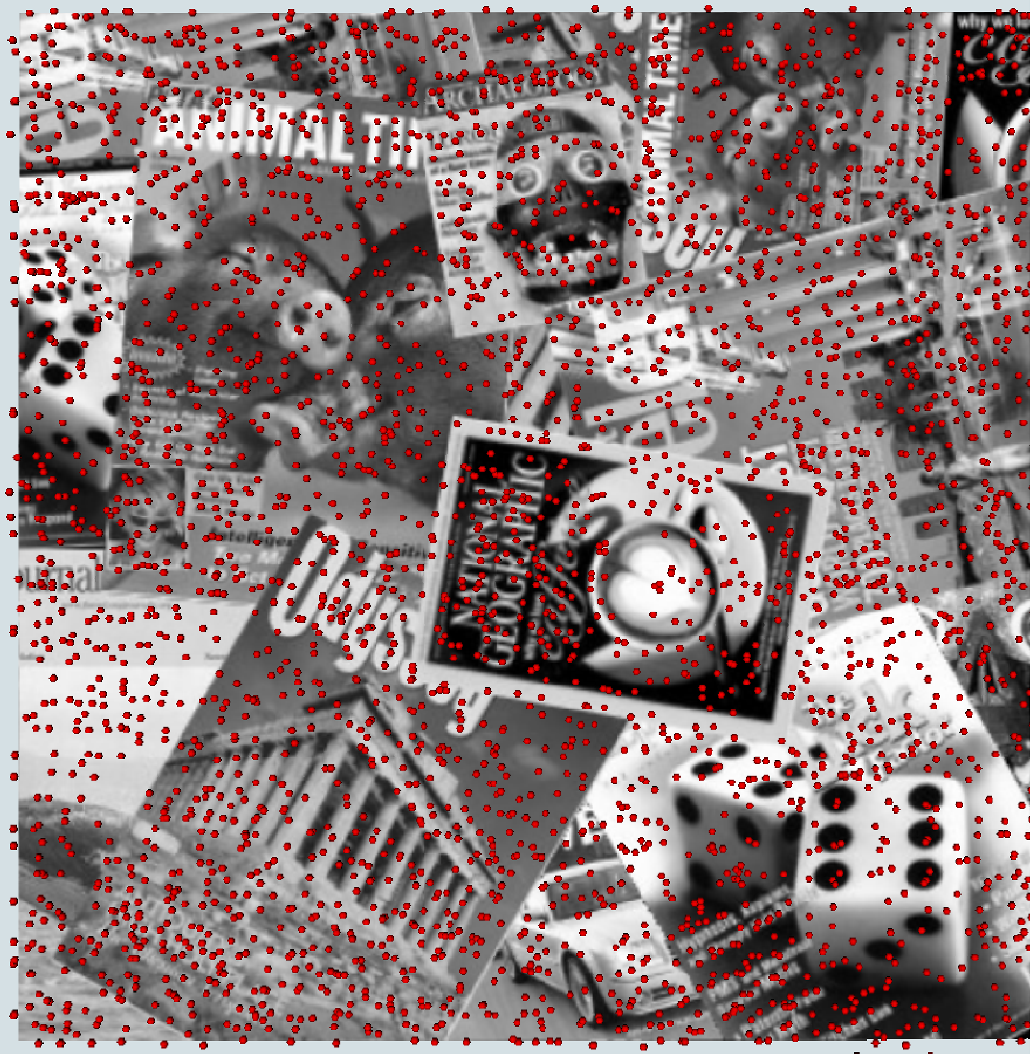
Matching



$(\overline{\Delta\theta_1}, \overline{\Delta\tau_1})$

$(\overline{\Delta\theta_2}, \overline{\Delta\tau_2})$

Rotate and scale according to the cluster means.

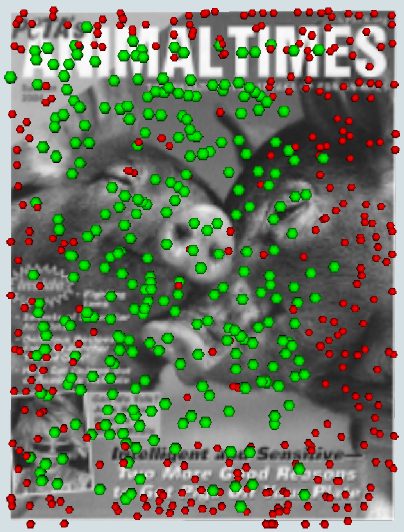


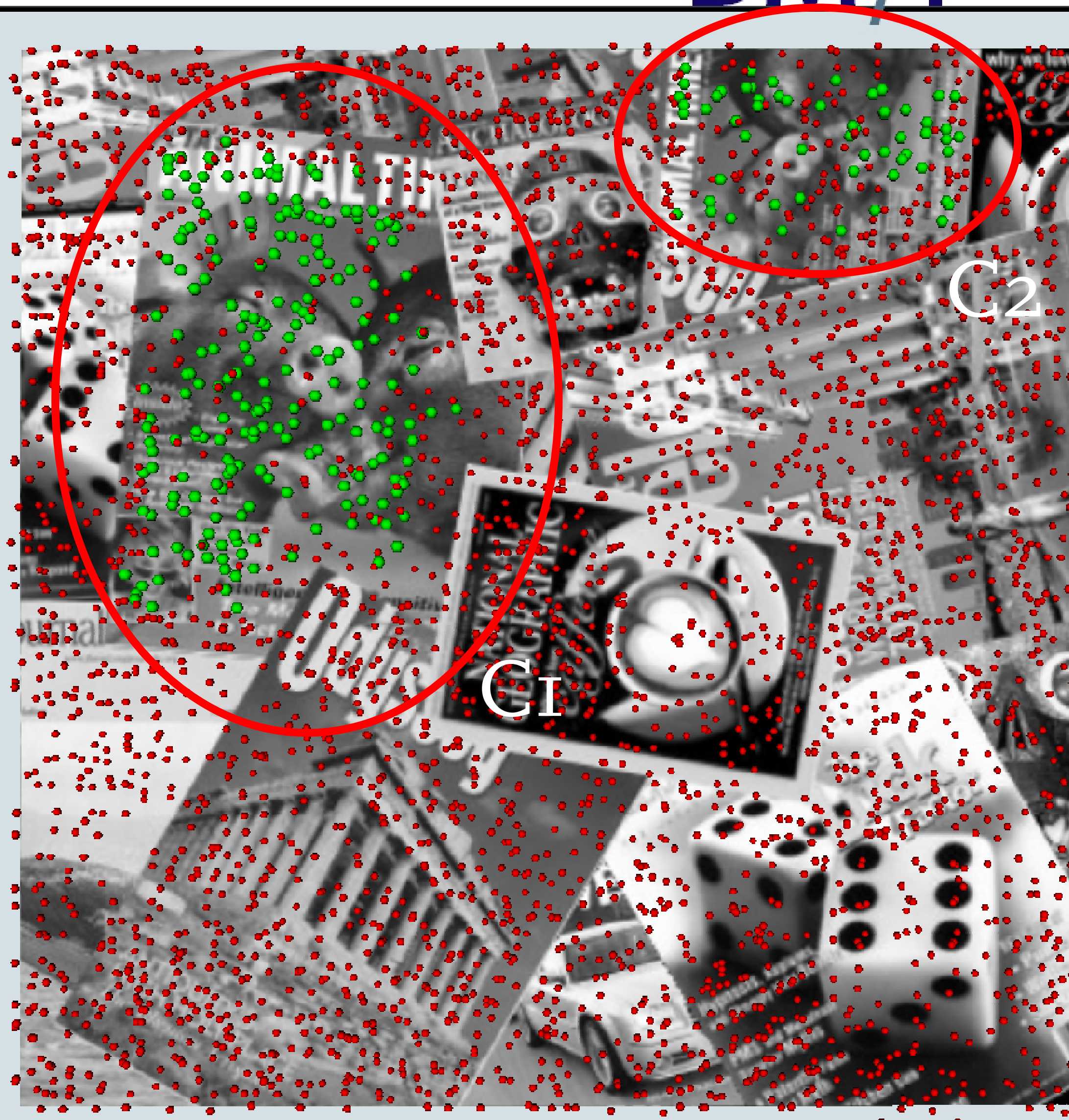
Differential Invariants

- Features are irreducible 3rd order differential invariants.
- These features are rotation and scale invariant.

$$\begin{pmatrix} \sigma\sqrt{L_i L_i / L} \\ \sigma L_{ii}/\sqrt{L_i L_i} \\ \sigma^2 L_{ij} L_{ij} / L_i L_i \\ \sigma L_i L_{ij} L_j / (L_i L_i)^{3/2} \\ \sigma^2 L_{ijk} L_i L_j L_k / (L_i L_i)^2 \\ \sigma^2 \varepsilon_{ij} L_{jkl} L_i L_k L_l / (L_i L_i)^2 \end{pmatrix}$$

In this example we have two clusters of correctly matched points.





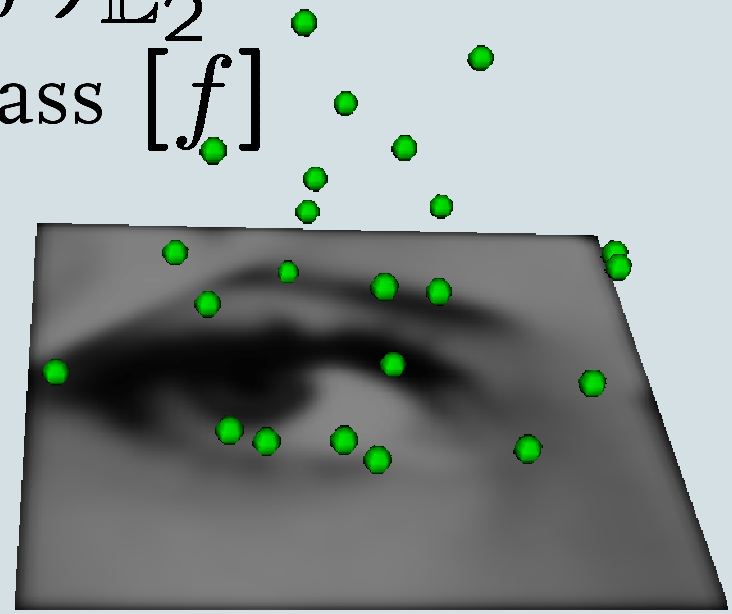
Outline

- Scale Space
- Toppoints
- Image Reconstruction
- Mathematica Implementation

Image Reconstruction

Given features $c_i = (\psi_i, f)_{\mathbb{L}_2}$

Select g from metameric class $[f]$

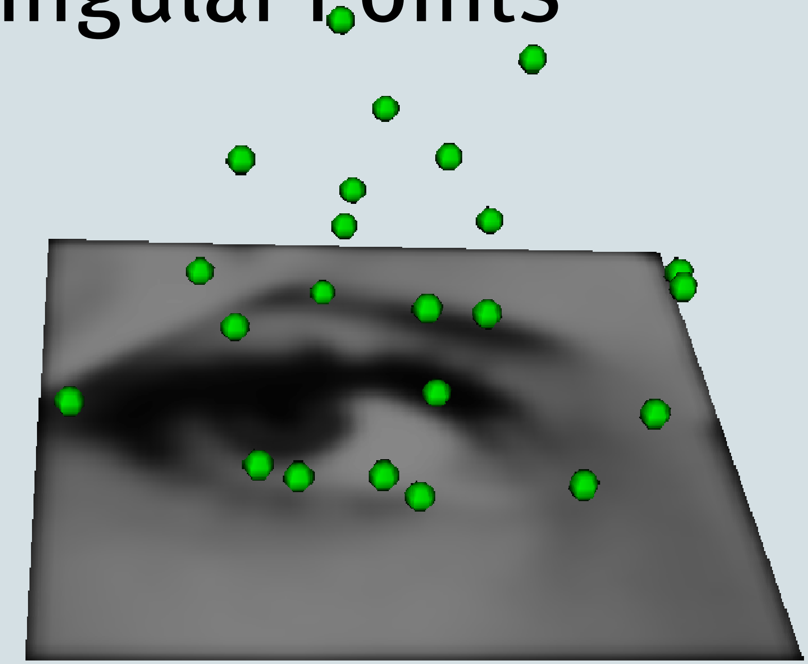


such that (consistent features)

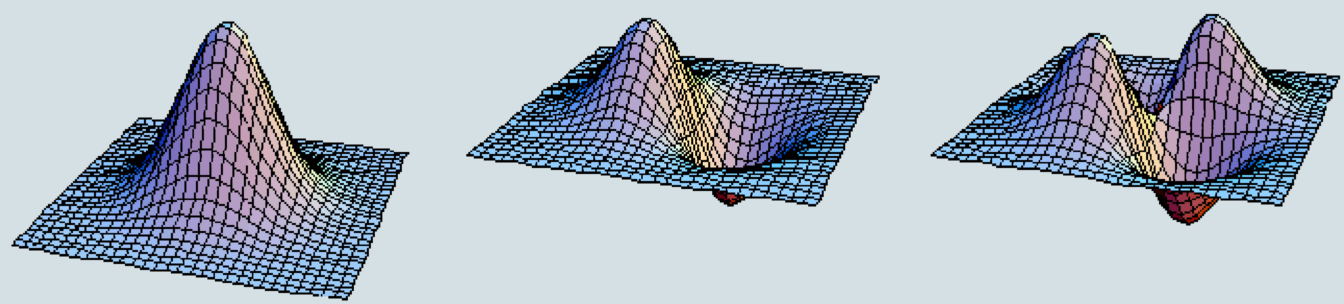
$$(\psi_i, g)_{\mathbb{L}_2} = c_i \ , (i = 1 \ldots N)$$

Reconstruction from Singular Points

Use differential structure
in singular points
as features.

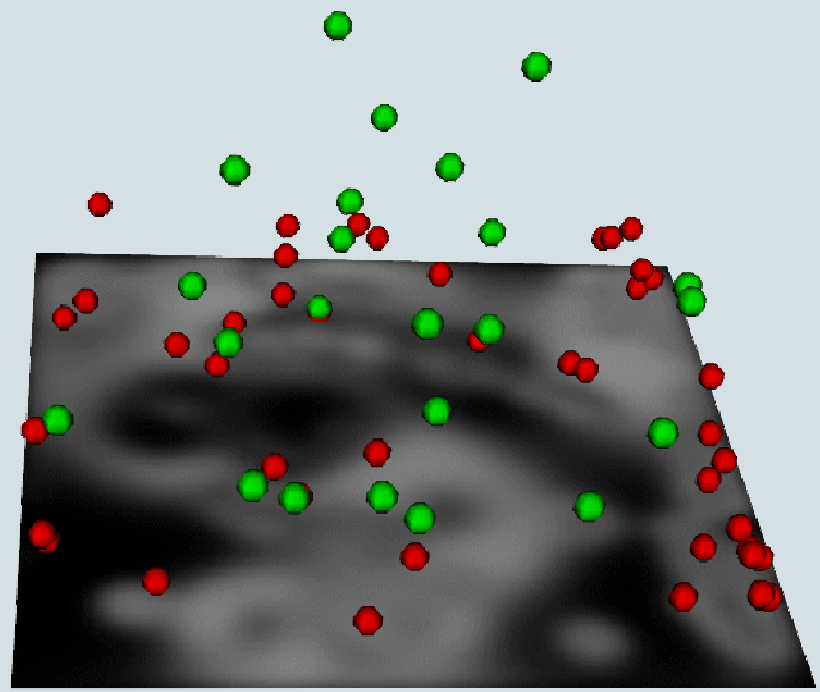


ψ_i =



Variational Approach

$$E(g) = \boxed{\frac{1}{2}(g,g)_{\mathbb{L}_2}} - \boxed{\lambda^i \left((\psi_i, g)_{\mathbb{L}_2} - c_i\right)}$$



Minimisation of $E(g) = \frac{1}{2}(g,g)_A$

under the constraints $((\kappa_i, g)_A - c_i) = 0$

$$(f,g)_A = (f,g)_{\mathbb{L}_2} + (Af, Ag)_{\mathbb{L}_2}$$

The solution is an A-orthogonal projection of f onto κ_i

Generalisation using gelfand triples (R. Duits)

Prior and Dual Filters

$$A = -\gamma\sqrt{-\Delta}$$

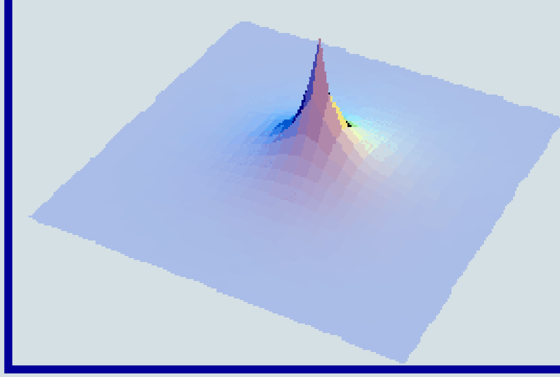
$$(f, g)_A = (f, g)_{\mathbb{L}_2} + (\gamma\nabla f, \gamma\nabla g)_{\mathbb{L}_2}$$

$$(\kappa_i, f)_A = (\psi_i, f)_{\mathbb{L}_2}$$

$$\kappa_i = (I + A^\dagger A)^{-1}\psi_i$$

Reconstruction from Singular Points

This means $\kappa_i = \boxed{\phi_\gamma} * \psi_i$

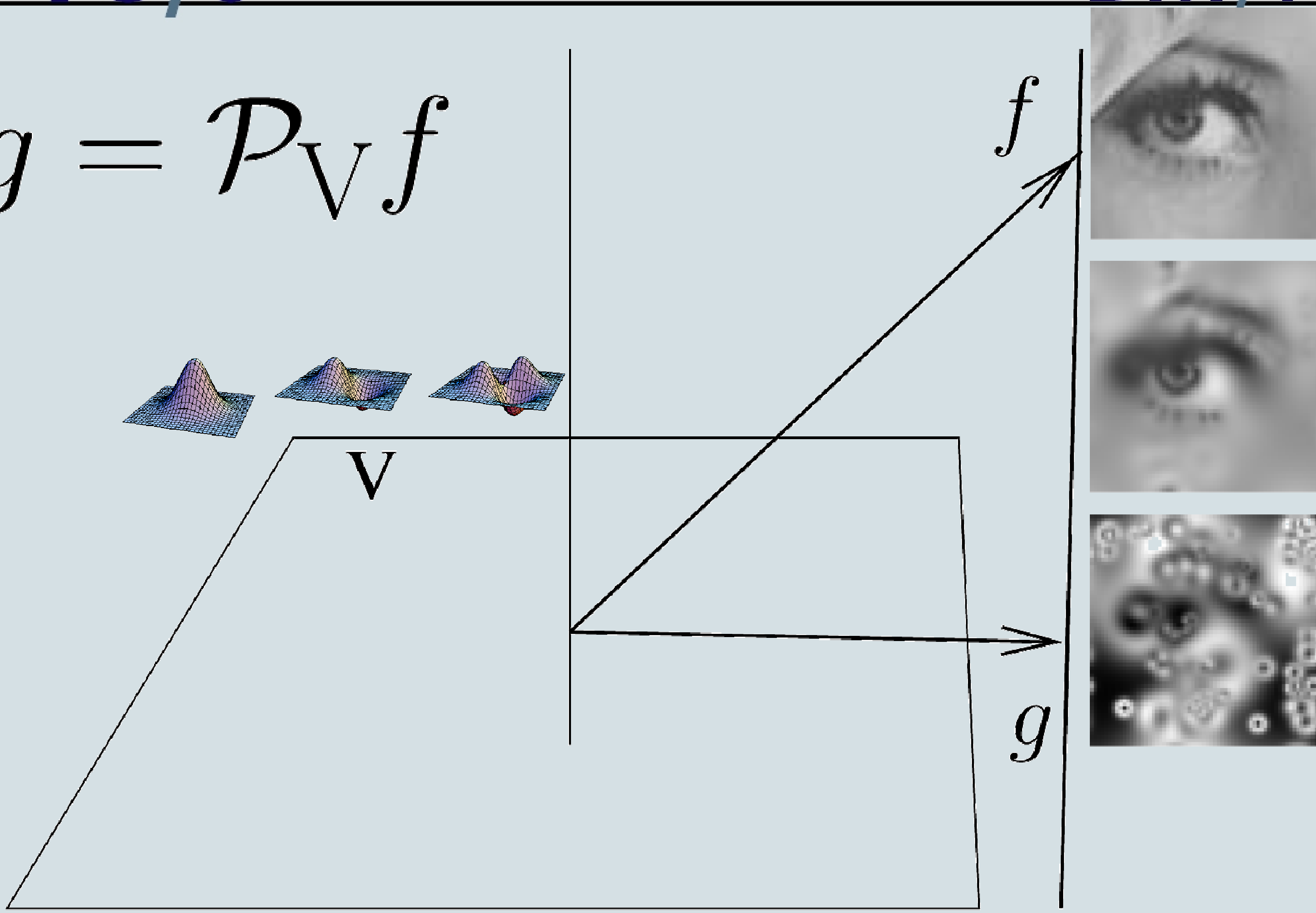


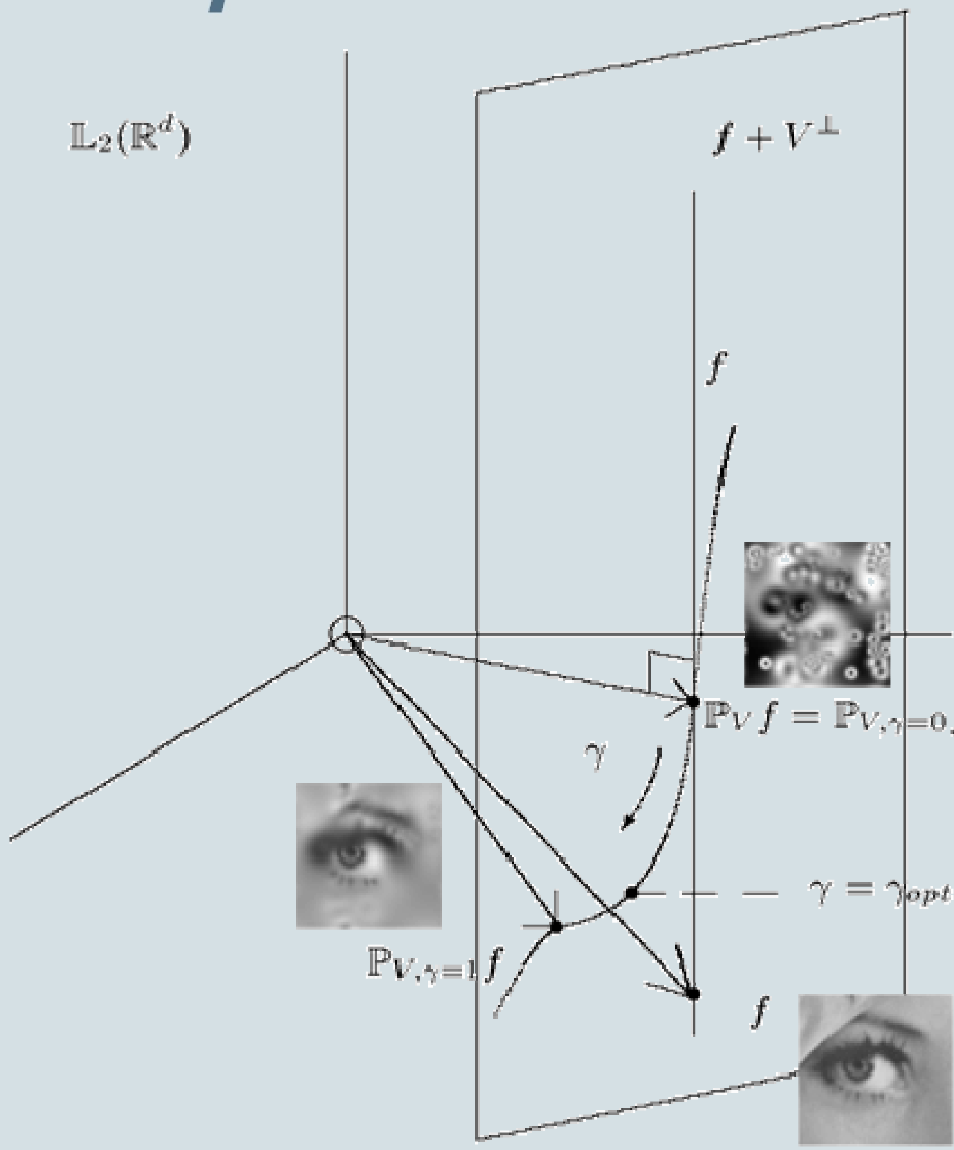
Gramm matrix:

$$G_{ij} = [(\kappa_i, \kappa_j)_A] = [(\phi_\gamma, \psi_i^* * \psi_j)_{\mathbb{L}_2}]$$

Projection: $g = G^{ij} c_j \kappa_i$

$$g = \mathcal{P}_V f$$




$\mathbb{L}_2(\mathbb{R}^d)$
$f + V^{\perp}$
f
$\mathbb{P}_V f = \mathbb{P}_{V,\gamma=0} f$
γ
$\gamma = \gamma_{opt}$
$\mathbb{P}_{V,\gamma=1} f$
f

Outline

- Scale Space
- Toppoints
- Image Reconstruction
- Mathematica Implementation
- Conclusions

Basic implementation

- Build Gramm Matrix
- Inversion of Gramm Matrix
- Building and Sampling Reconstruction

```
makeGramm[kernel_, orders_, points_, {xsize_, ysize_}] := Module[{Φtensor, gramm, signmx, signmy, subΦ, signs},

  subΦ[{st_, dx_, dy_}] :=   submatrix of innerproducts;

  Φtensor =
   Map[
    subΦ,
    Outer[Plus, points, {1, -1, -1} # & /@ points , 1],
    {2}
   ];


  gramm = Chop[Flatten[
     Map[
      Flatten,
      Transpose[Φtensor, {2, 4, 1, 3}],
      {2}
     ],
     1
    ]];

  gramm
 ]
```

Dynamic Programming

```
subΦ[{st_, dx_, dy_}] := If[Negative[dx],

   If[Negative[dy],
    signmx signmy subΦ[{st, -dx, -dy}],
    signmx subΦ[{st, -dx, dy}]
   ],

   If[Negative[dy],
    signmy subΦ[{st, dx, -dy}] ,
    subΦ[{st, dx, dy}] = submatrix of innerproducts;
   ]

  ];
```

Parallel Implementation

```
Φtensor =
  Map[
   subΦ,
   Outer[Plus, points, {1, -1, -1} # & /@ points , 1],
   {2}
  ];
```

```
ExportEnvironment[GaussianDerivativeAt, xsize, ysize,
  signmy, signmx, kernel, orders, subΦ];

Φtensor =
  ParallelMap[
   Map[subΦ, #] &,
   Outer[Plus, points, {1, -1, -1} # & /@ points , 1]
  ];
```

Sampling the Reconstruction

advantage of symbolic power

```
FourierK[gamma_, scale_, order_, {wx_, wy_}, {xi_, yi_}] := Module[{},
  Exp[i (wx xi + wy yi)] (-i wx)^order[[1]] (-i wy)^order[[2]] 1/(1 + gamma^2 (wx^2 + wy^2)) Exp[-scale (wx^2 + wy^2)]/1
]
```

```
FourierReconstructionFunction =
  Compile[
   {{x, _Real}, {y, _Real}},
   Evaluate[rf]
  ];
```

Parallel Sampling

```
ParallelSampleReconstruction[{xsize_, ysize_}] := Module[{x, y, FourierimageData, image, newfeaturepoints, n, wlist},

  wlist = list of frequencies;

  ExportEnvironment[FourierReconstructionFunction];
  FourierimageData = ParallelMap[Apply[FourierReconstructionFunction, #] &, wlist, {2}];

  image = Re[ InverseFourier[ FourierimageData , FourierParameters → {1, 1}]][[Range[xsize], Range[ysize]]];

  image
 ]
```

Mathematica Demo 1

MATHEMATICA® 5

Computing cluster

Mathlink for better performance

(sometimes)

```
:Begin:
:Function:MLSobolevReconstruction
:Pattern:MLSobolevReconstruction[X__?(VectorQ[#1,NumberQ]&), Y__?(VectorQ[#1,NumberQ]&),...
:Arguments:{X,Y,OX,OY,T,F,gridsize,gamma}
:ArgumentTypes:{RealList,RealList,IntegerList,...
:ReturnType:Manual
:End:
```

```
#include<math.h>...
#include "mathlink.h"
#include "your own stuff.h"

void MLSobolevReconstruction(double*X,long Xcount,double*Y,long Ycount,...
{
    malloc();
    do some computations;
    MLPutRealList(stdlink,data,size);
    free();
}
```

Mathematica Demo 2

MATHEMATICA® 5

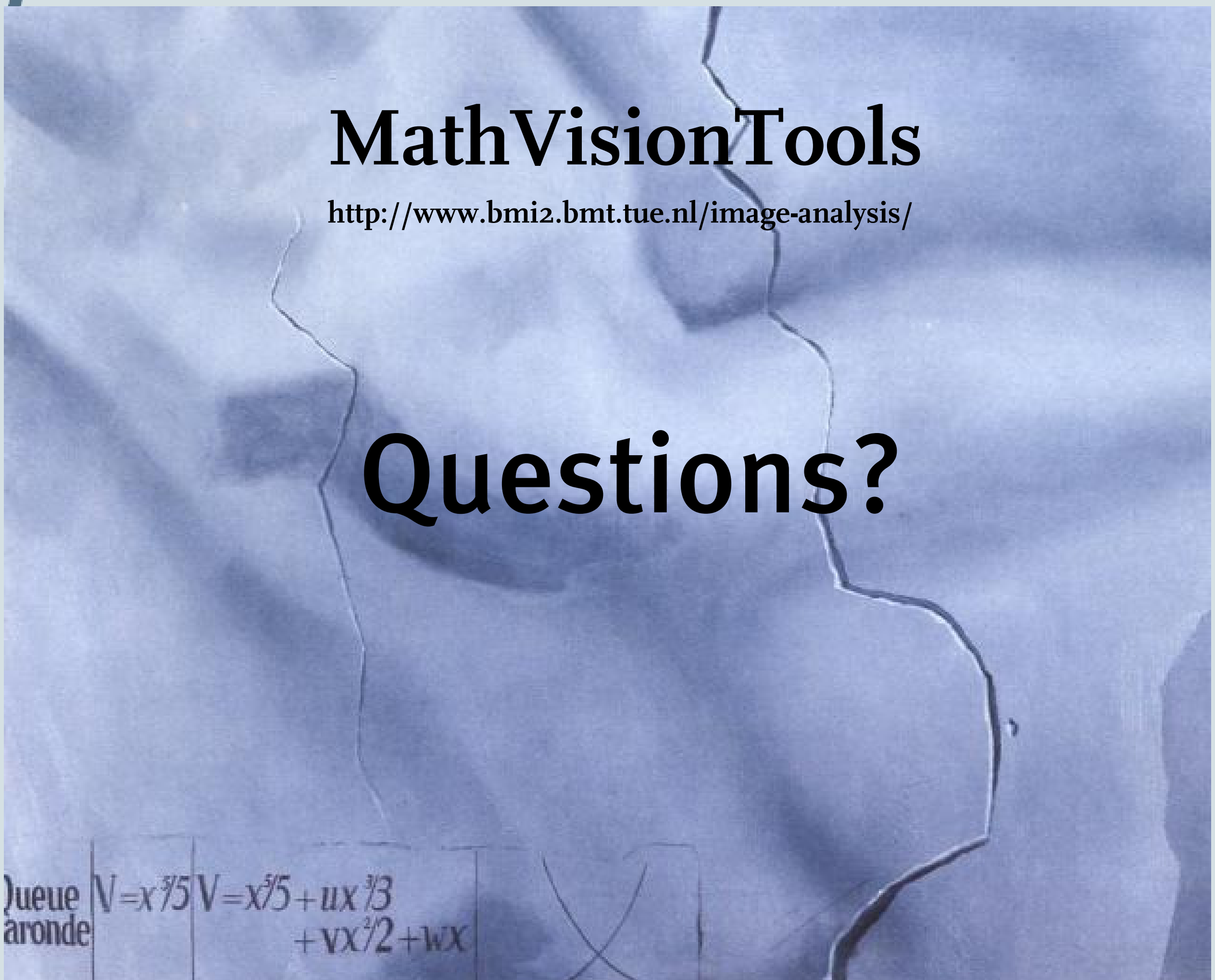
Computing cluster

(State of the) Art

MathVisionTools

http://www.bmi2.bmt.tue.nl/image-analysis/

Questions?



Topological Abduction of Europe - Homage to Rene Thom
Salvador Dali

Acknowledgements

- Bart ter Haar Romeny
- Evguenia Balmachnova
- Remco Duits
- Luc Florack
- Frans Kanters
- Bram Platel