Image enhancement

Computer Vision, Lecture 15

Michael Felsberg
Computer Vision Laboratory
Department of Electrical Engineering

LINKÖPINGS UNIVERSITET

Why image enhancement?

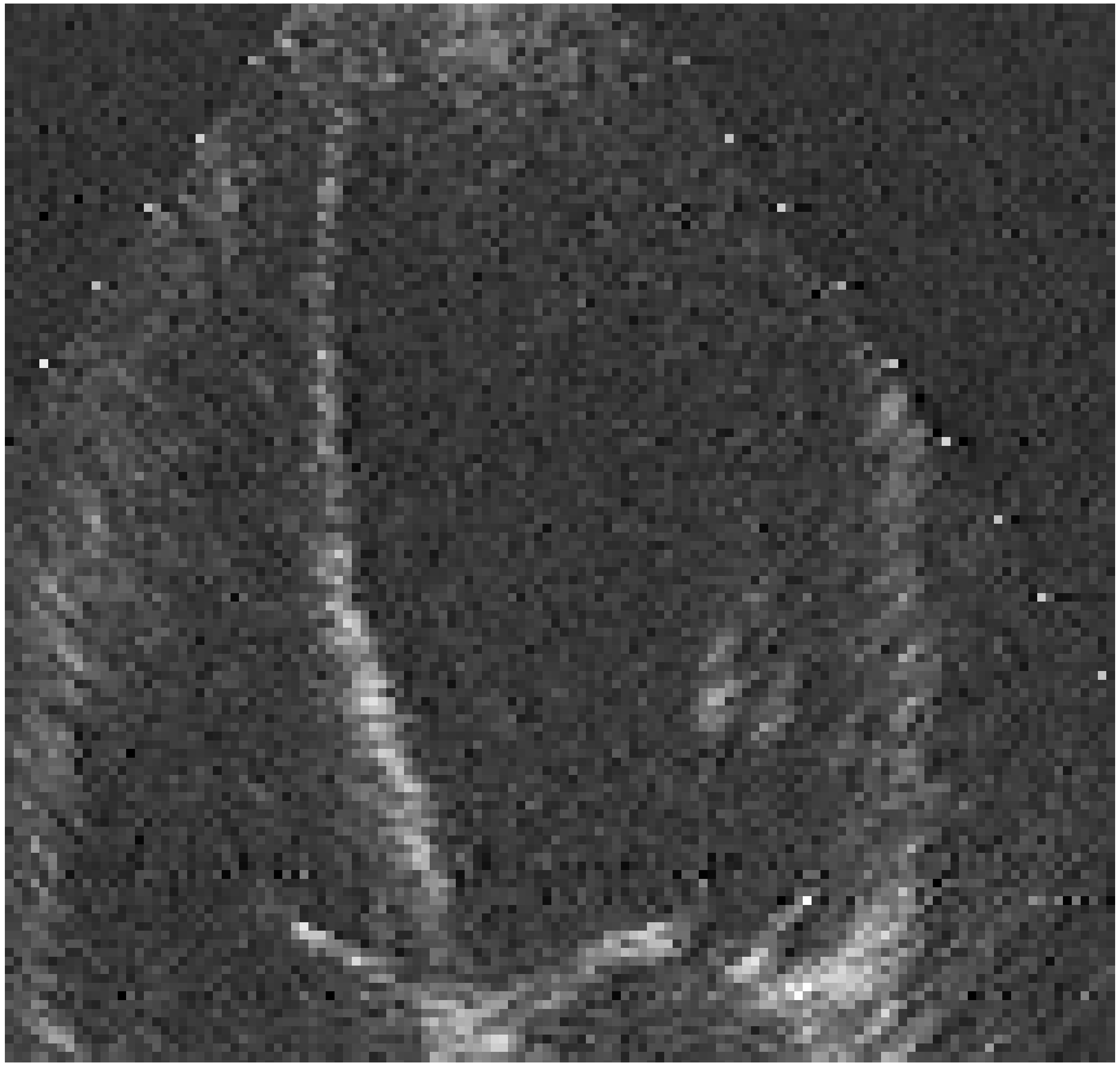
- Example of artifacts caused by image encoding

Why image enhancement?


Example of motion blur

Why image enhancement?

- Example of an image with sensor noise
  - ultrasound image of a beating heart



Why image enhancement?

- IR-image
  - fixed pattern noise = spatial variations in gain and offset
  - Possibly even variations over time!
  - Hot/dead pixels

- A digital camera with short exposure time
  - Shot noise (photon noise)

Methods for image enhancement

- Inverse filtering: the distortion process is modeled and estimated (e.g. motion blur) and the *inverse* process is applied to the image

- Image restoration: an *objective* quality (e.g. sharpness) is estimated in the image. The image is modified to increase the quality

- Image enhancement: modify the image to improve the visual quality, often with a subjective criteria

Additive noise

- Some types of image distortion can be described as
  - Noise added on each pixel intensity
  - The noise has the identical distribution and is independent at each pixel (i.i.d.)
- Not all type of image distortion are of this type:
  - Multiplicative noise
  - Data dependent noise
  - Position dependent
- The methods discussed initially assume additive i.i.d.-noise

What about pixel shot noise?

Removing additive noise

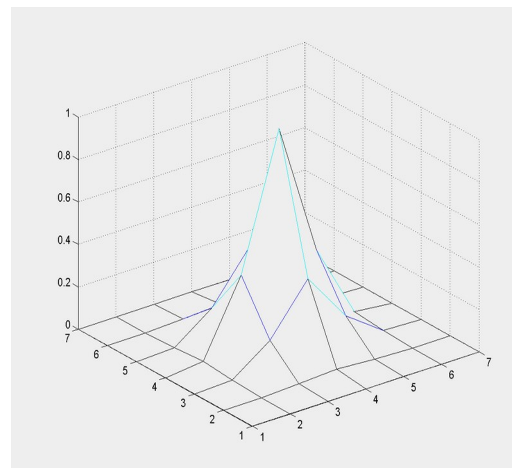
- Image noise typically contains higher frequencies than images generally do
  => a low-pass filter can reduce the noise

- BUT: we also remove high-frequency signal components, e.g. at edges and lines

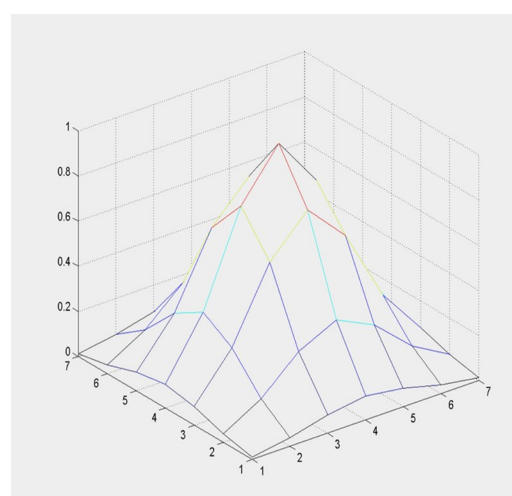
- HOWEVER: A low-pass filter works in regions without edges and lines (ergodicity)

Example: LP filter

Image with some noise



Filter, σ = 1



Filter, σ = 2

Basic idea

The problem of low-pass filters is that we apply the same filter on the whole image

We need a filter that locally adapts to the image structures

A space-variant filter

Ordinary filtering / convolution

- Ordinary filtering can be described as a convolution of the signal f and the filter g:

$$h(\mathbf{x}) = (f * g)(\mathbf{x}) = \int f(\mathbf{x} - \mathbf{y}) g(\mathbf{y})\, d\mathbf{y}$$

For each **x**, we compute the integral between the filter *g* and a shifted signal *f*

Adaptive filtering

- If we apply an adaptive (or position dependent, or space-variant) filter $g_{\mathbf{x}}$, the operation cannot be expressed as a convolution, but instead as

$$h(\mathbf{x}) = \int f(\mathbf{x}-\mathbf{y}) g_{\mathbf{x}}(\mathbf{y})\, d\mathbf{y}$$

For each **x**, we compute the integral between a shifted signal *f* and the filter $g_{\mathbf{x}}$ where the filter depends on **x**

Scale space recap (from lecture 2)

- The linear Gaussian scale space related to the image f is a family of images $L(x,y;s)$

$$L(x, y; s) = (g_s * f)(x, y)$$

Convolution over (x,y) only!

parameterized by the scale parameter s, where

$$g_s(x, y) = \frac{1}{2\pi s} e^{-\frac{x^2+y^2}{2s}}$$

A Gaussian LP-filter with $\sigma^2 = s$

Note: $g_s(x,y) = \delta(x,y)$ for $s = 0$

Scale space recap (from lecture 2)

- $L(x,y;s)$ can also be seen as the solution to the PDE

$$\frac{\partial}{\partial s} L = \frac{1}{2} \nabla^2 L$$

$$\frac{\partial}{\partial s} L = \frac{1}{2} \left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} \right) L$$

Left hand side:
the change in *L*
at (*x*,*y*) between
s and *s*+∂*s*

The diffusion equation

Example:

L = temperature

s = time

with boundary condition $L(x,y;0) = f(x,y)$

Repetition: Vector Analysis

- Nabla operator $$\nabla = \begin{bmatrix} \partial_x \\ \partial_y \end{bmatrix} = \begin{bmatrix} \frac{\partial}{\partial x} \\ \frac{\partial}{\partial y} \end{bmatrix}$$
- On a scalar function $$\nabla f = \operatorname{grad} f = \begin{bmatrix} \partial_x f \\ \partial_y f \end{bmatrix} = \begin{bmatrix} f_x \\ f_y \end{bmatrix}$$
- On a vector field $$\langle \nabla | \mathbf{f} \rangle = \nabla^T \mathbf{f} = \operatorname{div} \mathbf{f} = \partial_x f_1 + \partial_y f_2$$
- Laplace operator $$\Delta = \nabla^2 = \langle \nabla | \nabla \rangle = \operatorname{div} \operatorname{grad} = \partial_x^2 + \partial_y^2$$
  note: $$\partial_x^2 f = f_{xx} \neq f_x^2$$

Enhancement based on linear (homogeneous) diffusion

- This means that $L(x,y;s)$ is an LP-filtered version of $f(x,y)$ for $s > 0$.
- The larger s is, the more LP-filtered is f
  - High-frequency noise will be removed for larger s
- Also high-frequency image components (e.g. edges) will be removed
- We need to control the diffusion process such that edges remain - How?

Step 1

- Modify the PDE by introducing a parameter μ:

$$\frac{\partial}{\partial s} L = \frac{\mu}{2} \nabla^2 L$$

μ can be seen as a "diffusion speed":

Small μ: the diffusion process is slow when s increases

Large μ: the diffusion process is fast when s increases

- This PDE is solved by

$$L(x, y; s) = (g_s * f)(x, y)$$

Same as before

$$g_s(x, y) = \frac{1}{2\pi \mu s} e^{-\frac{x^2 + y^2}{2 \mu s}}$$

Slightly different

Step 2

- We want the image content to control μ
  - In flat regions: fast diffusion (large μ)
  - In non-flat region: slow diffusion (small μ)

- We need to do *space-variant* diffusion
  - μ is a function of position (x,y)

We will introduce another space- variant filter g_x in adaptive filtering

Inhomogeneous diffusion

- Perona & Malik suggested to use

$$\mu(x, y) = \frac{1}{1 + |\nabla f|^2 / \lambda^2}$$

where ∇f is the image gradient at (x,y)
and λ is a fixed parameter

  - Close to edges: $|\nabla f|$ is large) μ is small
  - In flat regions: $|\nabla f|$ is small) μ is large

Inhomogeneous diffusion

Funtional View: Variational Methods

- Minimize the local integral of a Lagrange function $L(f, f_x, f_y, x, y)$

$$\varepsilon(f) = \int_\Omega L(f, \nabla f, \mathbf{x})\, d\mathbf{x}$$

- gives the Euler-Lagrange equation on Ω

$$L_f - \operatorname{div} L_{\nabla f} = L_f - \partial_x L_{f_x} - \partial_y L_{f_y} = 0 \qquad \forall x, y$$

- if we require $\langle \nabla f | \mathbf{n} \rangle = 0 \quad \text{on } \partial\Omega$

Variational View: linear denoising

- Assume $\mathbf{f}_0 = \mathbf{f} + \text{noise}$. Minimizing

$$\varepsilon(f) = \frac{1}{2}\int_\Omega \underbrace{(f - f_0)^2 + \lambda(f_x^2 + f_y^2)}_{L(f, f_x, f_y, x, y)} \, dx \, dy$$

- Gives the Euler-Lagrange equation (note: $L_f = f - f_0,\ L_{f_x} = \lambda f_x,\ L_{f_y} = \lambda f_y$)

$$\underbrace{f - f_0}_{L_f} - \underbrace{\lambda \Delta f}_{\operatorname{div}(L_{f_x}, L_{f_y})} = 0$$

$$(\partial_x f_x + \partial_y f_y) = \Delta f$$

Non-Linear case

- Minimizing

$$\varepsilon(f) = \int_\Omega \frac{1}{2}(f - f_0)^2 + \lambda\Psi(|\nabla f|)\, dx\, dy$$

- Gives the Euler-Lagrange equation

$$f - f_0 - \lambda \operatorname{div}\left(\frac{\Psi'(|\nabla f|)}{|\nabla f|}\nabla f\right) = 0$$

where we exploited

$$\partial_x \frac{\Psi'(|\nabla f|)}{|\nabla f|} f_x + \partial_y \frac{\Psi'(|\nabla f|)}{|\nabla f|} f_y = \operatorname{div}\left(\frac{\Psi'(|\nabla f|)}{|\nabla f|}\nabla f\right)$$

Exemple: Perona-Malik Flow

- Special cases:

$$\Psi(|\nabla f|) = -K^2/2 \ \cdot \exp(-|\nabla f|^2/K^2)$$

$$\Rightarrow \Psi'(|\nabla f|) = |\nabla f| \exp(-|\nabla f|^2/K^2)$$

$$\Psi(|\nabla f|) = K^2/2 \ \cdot \log(K^2 + |\nabla f|^2)$$

$$\Rightarrow \Psi'(|\nabla f|) = |\nabla f|(1 + |\nabla f|^2/K^2)^{-1}$$

- Such that gradient descent gives Perona-Malik Flow

$$f^{(s+1)} = f^{(s)} + \alpha \operatorname{div}\left(\frac{\Psi'(|\nabla f^{(s)}|)}{|\nabla f^{(s)}|} \nabla f^{(s)}\right)$$

Total Variation (TV) / ROF

- Minimizing $$\min_f \frac{\|f-f_0\|^2}{2\lambda} + \sum_{i,j} |(\nabla f)_{i,j}|$$

  means $$\Psi() = \mathrm{Id}() \Rightarrow \Psi'() = 1$$
- Stationary point
$$f - f_0 - \lambda \operatorname{div}\left(\frac{1}{|\nabla f|}\nabla f\right) = 0$$

  and after some calculations
$$f - f_0 - \lambda \frac{f_{xx} f_y^2 - 2 f_{xy} f_x f_y + f_{yy} f_x^2}{|\nabla f|^3} = 0$$

Efficient TV Algorithms

- In 1D: Chambolle's algorithm (JMIV, 2004)
- In 2D:
  - Alternating direction method of multipliers (ADMM, variant of augmented Lagrangian): Split Bregman by Goldstein & Osher (SIAM 2009)
  - Based on threshold Landweber: Fast Iterative Shrinkage-Thresholding Algorithm (FISTA) by Beck & Teboulle (SIAM 2009)
  - Based on Lagrange multipliers: Primal Dual Algorithm by Chambolle & Pock (JMIV 2011)

Demo: TV Image Denoising

- Paramters: α =0.0005, λ=0.5, noise(0,0.001), TVdemo_script.m

Inhomogeneous diffusion

- Noise is effectively removed in flat regions
- Edges are preserved

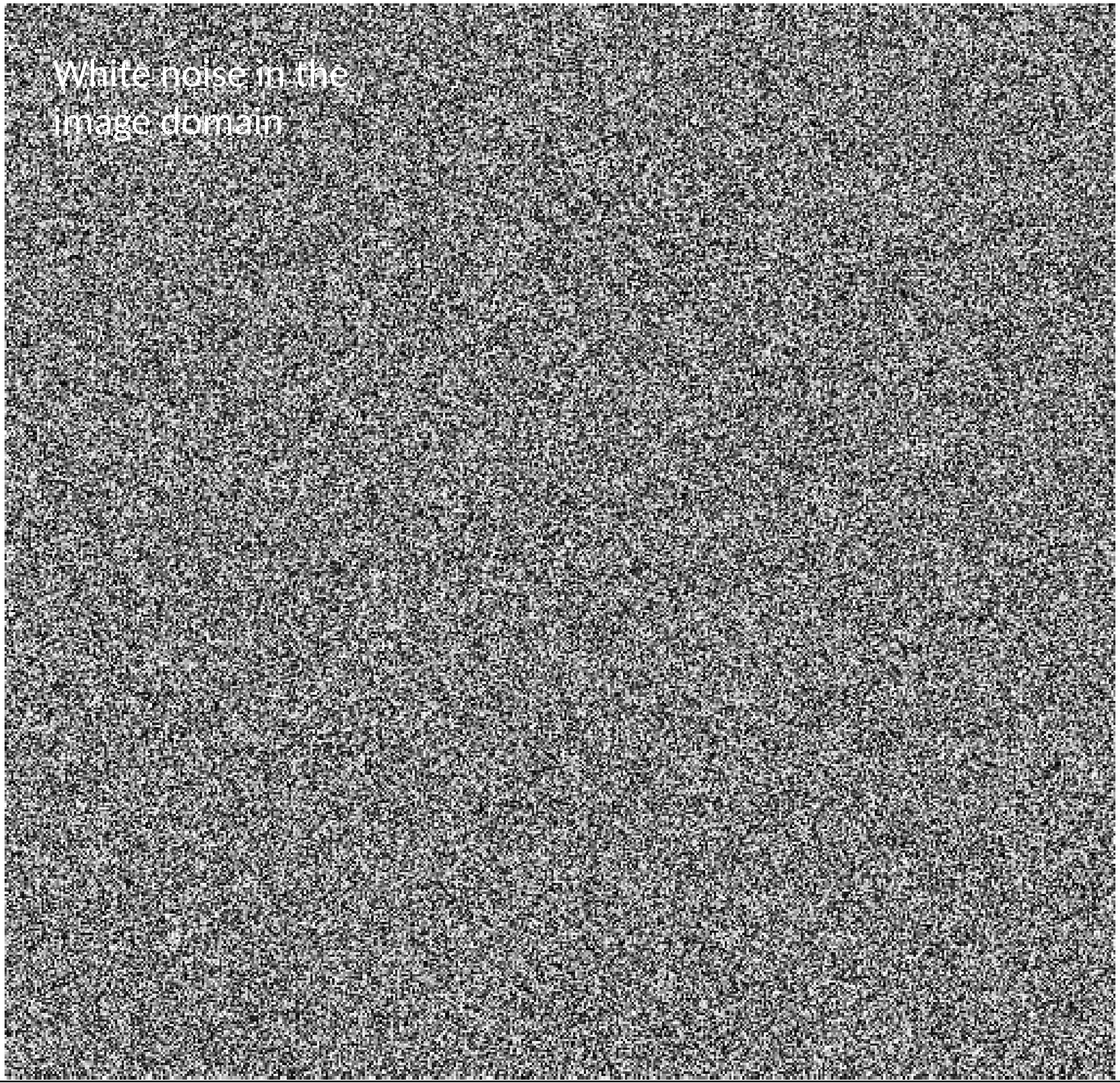
- Noise is preserved close to edges

We want to be able to LP-filter along but not across edges

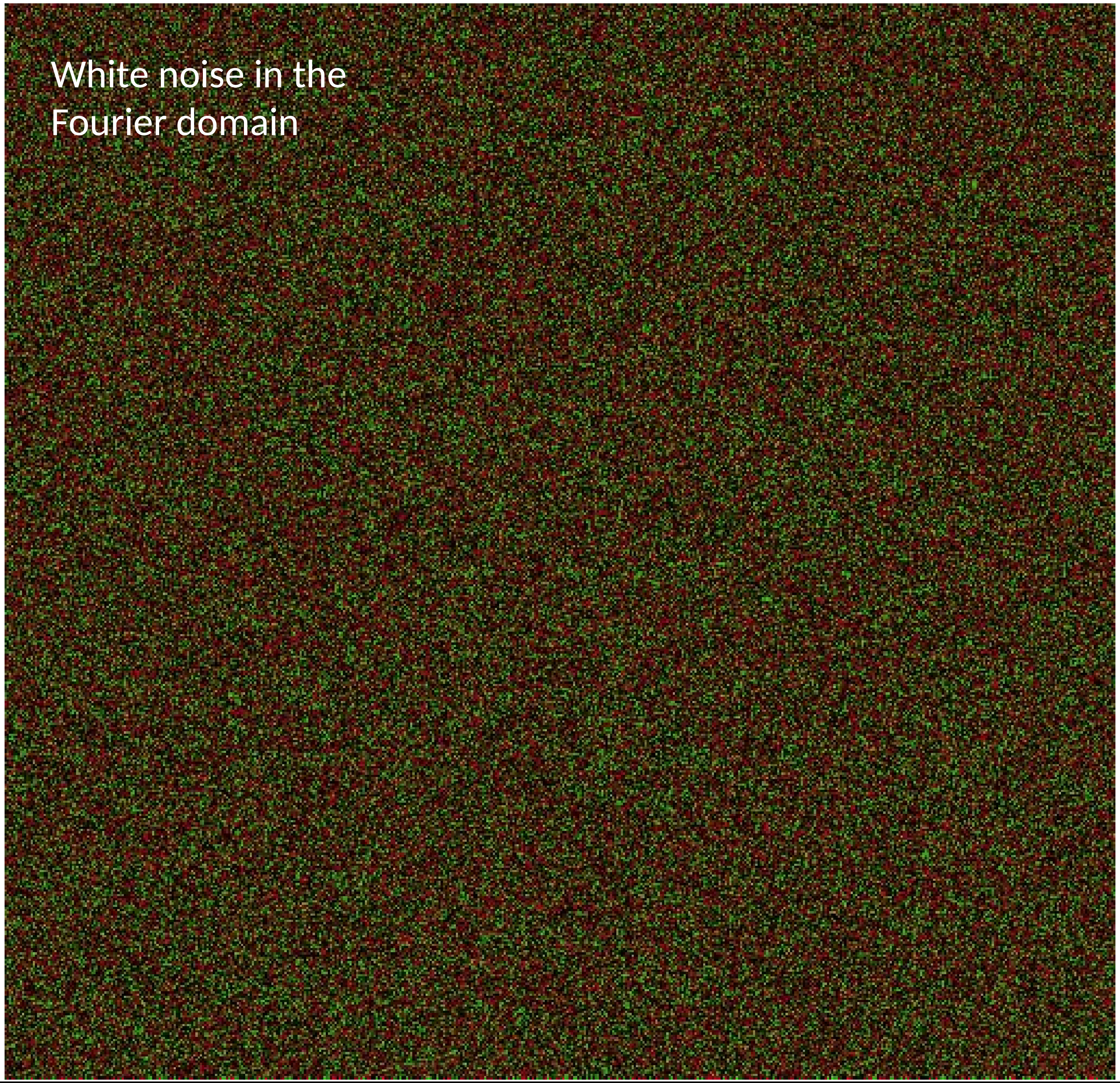
Orientation-selective g_x

- If the signal is ≈ i1D the filter can maintain the signal by reducing the frequency components orthogonal to the local structure
- The human visual system is less sensitive to noise along linear structures than to noise in the orthogonal direction
- Results in good subjective improvement of image quality

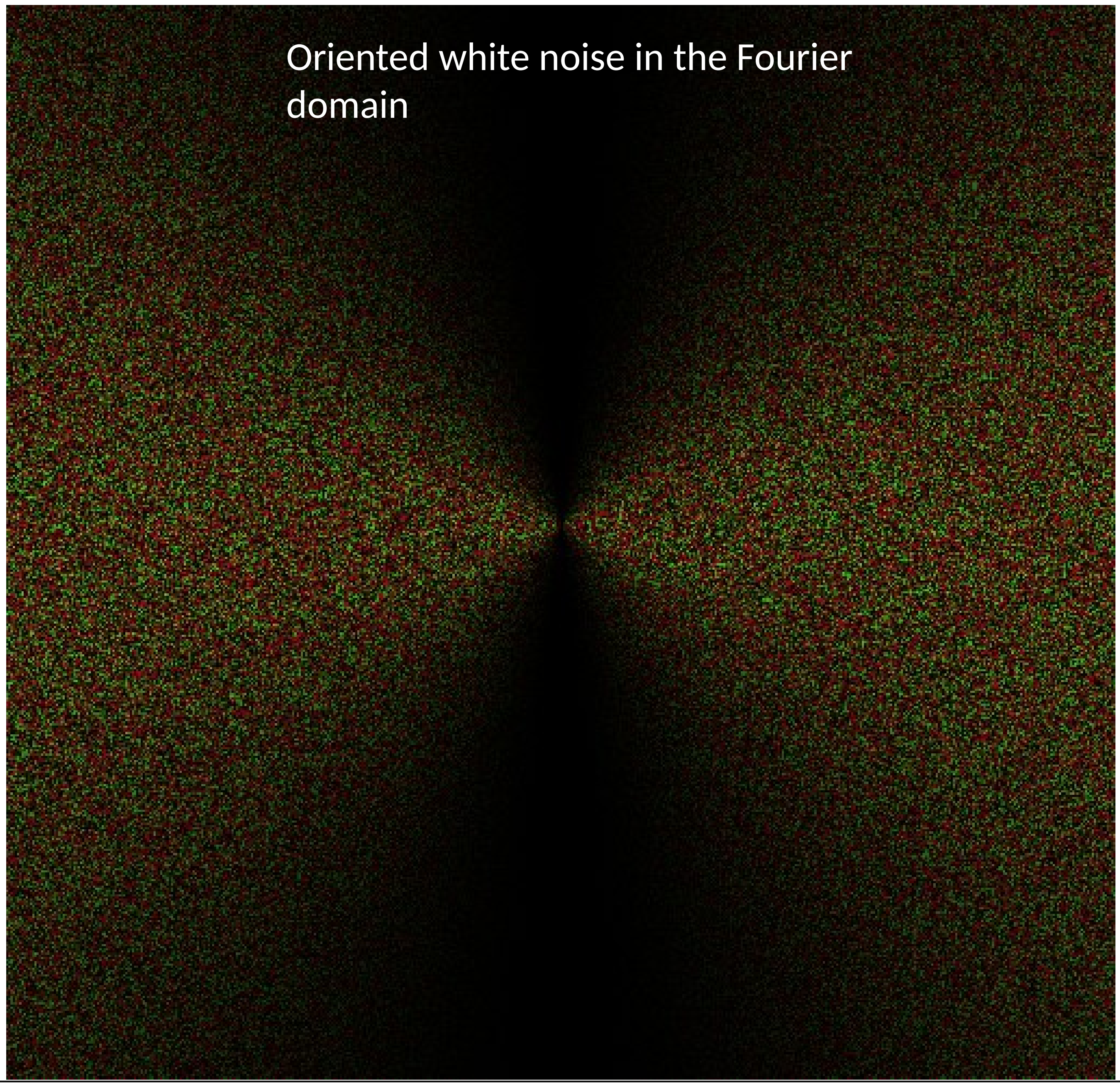
Oriented noise



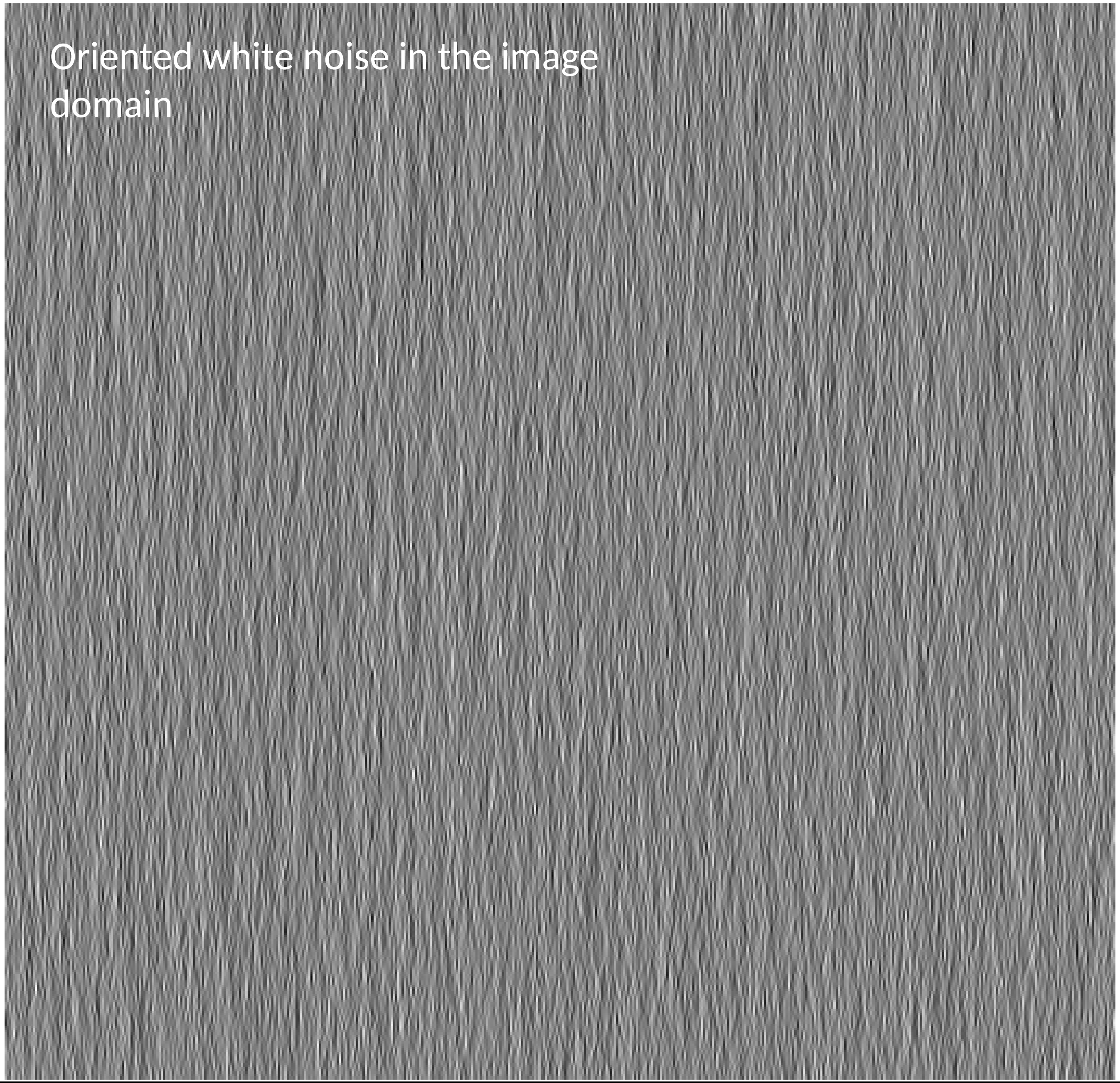
Oriented noise

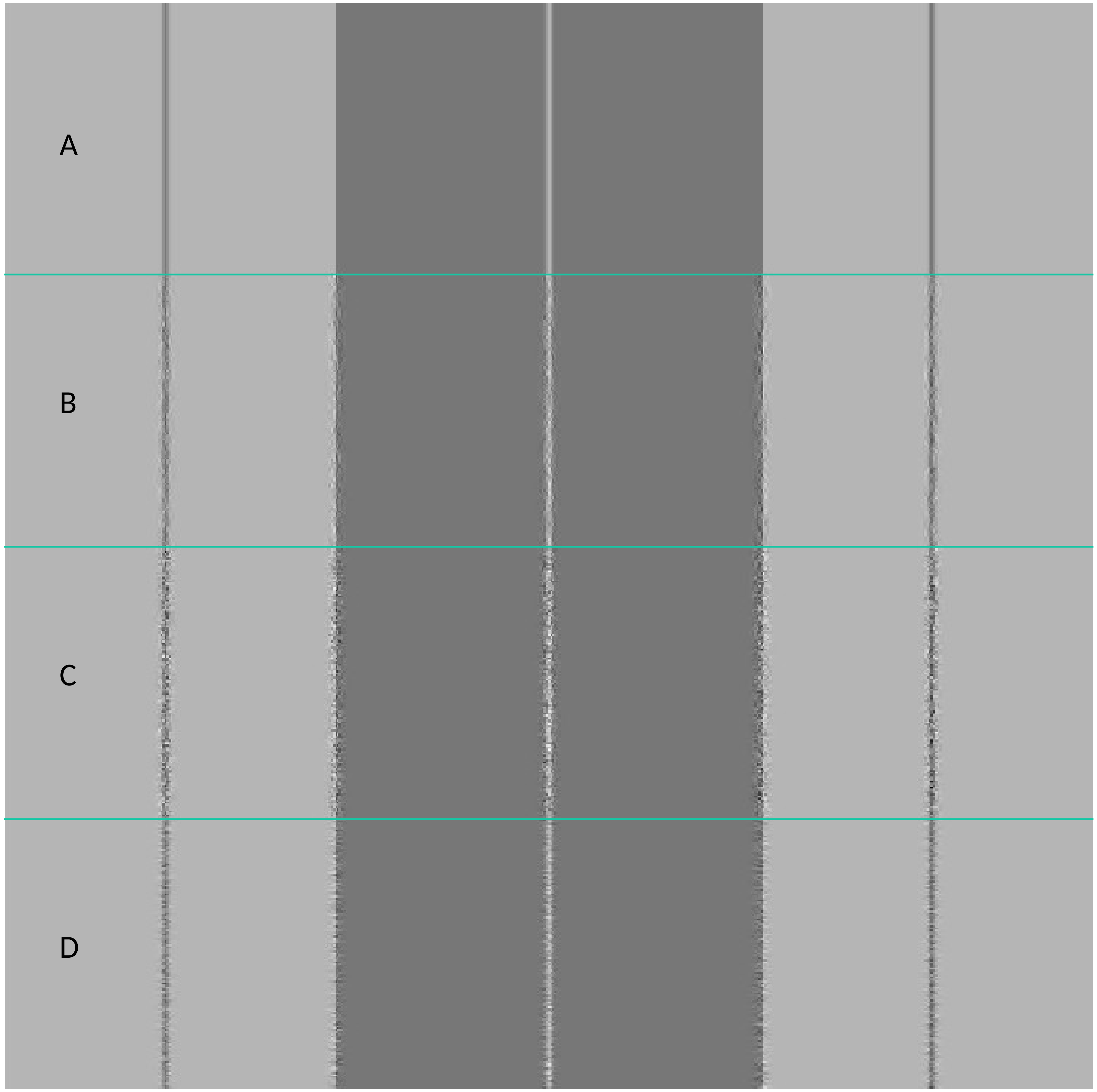


Oriented noise



Oriented noise





Oriented noise

Edges and lines

A. Without noise

B. With oriented noise along

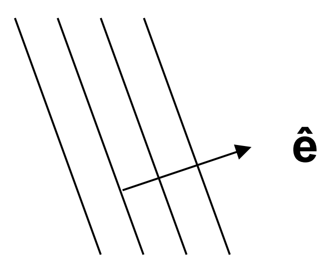
C. With isotropic noise

D. With oriented noise across

Local structure information

- We compute the local orientation tensor $\mathbf{T}(\mathbf{x})$ at all points $\mathbf{x}$ to control / steer $g_{\mathbf{x}}$
- At a point $\mathbf{x}$ that lies in a locally i1D region, we obtain

$$\mathbf{T}(\mathbf{x}) = A\hat{\mathbf{e}}\hat{\mathbf{e}}^T$$



ê is normal to the linear structure

Step 3 (making the PDE anisotropic)

- The previous PDEs are all isotropic

  => The resulting filter g is isotropic
- The Perona-Malik flow can be rewritten:

$$\boxed{\frac{\partial}{\partial s} L = \frac{\mu}{2} \nabla^2 L = \frac{1}{2} \operatorname{div}(\mu \operatorname{grad} L)}$$

Gradient of *L*,
a 2D vector field

Divergence of (…)
maps 2D vector field to scalar field

Step 3

- Change μ from a scalar to a 2x2 symmetric matrix $\mathbf{D}$

$$\frac{\partial}{\partial s} L = \frac{1}{2} \operatorname{div}(\mathbf{D} \operatorname{grad} L)$$

- The solution is now given by

$$L(\mathbf{x}; s) = (g_s * f)(\mathbf{x})$$

<= Same as before

$$g_s(\mathbf{x}) = \frac{1}{2\pi \det(\mathbf{D})^{1/2} s} e^{-\frac{1}{2s} \mathbf{x}^T \mathbf{D}^{-1} \mathbf{x}}$$

Ansiotropic diffusion

- The filter g is now anisotropic, i.e., not necessary circular symmetric
- The shape of g depends on $\mathbf{D}$
- $\mathbf{D}$ is called a *diffusion tensor*
  - Can be given a physical interpretation, e.g. for anisotropic heat diffusion

The diffusion tensor

- Since $\mathbf{D}$ is symmetric 2x2:

$$\boxed{\mathbf{D} = \alpha_1 \mathbf{e}_1 \mathbf{e}_1^T + \alpha_2 \mathbf{e}_2 \mathbf{e}_2^T}$$

where α_1, α_2 are the eigenvalues of $\mathbf{D}$, and $\mathbf{e}_1$ and $\mathbf{e}_2$ are corresponding eigenvectors

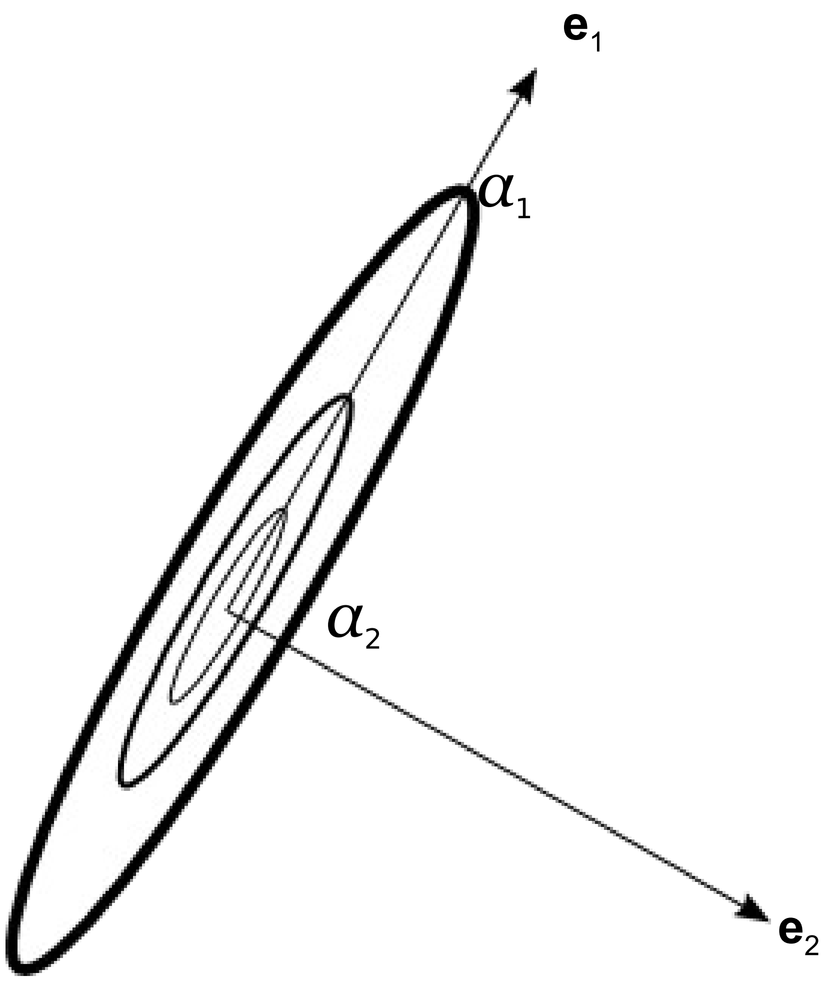
$\mathbf{e}_1$ and $\mathbf{e}_2$ form an ON-basis

The filter g

- The corresponding shape of g is given by

The width of the filter in direction $\mathbf{e}_k$ is given by α_k

Iso-curves for g =>



Step 4

- We want g to be narrow across edges and wide along edges
- This means: $\mathbf{D}$ should depend on (x,y)
  - A space-variant anisotropic diffusion

- This is referred to as *anisotropic diffusion* in the literature
- Introduced by Weickert

Anisotropic diffusion

- Information about edges and their orientation can be provided by an orientation tensor, e.g., the structure tensor $\mathbf{T}$ in terms of its eigenvalues λ_1, λ_2

- However:
  - We want α_k to be close to 0 when λ_k is large
  - We want α_k to be close to 1 when λ_k is close to 0

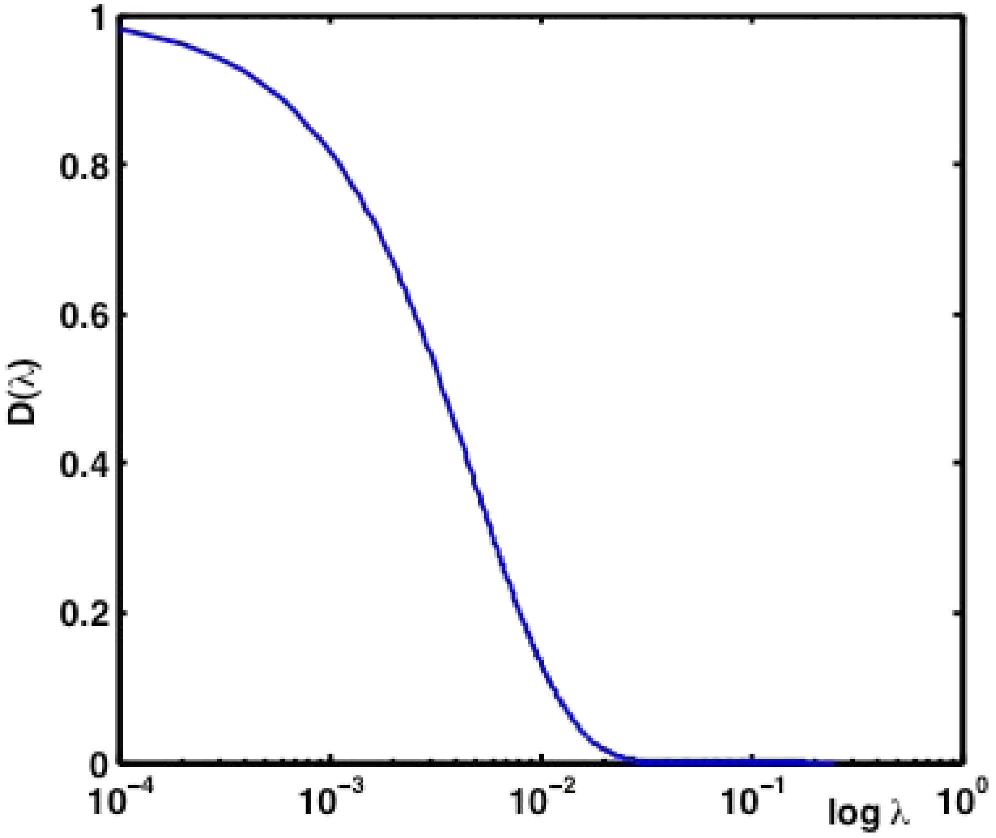
From **T** to **D**

- The diffusion tensor **D** is obtained from the orientation tensor **T** by modifying the eigenvalues and keeping the eigenvectors, e.g.

$$\alpha_k = \exp(-\lambda_k / m)$$

For example

m is a control parameter



Anisotropic diffusion: summary

1. At all points:
   1. compute a local orientation tensor $\mathbf{T}(\mathbf{x})$
   2. compute $\mathbf{D}(\mathbf{x})$ from $\mathbf{T}(\mathbf{x})$
2. Apply anisotropic diffusion onto the image by locally iterating

$$\frac{\partial}{\partial s} L = \frac{1}{2} \mathrm{div}(\mathbf{D}\, \mathrm{grad}\, L)$$

Right hand side:
can be computed
locally at each
point (x,y)

This defines how scale space level
$L(x,y;s+\partial s)$ is generated from $L(x,y;s)$

Implementation aspects

- The anisotropic diffusion iterations can be done with a constant diffusion tensor field $\mathbf{D}(\mathbf{x})$, computed once from the original image (faster)
- Alternatively: re-compute $\mathbf{D}(\mathbf{x})$ between every iteration (slower)

Simplification

- We assume $\mathbf{D}$ to have a slow variation with respect to $\mathbf{x}$
- This means (see [EDUPACK – ORIENTATION (22)])

$$\frac{\partial}{\partial s}L = \frac{1}{2}\nabla^T \mathbf{D} \nabla L \approx \frac{1}{2}\langle \mathbf{D} | \nabla\nabla^T \rangle L = \frac{1}{2}\mathrm{tr}[\mathbf{D}\,(\mathbf{H}L)]$$

The Hessian of *L* = second order derivatives of *L*

$$\mathbf{H}L = \begin{pmatrix} \frac{\partial^2}{\partial x^2}L & \frac{\partial^2}{\partial x \partial y}L \\ \frac{\partial^2}{\partial x \partial y}L & \frac{\partial^2}{\partial y^2}L \end{pmatrix}$$

Numerical implementation

- Several numerical schemes for implementing anisotropic diffusion exist
- Simplest one:
  - Replace all partial derivatives with finite differences (see also lecture 14)

$$L(x,y;s+\Delta s) = L(x,y;s) + \frac{\Delta s}{2}\mathrm{tr}[\mathbf{D}\,(\mathbf{H}L)]$$

The Hessian of *L* can be approximated by convolving *L* with:

$$H_{11} = \begin{pmatrix} 0 & 0 & 0 \\ 1 & -2 & 1 \\ 0 & 0 & 0 \end{pmatrix} \quad H_{12} = \begin{pmatrix} \frac{1}{2} & 0 & -\frac{1}{2} \\ 0 & 0 & 0 \\ -\frac{1}{2} & 0 & \frac{1}{2} \end{pmatrix} \quad H_{22} = \begin{pmatrix} 0 & 1 & 0 \\ 0 & -2 & 0 \\ 0 & 1 & 0 \end{pmatrix}$$

Algorithm Outline

1. Set parameters

   e.g.: k, Δs, number of iterations, …

2. Iterate
   1. Compute orientation tensor **T**
   2. Modify eigenvalues => **D**
   3. Computer Hessian **H** L
   4. Update L according to:

$$\boxed{L(x,y;s+\Delta s) = L(x,y;s) + \frac{\Delta s}{2}\mathrm{tr}[\mathbf{D}\,(\mathbf{H}L)]}$$

Comparison

Deblurring

- Minimizing

$$\varepsilon(f) = \frac{1}{2}\int_\Omega (g * f - f_0)^2 + \lambda(f_x^2 + f_y^2)\,dx\,dy$$

- Gives the Euler-Lagrange equation

$$g(-\cdot) * (g * f - f_0) - \lambda \Delta f = 0$$

- *g*: point spread function (PSF)
- *g*(-*x*): correlation operator / adjoint operator
- definition of adjoint operator $\langle x|Ay\rangle = \langle A^* x|y\rangle$
- cmp le14: $\mathbf{G}^T\mathbf{G}\mathbf{f} - \mathbf{G}^T\mathbf{f}_0 + \lambda(\mathbf{D}_x^T\mathbf{D}_x\mathbf{f} + \mathbf{D}_y^T\mathbf{D}_y\mathbf{f}) = 0$

Demo: Deblurring

- DBdemo.m

Optical Flow

- Minimizing (lecture 4) $\varepsilon(\mathbf{v}_h) = \sum_{\mathcal{R}} w|[\nabla^T f \ f_t]\mathbf{v}_h|^2$
- Under the constraint $|\mathbf{v}_h|^2 = 1$
- Using Lagrangian multiplier leads to the minimization problem

$$\varepsilon_T(\mathbf{v}_h, \lambda) = \varepsilon(\mathbf{v}_h) + \lambda(1 - |\mathbf{v}_h|^2)$$

- This is the *total least squares* formulation to determine the flow

Optical Flow

- Solution is given by the eigenvalue problem

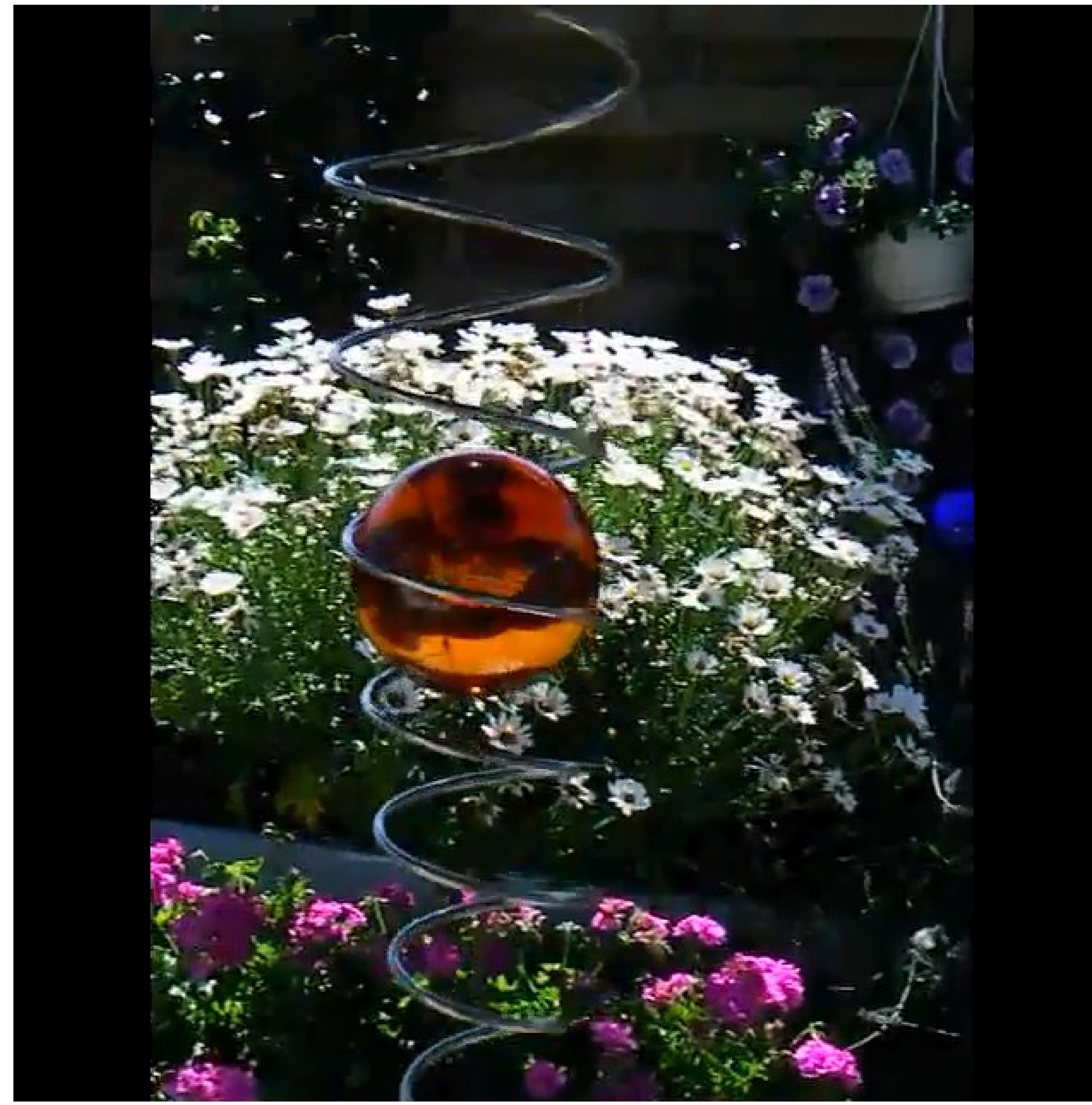
$$\left(\sum_{\mathcal{R}} w \begin{bmatrix} \nabla f \\ f_t \end{bmatrix} \begin{bmatrix} \nabla^T f & f_t \end{bmatrix} \right) \mathbf{v}_h = \lambda \mathbf{v}_h$$

$$\mathbf{T}\mathbf{v}_h = \lambda \mathbf{v}_h$$

- The matrix term **T** is the spatio-temporal structure tensor
- The eigenvector with the smallest eigenvalue is the solution (up to normalization of homogeneous element)

Optical Flow

- *Local* flow estimation
  - Design question: w and *R*
  - Aperture problem: motion at linear structures can only be estimated in normal direction (underdetermined)
  - Infilling limited
- *Global* flow instead



Optical Flow

- Minimizing BCCE over the whole image with additional smoothness term

$$\varepsilon(\mathbf{v}) = \frac{1}{2}\int_\Omega \overbrace{(\langle \mathbf{v}|\nabla f\rangle + f_t)^2}^{\text{BCCE}} + \lambda(|\nabla v_1|^2 + |\nabla v_2|^2)\,dx\,dy$$

$$\mathbf{v} = \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}$$

- Gives the Euler-Lagrange equation

$$(\langle \mathbf{v}|\nabla f\rangle + f_t)\nabla f - \lambda\Delta\mathbf{v} = 0$$

- Laplacian is approximately

$$\Delta\mathbf{v} \approx \bar{\mathbf{v}} - \mathbf{v}$$

$$\boxed{1}\boxed{1}\boxed{1} - 3 \cdot \boxed{0}\boxed{1}\boxed{0} = \boxed{1}\boxed{-2}\boxed{1}$$

Optical Flow

- Plugging into the EL-equation gives

$$(\lambda + \nabla f \nabla f^T)\mathbf{v} = \lambda \bar{\mathbf{v}} - f_t \nabla f$$

- Explicitly solving for **v** results in

$$\begin{aligned}
(\lambda + \nabla f \nabla f^T)\mathbf{v} &= (\lambda + \nabla f \nabla f^T)\bar{\mathbf{v}} - (\nabla f \nabla f^T \bar{\mathbf{v}} + \nabla f f_t) \\
&= (\lambda + \nabla f \nabla f^T)\bar{\mathbf{v}} - \nabla f (\nabla f^T \bar{\mathbf{v}} + f_t) \\
&= (\lambda + \nabla f \nabla f^T)\bar{\mathbf{v}} - \frac{\lambda + \nabla f^T \nabla f}{\lambda + \nabla f^T \nabla f} \nabla f (\nabla f^T \bar{\mathbf{v}} + f_t) \\
&= (\lambda + \nabla f \nabla f^T)\bar{\mathbf{v}} - \frac{\lambda + \nabla f \nabla f^T}{\lambda + \nabla f^T \nabla f} \nabla f (\nabla f^T \bar{\mathbf{v}} + f_t) \\
\mathbf{v} &= \bar{\mathbf{v}} - \frac{1}{\lambda + \nabla f^T \nabla f} \nabla f (\nabla f^T \bar{\mathbf{v}} + f_t)
\end{aligned}$$

Optical Flow

- Iterating the solution

$$\mathbf{v} = \bar{\mathbf{v}} - \frac{1}{\lambda + \nabla f^T \nabla f} \nabla f (\nabla f^T \bar{\mathbf{v}} + f_t)$$

results in the Horn & Schunck iteration

$$\mathbf{v}^{(s+1)} = \bar{\mathbf{v}}^{(s)} - \frac{1}{\lambda + |\nabla f|^2} (\langle \bar{\mathbf{v}}^{(s)} | \nabla f \rangle + f_t) \nabla f$$

- Significant improvement: use median instead of $\bar{\mathbf{v}}$!

Demo: Horn & Schunck

- HSdemo.m

Michael Felsberg
michael.felsberg@liu.se

LINKÖPINGS UNIVERSITET