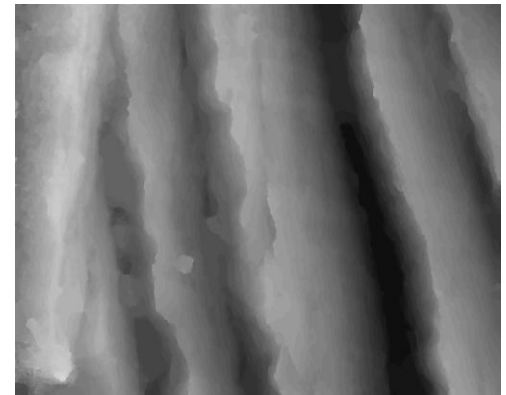
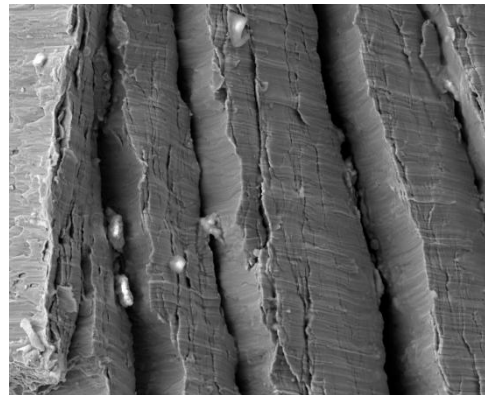


# Bildverarbeitung und Mustererkennung

## Image Processing and Pattern Recognition

**710.080 2VO**  
**710.081 1KU**

$$\int_{\Omega} f(x, u(x), \nabla u(x)) dx \Leftrightarrow \sup_{\phi \in K} \int_{\Omega \times R} \phi \cdot D1_u$$



# Bildverarbeitung und Mustererkennung

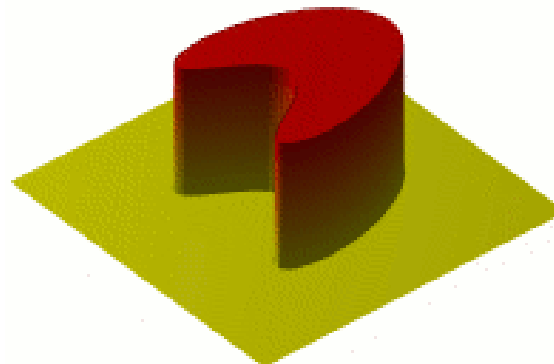
## Image Processing and Pattern Recognition

---

- PDE and Variational Methods
  - Linear Diffusion
  - Nonlinear Diffusion
  - Discretization and Numerical Solution

# Intuitive explanation of diffusion

- A physical process that equilibrates concentration differences without creating or destroying mass
- Heat equation: describes the distribution of heat over time
- Example: a cool metal plate that has been heated in a certain region. The temperature of the heated region gradually decreases while the temperature of the rest of the plate increases. In the equilibrium, there is a constant temperature of the complete plate



# Physical background

- *Fick's law:*

$$j = -D \cdot \nabla u$$

- The *concentration gradient*  $\nabla u$  causes a *flux*  $j$  which aims to compensate for this gradient.
- The relation between  $\nabla u$  and  $j$  is described by the so-called *diffusion tensor*  $D$
- The *continuity equation* describes the fact that diffusion does only transport mass without destroying it

$$\partial_t u = -\operatorname{div} j$$

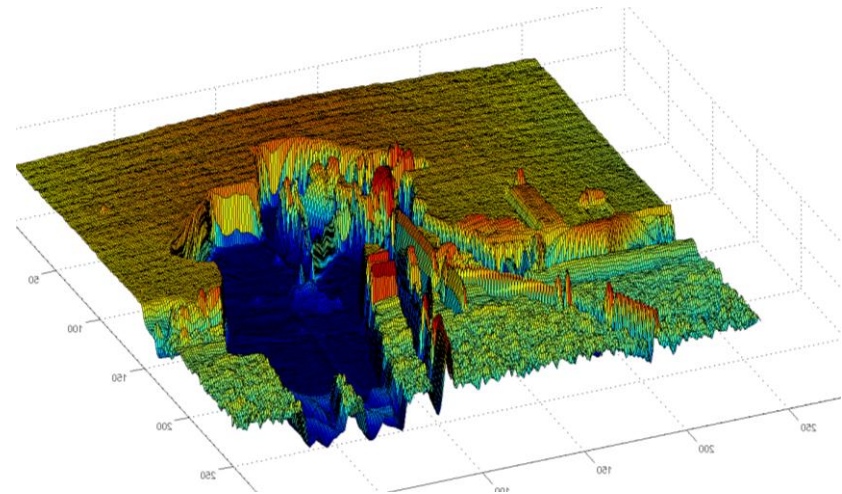
- Putting both equations together, we obtain the *diffusion equation*

$$\partial_t u = \operatorname{div} (D \cdot \nabla u)$$

Joachim Weickert: „Anisotropic Diffusion in Image Processing“ (Free for download)

# Diffusion in image processing

- Gray values are interpreted as the concentration
- Large brightness values indicate a high concentration
- Low brightness indicates a low concentration
- Diffusion of grey values means image filtering





# Classification of diffusion equations

- Homogeneous diffusion
  - Diffusion tensor  $D$  is constant
- Inhomogeneous diffusion
  - Diffusion tensor  $D$  is spatially varying
- Linear diffusion
  - Diffusion tensor  $D$  does not depend on  $u$
- Non-linear diffusion
  - Diffusion tensor  $D$  depends on  $u$

# Linear, homogeneous diffusion

- Simplest and best investigated PDE in image processing
  - Diffusion tensor is the identity:  $D = I$

- Linear diffusion equation:

$$\begin{aligned}\partial_t u &= \Delta u \\ u(x, 0) &= f(x)\end{aligned}$$

- Represents a second order initial value problem
- Relation to Gaussian filtering: Solution is given by

$$u(x, t) = \begin{cases} f(x) & \text{if } t = 0 \\ (K_{\sqrt{2t}} * f)(x) & \text{if } t > 0 \end{cases}$$

- $K_{\sqrt{2t}}$  is a Gaussian filter with standard deviation  $\sigma = \sqrt{2t}$

# Example

t = 0.000000



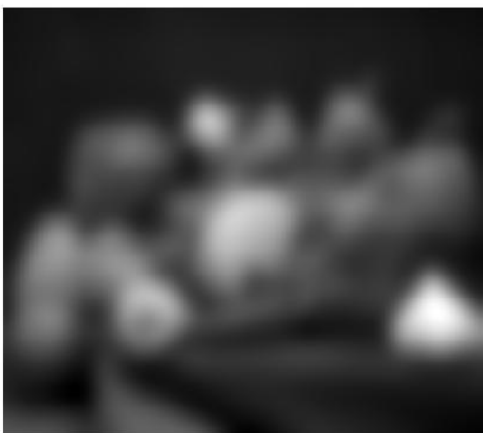
t = 20.000000



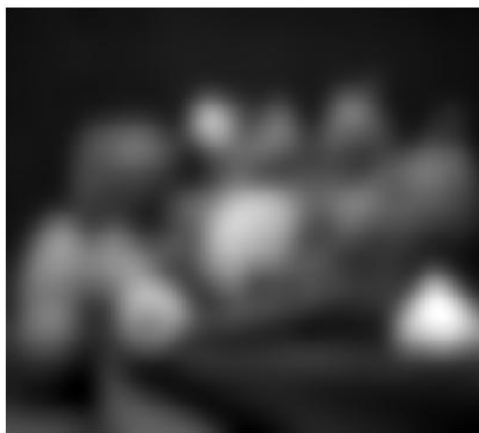
t = 40.000000



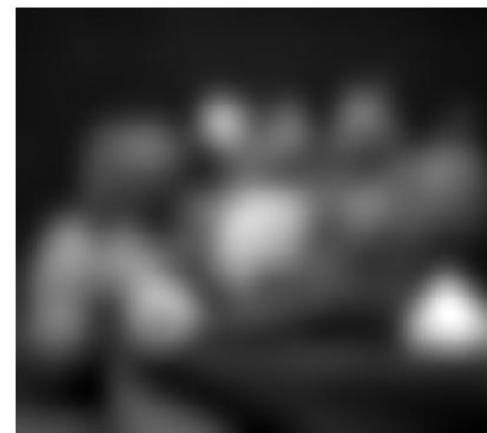
t = 60.000000



t = 80.000000



t = 100.000000



# Scale space

- The linear diffusion equation can be used to generate a linear *scale space*, providing a continuous scale parameter  $t$
- Recursive definition

$$\begin{aligned} T_0 f &= f \\ T_{t+s} f &= T_t (T_s f) , \quad \forall s, t \geq 0 \end{aligned}$$

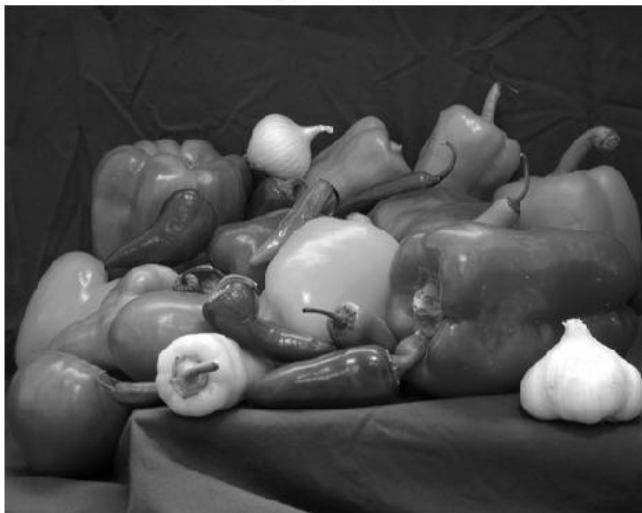
- Scale space analysis: Allows to analyze an image at different scales
- Important for edge detection (Canny), Feature detectors (SIFT), and image segmentation

# Inhomogeneous, linear diffusion

- Use the diffusion tensor  $D(x)$  to locally change the diffusion
- Example: decrease diffusivity at edges

$$D(x) = \begin{bmatrix} g(x) & 0 \\ 0 & g(x) \end{bmatrix}, \quad g(x) = e^{-\alpha \|\nabla f(x)\|^2}$$

Original image

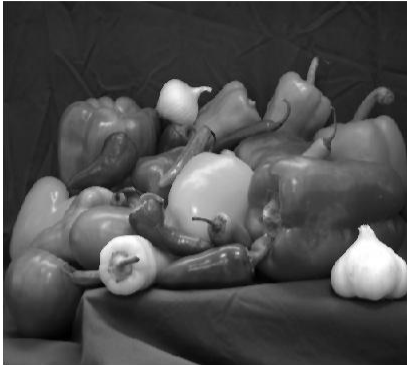


Diffusivity coefficient  $g(x)$



# Example

t = 0.000000



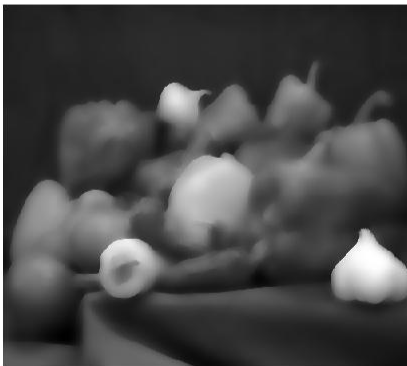
t = 20.000000



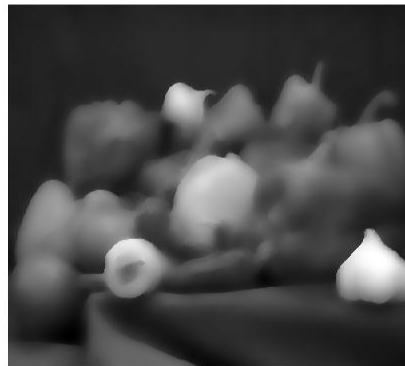
t = 40.000000



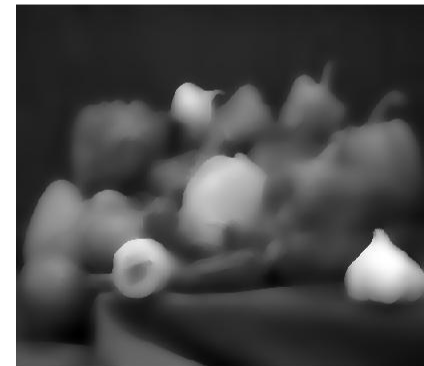
t = 60.000000



t = 80.000000



t = 100.000000



# Non-linear diffusion

- The Perona-Malik model

$$\partial_t u = \operatorname{div} (g(|\nabla u|^2) \nabla u)$$

- With diffusivities defined by

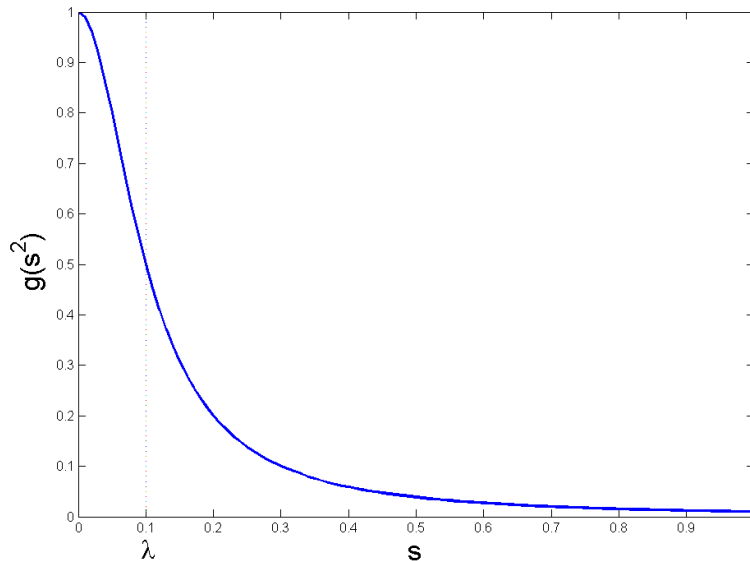
$$g(s^2) = \frac{1}{1 + s^2/\lambda^2}, \quad \lambda > 0$$

- Idea is to adaptively decrease the diffusivity at locations which are likely to be edges
- Edge detection based on this process clearly outperforms the Canny edge detector
- Edges are implicitly recognized and enhanced by the model

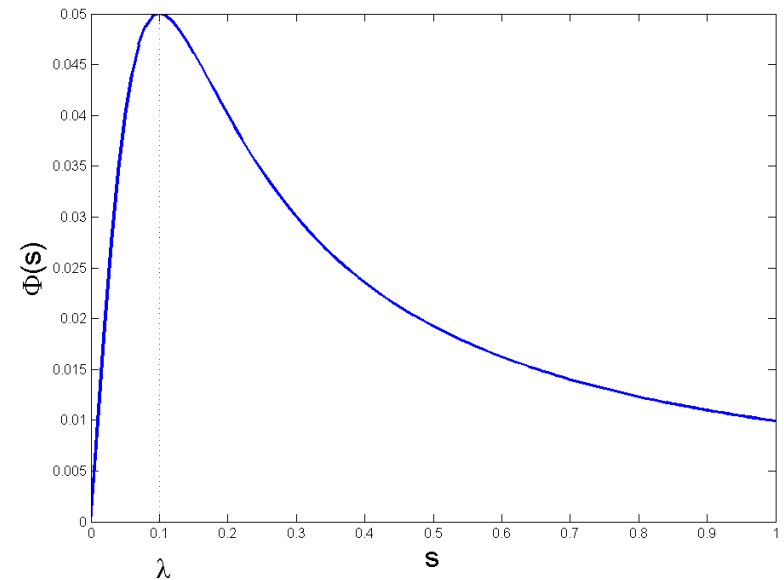


# The flux function

- In 1D:  $\partial_t u = \Phi'(u_x)u_{xx}$ ,  $\Phi(s) = sg(s^2)$
- $\Phi(s)$  is called the flux function



$$g(s^2) = \frac{1}{1 + s^2/\lambda^2}$$



$$\Phi(s) = sg(s^2) = \frac{s}{1 + s^2/\lambda^2}$$

- The parameter  $\lambda$  defines a threshold for edge detection



# Regularization

- Unfortunately, the Perona-Malik model is ill posed:
- No theoretical framework to show existence, uniqueness and regularity
- Diffusion process can become unstable (staircasing)
- Solution: Regularization by Gaussian smoothing:

Replace  $g(|\nabla u|^2)$  by  $g(|\nabla u_\sigma|^2)$  , where  $u_\sigma = K_\sigma * u$

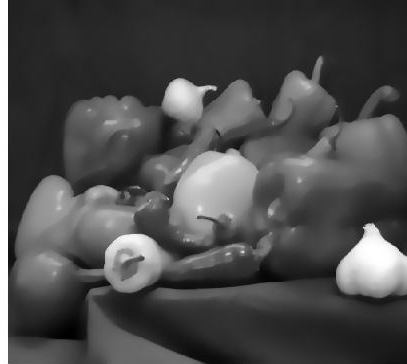
$$\partial_t u = \operatorname{div} \left( g(|\nabla u_\sigma|^2) \nabla u \right)$$

# Example

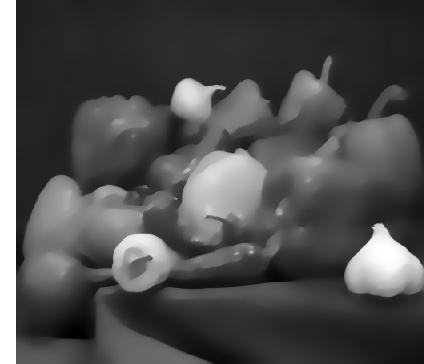
t = 0.000000



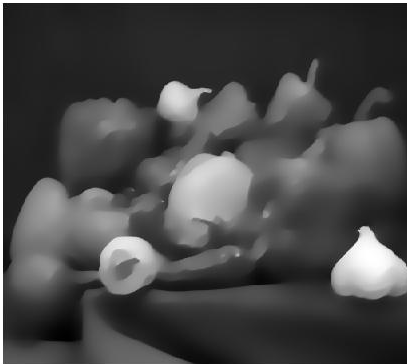
t = 20.000000



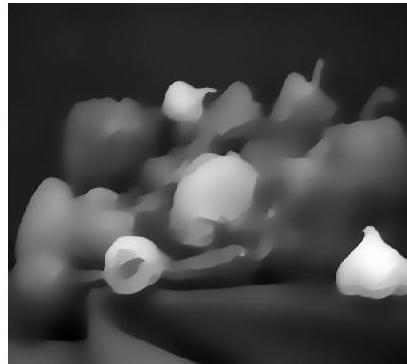
t = 40.000000



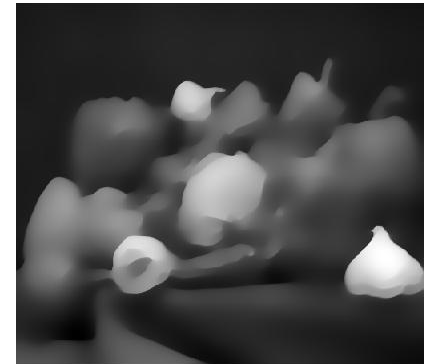
t = 60.000000



t = 80.000000



t = 100.000000





# Anisotropic nonlinear models

- Generalization of the scalar diffusivity  $g(|\nabla u|^2)$  in the Perona-Malik model to a full diffusion tensor  $D(\nabla u)$
- Does not only take the edge strength but also the edge direction into account
- The diffusion tensor can be written as

$$D(\nabla u) = \lambda_1 v_1 v_1^T + \lambda_2 v_2 v_2^T$$

where  $v_{1,2}$  and  $\lambda_{1,2}$  are the eigenvectors and eigenvalues of  $D(\nabla u)$

- One popular choice (Edge Enhancing Diffusion) is

$$v_1 \parallel \nabla u_\sigma, \quad v_2 \perp \nabla u_\sigma \quad \lambda_1(\nabla u_\sigma) = g(|\nabla u|^2), \quad \lambda_2 = 1$$

# Examples

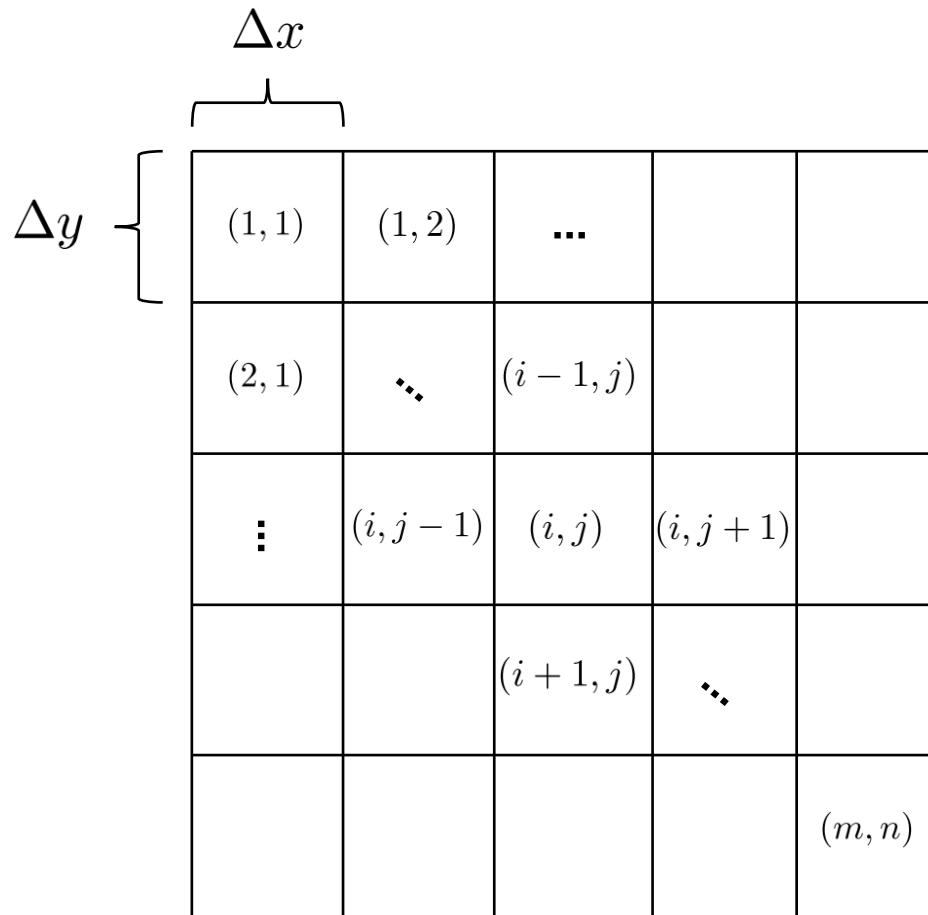


Coherence enhancing diffusion (Weickert et al.)



# Discretization and numerical solution

- Discretization of images (Matlab style)



# Finite differences

- Forward differences

$$(\delta_x^+ u)_{i,j} = \frac{u_{i,j+1} - u_{i,j}}{\Delta x}$$

$$(\delta_y^+ u)_{i,j} = \frac{u_{i+1,j} - u_{i,j}}{\Delta y}$$

$$(\delta_t^+ u)_{i,j} = \frac{u_{i,j}^{t+1} - u_{i,j}^t}{\Delta t}$$

- Backward differences

$$(\delta_x^- u)_{i,j} = \frac{u_{i,j} - u_{i,j-1}}{\Delta x}$$

$$(\delta_y^- u)_{i,j} = \frac{u_{i,j} - u_{i-1,j}}{\Delta y}$$

- Continuous PDE:

$$\partial_t u = \text{div} (\nabla u)$$

forward      backward      forward

Note:

$$\text{div} (\nabla u) = \partial_x (\partial_x u) + \partial_y (\partial_y u)$$

- Discretized PDE:

$$(\delta_t^+ u)_{i,j} = (\delta_x^- (\delta_x^+ u)_{i,j})_{i,j} + (\delta_y^- (\delta_y^+ u)_{i,j})_{i,j}$$

$$\frac{u_{i,j}^{t+1} - u_{i,j}^t}{\Delta t} = \left( \delta_x^- \left( \frac{u_{i,j+1} - u_{i,j}}{\Delta x} \right) \right)_{i,j} + \left( \delta_y^- \left( \frac{u_{i+1,j} - u_{i,j}}{\Delta y} \right) \right)_{i,j}$$

$$\frac{u_{i,j}^{t+1} - u_{i,j}^t}{\Delta t} = \frac{\frac{u_{i,j+1} - u_{i,j}}{\Delta x} - \frac{u_{i,j} - u_{i,j-1}}{\Delta x}}{\Delta x} + \frac{\frac{u_{i+1,j} - u_{i,j}}{\Delta y} - \frac{u_{i,j} - u_{i-1,j}}{\Delta y}}{\Delta y}$$

$$\frac{u_{i,j}^{t+1} - u_{i,j}^t}{\Delta t} = \frac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{(\Delta x)^2} + \frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{(\Delta y)^2}$$

# Computing the solution

- Simplification:  $\Delta x = \Delta y = 1$

$$\frac{u_{i,j}^{t+1} - u_{i,j}^t}{\Delta t} = u_{i,j+1} + u_{i,j-1} + u_{i+1,j} + u_{i-1,j} - 4u_{i,j} = (\Delta u)_{i,j}$$

- Matrix-vector representation:

$$\frac{u^{t+1} - u^t}{\Delta t} = \Delta u$$

- Explicit scheme

$$\frac{u^{t+1} - u^t}{\Delta t} = \Delta u^t \implies u^{t+1} = u^t + \Delta t \cdot \Delta u^t, \quad \Delta t < 1/4$$

- Implicit scheme

$$\frac{u^{t+1} - u^t}{\Delta t} = \Delta u^{t+1} \implies u^{t+1} = (I - \Delta t \cdot \Delta)^{-1} u^t, \quad \forall \Delta t \geq 0$$



# Perona-Malik diffusion

-