# Image Evolutions

**Image evolutions**

Consider images which *evolve over time*

$$u : \Omega \times [0, T] \to \mathbb{R}^n$$

The image has now three parameters: $u(x, y, t)$.

**Discretized view**

Generate a *sequence* of images $u^k : \Omega \to \mathbb{R}^n$ starting with some $u^0$:

$$u^0, u^1, u^2, u^3, \ldots$$

by a specific algorithm. Only the result $u^{k_0}$ for some $k_0 \geq 1$ is of interest.

# Diffusion

We will first consider grayscale images $u : \Omega \times [0, T] \to \mathbb{R}$,
and later generalize to multi-channel images.

**Diffusion**
Continuous-time update equation

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

Starting with some image $u(t = 0) = u^0$, this tells how the image must be changed over time. $\nabla$ and div are only w.r.t. spatial variables $x, y$.

**Diffusion tensor**
$D : \Omega \times [0, T] \to \mathbb{R}^{2\times 2}$ is called the *diffusion tensor*. It gives a *symmetric, positive definite* $2 \times 2$ *matrix* $D(x, y, t)$ for all $(x, y, t)$.
It may be different for every $(x, y, t)$, and may depend on $u$.

**Intuitively**
Diffusion tries to *locally* cancel out any existing color differences,
the image $u$ gradually becomes *more and more smooth* over time.

# Diffusion: Computation of the Update

**Diffusion**

$$(\partial_t u)(x, y, t) = \big(\operatorname{div}(D\nabla u)\big)(x, y, t)$$

1. Start with image $u : \Omega \times [0, T] \to \mathbb{R}$, values $u(x, y, t) \in \mathbb{R}$
2. Compute the gradient

$$g(x, y, t) := (\nabla u)(x, y, t) = \begin{pmatrix} (\partial_x u)(x, y, t) \\ (\partial_y u)(x, y, t) \end{pmatrix} \in \mathbb{R}^2$$

3. Multiply the diffusion tensor $D(x, y, t) \in \mathbb{R}^{2\times 2}$ with the gradient $g(x, y, t) \in \mathbb{R}^2$:

$$v(x, y, t) := D(x, y, t) g(x, y, t) \in \mathbb{R}^2$$

4. Take divergence of $v$:

$$d(x, y, t) := (\operatorname{div} v)(x, y, t) = (\partial_x v_1)(x, y, t) + (\partial_y v_2)(x, y, t) \in \mathbb{R}$$

# Diffusion: Types

**Diffusion**

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

**Linear/Nonlinear**

- Linear: $D$ *does not* depend on $u$
- Nonlinear: $D$ depends on $u$

*Additivity* property of *linear* diffusion:
Given the solutions $u$ and $v$ for starting images $u^0$ and $v^0$, respectively, the solution for the starting image $u^0 + v^0$ is given by $u + v$.

# Diffusion: Types

**Diffusion**

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

**Isotropic/Anisotropic**

- Isotropic: Diffusivity matrix $D$ is a scaled identity matrix:

  $$D(x, y, t) = g(x, y, t) \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}.$$

  $g(x, y, t) \in \mathbb{R}$ is called **diffusivity**. The diffusion equation becomes

  $$\partial_t u = \mathrm{div}(g \nabla u)$$

- Anisotropic: Any diffusion which is not isotropic.

Isotropic diffusion spreads out the values $u$ *equally in every direction.*

Anisotropic diffusion can selectively suppress information flow in certain directions, e.g. only smooth out $u$ *along* potential edges, and *not across*.

# Diffusion: Types

Each diffusion is either linear or nonlinear, and either isotropic or anisotropic:

| | isotropic | anisotropic |
|---|---|---|
| linear | linear isotropic | linear anisotropic |
| nonlinear | nonlinear isotropic | nonlinear anisotropic |

# Example: Laplace Diffusion

Diffusion tensor is constant:

$$D(x, y, t) := \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

Diffusion equation becomes

$$\partial_t u = \mathrm{div}(D\nabla u) = \Delta u$$

This is a *linear* and *isotropic* diffusion.

**Effect: Blurry version of the input image**
For $\Omega = \mathbb{R}^2$ one can show the explicit formula

$$u(x, y, t) = (G_{\sqrt{2t}} * u^0)(x, y).$$

This formula is **not** valid for rectangular domains $\Omega$, only for $\Omega = \mathbb{R}^2$, but the Laplace diffusion results are still similar to Gaussian convolution.

**Multi-channel images**
Process channel-wise.

# Example: Laplace Diffusion

Input at $t = 0$

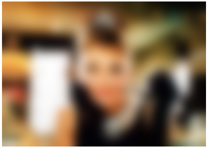
$t = 2$

$t = 4$

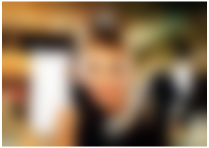
$t = 10$

$t = 20$

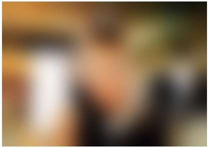
$t = 40$



$t = 100$



$t = 200$



$t = 400$

# Example: Huber Diffusion

Diffusion tensor depends on the image $u$ (or, more precicely, on $\nabla u$):

$$D(x,y,t) = g(x,y,t)\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

$$\text{with} \quad g(x,y,t) := \widehat{g}\Big(|\nabla u(x,y,t)|\Big) \quad \text{and} \quad \widehat{g}(s) := \frac{1}{\max(\varepsilon, s)}.$$

Diffusion equation becomes

$$\partial_t u = \operatorname{div}(D\nabla u) = \operatorname{div}\Big(\widehat{g}(|\nabla u|)\nabla u\Big)$$

This is a *nonlinear* and *isotropic* diffusion.

**Effect: Smoothing with better edge preservation:**
Edges of $u$ are points $(x,y)$ with large gradient value $|\nabla u(x,y)|$.
The diffusivity $g$ is small in these points, so there will be less smoothing.

**Multi-channel images**
Channel-wise, but with *one common* diffusivity $\widehat{g}(|\nabla u|)$ for *all* channels.

# Example: Huber Diffusion with $\varepsilon = 0.01$

Input at $t = 0$

$t = 0.04$

0.1

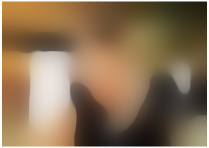
$t = 0.2$

$t = 0.4$

$t = 1$

$t = 2$

$t = 4$



$t = 10$

# Example: Linear Anisotropic Diffusion

Diffusion tensor depends on the structure tensor $T$ of the input image $f$.

$$D(x,y,t) = \begin{pmatrix} G_{11}(x,y) & G_{12}(x,y) \\ G_{21}(x,y) & G_{22}(x,y) \end{pmatrix}$$

where $G(x,y) = \mu_1 e_1 e_1^T + \mu_2 e_2 e_2^T \in \mathbb{R}^{2\times 2}$ is constructed from the eigenvalues and eigenvectors of the structure tensor $T(x,y)$ [1]. In particular

$$\mu_1 = \alpha, \quad \mu_2 = \alpha + (1-\alpha)\exp\Big(-\frac{C}{(\lambda_1-\lambda_2)^2}\Big).$$

Diffusion equation becomes

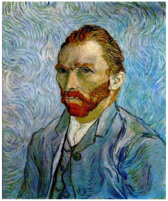
$$\partial_t u = \operatorname{div}(G\nabla u).$$

This is a *linear* and *anisotropic* diffusion.

**Effect: Smoothing along the direction of the image structures.**
If $(\lambda_1-\lambda_2)^2$ is big, $\mu_2$ is chosen big $\Rightarrow$ more smoothing along $e_2$.

[1] Weickert, Coherence-Enhancing Diffusion Filtering, '99

# Example: Linear Isotropic Diffusion



Input at $t = 0$

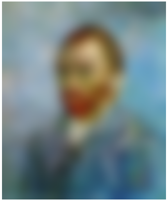
$t = 2$

$t = 4$

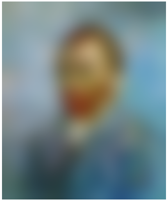
$t = 10$

$t = 20$

$t = 40$

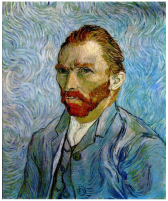


$t = 100$

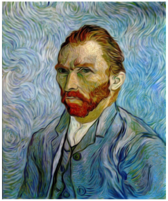


$t = 200$

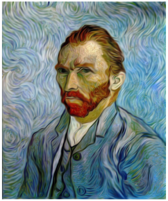
# Example: Linear Anisotropic Diffusion



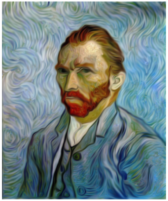
Input at $t = 0$



$t = 2$



$t = 4$



$t = 10$

$t = 20$

$t = 40$

$t = 100$

$t = 200$

# Discretization: General Isotropic Diffusion

**Temporal derivative**

Forward differences for $\partial_t$ with a time step $\tau > 0$:

$$(\partial_t^+ u)(x, y, t) = \frac{u(x, y, t+\tau) - u(x, y, t)}{\tau}$$

**Spatial derivatives**

Forward differences for $\nabla$, backward differences for div:

$$\mathrm{div}^-\Big(g\,\nabla^+ u\Big) = \partial_x^-\left(g\,\partial_x^+ u\right) + \partial_y^-\left(g\,\partial_y^+ u\right)$$

**Diffusivity**

Forward differences:

$$g = \widehat{g}\left(|\nabla^+ u|\right)$$

The current image $u(t)$ is used to compute $g$.

# Discretization: General Isotropic Diffusion

**Final scheme for general isotropic diffusion**

$$u(x,y,t+\tau) = u(x,y,t) + \tau\,\mathrm{div}^-\left(g\,\nabla^+ u\right) \qquad \text{with} \quad g = \widehat{g}\big(|\nabla^+ u|\big)$$

**Computation in several steps**

1. Compute the gradient $G := \nabla^+ u$
2. Compute the diffusivity $g = \widehat{g}(|G|)$
3. Compute the product $P := g \cdot G$
4. Compute the divergence $\mathrm{div}^-(P)$
5. Multiply by $\tau$ and add to $u$

**Time step restriction**
Only small $\tau$ possible. For monotonically decreasing $\widehat{g}$: $\tau < 0.25/\widehat{g}(0)$.

# Discretization: Laplace Diffusion

Two ways to discretize the special case of Laplace diffusion, i.e. $g = 1$.

**Final scheme for Laplace diffusion: Multi-step**
One way is to use the above general multi-step procedure.

**Final scheme for Laplace diffusion: Direct**
Another way is to compute the update $\operatorname{div}^-(\nabla^+ u) = \Delta\, u$ directly in a single step, using the discretization from the previous lecture:

$$u(x, y, t + \tau) = u(x, y, t) + \tau\, (\Delta\, u)(x, y, t)$$

with

$$\begin{aligned}(\Delta\, u)(x, y, t) = \quad & \mathbf{1}_{x+1<W} \cdot u(x+1, y, t) \ + \ \mathbf{1}_{x>0} \cdot u(x-1, y, t) \\ + \ & \mathbf{1}_{y+1<H} \cdot u(x, y+1, t) \ + \ \mathbf{1}_{y>0} \cdot u(x, y-1, t) \\ - \ & \Big( (\mathbf{1}_{x+1<W}) + (\mathbf{1}_{y+1<H}) + (\mathbf{1}_{x>0}) + (\mathbf{1}_{y>0}) \Big) \cdot u(x, y, t).\end{aligned}$$

**Time step restriction**
Only small $\tau$ possible: $\tau < 0.25$.