Lecture 23:

In the continuous case, consider:

$$E(f) = \int_\Omega (f-g)^2 \, dx\, dy + \lambda \int |\nabla f| \, dx\, dy$$

We want to find a sequence $f_0 \overset{=}{\approx} g$, $f_1, \ldots, f_n, \ldots$ such that:

$$E(f_0) \geq E(f_1) \geq \ldots \geq E(f_n) \geq E(f_{n+1}) \geq \ldots$$

Define $s(\varepsilon) \overset{\text{def}}{=} E(\underbrace{f_n + \varepsilon v}_{f_{n+1}})$ for some suitable v. (Assuming that $f_n = 0$ on $\partial\Omega$ with $v|_{\partial\Omega} = 0$)

For small $\varepsilon \geq 0$ by Taylor expansion,

$$s(\varepsilon) = s(0) + s'(0)\varepsilon + \underbrace{\mathcal{O}(\varepsilon^2)}_{\text{neglible}}$$

$s(\varepsilon) = E(f_{n+1})$, $s(0) = E(f_n)$, $s'(0)\varepsilon \to 0$

If $s'(0) \leq 0$, then $E(f_{n+1}) \leq E(f_n)$

$$\sqrt{(\nabla f_n + \varepsilon \nabla v)\cdot(\nabla f_n + \varepsilon \nabla v)} = |\nabla f_n + \varepsilon \nabla v|$$

Now, $\left.\frac{d}{d\varepsilon}\right|_{\varepsilon=0} s(\varepsilon) = \left.\frac{d}{d\varepsilon}\right|_{\varepsilon=0} \left[\int_\Omega (f_n + \varepsilon v - g)^2 \, dx\, dy + \lambda \int_\Omega |\nabla f_n + \varepsilon \nabla v| \, dx\, dy \right]$

$$\therefore\ S'(0) = \int_\Omega (f_n - g)\, v \, dx\,dy + \lambda \int_\Omega \frac{\nabla f_n \cdot \nabla v}{\sqrt{\nabla f_n \cdot \nabla f_n}}\, dx\,dy$$

$$= \int_\Omega (f_n - g)\, v \, dx\,dy - \lambda \int_\Omega \nabla \cdot \left(\frac{\nabla f_n}{|\nabla f_n|} \right) v \, dx\,dy + \lambda \int_{\partial\Omega} \frac{\nabla f_n}{|\nabla f_n|} \cdot \vec{n}\, v \, ds \quad (= 0)$$

$$= \int_\Omega \left[(f_n - g) - \lambda \nabla \cdot \left(\frac{\nabla f_n}{|\nabla f_n|} \right) \right] v \, dx\,dy$$

Put $v = -\left[(f_n - g) - \lambda \nabla \cdot \left(\frac{\nabla f_n}{|\nabla f_n|} \right) \right]$. Then: $S'(0) \leq 0$.

$\therefore$ Gradient descent algorithm:

$$f^{n+1} = f^n + \varepsilon \underbrace{\left[-\left((f_n - g) - \lambda \nabla \cdot \left(\frac{\nabla f_n}{|\nabla f_n|} \right) \right) \right]}_{v} \quad \text{for } n = 0, 1, 2, \ldots$$

This is called the gradient descent algorithm.

Image segmentation

Basic idea of Image Segmentation:

Task 1: extract sets of points describing the boundaries/edges of objects;

Task 2: Find a binary image ("black and white") so that "white" color represents the objects.

Information from the image: ($I: \Omega \to \mathbb{R}$; Ω = image domain)

Edge detector: $V: \Omega \to \mathbb{R}$ such that $V(\vec{x})$ is small if $\vec{x}$ is on the edges of the object.

Example 1: $V(\vec{x}) = -|\nabla I(\vec{x})|$

On edges, $\nabla I(\vec{x})$ is big $\Rightarrow$ $-|\nabla I(\vec{x})|$ is small

$V(\vec{x}) = -|\nabla I(\vec{x})| = 0$ in the interior of the object.

In the discrete case, $\frac{\partial I}{\partial x}$ and $\frac{\partial I}{\partial y}$ (hence ∇I) can be computed by linear filtering with filters: $\begin{pmatrix} 0 & -1 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}$ and $\begin{pmatrix} 0 & 0 & 0 \\ -1 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix}$

Example 2: $V(\vec{x}) = \frac{1}{1+|\nabla I(\vec{x})|}$

Segmentation models:

1. (Explicit) Parameterized curve evolution (Active contour model)

   Goal: Find a parameterized curve $\gamma : [0, 2\pi] \to \Omega$ such that it represents the boundary of the object.

2. (Implicit) Level set model:

   Find a function $\varphi : \Omega \to \mathbb{R}$ such that $\varphi(\vec{x}) > 0$ if $\vec{x}$ is inside the object and $\varphi(\vec{x}) < 0$ if $\vec{x}$ is outside the object.

$\therefore \varphi^{-1}(\{0\}) = \{\vec{x} \in \Omega : \varphi(\vec{x}) = 0\}$ is a set of points on the boundary of the object.

zero level set of φ.

This segmentation method is called the Level set method!

Active contour model (Kass, Witkin, Terzopoulous)

Let $I : \Omega \to \mathbb{R}$ be the image.

Goal: Find $\gamma : [0, 2\pi] \to \Omega \subseteq \mathbb{R}^2$, which lies on the boundary of the object.

Assume the boundary is a simple closed curve. Then: $\gamma(0) = \gamma(2\pi)$

Let $V : \Omega \to \mathbb{R}$ be the edge detector. We consider the snake model to find γ that minimizes the snake energy:

$$E_{snake}(\gamma) = \int_0^{2\pi} \underbrace{\frac{1}{2}|\gamma'(s)|^2}_{\text{enhance smoothness of } \gamma(s)} ds + \beta \int_0^{2\pi} \underbrace{V(\gamma(s))}_{\text{Find } \gamma(s) \text{ that lies on the boundary.}} ds$$

Goal: Use gradient descent algorithm to obtain γ.

Remark: $\gamma = (\varphi_1, \varphi_2)$. $\therefore \gamma'(s) = (\varphi_1'(s), \varphi_2'(s)) \Rightarrow |\gamma'(s)|^2 = (\varphi_1'(s))^2 + (\varphi_2'(s))^2$.

Starting from γ^0 = initial curve (e.g. circle)

Iteratively look for $\gamma^1, \gamma^2, \dots, \gamma^n, \gamma^{n+1}, \dots$ such that:

$$E_{\text{snake}}(\gamma^{n+1}) \le E_{\text{snake}}(\gamma^n).$$

Given γ^n, define $\gamma^{n+1} = \gamma^n + \varepsilon\varphi$ (Perturbation of γ^n)

with $\varphi(0) = \varphi(2\pi)$

Need to find φ such that $\left.\frac{d}{d\varepsilon}\right|_{\varepsilon=0} E_{\text{snake}}(\underbrace{\gamma^n + \varepsilon\varphi}_{\gamma^{n+1}}) < 0.$

(Then: $E(\gamma^{n+1}) = E(\gamma^n) + \varepsilon\left(\left.\frac{d}{d\varepsilon}\right|_{\varepsilon=0} E_{\text{snake}}(\gamma^n + \varepsilon\varphi) + \mathcal{O}(\varepsilon^2)\right)$

$$\left.\frac{d}{d\varepsilon}\right|_{\varepsilon=0} E_{\text{snake}}(\gamma^n + \varepsilon\varphi) = \left.\frac{d}{d\varepsilon}\right|_{\varepsilon=0} \int_0^{2\pi} \frac{1}{2}\underbrace{\left|(\gamma^n)'(s) + \varepsilon\varphi'(s)\right|^2}_{((\gamma^n)'(s) + \varepsilon\varphi'(s))\cdot((\gamma^n)'(s) + \varepsilon\varphi'(s))} ds + \beta \left.\frac{d}{d\varepsilon}\right|_{\varepsilon=0} \int_0^{2\pi} V(\gamma^n(s) + \varepsilon\varphi(s))\, ds$$

$$= \int_0^{2\pi} (\gamma^n)'(s) \cdot \varphi'(s)\, ds + \beta \int_0^{2\pi} \nabla V(\gamma^n(s)) \cdot \varphi(s)\, ds$$

$$= -\int_0^{2\pi} (\gamma^n)''(s) \cdot \varphi(s)\, ds + \underbrace{(\gamma^n)'(s)\,\varphi(s)\Big|_0^{2\pi}}_{0} + \beta \int_0^{2\pi} \nabla V(\gamma^n(s)) \cdot \varphi(s)\, ds$$

$$= \int_0^{2\pi} \left(-(\gamma^n)''(s) + \beta \nabla V(\gamma^n(s))\right) \cdot \varphi(s)\, ds$$

In order that $\frac{d}{d\varepsilon}\Big|_{\varepsilon=0} E_{snake}(\gamma^{n+1}) < 0$ (decreasing), we choose:

$$\varphi(s) = (\gamma^n)''(s) - \beta \nabla V(\gamma^n(s))$$

Hence, we modify γ^n by:

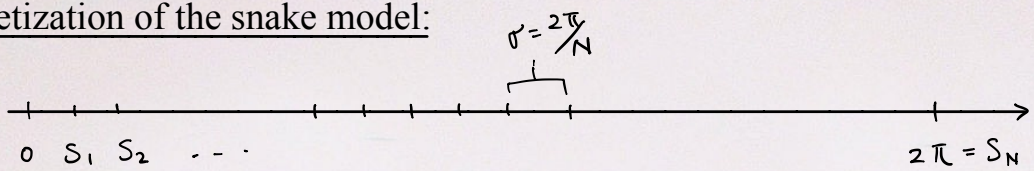
$$\gamma^{n+1}(s) = \gamma^n(s) + \varepsilon\left[(\gamma^n)''(s) - \beta \nabla V(\gamma^n(s))\right] \quad \text{for some } \varepsilon > 0.$$

$$\frac{\gamma^{n+1}(s) - \gamma^n(s)}{\varepsilon} = \underbrace{(\gamma^n)''(s) - \beta \nabla V(\gamma^n(s))}_{\text{denote it by: } -\nabla E(\gamma^n(s))}$$

Remark: In the continuous setting, we aim to obtain a time-dependent contour: $\gamma_t(s) = \gamma(s;t)$ such that:

$$\frac{d}{dt}\gamma_t(s) = -\nabla E(\gamma_t(s)).$$

Discretization of the snake model:



Let N = number of discrete points in $[0, 2\pi]$

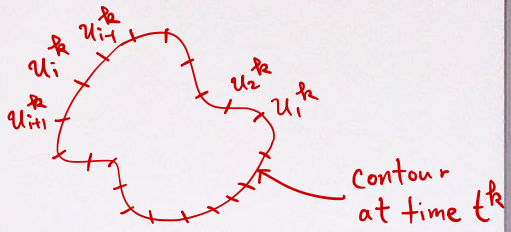
Let $\sigma = \frac{2\pi}{N}$ = step length.

$s_i = i\sigma$ for $i = 1, 2, \dots, N$

$t^k = k\tau$ for $k = 1, 2, \dots$ (τ = time step)

$u_i^k = \gamma(s_i; t^k) = \gamma(i\sigma; k\tau) = i^{th}$ node of the contour at time t^k

$= \begin{pmatrix} (u_i^k)_x \\ (u_i^k)_y \end{pmatrix}$



Define: $u^k = \begin{pmatrix} (u_1^k)_x & (u_1^k)_y \\ \vdots & \\ (u_i^k)_x & (u_i^k)_y \\ \vdots & \\ (u_N^k)_x & (u_N^k)_y \end{pmatrix} = \begin{pmatrix} u_1^k & u_2^k & \dots & u_N^k \\ | & | & & | \end{pmatrix}^T \in M_{N\times 2}(\mathbb{R})$

u^k is called the discrete closed curve at k iterations.

The discrete derivative can be approximated by:

$$\gamma_k'(s_i) \approx \frac{u_{i+1}^k - u_i^k}{\sigma} \qquad i = 1, 2, \ldots, N$$

$i\sigma$

(Here, $u_{N+1}^k = u_1^k$ and $u_0^k = u_N^k$. ($\because$ contour is closed)

Thus, the discrete snake energy can be written as:

$$E_{snake}(u) = \sum_{i=1}^{N} \frac{1}{2} \left| \frac{u_{i+1} - u_i}{\sigma} \right|^2 \underbrace{\sigma}_{ds} + \beta \sum_{i=1}^{N} V(u_i)\,\sigma$$

$u \in M_{N\times 2}(\mathbb{R})$

where $u = (u_1\ u_2 \ldots u_N)^T \in M_{N\times 2}(\mathbb{R})$ is a discrete closed curve.

To simplify, we throw away σ to obtain:

$$E_{snake}(u) = \sum_{i=1}^{N} \frac{1}{2} \left| \frac{\overset{\in\mathbb{R}^2}{u_{i+1}} - \overset{\in\mathbb{R}^2}{u_i}}{\sigma} \right|^2 + \beta \sum_{i=1}^{N} V(u_i)$$

Note that E_{snake} is a multi-variable function depending on:

$u_{1x}, u_{1y}, u_{2x}, u_{2y}, \ldots, u_{Nx}, u_{Ny}$, where $u_i = \begin{pmatrix} u_{ix} \\ u_{iy} \end{pmatrix} \in \mathbb{R}^2$.

To minimize E_{snake}, we compute the gradient of E_{snake}.

$$\text{Gradient of } E_{snake} = \nabla E_{snake} = \begin{pmatrix} \frac{\partial E_{snake}}{\partial u_{1x}} \\ \frac{\partial E_{snake}}{\partial u_{1y}} \\ \vdots \\ \frac{\partial E_{snake}}{\partial u_{ix}} \\ \frac{\partial E_{snake}}{\partial u_{iy}} \\ \vdots \end{pmatrix}$$

For simplicity, we can rewrite ∇E_{snake} as:

$$\nabla E_{snake} = \left(\frac{\partial E_{snake}}{\partial u_1}, \frac{\partial E_{snake}}{\partial u_2}, \cdots, \frac{\partial E_{snake}}{\partial u_N} \right)^T = \begin{pmatrix} \frac{\partial E_{snake}}{\partial u_{1x}} & \frac{\partial E_{snake}}{\partial u_{1y}} \\ \vdots & \vdots \\ \frac{\partial E_{snake}}{\partial u_{ix}} & \frac{\partial E_{snake}}{\partial u_{iy}} \\ \vdots & \vdots \end{pmatrix}$$

$\frac{\partial E_{snake}}{\partial u_1}$ // def $\begin{pmatrix} \frac{\partial E_{snake}}{\partial u_{1x}} \\ \frac{\partial E_{snake}}{\partial u_{1y}} \end{pmatrix}$, $\frac{\partial E_{snake}}{\partial u_2}$ \\ def $\begin{pmatrix} \frac{\partial E_{snake}}{\partial u_{2x}} \\ \frac{\partial E_{snake}}{\partial u_{2y}} \end{pmatrix}$

Here,

$$\frac{\partial E_{snake}}{\partial u_i} = \begin{pmatrix} \frac{\partial E_{snake}}{\partial u_{ix}} \\ \frac{\partial E_{snake}}{\partial u_{iy}} \end{pmatrix}.$$

Gradient descent:

$$M_{N\times 2} \ni \frac{u^{k+1} - u^k}{\tau} \in M_{N\times 2} = -\nabla E_{snake}$$

$\nabla E_{snake} \in M_{N\times 2}$

(Dimension agrees)

Recall that: $E_{snake}(u) = \sum_{i=1}^{N} \frac{1}{2} \left| \frac{u_{i+1} - u_i}{\sigma} \right|^2 + \beta \sum_{i=1}^{N} V(u_i)$

$$\therefore \frac{\partial E_{snake}}{\partial u_i} = -\left(\frac{u_{i+1} - u_i}{\sigma^2} \right) + \left(\frac{u_i - u_{i-1}}{\sigma^2} \right) + \beta \nabla V(u_i)$$

$$= \frac{-u_{i+1} + 2u_i - u_{i-1}}{\sigma^2} + \beta \nabla V(u_i)$$

The gradient descent algorithm can be written as:

$$\frac{u^{k+1} - u^k}{\tau} = -\nabla E_{snake}(u^k) = -\begin{pmatrix} \frac{\partial L_{snake}}{\partial u_{ix}^k} & \frac{\partial E_{snake}}{\partial u_{iy}^k} \\ \vdots & \\ \frac{\partial E_{snake}}{\partial u_{ix}^k} & \frac{\partial E_{snake}}{\partial u_{iy}^k} \\ \vdots & \end{pmatrix}$$

or $\frac{u_i^{k+1} - u_i^k}{\tau} = \frac{u_{i+1}^k - 2u_i^k + u_{i-1}^k}{\sigma^2} - \beta \nabla V(u_i)$ for $i = 1, 2, \ldots, N$.

(Explicit Euler Scheme)