图像分割

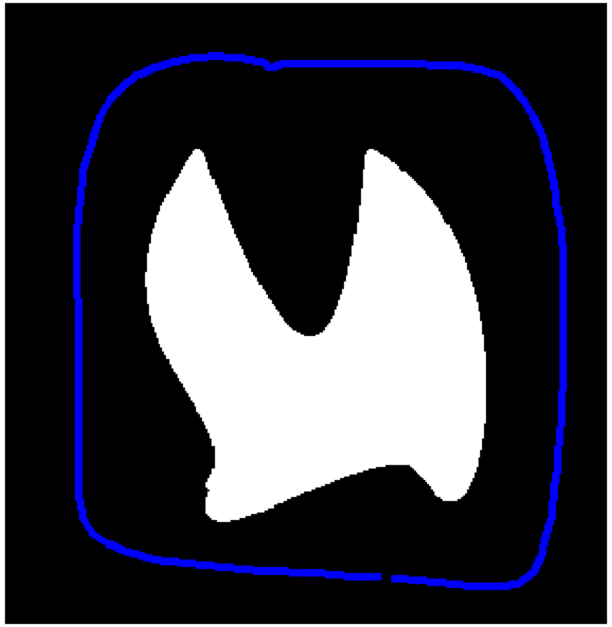
第二节：基于主动轮廓的图像分割

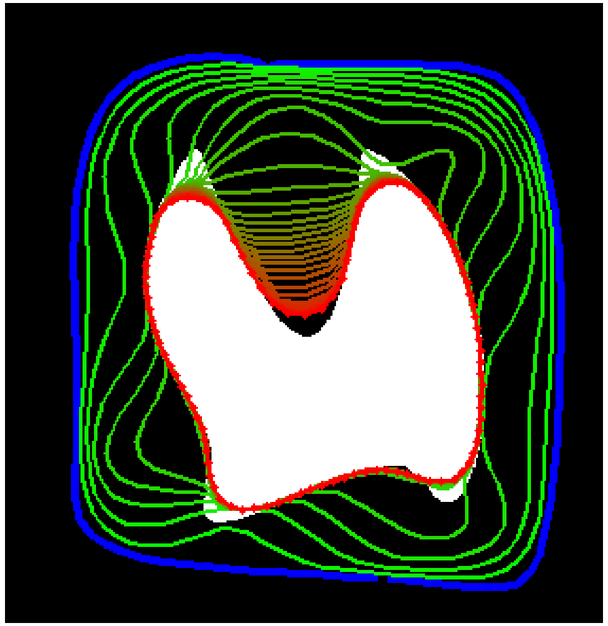
课程目录

基本思想

图像分割问题 → 轮廓进化问题

1) 工作原理：将一条曲线，在内外力的共同作用下，使得曲线逐步收敛到目标轮廓。





算法流程

1. 初始化曲线
2. 利用内外力更新曲线
3. 判断曲线是否收敛，
   否，转到步骤2。

基于主动轮廓的图像分割

基本思想

图像分割问题 → 轮廓进化问题 → 能量最小化问题

2) 如何表示曲线

3) 如何定义内外力

基本思想

图像分割问题 → 轮廓进化问题 → 能量最小化问题

2) 如何表示曲线

参数化曲线表示 → 参数主动轮廓模型（Parametric Active Contour Model） **Snake模型**

几何化曲线表示 → 几何活动轮廓模型（Geometric Active Contour Model） **Level Set模型**

3) 如何定义内外力

基于主动轮廓的图像分割

基本思想

图像分割问题 → 轮廓进化问题 → 能量最小化问题

2) 如何表示曲线

参数化曲线表示 → 参数主动轮廓模型（Parametric Active Contour Model）　**Snake模型**

几何化曲线表示 → 几何活动轮廓模型（Geometric Active Contour Model）　**Level Set模型**

3) 如何定义内外力

内力：控制曲线的弯曲与拉伸（曲线的平滑程度）；

外力（也称图像力）：曲线与图像边缘的吻合程度。

课程目录

- 基于主动轮廓的图像分割
- Snake算法
- Snake算法实现
- GVFSnake算法

模型定义

思考题：如果只有内力，分割曲线会怎么样？
如果只有外力，分割曲线又会怎么样？

(1) 曲线定义

- 参数化曲线表示 $C(s)=[x(s)\ y(s)]$ $s\in[0,1]$

(2) 目标函数

- 基于内外力的目标函数 $\min_C E(C)=\underbrace{\boxed{\alpha\int_0^1\|C'(s)\|_2^2ds+\beta\int_0^1\|C''(s)\|_2^2ds}}_{\text{内力}}+\underbrace{\boxed{\gamma\int_0^1 E_{ext}(C(s))ds}}_{\text{外力}}$

$$E_{ext}(C(s))=\frac{1}{1+\|\nabla(G_\sigma\otimes I(x(s),y(s)))\|}$$

梯度倒数

$$E_{ext}(C(s))=-\|\nabla(G_\sigma\otimes I(x(s),y(s)))\|$$

梯度负数

Snake算法：模型

模型优化

$$\min_{C} E(C) = \alpha \int_0^1 \|C'(s)\|_2^2 ds + \beta \int_0^1 \|C''(s)\|_2^2 ds + \gamma \int_0^1 E_{ext}(C(s)) ds$$

变分求解

Euler-Lagrange（欧拉拉格朗日）方程 $\alpha C''(s) - \beta C''''(s) - \gamma \nabla E_{ext} = 0$

引入时间t，将其转换为演化方程来迭代求解

$$\frac{dC(s,t)}{dt} = \alpha C''(s,t) - \beta C''''(s,t) - \gamma \nabla E_{ext}$$

https://en.wikipedia.org/wiki/Euler-Lagrange_equation

算法分析：优点

算法优点分析

（1）将分割问题统一于能量函数，模型简洁，优化方便

- 转化为**目标优化问题**
- **优化方法**：**变分法**；动态规划；贪婪算法；有限差分法；有限元法

（2）将分割问题统一于能量函数，便于引入新的模型认知

- 曲线自身的复杂度描述（**Snake用光滑度**；还包括曲线长度和所包含的面积）
- 曲线所对应的图像的描述（**Snake用梯度**；还包括区域统计一致性等）
- 将**高层知识**和**底层图像特征**结合

算法分析：缺点

深蓝学院 shenlanxueyuan.com

算法缺点分析

（1）初始化敏感、初始化要求高（在目标附近）

（2）容易陷入局部最优

→ **引入气球力、梯度矢量流**

（3）弱边缘难以处理 → 基于区域的方法

（4）曲线的拓扑结构无法变化 → 基于水平集的方法

（5）分割目标的形状无法保持 → 引入形状先验（如：PCA等流形）

算法分析：缺点

算法缺点的解决思路1：气球力（balloon）

该模型由Cohen提出

（1）核心思想

- 应用压力模型和高斯平滑，增大梯度场的吸引范围。因此不再要求将模型初始化在所期望的对象边界附近。

（2）形式化表达

$$\frac{dC(s,t)}{dt} = \alpha C''(s,t) - \beta C''''(s,t) + k_1 n(s) - k_2 \frac{\nabla E_{ext}}{\|\nabla E_{ext}\|}$$

- 如果 $k_1 > 0$ ，为膨胀力，曲线膨胀。
- 如果 $k_1 < 0$ ，为收缩力，曲线收缩。

（3）优点：对初始边界不敏感；**缺点**：对弱边界，存在漏出边界间隙等问题。

算法分析：缺点

算法缺点的解决思路2：梯度矢量流（Gradient Vector Flow：GVF）

该模型由Cohen提出

（1）核心思想

- 用GVF场代替经典外力场，使模型捕捉的范围得到了提高，而且能使主动轮廓进入凹陷区

（2）形式化表达

$$\frac{dC(s,t)}{dt} = \alpha C''(s,t) - \beta C''''(s,t) + U$$

-其中，$U = [u(x,y)\ v(x,y)]$ 为梯度矢量流，通过如下扩散方程求解得到

$$\iint_{x,y} \left[\mu(u_x^2 + u_y^2 + v_x^2 + v_y^2) + \|\nabla f\|^2 \|U - \nabla f\|^2\right] dxdy$$

（3）优点：对初始边界不敏感，良好的收敛性，深入目标边缘的凹陷区域

课程目录

算法分析：数值实现

$$\boxed{\frac{dC(s,t)}{dt} = \alpha C''(s,t) - \beta C''''(s,t) + U}$$

数值实现方法

(1) 进化方程（这里以GVF模型为例）

$$\frac{dC(s,t)}{dt} = \alpha C''(s,t) - \beta C''''(s,t) + U$$

求差分

$$\frac{C(s,t+dt) - C(s,t)}{dt} = \alpha C''(s,t) - \beta C''''(s,t) + U$$

$dt = 1$

$$\boxed{C(s,t+1) = \alpha C''(s,t) - \beta C''''(s,t) + U + C(s,t)}$$

曲线C

如何表达？

数值实现方法

(2) 曲线表达

曲线 C 用一组点组成（控制点通过插值得到）$C_1, C_2, \ldots\ldots, C_P$

(3) 各项

$$C(s,t) = \underbrace{\begin{bmatrix} C_1(t) \\ C_2(t) \\ \ldots\ldots \\ C_P(t) \end{bmatrix}}_{\text{曲线}} = \begin{bmatrix} 1 & 0 & \ldots & 0 \\ 0 & 1 & \ldots & 0 \\ \ldots & \ldots & \ldots & \ldots \\ 0 & 0 & \ldots & 1 \end{bmatrix} \begin{bmatrix} C_1(t) \\ C_2(t) \\ \ldots\ldots \\ C_P(t) \end{bmatrix}$$

算法分析：数值实现

数值实现方法

(2）曲线表达

曲线 C 用一组点组成（控制点通过插值得到）$C_1, C_2, \ldots\ldots, C_P$

(3）各项

$$C'(s,t) = \begin{bmatrix} C_1(t) \\ C_2(t) \\ \cdots\cdots \\ C_P(t) \end{bmatrix} - \begin{bmatrix} C_P(t) \\ C_1(t) \\ \cdots\cdots \\ C_{P-1}(t) \end{bmatrix} = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & 1 \end{bmatrix} \begin{bmatrix} C_1(t) \\ C_2(t) \\ \cdots\cdots \\ C_P(t) \end{bmatrix} - \begin{bmatrix} 0 & 0 & \cdots & 1 \\ 1 & 0 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & 0 \end{bmatrix} \begin{bmatrix} C_1(t) \\ C_2(t) \\ \cdots\cdots \\ C_P(t) \end{bmatrix}$$

曲线1阶导数

数值实现方法

（2）曲线表达

曲线 C 用一组点组成（控制点通过插值得到）$C_1, C_2, \ldots\ldots, C_P$

（3）各项

$$C''(s,t) = \underbrace{\begin{bmatrix} C_1(t) - C_P(t) \\ C_2(t) - C_1(t) \\ \ldots\ldots \\ C_P(t) - C_{P-1}(t) \end{bmatrix}}_{C'(s,t)} - \begin{bmatrix} C_2(t) - C_1(t) \\ C_3(t) - C_2(t) \\ \ldots\ldots \\ C_1(t) - C_P(t) \end{bmatrix} = \begin{bmatrix} 2 & -1 & \ldots & -1 \\ -1 & 2 & \ldots & 0 \\ \ldots & \ldots & \ldots & \ldots \\ -1 & 0 & \ldots & 2 \end{bmatrix} \begin{bmatrix} C_1(t) \\ C_2(t) \\ \ldots\ldots \\ C_P(t) \end{bmatrix}$$

曲线2阶导数

对第p个点：$(C_p - C_{p-1}) - (C_{p+1} - C_p) = (2C_p - C_{p+1} - C_{p-1})$

深蓝学院 shenlanxueyuan.com

数值实现方法

（2）曲线表达

曲线 C 用一组点组成（控制点通过插值得到）

（3）各项

$$C''''(s,t) = B * \begin{bmatrix} C_1(t) \\ C_2(t) \\ \cdots\cdots \\ C_P(t) \end{bmatrix}$$

曲线4阶导数

对第p个点：

$$(C_{p+2} + C_p - 2C_{p+1}) + (C_p + C_{p-2} - 2C_{p-1}) - 2(C_{p+1} + C_{p-1} - 2C_p)$$
$$= C_{p+2} - 4C_{p+1} + 6C_p - 4C_{p-1} + C_{p-2}$$

算法分析：数值实现

数值实现方法

(4) 目标函数转换

$$\boxed{C(s,t+1)=\alpha C''(s,t)-\beta C''''(s,t)+U+C(s,t)}$$

令 $V(t)=\begin{bmatrix} C_1(t) \\ C_2(t) \\ \cdots \\ C_P(t) \end{bmatrix}$，

$C(s,t)=V(t)=Id*V(t)$，其中 Id 是单位矩阵

$C''(s,t)=A*V(t)$

$C''''(s,t)=B*V(t)$

$$\boxed{\begin{aligned} C(s,t+1)=V(t+1) \\ &=\alpha A*V(t)-\beta B*V(t)+U+V(t) \\ &=(\alpha A-\beta B+Id)*V(t)+U \end{aligned}}$$

算法分析：数值实现

问题与解决方案

$$V(t+1) = (\alpha A - \beta B + Id) * V(t) + U$$

（1）对步长要求非常高

- 过大（不收敛），过小（收敛速度极其慢，最终还是不收敛）

（2）半隐解决方案

思考题：此处的步长是多少，注意*dt*?

$$C(s,t+1) = \alpha C''(s,t) - \beta C''''(s,t) + U + C(s,t)$$

半隐表示

参考代码SnakeInternalForceMatrix2D以及SnakeMoveIteration2D。

$$C(s,t+1) = \alpha C''(s,t+1) - \beta C''''(s,t+1) + U + C(s,t)$$

$$V(t+1) = (\alpha A - \beta B) * V(t+1) + U + V(t)$$

$$(Id - (\alpha A - \beta B))V(t+1) = U + V(t)$$

课程目录

- 基于主动轮廓的图像分割
- Snake算法
- Snake算法实现
- **GVFSnake算法**

GVFSnake算法

核心思想

利用“扩散方程”从原始外力∇f求出新的扩散后的外力项 $U=[u(x,y)\ v(x,y)]$

$$\iint_{x,y}\left[\boxed{\mu}\boxed{(u_x^2+u_y^2+v_x^2+v_y^2)}+\boxed{\|\nabla f\|^2\|U-\nabla f\|^2}\right]dxdy$$

第一项：光滑项（先验项）

第二项：保真项，数据项（似然项）

第三项：加权项

问题1：为什么第一项能确保新的外力项是光滑的？

问题2：第二项前面的 $\|\nabla f\|^2$ 有什么作用？

求解方法

$$\boxed{\|\nabla f\|^2 = (f_x^2 + f_y^2)}$$

利用变分法中的“欧拉方程”，可得

$$\mu\nabla^2 u - (u - f_x)(f_x^2 + f_y^2) = 0 \qquad \mu\nabla^2 v - (v - f_y)(f_x^2 + f_y^2) = 0$$

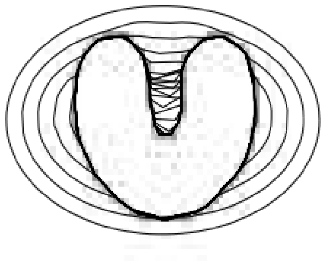
梯度流更新 引入时间t，将其转换为演化方程来迭代求解

$$u(t+1) = u(t) + \mu\nabla^2 u(t) - (u(t) - f_x)(f_x^2 + f_y^2)$$
$$v(t+1) = v(t) + \mu\nabla^2 v(t) - (v(t) - f_y)(f_x^2 + f_y^2)$$

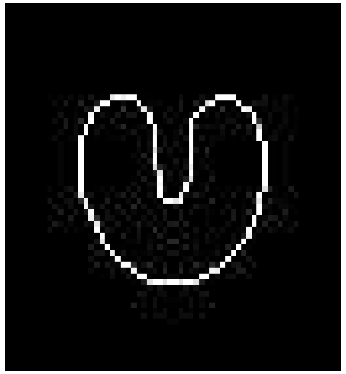
参考代码GVFOptimizeImageForces2D

GVFSnake算法

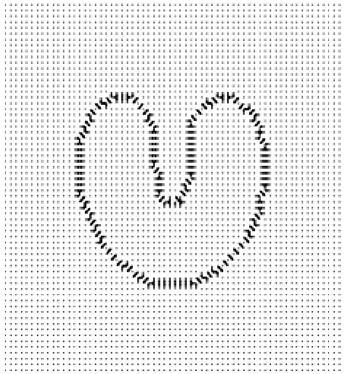
实验对比



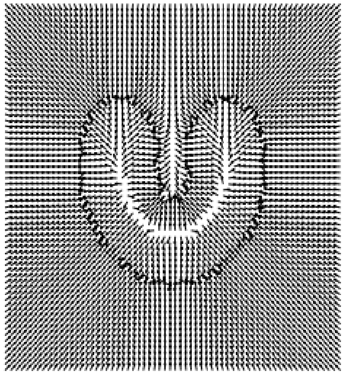
Edge Map



Edge Map Gradient



Normalized GVF field



优点：

1. 迭代加速
2. 对初始化略微不敏感

作业

作业1：推导Snake模型中的A和B矩阵，并自己用代码实现 $(Id-(\alpha A-\beta B))^{-1}$。

作业2：自己实现Snake模型

根据提供的代码框架参考，实现Snake模型。

作业3：自己实现GVFSnake模型

根据提供的代码框架参考，在实现Snake模型的基础上，进一步实现GVFSnake。

作业4：假设一模型的梯度是 $f(x)=Ax-b$ **，**

1. 试用梯度下降以及半隐求解，用代码实现。
2. 尝试不同的更新率（梯度下降），比较算法稳定性，收敛速度以及复杂度。

深蓝学院
shenlanxueyuan.com

感谢各位聆听!

Thanks for Listening