Horn-Schunck(HS) 光流法

Dezeming Family

2023 年 3 月 13 日

DezemingFamily 系列文章和电子书**全部都有免费公开的电子版**，可以很方便地进行修改和重新发布。如果您获得了 DezemingFamily 的系列电子书，可以从我们的网站 [https://dezeming.top/] 找到最新的版本。对文章的内容建议和出现的错误也欢迎在网站留言。

目录

一 建立光流估计极小化方程 1

二 偏导的估计与光流速度的拉普拉斯估计 2

三 求解极小化方程 2

参考文献 4

一 建立光流估计极小化方程

HS 光流估计的年代较早，我们的叙述就不按照论文原文 [1] 的描述顺序，而是多参考自网上的博客和随笔文章。

HS 光流计算基于物体移动的光学特性的两个假设：

- 运动物体的灰度在很短的间隔时间内保持不变。
- 给定邻域内的速度向量场变化是缓慢的。

因此对于时间 t 上的 (x, y) 处的像素值 $E(x, y, t)$，在短暂的时间 δt 后对应于图像上的点位置为：

$$E(x+\delta x, y+\delta y, t+\delta t) = E(x, y, t) \tag{一.1}$$

上式左边部分在 x, y 处泰勒展开，得到：

$$E(x, y, t) = E(x, y, t) + \delta x\frac{\partial E}{\partial x} + \delta y\frac{\partial E}{\partial y} + \delta t\frac{\partial E}{\partial t} + \epsilon \tag{一.2}$$

上式两边减去 $E(x, y, t)$，然后再除以 δt 就能得到：

$$\frac{\delta x}{\delta t}\frac{\partial E}{\partial x} + \frac{\delta y}{\delta t}\frac{\partial E}{\partial y} + \frac{\partial E}{\partial t} + O(\delta t) = 0 \tag{一.3}$$

$$\delta t \to 0: \quad \frac{\partial E}{\partial x}\frac{dx}{dt} + \frac{\partial E}{\partial y}\frac{dy}{dt} + \frac{\partial E}{\partial t} = 0 \tag{一.4}$$

我们表示 x 方向的速度的 u，y 方向的速度为 v，就能得到：

$$I_x u + I_y v + I_t = 0 \tag{一.5}$$

该式就是光流约束方程，有两个变量 u 和 v，因此还需要其他约束条件。

这里引入的是光流的平滑约束条件（设光流场是一个分段光滑光流场），即在局部区域 u 和 v 是基本一致的，改变不大（假设物体是无变形的刚体）。局部平滑项写为（下式即速度在各个方向的变化率大小）速度的梯度的模的平方：

$$\zeta_c^2 = (\frac{\partial u}{\partial x})^2 + (\frac{\partial u}{\partial y})^2 + (\frac{\partial v}{\partial x})^2 + (\frac{\partial v}{\partial y})^2 \tag{一.6}$$

对于所有的像素点，要满足该式的和最小。

另一种衡量光流场平滑性的方法是光流在 x 和 y 方向的拉普拉斯。u 和 v 的拉普拉斯定义为：

$$\begin{aligned}\nabla^2 u &= \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \\ \nabla^2 v &= \frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}\end{aligned} \tag{一.7}$$

在简单的情况下，这两个拉普拉斯都是 0（比如相机沿直线匀速移动），如果观察者平行于平面物体平移，或者围绕垂直于表面的线旋转或垂直于表面行进，则 u 和 v 的二次偏导数消失（假设图像生成是透视投影）。

在这里我们使用的是梯度的模作为平滑项，列出两个约束条件：

$$\begin{aligned}\zeta_b &= \frac{\partial E}{\partial x}\frac{dx}{dt} + \frac{\partial E}{\partial y}\frac{dy}{dt} + \frac{\partial E}{\partial t} \\ \zeta_c^2 &= (\frac{\partial u}{\partial x})^2 + (\frac{\partial u}{\partial y})^2 + (\frac{\partial v}{\partial x})^2 + (\frac{\partial v}{\partial y})^2\end{aligned}$$

联合起来可以建立极小化方程：

$$\zeta^2 = \int\int(\alpha^2\zeta_c^2 + \zeta_b^2)dxdy \tag{一.8}$$

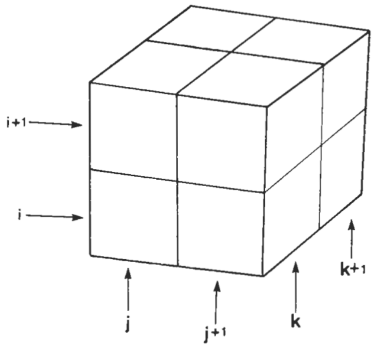
ζ_c^2 是空间相关性，ζ_b^2 是亮度约束。通过找到合适的光流速度 (u, v) 来极小化上式。

在求解它之前，先介绍一下图像中常用的偏导和拉普拉斯估计的方法。

二　偏导的估计与光流速度的拉普拉斯估计

偏导的估计

假设下图中，i 索引表示图像 y 方向，j 索引表示图像 x 方向，k 索引表示图像时序方向：



我们用下式来估计微分（其实就是一侧的四个值减去另一侧的四个值，然后求平均）：

$$
\begin{aligned}
E_x &\approx \frac{1}{4}\left\{E_{i,j+1,k} - E_{i,j,k} + E_{i+1,j+1,k} - E_{i+1,j,k} + E_{i,j+1,k+1} - E_{i,j,k+1} + E_{i+1,j+1,k+1} - E_{i+1,j,k+1}\right\} \\
E_y &\approx \frac{1}{4}\left\{E_{i+1,j,k} - E_{i,j,k} + E_{i+1,j+1,k} - E_{i,j+1,k} + E_{i+1,j,k+1} - E_{i,j,k+1} + E_{i+1,j+1,k+1} - E_{i,j+1,k+1}\right\} \\
E_t &\approx \frac{1}{4}\left\{E_{i,j,k+1} - E_{i,j,k} + E_{i+1,j,k+1} - E_{i+1,j,k} + E_{i,j+1,k+1} - E_{i,j+1,k} + E_{i+1,j+1,k+1} - E_{i+1,j+1,k}\right\}
\end{aligned} \tag{二.1}
$$

光流速度的拉普拉斯估计

我们也需要近似 u 和 v 的拉普拉斯，因为后面求最小化的时候会用到。

一个方便的近似是如下形式：

$$
\begin{aligned}
\nabla^2 u &\approx \kappa(\overline{u}_{i,j,k} - u_{i,j,k}) \\
\nabla^2 v &\approx \kappa(\overline{v}_{i,j,k} - v_{i,j,k})
\end{aligned} \tag{二.2}
$$

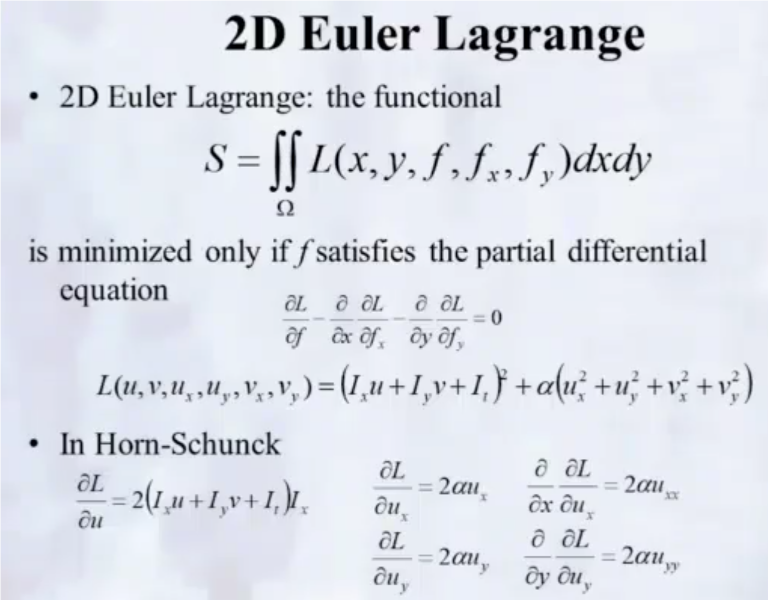
其中，局部均值 $\overline{u}$ 和 $\overline{v}$ 是用下面的卷积核在空间域计算的（计算第 k 帧的 $\overline{u}$ 和 $\overline{v}$）：

1/12	1/6	1/12
1/6	−1	1/6
1/12	1/6	1/12

三　求解极小化方程

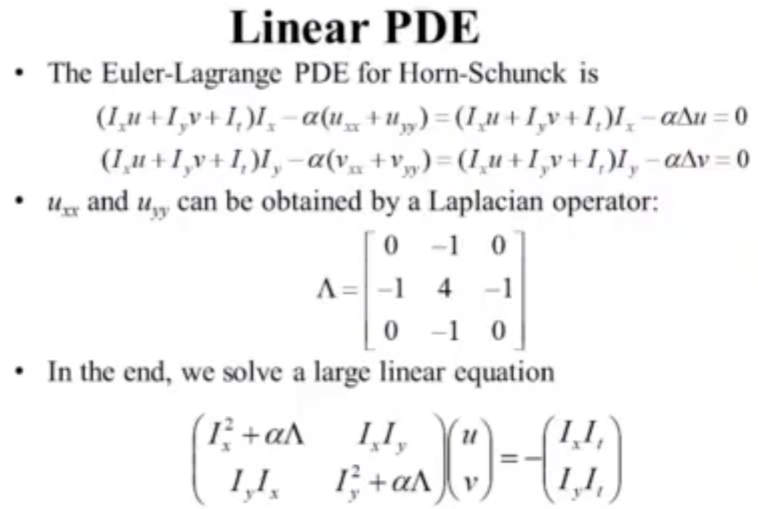
求解的目标是找到合适的光流速度 u, v，使得式一.8的值最小。论文中使用的是变分-迭代求解的过程。这里我们也是使用欧拉-拉格朗日变分法来求解，我先根据浙大陆系群教授的 PPT [3] 来描述。

最小化一.8最常用的方式就是 $2D$ 欧拉-拉格朗日方程方法 [3]。（其中，I 就是论文里的 E，注意这里假设 α 是一个正数，论文 [1] 里用的是 α^2，这点区别大家一定要注意）：



这里的 f 是二维的，意味着跟 u 和 v 有关。

经过化简，得到最终要求解的方程（这里没有用论文原文 [1] 中的拉普拉斯计算式）：



由于每个像素都要求解上式条件，所以这是一个超大规模的线性系统。可以用 python 或者 matlab 里的线性系统求解公式来求解。

下图表示输入两张连续帧图像（只显示了一张），得到的 HS 光流图，其中，颜色越深，表示运动距离越大：

不过对于编程的话，论文原文更容易实现，因为它用的是像素迭代方案。从 [4] 中摘录的 matlab 源码参考如下。

代码 1，计算图像偏导：

```
function [Ex, Ey, Et] = derivative(Im1, Im2)
% function DERIVATIVE computes partial derivatives Ex, Ey, Et
% of a sequence of 2 images Im1, Im2 of double class.

% kernels for convolution
```

```
Kx = 0.25 * [-1 1; -1 1];
Ky = 0.25 * [-1 -1; 1 1];
Kt = 0.25 * [-1 -1; -1 -1]; % kt1 = Kt, kt2 = -Kt

% compute derivatives
Ex = conv2(Im1, Kx, 'same') + conv2(Im2, Kx, 'same');
Ey = conv2(Im1, Ky, 'same') + conv2(Im2, Ky, 'same');
Et = conv2(Im1, Kt, 'same') + conv2(Im2, -Kt, 'same');
```

代码 2，计算图像光流：

```
function [U, V] = HS(Im1, Im2, alpha, N)
% function HS computes flow velocities U, V of a sequence of 2 images
% Im1, Im2 of double class based on Horn-Schunck algorithm. alpha is the
% weighting factor and N is the number of iteration.

% compute partial derivatives
[Ex, Ey, Et] = derivative(Im1, Im2);

% intial U, V
[l, c] = size(Im1);
U = zeros(l, c);
V = zeros(l, c);

K = [1/12 1/6 1/12; 1/6 -1 1/6; 1/12 1/6 1/12]; % Laplacian kernel
A = alpha^2 + Ex.^2 + Ey.^2;

for i = 1:N
    % compute U,V averages
    U_avg = conv2(U, K, 'same');
    V_avg = conv2(V, K, 'same');
    B = (Ex.*U_avg + Ey.*V_avg + Et);

    % compute U, V at current iteration
    U = U_avg - Ex.*B./A;
    V = V_avg - Ey.*B./A;
end;
```

因为算法确实比较早了，所以也无需过于深究，想要了解求解的变分原理可以参考其他书目（变分和拉格朗日乘子法等都可以认为是运筹学方面的内容，掌握难度较大，一般工程上都是知道公式形式，能够在求解约束时使用即可）。

参考文献

[1] Horn B K P, Schunck B G. Determining optical flow[J]. Artificial intelligence, 1981, 17(1-3): 185-203.

[2] https://www.bbsmax.com/A/WpdKavKZJV/

[3] https://haokan.baidu.com/v?pd=wisenatural&vid=16865900605827872877

[4] https://github.com/vuanhtuan1012/determining-optical-flow