运动补偿技术研究

learning note For reading translation

我真的不懂忧郁

Delft University of Technology

TU Delft

运动补偿技术研究

learning note For reading translation

by

我真的不懂忧郁

Student Name	Student Number
First Surname	1234567

Instructor:	I. Surname
Teaching Assistant:	I. Surname
Project Duration:	Month, Year - Month, Year
Faculty:	Faculty of Aerospace Engineering, Delft

TU Delft

Preface

A preface...

我真的不懂忧郁

Delft, September 2024

Summary

A summary...

目录

Nomenclature

If a nomenclature is required, a simple template can be found below for convenience. Feel free to use, adapt or completely remove.

Abbreviations

Abbreviation	Definition
ISA	International Standard Atmosphere
...	

Symbols

Symbol	Definition	Unit
V	Velocity	[m/s]
...		
ρ	Density	[kg/m^3]
...		

Chapter 1

Photoslip Algorithm：传统算法

本篇文章来自https://zhuanlan.zhihu.com/p/660866515?utm_campaign=shareopn&utm_medium=social&utm_psn=1815096437689683968&utm_source=wechat_session

序列图像的光流信息是指一幅图像中的许多点在第二幅图像中移动的位置，通常假设第一幅图像中的大部分点框架都可以在第二幅图像中找到。

在理想情况下，光流场与图像中的二维运动场吻合，光流法利用图像序列中像素在时间域上的变化以及相邻帧之间的相关性来找到上一帧跟当前帧之间存在的对应关系，一幅图像中每个像素的位移矢量，指示该像素在另一幅图像中的相对位置，从而计算出相邻帧之间物体的运动信息，因此可用于对物体运动的情况进行估测与分析。

如果图像中的每个像素都采用这种方法通常称为“**稠密光流**”。还有一种算法称为“**稀疏光流**”，仅仅只跟踪图像中某些点的子集，该算法通常是快速且可靠的，因为其将注意力只放在容易跟踪的特定点上，稀疏跟踪的计算成本远远低于稠密跟踪

1.1. *Lucas-Kanade* Algorithm

Lucas-Kanade 基于两个重要假设：

亮度不变假设

亮度不变假设是指待估计的光流的两帧图像的同一物体的亮度不变。

$$I(x,y,t) = I(x+u, y+v, t+\Delta t) \tag{1.1}$$

将右边的等式作一阶泰勒展开得

$$I(x+u, y+v, t+\Delta t) = I(x,y,t) + I'_x u + I'_y v + I'_t \Delta t \tag{1.2}$$

根据假设

$$I'_x u + I'_y v + I'_t \Delta t = 0 \tag{1.3}$$

$$[I'_x, I'_y] \begin{bmatrix} u \\ v \end{bmatrix} = -I'_t \Delta t \tag{1.4}$$

I'_x, I'_y 分别为 (x, y) 像素点处图像亮度在 x 方向和 y 方向的梯度，即图像 x 和 y。$I'_t \Delta t$ 即为两图之间的 (x, y) 坐标位置之间的亮度差。表示为 $\Delta I_t = I'_t \Delta_t$，即

$$[I'_x, I'_y] \begin{bmatrix} u \\ v \end{bmatrix} = -\Delta I_t \tag{1.5}$$

给定两张图像，$I'_x, I'_y, \Delta I_t$ 均为已知量，u, v 即待求的光流。

邻域光流相似假设

算法认为在每一个滑动窗口内每一个点的移动速度 u, v 一致。假设在一个大小为 $m \times m$ 的窗口内，图像的光流是一个恒定值，那么就可以得到以下条件方程

$$\begin{aligned} I_{x1} u + I_{y1} v &= -I_{t1} \\ I_{x2} u + I_{y2} v &= -I_{t2} \\ &\vdots \\ I_{xn} u + I_{yn} v &= -I_{tn} \end{aligned} \tag{1.6}$$

用矩阵形式表示为

$$\begin{bmatrix} I_{x1} & I_{y1} \\ I_{x2} & I_{y2} \\ \vdots & \vdots \\ I_{xn} & I_{yn} \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} -I_{t1} \\ -I_{t2} \\ \vdots \\ -I_{tn} \end{bmatrix} \tag{1.7}$$

记作 $Ax = -b$，可求得光流 $x = (u, v)$，根据最小二乘法

$$A^T A x = A^T (-b) \tag{1.8}$$

则最终所有求解的光流速度矢量为

$$x = (A^T A)^{-1} A^T (-b) \tag{1.9}$$

其中，要求 $(A^T A)$ 可逆，如果 $(A^T A)$ 不可逆，方程可能出现多解，即**孔径问题** (从圆孔中观察三种移动的条纹的变化，是一致的，从而无法通过圆控得到条纹的真实移动方向，即光流方向)。

缺点

这个算法不足在于它不能产生一个密度很高的流向量，例如在运动的边缘和黑大的同质区域中的微小移动方面流信息会很快褪去，它的优点在于有噪声存在的鲁棒性还是可以的。

改进算法

Lucas-Kanade 的改进算法为 *LK 金字塔算法*。*LK 金字塔算法*算法的约束条件为：*微小运动，亮度不变以及区域一致性*。

1.2. *Dis-FLow* Algorithm

Dis Flow 是在 *LK* 金字塔算法的基础上，提出了一个**逆向搜索**，从而节省计算，提高性能。

1.3. *Horn-Schunck* Algorithm

HS 算法用一种全局方法估计图像的稠密光流场。*HS* 光流计算基于两个假说

亮度不变假设

运动物体的灰度在很短的时间间隔内保持不变；

光流场平滑假设

场景中属于同一个物体的像素形成光流场应当十分平滑，只有在物体边界的地方才会出现光流的突变，但这只占图像的一小部分，总体来看图像的光流场应当是平滑的。

缺点

鲁棒性较差，无法准确处理原本图像梯度较大的边界区域；并没有解决针对变换幅度比较大情况如何准确求解光流的问题。

1.4. *Brox-FLow* Algorithm

Brox-FLow 在 *HS* 的算法基础上提出了一个新的变分方法。基于三个假设

灰度值恒定假设

梯度恒定假设

光流场平滑假设

1.5. 总结

传统光流算法主要分为稀疏光流和稠密光流。

1. 稀疏光流仅仅只跟踪图像中某些点的子集，计算方便，但对于特征点的选取和精确匹配困难；
2. 稠密光流对区域内的所有像素点进行跟踪，计算量大；

传统光流算法通过若干假设对光流估计进行建模，其中亮度不变假设为光流算法中的基础假设，通过假设对模型进行约束，利用变分法求取最优值，从而得到光流信息。但传统光流算法对于最优值的求解，大多需要多次迭代，计算量很大。

Chapter 2

Photoslip Algorithm：基于深度学习

本篇文章来自https://zhuanlan.zhihu.com/p/660866515?utm_campaign=shareopn&utm_medium=social&utm_psn=1815096437689683968&utm_source=wechat_session

2.1. Flow Net

采用端对端的学习方法来预测光流：给定一个由图像对和光流的 *ground truth* 组成的数据集，训练一个卷积神经网络直接从图像对中预测光流信息。

假设输入端为两幅序列图像 I_1, I_2，输出端为预测的光流 F

$$F = CNN(\theta, I_1, I_2) \tag{2.1}$$

θ 为待学习参数。

网络结构

在 *FlowNet* 中，使用的网络为 *Unet*，分为 *Decoder* 和 *Encoder*。

根据输入方式的不同，*FlowNet* 提出了两种网络结构

1. *FLowNetSimple*

直接将两张图像按通道重叠后输入，*channel* 为 6。

2. *FlowNetCorr*

解码器

总结

首个将 CNN 网络用于光流估计，精度和效果比不上传统基于能量最小化的方法。

2.2. PWC Net

原理

采用特征金字塔结构作为特征提取网络，其包含三个部分

1. *pyramid*;
2. *wrap*;
3. *cost*;
4. *volume*;

网络结构

1. *pyramid feature extractor*;
2. *wraping layer*;
3. *cost volume layer*;
4. *optical flow estimator*;
5. *context networks*

损失函数

训练使用 L_2 损失

$$\mathcal{L}(\Theta) = \sum_{l=l_0}^{L} \alpha_l \sum_{\mathbf{x}} \left| \mathbf{w}_\Theta^l(\mathbf{x}) - \mathbf{w}_{GT}^l(\mathbf{x}) \right|_2 + \gamma |\Theta|_2 \tag{2.2}$$

其中，Θ 为不同金字塔层的特征提取参数和光流估计参数，$\mathbf{w}_\Theta^l(\mathbf{x})$ 表示网络预测的第 l 金字塔层的光流，$\mathbf{w}_{GT}^l(\mathbf{x})$ 表示标签值。

总结

结合了 *coarse-to-fine* 与传统方法中的多种思想，效果优于传统方法，但是模型参数多，计算量大。

2.3. IRR

原理

受传统方法能量最小化的启发，基于共享权重，对光流进行迭代化细化，利用**双向估计**和**遮挡推理**提高光流精度。

网络结构

1. *FlowNetS*：在不增加参数或使训练过程复杂化的情况下，通过迭代地重复使用单个网络来不断优化先前的光流估计

$$\mathbf{f}_{fw}^i = D(E(\mathbf{I}_1), \omega(E(\mathbf{I}_2), \mathbf{f}_{fw}^{i-1})) + \mathbf{f}_{fw}^{i-1} \tag{2.3}$$

2. *Based PWC-Net*：用一个共享解码器来替代多个解码器，该解码器迭代地改进所有金字塔级别的输出；

3. 遮挡估计：在 *encoder* 末端的第一帧增加了一个额外的 *decoder*$\mathbf{o}_1^i$ 估计遮挡，与光流 decoder 平行，$\mathbf{o}_1^i$ 的输入与光流的 *decoder* 相同，输出通道的数量为 1；

4. 双向估计：

5. 对光流和遮挡的双边优化：

6. *occlusion* 上采样层

损失函数

分别对光流和遮挡定义 L_2 损失和交叉熵损失

总结

双向估计对于光流精度的提升不高，但利用正向和反向的一致性能够估计更加准确的遮挡；

2.4. RAFT

为了优化光流估计，RAFT 提取像素特征，对所有像素建立多尺度 correlation volume，在 correlation volume 上进行邻域采样得到 lookups 特征（增强特征相关性，也可以理解为感受野），之后直接使用以 CNN-GRU 为基础的迭代优化网络，在完整尺寸上对光流估计迭代优化。

2.5. GMA

遮挡问题是指像素点在第一帧中存在，但在第二帧（匹配帧）中不存在的问题。以往的算法使用 cost volume 去计算两帧之间的相关性，但这种相关性对于被遮挡区域来说是毫无意义的。当两帧像素之间不存在匹配信息时（第二帧像素在第一帧中不存在），单个物体的运动通常时均匀的，运动信息必须从其他像素开始传播。也就是说，未被遮挡的像素点的运动信息可以传播到被遮挡的像素点。但利用传统卷积进行局部传播的缺点在于其范围有限，因此提出 global motion aggregation GMA 模块（全局聚合模块），以 Raft 为基础，利用图像自相似解决遮挡问题，该模块基于 transformer，可以更好的去学习两帧像素之间的依赖关系（哪个像素与它有关或它属于哪个物体）。

References

[1] I. Surname, I. Surname, and I. Surname. “The Title of the Article”. In: *The Title of the Journal* 1.2 (2000), pp. 123–456.

Chapter A

Source Code Example

Adding source code to your report/thesis is supported with the package `listings`. *An example can be found below. Files can be added using* `\lstinputlisting[language=<language>]{<filename>}`.

```
"""
ISA Calculator: import the function, specify the height and it will return a
list in the following format: [Temperature,Density,Pressure,Speed of Sound].
Note that there is no check to see if the maximum altitude is reached.
"""

import math
g0 = 9.80665
R = 287.0
layer1 = [0, 288.15, 101325.0]
alt = [0,11000,20000,32000,47000,51000,71000,86000]
a = [-.0065,0,.0010,.0028,0,-.0028,-.0020]

def atmosphere(h):
    for i in range(0,len(alt)-1):
        if h >= alt[i]:
            layer0 = layer1[:]
            layer1[0] = min(h,alt[i+1])
            if a[i] != 0:
                layer1[1] = layer0[1] + a[i]*(layer1[0]-layer0[0])
                layer1[2] = layer0[2] * (layer1[1]/layer0[1])**(-g0/(a[i]*R))
            else:
                layer1[2] = layer0[2]*math.exp((-g0/(R*layer1[1]))*(layer1[0]-layer0[0]))
    return [layer1[1],layer1[2]/(R*layer1[1]),layer1[2],math.sqrt(1.4*R*layer1[1])]
```

Chapter B

Task Division Example

If a task division is required, a simple template can be found below for convenience. Feel free to use, adapt or completely remove.

表 B.1: Distribution of the workload

	Task	Student Name(s)
	Summary	
Chapter 1	Introduction	
Chapter 2		
Chapter 3		
Chapter *		
Chapter *	Conclusion	
	Editors	
	CAD and Figures	
	Document Design and Layout	