（论文解读）RAFT: Recurrent All-Pairs Field Transforms for Optical Flow

目录

论文解读之：

RAFT: Recurrent All-Pairs Field Transforms for Optical Flow

由于前段时间太忙了，这篇博客拖了很久才发布出来，一直只写了一半扔在草稿箱里。
今天我来试着分析一下近期光流最强论文：RAFT: Recurrent All-Pairs Field Transforms for Optical Flow，该论文发表于ECCV2020并获得了best paper。在阅读过后就能发现它当之无愧。下面我们就来一起看叭。
该文章地址：https://arxiv.org/pdf/2003.12039.pdf
作者公开了代码：https://github.com/princeton-vl/RAFT

写在前面

光流估计领域从FlowNet开始就认定了类似于UNet的结构，即下采样加上采样的结构。但这样的结构是不是真的合适？答案并不一定。我们设计一个网络需要能够解释为什么这样设计，可是使用这种UNet的结构，我们只能把它看做一个变换器，将数据从空间像素域变换到光流值域。恰好UNet这样的结构可以以图片为输入，以图片为输出，同时下采样的过程可以注意到局部的信息，所以很快成为光流估计界的主流网络风格。

在我看来，光流估计需要注意以下一些点：

- 网络输入为图片，输出为图片。
- 同时注意局部与全局的特征（由于光流是一个比较稠密的估计任务，如果不对整个图片内局部的光流估计值进行约束或监督，可能就会导致网络沿着其他方向去拟合总的损失函数，最终损失虽然降低，但效果并不理想）
- 虽然光流的信息来自于前后帧间的信息，而光流估计也需要一定的纹理信息和上下文信息（由于光流图与原图轮廓基本一致）。
- 在前向传播过程中不能丢失太多信息，因为光流估计需要的信息量较大。这也限制了下采样层数不能太深，而且跳层连接能够提升性能。
- **迭代优化**

而这篇论文所提出的方法，注意到了所有的点。

回顾光流领域

光流估计领域先后有几种主流网络。一开始是FlowNet以及FlowNet2，后来被PWCNet所替代，后来又有了IRR这样的迭代式网络，最后到我们今天所说的RAFT。其实光流估计方法的发展过程就是从UNet的结构逐渐跳出的过程。
而且，由于光流所描述的是两帧图像间的关系，通过已估计出的光流可以把后一帧向前一帧映射，映射之后的两帧可以继续估计光流的残差，如此循环，可以将残差不断叠加，所以，加入迭代优化的结构是很好的思路。
PWCNet试图加入迭代优化，熟悉PWCNet的朋友都知道，PWCNet中间的那个从下到上的过程便是在迭代优化。但由于其思路仍然被UNet限制，没能走出上下采样的结构，限制了迭代的次数，即下采样几层，迭代上采样也只能设计几层。

后来，IRR这样的网络想进一步推动迭代优化的过程，将整个PWCNet网络迭代很多次，不断地计算光流的残差，再求和，达到了更好的效果。
本篇论文巧妙地将下采样与迭代优化结合，完美地跳出来UNet对思想的限制。我认为将成为接下来一段时间的光流估计方法的主要思想，即使用某些信息（局部与全局都要考虑到），输入到循环网络里将光流的估计值迭代优化。

RAFT

文章所提出的网络结构如下图所示：

网络整体分为三个部分：Feature Encoder、Context Encoder、Update。
在前端，网络分为两支，即Feature Encoder以及Context Encoder，分别将图片特征以及上下文进行编码，在后面，通过将前面两支的编码融合，并使用GRU进行迭代，最终得到光流的估计值。下面，我们分块对这些网络进行解说。

Feature Encoder与Context Encoder

这两个编码层采用如UNet的编码层相同的结构，即使用几层卷积层将原图缩小为原来的八分之一，减小后续网络的计算量，同时相当于进行编码操作。不同的是，两个编码器输入图片不同，Feature Encoder需要输入前后两帧的照片，而Context Encoder只输入前一帧的照片。这样的想法很显而易见：Feature Encoder用于提取光流的特征，所以从前后两帧图片中提取特征，用于后续的光流估计，而Context Encoder顾名思义提取上下文信息，保证估计出的光流图保持与原图相同的上下文信息以及位置对应。这样的思想简洁而又强大，乍一看很简单，但不得不承认这样非常实用。

Update

在这个部分，作者将上述的两个编码器输出的结果进行融合，同时进行GRU迭代。我们知道光流在迭代时相当于逐渐优化，不断逼近真实光流。而且这样的结构有利于适应不同环境，如果 硬件　条件限制，可以减少迭代次数，舍弃估计准确率，增快速度，而当硬件环境富余时，可以增加迭代次数，提高准确率。
迭代时，作者先把Feature Encoder对两张图片编码的结果进行相似度的提取。由于光流即使找到前两帧之间相似度最大的像素并进行对应，该相似度并不仅仅是像素值的相似，是描述子的相似，所以可以看做两张图提取出的特征相似，即Feature Encoder输出的编码结果中寻找相似的位置进行对应。所以作者使用了最简单的点积相似度衡量。对两张图两两像素之间进行点积相似度的计算，得到一个大小为H*W*H*W的相似度块，如下图所示：

写成公式如下所示：

其中C为相似度块，g_θ 为Feature Encoder。
同时，作者使用了相似度金字塔，用于关注到不同尺度的相似度，这样的操作可以同时保证微小运动和剧烈运动同时被观测到。在论文中，作者使用了四层金字塔，即通过Pool的方式将上述得到的相似度块分别缩小。若将四层金字塔表示为C^1, C^2, C^3, C^4，则其大小分别为$H \times W \times H/2^k \times W/2^k$。
接着，这篇文章最灵魂的地方到了：查表。
对于已经估计到的光流(f^1, f^2)，可以将每个像素点(u, v)按照已经估计到的光流移动到其在下一帧对应位置$(u + f^1(u), v + f^2(v))$。如果光流估计准确， 这个位置上的相似度应该较大。而在迭代过程中，我们将进一步准确这个位置，所以我们将相似度块中在这个位置周围的数全部取出来按顺序排好。这个周围是多大呢？这个可以定义，即如下公式：

将这些值取出来后拉成一个向量，作为这一个点的特征，每个点的特征都按这样查表进行填充。这样查表的操作只需要计算一次相似度块（由于相似度块太大，计算其实比较耗时），节省了计算时间。而且在原理上合情合理，这样得到的新特征图可以作为这一层的GRU输入对上一层的光流进行进一步修正。而上面Context Encoder得到的特征图将与这个特征图拼接，作为GRU的输入。GRU的hidden　state作为光流残差的估计值，加到上一层估计出的光流中。这样一层一层加下去得到最后迭代出的光流。光流初始化为0。

总结

这篇文章理解起来非常容易，思路也非常清晰，简单。但是每一步都是可以解释为什么这样设计的，不像UNet那样茫然。所以这也是这个网络能获得成功的原因。反观我上文中提到的光流网络需要具备一些点，RAFT巧妙地将它们都融合在了一起。同时，这个网络也成功跳出了UNet这样结构的思想限制，真正的产生了新的光流网络，应该会成为今后光流网络研究的热点方式：即特征匹配与迭代的方式。