【论文简述及翻译】RAFT: Recurrent All-Pairs Field Transforms for Optical Flow（ECCV 2020）

一、论文简述

1. 第一作者：Zachary Teed

2. 发表年份：2020

3. 发表期刊：ECCV，best paper

4. 关键词：光流估计、端到端训练、迭代优化、GRU

5. 探索动机：光流是估计视频帧之间的逐像素运动的任务。这是一个长期存在但仍未解决的视觉问题。目前最好的系统仍被一些难题所制约，包括快速移动的物体、遮挡、运动模糊和无纹理表面等。不管是传统的还是基于深度学习的经典光流预测算法，都存在如上缺点，而且无论怎么优化，这些缺点都会因为框架的设计而一直存在。

6. 工作目标：是否可以跳出原先的设计思路，设计一个性能更好、训练更容易并能很好地推广到新场景的新结构？同时实现如下要求：

- 网络输入为图片，输出为图片（端到端网络）；
- 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，可能就会导致网络沿着其他方向去拟合损失函数，最终损失虽然降低，但效果并不理想，因此需要同时考虑局部与全局的特征；
- 虽然光流的信息来自于前后帧间的运动信息，但光流估计也需要一定的纹理信息和上下文信息用于匹配像素点，光流图与原图轮廓也基本一致；
- 网络模块参数量太大，堆叠多个模块会导致网络计算成本太高，因此限制了层数不能太深；
- 延续经典的迭代优化思路。

7. 核心思想：RAFT以高分辨率保持和更新单个固定的光流场；RAFT的更新算子是循环的、轻量级的，并且共享权重；更新算子由一个卷积GRU组成，在4D多尺度相关体上进行查找。

8. 实现方法：

- **特征提取：**通过特征编码器从两个输入图像中提取每个像素的特征，通过上下文编码器从I1中提取特征；
- **计算视觉相似度：**通过计算所有特征向量对的内积，构造一个4D W×H×W×H相关体，在4D体最后2维上进行多尺度池化，以构建一组多尺度体；
- **相关查找：**定义查找算子LC，它通过从相关金字塔中索引生成特征图；
- **迭代更新：**基于GRU的循环更新算子，从相关体中检索值并迭代更新初始化为零的光流场；
- **上采样：**一个新颖的通过卷积层学习的上采样方式；
- **监督：**计算伴随权重呈指数增长的预测光流和真实光流之间的l1距离。

9. 实验结果：

- **最好的精度：**在KITTI上，RAFT实现了5.10% 的F1-all误差，比已公布的最佳结果(6.10%)减少了16%的误差。在Sintel（final pass）上，RAFT获得了2.855 像素的EPE误差，比已公布的最佳结果（4.098像素）减少了30%的误差。
- **强泛化性：**仅在合成数据上训练时，在KITTI上RAFT取得了5.04像素的端点误差，与在相同数据上训练的之前的最佳深度网络（8.36像素）相比，误差减少了40%。
- **高效率：**在1080Ti GPU上RAFT以每秒10帧的速度处理1088×436视频。它训练的迭代次数比其他结构少10倍。具有1/5参数的较小版本的RAFT以每秒20帧的速度运行，然而在Sintel上仍然优于以前所有的方法。

10.论文&代码下载：

https://arxiv.org/pdf/2003.12039.pdf

https://github.com/princeton-vl/RAFT

二、论文翻译

RAFT: Recurrent All-Pairs Field Transforms for Optical Flow

摘要

我们首次提出了RAFT，这是一种用于光流的全新的深度网络结构。RAFT 逐像素提取特征，为所有像素对构建多尺度4D相关体，并通过循环单元在相关体上进行查找，以迭代更新光流场。RAFT实现了最先进的性能。在KITTI上，RAFT取得了5.10%的F1-all误差，比已公布的最佳结果 (6.10%)降低了16%。在Sintel（final pass）上，RAFT获得了2.855 EPE误差，比已发布的最佳结果（4.098像素）减少了30%的误差。此外，RAFT具有很强的跨数据集泛化能力，以及在推理时间、训练速度和参数量方面具有很高效率。

1. 介绍

光流是估计视频帧之间的逐像素运动的任务。这是一个长期存在但仍未解决的视觉问题。最好的系统仍被一些难题所制约，包括快速移动的物体、遮挡、运动模糊和无纹理表面。

传统上，光流被视为在一对图像间的密集位移场的空间上的人工优化问题。通常，优化的目标是平衡数据项和正则化项，数据项促进视觉上相似的图像区域的对齐，正则化项对运动的合理性施加先验。这种方法取得了相当大的成功，但由于人工设计的优化目标很难对各种极端情况具有鲁棒性，因此很难进一步发展。

最近，深度学习已被证明是一种很有前途的可以替代传统方法的方案。深度学习可以回避将优化问题公式化并且训练网络直接预测光流。当前的深度学习方法已取得与最佳传统方法相匹敌的性能，同时在推断时速度明显更快。下一步研究的一个关键问题是设计一个性能更好、训练更容易并能很好地推广到新场景的新结构。

我们首次提出了RAFT，这是一种用于光流的新的深度网络结构。RAFT拥有以下优势：

最好的精度：在KITTI上，RAFT实现了5.10% 的F1-all误差，比已公布的最佳结果(6.10%)减少了16%的误差。在Sintel（final pass）上，RAFT获得了2.855 像素的EPE误差，比已公布的最佳结果（4.098像素）减少了30%的误差。

强泛化性：仅在合成数据上训练时，在KITTI上RAFT取得了5.04像素的端点误差，与在相同数据上训练的之前的最佳深度网络（8.36像素）相比，误差减少了40%。

高效率：在1080Ti GPU上RAFT以每秒10帧的速度处理1088×436视频。它训练的迭代次数比其他结构少10倍。具有1/5参数的较小版本的RAFT以每秒20帧的速度运行，然而在Sintel上仍然优于以前所有的方法。

RAFT有三个主要组成部分：（1）为每个像素提取特征向量的特征编码器；(2)相关层产生所有像素对的4D相关体，接着进行池化以产生较低分辨率的体；(3)基于GRU的循环更新算子，从相关体中检索值并迭代更新初始化为零的光流场。图1说明了RAFT的设计。

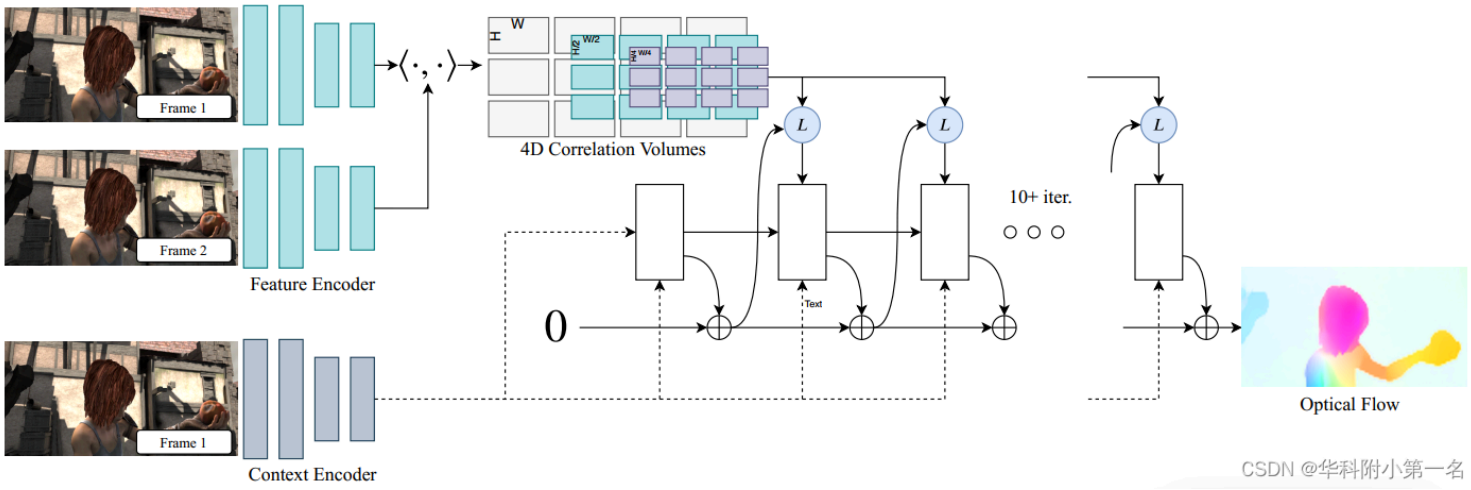


图1：RAFT由3个主要部分组成：(1) 特征编码器，从两个输入图像中提取每个像素的特征，以及仅从I1中提取特征的上下文编码器。(2)相关层，通过计算所有特征向量对的内积，构造一个4D W×H×W×H相关体。在4D体最后2维上进行多尺度池化，以构建一组多尺度体。(3)更新算子，通过使用当前估计从一组相关体中查找值来循环更新光流。

RAFT结构是由传统的基于优化的方法推动的。特征编码器提取每个像素的特征。相关层计算像素之间的视觉相似性。更新算子模拟迭代优化算法的步骤。但与传统方法不同，特征和运动先验不是人工制作的，而是分别由特征编码器和更新算子学习的。

RAFT的设计灵感来自许多现有的工作，但本质上是新的。首先，RAFT以高分辨率保持和更新单个固定的光流场。这与之前工作中流行的从粗糙到精细的设计不同，这类工作首先在低分辨率下估计光流，然后在高分辨率下进行上采样和改进。通过在单个高分辨率的光流场上运行，RAFT克服了从从粗糙到精细级联的几个限制：在粗分辨率下难以从误差中恢复，容易错过快速移动的小物体，以及为了训练多阶段级联通常需要多次训练迭代（通常超过1M）。

其次，RAFT的更新算子是循环的、轻量级的。近期许多的工作都包括了某种形式的迭代改进，但没有在迭代之间使权重发生联系，并且因此被限制在固定次数的迭代中。据我们所知，IRR是唯一循环的深度学习方法。它使用FlowNetS或PWC-Net作为循环单元。使用FlowNetS 时，受网络大小（38M参数）的限制，最多只能应用5次迭代。使用PWC-Net时，迭代受到金字塔层数的限制。相比之下，我们的更新算子只有270万个参数，并且可以在推断过程中应用100多次而不会发散。

第三，更新算子有一个新颖的设计，它由一个卷积GRU组成，它在4D多尺度相关体上进行查找；相比之下，先前工作中的改进模块通常仅使用普通卷积或相关层。

我们在Sintel和 KITTI上进行了实验。结果表明，RAFT在两个数据集上都达到了最先进的性能。此外，我们通过广泛的消融研究验证了RAFT的各种设计选择。

2. 相关工作

光流作为能量最小化。传统上光流视作能量最小化问题，它在数据项和正则化项之间进行平衡。Horn和Schnuck使用变分框架将光流表述为连续优化问题，并且能够通过执行梯度步骤来估计稠密的光流场。Black和Anandan首次使用鲁棒的估计框架解决了过度平滑和噪声敏感性的问题。TV-L1用L1数据项和完全变分正则化代替了二次惩罚，这使得运动可以不连续性，并且能更好地处理异常值。通过定义更好的匹配代价和正则化项也有了提高。

这种连续的公式保持对光流的单一估计，该估计在每次迭代中都会被改进。为了确保是平滑的目标函数，使用一阶泰勒近似来对数据项进行建模。因此，它们仅适用于小位移。为了处理大位移，很多方法使用了从粗糙到精细的策略，该策略使用图像金字塔在低分辨率下估计大位移，然后在高分辨率下改进小位移。但是这种从粗糙到精细的策略可能会错过快速移动的小物体，并且难以从早期错误中恢复。与连续方法一样，我们保持对光流的单一估计，每次迭代都会对其进行改进。然而，由于我们在高分辨率和低分辨率下为所有像素对建立相关体，因此每次局部更新都使用了小位移和大位移相关的信息。此外，我们的更新算子不使用数据项的亚像素泰勒近似，而是学习去提出下降方向。

最近，光流也视为使用全局目标的离散优化问题。这种方法的一个挑战是搜索空间的尺寸过大，因为每个像素按理来说都可以与另一帧中的数千个点配对。Menez等人使用特征描述子进行空间搜索，并使用消息传递来近似全局MAP估计。Chen等人表明，通过使用距离变换，可以轻松解决光流场全空间的全局优化问题。DCFlow通过使用神经网络作为特征描述子有了进一步的提高，并构建了所有特征对的4D代价体。然后使用半全局匹配 (SGM) 算法处理4D代价体。与DCFlow一样，我们也构建了学习得到的特征的4D代价体。然而，我们没有使用SGM处理代价体，而是使用神经网络来估计光流。我们的方法是端到端可微分的，这意味着特征编码器可以与网络的其余部分一起训练，将最终光流估计的误差直接最小化。相比之下，DCFlow需要网络使用像素之间的嵌入损失进行训练；它不能直接在光流上进行训练，因为它们的代价体处理过程是不可微的。

直接光流预测。已经训练过的神经网络可以直接预测一对帧之间的光流，完全避开了优化问题。在近期许多工作中，从粗糙到精细的过程已成为一种流行的组成部分。相比之下，我们的方法保持和更新单个高分辨率的光流场。

光流的迭代改进。近期许多工作都使用迭代改进来提升光流和相关任务的结果。Ilg等人通过将多个FlowNetS和FlowNetC模块连续堆叠，将迭代改进应用于光流。SpyNet、PWC-Net、LiteFlowNet和VCN使用从粗糙到精细的金字塔进行迭代改进。这些方法与我们方法主要的区别在于它们在迭代之间不共享权重。

与我们的方法更密切相关的是IRR，它建立在FlownetS和PWC-Net结构之上，但在改进网络之间共享权重。使用FlowNetS时，受网络大小（38M参数）的限制，最多只能进行5次迭代。在使用PWC-Net时，迭代受到金字塔层数的限制。相比之下，我们使用了一个更简单的改进模块（2.7M参数），它可以在推断阶段进行100多次迭代而不会发散。我们的方法也与Devon有相似之处，即在没有形变和固定分辨率更新的情况下构建代价体。但是，Devon没有任何循环单元。在较大的位移方面，它也与我们的不同。Devon使用扩张的代价体处理大位移，而我们的方法池化多个分辨率的相关体。

我们的方法还与TrellisNet和深度平衡模型 (DEQ)有联系。TrellisNet在大量层上使用深度绑定权重，DEQ通过直接求解固定点来模拟无限数量的层。TrellisNet和DEQ是为序列建模任务而设计的，但我们采用了使用大量权重绑定（权重共享）单元的想法。我们的更新算子使用修正后的GRU块，它类似于在TrellisNet中使用的LSTM块。我们发现这种结构使我们的更新算子更容易收敛到一个固定的光流场。

学习优化。视觉中的许多问题都可以表述为优化问题。这促使一些工作将优化问题加入到网络结构中。这些工作通常使用网络来预测优化问题的输入或参数，然后通过求解器隐式地或展开每个步骤，反向传播梯度来训练网络权重。然而，这种技术仅限于目标容易定义的问题。

另一种方法是直接从数据中学习迭代更新。这些方法是由一阶优化器所推动的，如原始双混合梯度(PDHG)可以表示为一系列迭代更新的步骤。Adler等人没有直接使用优化器，而是提出构建一个模拟一阶算法更新的网络。这种方法已应用于图像去噪、断层重建和新视图合成等可逆问题。TVNet将TV-L1算法实现为计算图，从而可以训练TV-L1参数。然而，TVNet的操作直接基于强度梯度而不是学习的特征，这限制了在诸如Sintel等具有挑战性的数据集上可实现的精度。

我们的方法可以看作是学习优化：我们的网络使用大量更新块来模拟一阶优化算法的步骤。然而，与之前的工作不同，我们从未明确定义某个优化目标的梯度。相反，我们的网络从相关体中检索特征以提出下降方向。

3. 方法

给定一对连续的RGB图像，I1，I2，我们估计一个密集的位移场(f1，f2)，它将I2中的每个像素(u，v)映射到在I2中其对应的坐标(u‘，v’) = (u+f1 (u)，v+f2(v))。图1给出了我们方法的概述。我们的方法可以提炼为三个阶段：(1)特征提取，(2)计算视觉相似性，以及(3)迭代更新，其中所有阶段都是可微分的，并且组成一个端到端的可训练的结构。

3.1. 特征提取

我们使用卷积网络从输入图像中提取特征。特征编码器网络应用于I1和I2，并将输入图像映射为较低分辨率的稠密特征图。我们的编码器gθ输出1/8分辨率的特征gθ：$\mathbb{R}^{H\times W\times 3}\mapsto\mathbb{R}^{H/8\times W/8\times D}$，我们设置D=256。特征编码器由6个残差块组成，2个分辨率为1/2，2个分辨率为1/4，2个分辨率为1/8（更多细节详见补充材料）。

我们额外使用上下文网络。上下文网络仅从第一个输入图像I1中提取特征。上下文网络的结构hθ与特征提取网络相同。特征网络gθ和上下文网络hθ共同构成了我们方法的第一阶段，只需执行一次。

3.2. 计算视觉相似性

我们通过在所有对之间构建一个全相关体来计算视觉相似性。给定图像特征**g**θ(**I**1) ∈*R***H**×**W** ×**D** 和**g**θ(**I**2) ∈*R***H**×**W** ×**D** ，通过计算所有特征向量对之间的点积形成相关体。相关体C可以作为单个矩阵乘法有效地计算：

$$\mathbf{C}(g_\theta(I_1), g_\theta(I_2)) \in \mathbb{R}^{H\times W\times H\times W}, \qquad C_{ijkl} = \sum_h g_\theta(I_1)_{ijh} \cdot g_\theta(I_2)_{klh} \tag{1}$$

相关金字塔：我们通过以卷积核大小1、2、4、8以及同等步幅（如图2）进行池化相关体的最后两个维度，构建了一个4层金字塔{C1，C2，C3，C4}。因此，体Ck的维度为H×W×H/2k×W/2k。这组体提供了关于大位移和小位移的信息；但是，通过保持前2个维度（I1的维度），我们可以保持高分辨率信息，从而使我们的方法能够恢复小型快速移动物体的运动。

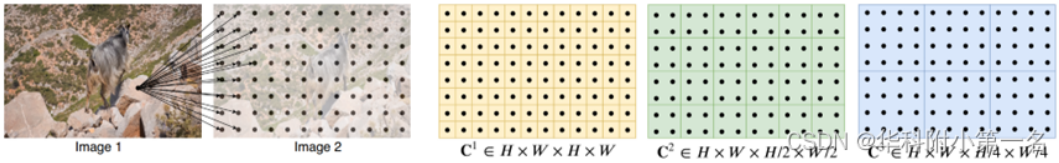


图2: 构建相关体。这里我们描绘了一个完整的4D体的二维切片。对于I1中的特征向量，我们取其与I2中所有对的内积，生成一个4D W×H×W×H体 (I2中的每个像素生成一个2D 响应图)。使用卷积核大小为1、2、4、8的平均池化对相关体池化。

相关查找：我们定义了一个查找算子LC，它通过从相关金字塔中索引来生成特征图。给定一个当前光流的估计(f1，f2)，我们将I1中的每个像素x= (u，v)映射到I2中的估计的对应：I_2: $\mathbf{x}' = (u + f^1(u), v + f^2(u))$，然后我们在x'周围定义一个局部网格

$$\mathcal{N}(\mathbf{x}')_r = \{\mathbf{x}' + \mathbf{dx} \mid \mathbf{dx} \in \mathbb{Z}^2, ||\mathbf{dx}||_1 \leq r\} \tag{2}$$

作为使用L1距离在x'的r个单位的半径内的整数偏移量的集合。我们使用局部邻域N(x')r从相关体中进行索引。由于N (x')r是实数网格，我们使用双线性采样。

我们在金字塔上的所有层上执行查找，使用网格N (x'/ 2k)r对层级k的相关体Ck进行索引。跨层的恒定半径意味着在较低层次有更大的上下文：在最低层级k=4，使用半径4对应原始分辨率下256像素的范围。然后将每个层级的值连接到单个特征图中。

高分辨率图像的高效计算：所有对的相关尺寸为O(N2)，其中N是像素数，但只需要计算一次，并且在迭代次数M中是恒定的。但是，存在一种与我们的方法等效的实现，即利用内积和平均池化的线性来改变尺寸O(NM)。考虑层级m的代价体，Cmijkl和特征图g(1)=gθ(I1)，g(2)=gθ(I2)：

$$\mathbf{C}^m_{ijkl} = \frac{1}{2^{2m}} \sum_p^{2^m} \sum_q^{2^m} \langle g^{(1)}_{i,j}, g^{(2)}_{2^m k+p, 2^m l+q} \rangle = \langle g^{(1)}_{i,j}, \frac{1}{2^{2m}} (\sum_p^{2^m} \sum_q^{2^m} g^{(2)}_{2^m k+p, 2^m l+q}) \rangle$$

这是2m×2m网格中相关响应的平均值。这意味着Cmijkl的值可以通过计算用核大小为2m×2m池化过的特征向量gθ(I1)ij和gθ(I2)之间的内积所得到。

在这个迭代实现中，我们不预先计算相关，而是预先计算池化后的图像特征图。在每次迭代中，我们按需计算每个相关值——仅在查找时。因此是O(NM)的复杂度。

我们凭经验发现，预先计算所有对很容易实现并且没有瓶颈，因为GPU上的矩阵线程经过高度优化——即使对于1088x1920视频，它也只需要总推断时间的17%。请注意，如果它成为瓶颈，我们总是可以切换到替代的实现。

3.3. 迭代更新

我们的更新算子从初始起点f0=0开始，估计了一系列光流估计{ f1，…，fN }。每次迭代，它都会产生一个更新方向Δf，并应用于当前估计：fk+1=Δf+fk+1。

更新算子将光流、相关和潜在隐藏状态作为输入，并输出更新Δf和更新后的隐藏状态。我们设计更新算子结构的目的是模仿优化算法的步骤。因此，我们使用绑定权重并使用有界激活来促进收敛到一个固定点。我们训练更新算子来执行更新，使得序列收敛到一个固定点fk→f*。

初始化：默认情况下，我们将光流场全部初始化为0，但我们的迭代方法让我们可以灵活地尝试替代方案。当应用于视频时，我们测试了热启动初始化，其中来自前一对帧的光流正向投影到下一对帧，并使用最近领域插值填充了遮挡间隙。

输入：给定当前光流估计fk，我们使用它从如第3.2节所述的相关金字塔中检索相关特征。然后由2个卷积层处理相关特征。此外，我们将2个卷积层用于光流估计本身以生成光流特征。最后，我们直接从上下文网络注入输入。然后将相关、光流和上下文特征进行连接作为输入特征图。

更新：更新算子的核心组成部分是一个基于GRU单元的门控激活单元，其中将全连接层替换为卷积：

$$z_t = \sigma(\text{Conv}_{3\text{x}3}([h_{t-1}, x_t], W_z)) \tag{3}$$
$$r_t = \sigma(\text{Conv}_{3\text{x}3}([h_{t-1}, x_t], W_r)) \tag{4}$$
$$\tilde{h}_t = \tanh(\text{Conv}_{3\text{x}3}([r_t \odot h_{t-1}, x_t], W_h)) \tag{5}$$
$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t \tag{6}$$

其中xt是先前定义的光流、相关和上下文特征的连接。我们还尝试了一个可分离的ConvGRU单元，我们用两个GRU替换3×3卷积：一个具有1×5卷积，一个具有5×1卷积，用来增加感受野，而不会显着增加模型的大小。

光流预测：GRU输出的隐藏状态通过两个卷积层来预测光流更新Δf。输出光流是输入图像分辨率的1/8。在训练和评估期间，我们对预测的光流场进行上采样以匹配真实值的分辨率。

上采样：网络以1/8分辨率输出光流。我们通过将每个像素的全分辨率光流作为其粗糙的分辨率领域的3x3网格的凸组合，将光流上采样到全分辨率。我们使用两个卷积层来预测H/8×W/8×(8×8×9)掩码，并在9格的领域的权重上执行softmax。通过使用掩码对领域进行加权组合，然后排列和重塑为一个H×W×2维光流场，得到最终的高分辨率光流场。该层可以在PyTorch中使用展开功能直接实现。

3.4 监督

在整个预测序列{ f1，…，fN }上，通过计算伴随权重呈指数增长的预测光流和真实光流之间的l1距离，我们监督该网络。给定真实光流fgt，损失定义为

$$\mathcal{L} = \sum_{i=1}^{N} \gamma^{N-i} ||\mathbf{f}_{gt} - \mathbf{f}_i||_1 \tag{7}$$

在实验中我们设置γ=0.8。

4. 实验

我们在Sintel和KITTI上评估RAFT。延续先前的工作，我们在FlyingChairs和FlyingThing上预训练我们的网络，然后在数据集上进行特定的微调。我们的方法在Sintel（clean和final pass）和KITTI上都取得了最好的性能。此外，我们在DAVIS数据集的1080p视频上测试了我们的方法，以证明我们的方法可以扩展到更高分辨率的视频中。

实现细节：RAFT在PyTorch中实现。所有模块都使用随机权重从头开始初始化。在训练期间，我们使用AdamW优化器并将梯度限制在[−1; 1]。除非另有说明，否则我们会在Sintel上进行32次光流更新并在KITTI上进行24次光流更新后评估。对于每次更新Δf + fk，我们只通过Δf分支反向传播梯度，并按照[20]中的建议通过fk分支将梯度归零。

训练计划：我们使用两个2080Ti GPU训练RAFT。我们在FlyingThings上进行100k次迭代预训练，批量大小为12，然后在FlyingThings3D上迭代训练100k次，批量大小为6。我们对RAFT在Sintel上再微调100k，数据集结合了来自Sintel、KITTI-2015和 HD1K的数据，这类似于MaskFlowNet和PWC-Net+的做法。最后，我们使用在Sintel上微调后模型的权重，在KITTI-2015上进行了额外的50k次迭代微调。补充材料中提供了有关训练和数据增强的详细信息。为了与之前的工作进行比较，我们还包括了仅在Sintel和仅在KITTI上进行微调时模型的结果。

4.1. Sintel

我们使用FlyingChairs→FlyingThings计划训练我们的模型，然后在Sintel数据集进行评估，使用拆分的训练数据作为验证集。结果如表1和图3所示，我们根据用于训练的数据拆分结果。C+T表示模型在FlyingChairs(C)和FlyingThings(T)上进行训练，而+ft表示模型在Sintel数据上进行了微调。与PWC-Net+和MaskFlowNet一样，在微调时我们使用包含来自KITTI和HD1K的数据。我们用不同的种子训练3次，并公布了使用在Sintel（训练）的clean pass上具中位数精度的模型的结果。

Training Data	Method	Sintel (train)		KITTI-15 (train)		Sintel (test)		KITTI-15 (test)
		Clean	Final	F1-epe	F1-all	Clean	Final	F1-all
-	FlowFields [7]	-	-	-	-	3.75	5.81	15.31
-	FlowFields++ [40]	-	-	-	-	2.94	5.49	14.82
S	DCFlow [47]	-	-	-	-	3.54	5.12	14.86
S	MRFlow [46]	-	-	-	-	2.53	5.38	12.19
C + T	HD3 [50]	3.84	8.77	13.17	24.0	-	-	-
	LiteFlowNet [22]	2.48	4.04	10.39	28.5	-	-	-
	PWC-Net [42]	2.55	3.93	10.35	33.7	-	-	-
	LiteFlowNet2 [23]	2.24	3.78	8.97	25.9	-	-	-
	VCN [49]	2.21	3.68	8.36	25.1	-	-	-
	MaskFlowNet [52]	2.25	3.61	-	23.1	-	-	-
	FlowNet2 [25]	2.02	3.54[1]	10.08	30.0	3.96	6.02	-
	Ours (small)	2.21	3.35	7.51	26.9	-	-	-
	Ours (2-view)	**1.43**	**2.71**	**5.04**	**17.4**	-	-	-
C+T+S/K	FlowNet2 [25]	(1.45)	(2.01)	(2.30)	(6.8)	4.16	5.74	11.48
	HD3 [50]	(1.87)	(1.17)	(1.31)	(4.1)	4.79	4.67	6.55
	IRR-PWC [24]	(1.92)	(2.51)	(1.63)	(5.3)	3.84	4.58	7.65
	ScopeFlow [8]	-	-	-	-	3.59	4.10	6.82
	Ours (2-view)	(0.77)	(1.20)	(0.64)	(1.5)	**2.08**	**3.41**	**5.27**
C+T+S+K+H	LiteFlowNet2[2] [23]	(1.30)	(1.62)	(1.47)	(4.8)	3.48	4.69	7.74
	PWC-Net+ [41]	(1.71)	(2.34)	(1.50)	(5.3)	3.45	4.60	7.72
	VCN [49]	(1.66)	(2.24)	(1.16)	(4.1)	2.81	4.40	6.30
	MaskFlowNet [52]	-	-	-	-	2.52	4.17	6.10
	Ours (2-view)	(0.76)	(1.22)	(0.63)	(1.5)	1.94	3.18	**5.10**
	Ours (warm-start)	(0.77)	(1.27)	-	-	**1.61**	**2.86**	

表1：Sintel和KITTI数据集的结果。在FlyingChairs(C)和FlyingThing(T)进行训练后，我们在Sintel(train)上测试了泛化性能，在clean和final pass上都优于现有所有的方法。底部两部分显示了我们的模型在数据集特定微调后在公共排行榜上的性能。S/K包括在Sintel上进行微调时仅使用Sintel数据和在KITTI上进行微调时仅使用KITTI数据的方法。+S+K+H包括在Sintel进行微调时结合了KITTI、HD1K和Sintel的数据的方法。我们的（warm-start）在Sintel的clean和final pass中均排名第一，在KITTI的所有光流方法中排名第一。（FlowNet2最初报告了Sintel视差分割的结果，3.54是他们的模型在标准数据上评估时的EPE。[23]发现HD1K数据在Sintel微调期间没有显着帮助，并在没有用它的情况下报告了结果。

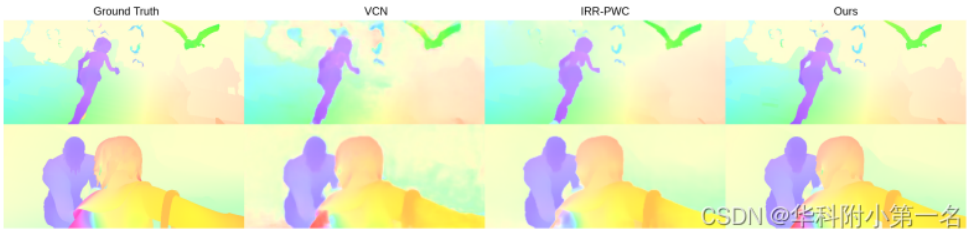


图3：在Sintel测试集上的光流预测。

当使用C+T进行训练时，我们的方法优于所有现有方法，尽管使用了明显更短的训练计划。我们的方法在Sintel(train) clean pass上实现了1.43的平均EPE（端点误差），比FlowNet2的误差低29%。这些结果证明了很好的跨数据集的泛化性。泛化性更好的原因之一是我们的网络结构。通过将光流限制为一系列相同更新步骤的产物，我们让网络学习一个更新算子，该算子模仿一阶下降算法的更新。这限制了搜索空间，降低了过拟合的风险，并实现了更快的训练和更好的泛化性。

在Sintel(test)集上进行评估时，我们在结合了KITTI、HD1K数据和训练集中clean pass和final pass组合上进行微调。我们的方法在Sintel clean和final pass中均排名第一，并且在clean pass上比之前的所有工作高0.9像素(36%)，在最终pass中高出1.2像素(30%)。我们评估我们模型的两个版本，Ours（two-frame）使用零初始化，而Ours（warp-start）通过向前投影的前一帧光流估计来初始化光流。由于我们的方法以单一分辨率运行，我们可以初始化光流估计以利用过去帧的运动平滑，这是使用粗糙到精细的模型不容易做到的。

4.2. KITTI

我们还在KITTI上评估了RAFT，并在表1和图4中提供了结果。我们首先通过在Chairs(C)和FlyingThings(T) 训练后的模型在KITTI-15（train）划分集上验证来评估跨数据集的泛化性。我们的方法明显优于先前的工作，将EPE（端点误差）从8.36降低到5.04，这表明我们网络的基底结构有助于泛化性。在KITTI排行榜上在所有光流方法中我们

的方法排名第一。

图4：在KITTI测试集上的光流预测。

4.3. 消融

我们进行了一组消融实验来说明每个组成部分的相对重要性。所有消融版本都在FlyingChairs(C) + FlyingThings(T)上进行训练。消融的结果在表2中显示。在表的每个部分中，我们分别测试了方法的特定组成部分，在我们的最终模型中使用的设置带有下划线。下面我们更详细地描述每个实验。

更新算子的结构：我们使用基于GRU单元的门控激活单元。我们使用一组具有ReLU激活的3个卷积层替换卷积GRU。通过使用GRU块我们获得了更好的性能，这可能是因为门控激活使一系列的光流估计更容易收敛。

权重绑定：默认情况下，我们在更新算子的所有实例中绑定权重。在这里，我们测试了另一个版本，其中每个更新算子分别学习权重。当权重绑定时精度会更好，并且参数总数明显降低。

上下文：我们通过训练没有上下文网络的模型来测试上下文的重要性。在没有上下文的情况下，我们仍然取得了不错的结果，优于在Sintel和KITTI上现有的所有工作。但是上下文很有帮助。将图像特征直接添加进更新算子可能在运动边界内能更好地聚合空间信息。

特征尺度：默认情况下，我们以单一分辨率提取特征。我们还尝试通过在每个尺度上分别构建相关体来提取多个分辨率的特征。单分辨率特征简化了网络结构，即使在大位移下也可以进行细粒度的匹配。

查找半径：在查找操作中查找半径指定使用的网格的尺寸。当使用半径为0时，在单个点检索相关体。令人惊讶的是，当半径为0时，我们仍然可以粗略估计光流，这意味着网络正在学习使用0阶信息。然而，随着半径的增加，我们看到了更好的结果。

相关池化：我们以单一分辨率输出特征，然后执行池化以生成多尺度体。在此，我们测试了移除池化时的影响。有池化的结果会更好，因为同时获得了大位移和小位移。

相关范围：我们还尝试仅为每个像素周围的局部邻域构建相关体，而不是所有对相关。我们尝试了32像素、64像素和128像素的范围。总体而言，当使用所有对时，我们得到了最好的结果，尽管128px的范围足以在Sintel上表现很好，因为大多数位移都在这个范围内。也就是说，all-pairs仍然是更合适的，因为它排除了指定范围的需要。实现起来也更方便：它可以使用矩阵乘法计算，从而让我们的方法在PyTorch中可以完全实现。

改进特征：我们通过在所有像素对之间构建相关体来计算视觉相似性。在这个实验中，我们尝试用形变层替换相关体，形变层使用当前的光流估计将特征从I2形变到I1，然后估计残差位移。与之前在Sintel上的工作相比虽然形变仍然具有竞争力，但相关的表现要好得多，尤其是在KITTI上。

上采样：RAFT以1/8分辨率输出光流场。我们比较了双线性上采样与我们学习的上采样模块。上采样模块产生更好的结果，尤其是在运动边界附近。

推断更新：虽然我们在训练期间进行了12次更新，但我们可以在推断期间应用任意数量的更新。在表2中，我们提供了选定更新次数的数值结果，并测试了200次的极端情况，以表明我们的方法没有发散。我们的方法快速收敛，在3次更新后超过了PWC-Net，在6次更新后超过了FlowNet2，随着更多的更新效果继续提高。

Experiment	Method	Sintel (train) Clean	Sintel (train) Final	KITTI-15 (train) F1-epe	KITTI-15 (train) F1-all	Parameters
Reference Model (bilinear upsampling), Training: 100k(C) → 60k(T)						
Update Op.	ConvGRU	1.63	2.83	5.54	19.8	4.8M
	Conv	2.04	3.21	7.66	26.1	4.1M
Tying	Tied Weights	1.63	2.83	5.54	19.8	4.8M
	Untied Weights	1.96	3.20	7.64	24.1	32.5M
Context	Context	1.63	2.83	5.54	19.8	4.8M
	No Context	1.93	3.06	6.25	23.1	3.3M
Feature Scale	Single-Scale	1.63	2.83	5.54	19.8	4.8M
	Multi-Scale	2.08	3.12	6.91	23.2	6.6M
Lookup Radius	0	3.41	4.53	23.6	44.8	4.7M
	1	1.80	2.99	6.27	21.5	4.7M
	2	1.78	2.82	5.84	21.1	4.8M
	4	1.63	2.83	5.54	19.8	4.8M
Correlation Pooling	No	1.95	3.02	6.07	23.2	4.7M
	Yes	1.63	2.83	5.54	19.8	4.8M
Correlation Range	32px	2.91	4.48	10.4	28.8	4.8M
	64px	2.06	3.16	6.24	20.9	4.8M
	128px	1.64	2.81	6.00	19.9	4.8M
	All-Pairs	1.63	2.83	5.54	19.8	4.8M
Features for Refinement	Correlation	1.63	2.83	5.54	19.8	4.8M
	Warping	2.27	3.73	11.83	32.1	2.8M
Reference Model (convex upsampling), Training: 100k(C) → 100k(T)						
Upsampling	Convex	1.43	2.71	5.04	17.4	5.3M
	Bilinear	1.60	2.79	5.17	19.2	4.8M
Inference Updates	1	4.04	5.45	15.30	44.5	5.3M
	3	2.14	3.52	8.98	29.9	5.3M
	8	1.61	2.88	5.99	19.6	5.3M
	32	1.43	2.71	5.00	17.4	5.3M
	100	1.41	2.72	4.95	17.4	5.3M
	200	1.40	2.73	4.94	17.4	5.3M

表2：消融实验。最终模型中使用的设置带有下划线。细节见4.3节。

4.4. 时间和参数总数

推断时间和参数总数如图5所示。精度取决于在FlyingChairs和FlyingThings (C+T)训练后的Sintel(train) final pass的表现。在这些图中，我们公布了10次迭代后的精度和时间，我们在GTX 1080Ti GPU上我们的方法进行计时。我们使用其他方法在他们的论文中公布的参数总数，并公布在我们的硬件上运行的时间。RAFT在参数总数、推断时间和训练迭代方面更有效。Ours-S仅使用1M参数，但优于PWC-Net和VCN，后者大6倍以上。我们在补充材料中提供了一个附加表格，其中包含参数、时间和训练迭代的数值。

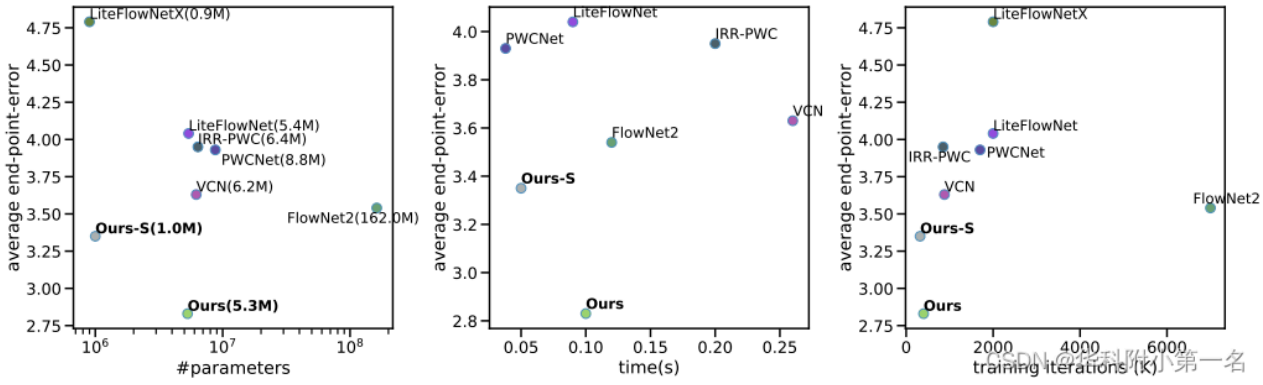


图 5：比较参数总数、推断时间和训练迭代与精度的图表。精度是在C+T训练后的Sintel(train) final pass上测量EPE得到的。左：与其他方法相比，参数总数与准确性。RAFT在取得更低EPE的同时参数效率更高。中间：使用我们的硬件计时下推断时间与精度。右：训练迭代与精度（作为迭代和使用的GPU的产物）。

4.5. 非常高分辨率的视频

为了证明我们的方法可以很好地扩展到非常高分辨率的视频，我们将网络应用于来自DAVIS数据集的高清视频。我们使用1080p (1088x1920) 分辨率的视频并将我们的方法迭代12次。在1080p视频上进行12次迭代的推断需要550毫秒，其中所有对相关需要95毫秒。图6可视化了在DAVIS上的例子结果。

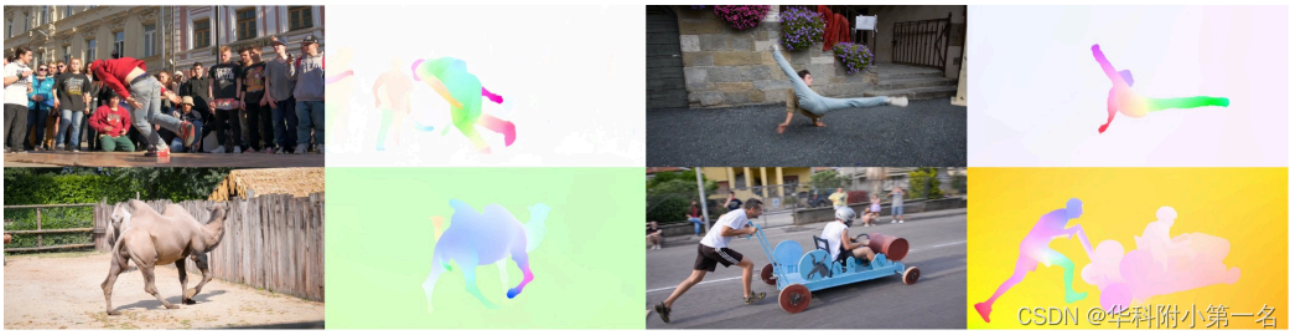


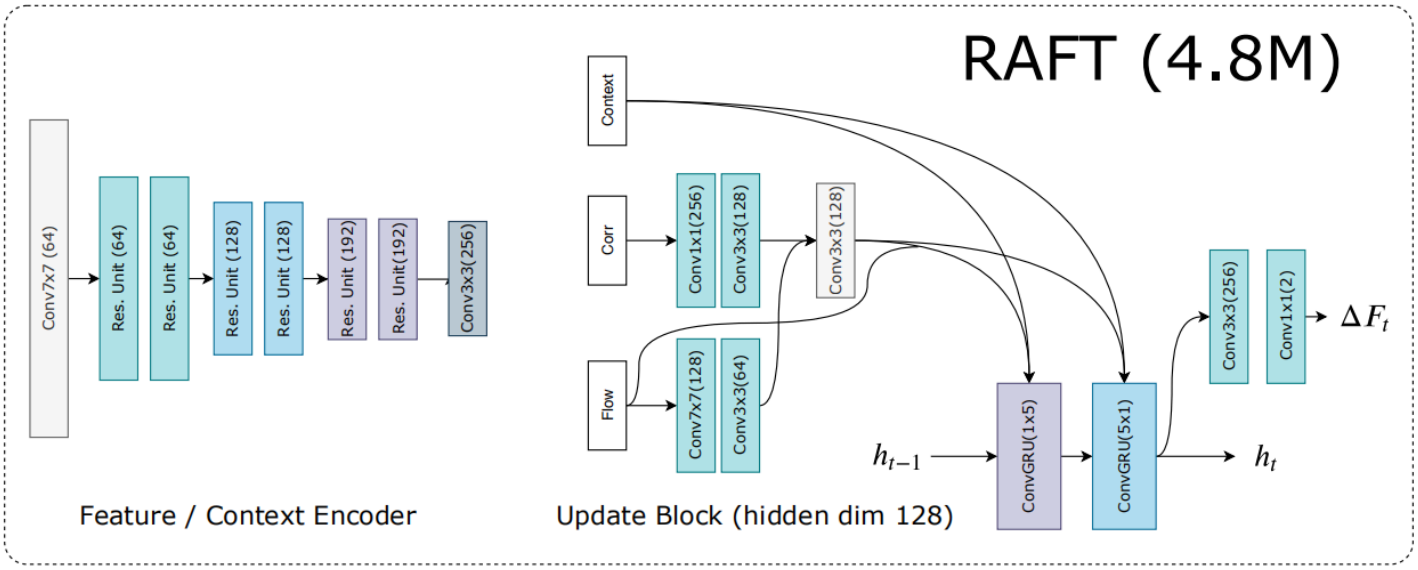
图6：来自DAVIS的1080p (1088x1920)视频的结果（每帧550毫秒）。

5. 结论

我们提出了RAFT—一种新的适用于光流的端到端可训练的模型。RAFT的独特之处在于它使用大量轻量级、循环更新算子以单一分辨率运行。我们的方法在各种数据集上实现了最好的精度，强大的跨数据集泛化能力，并且在推断时间、参数总数和训练迭代方面非常有效。

补充材料：

A 网络结构



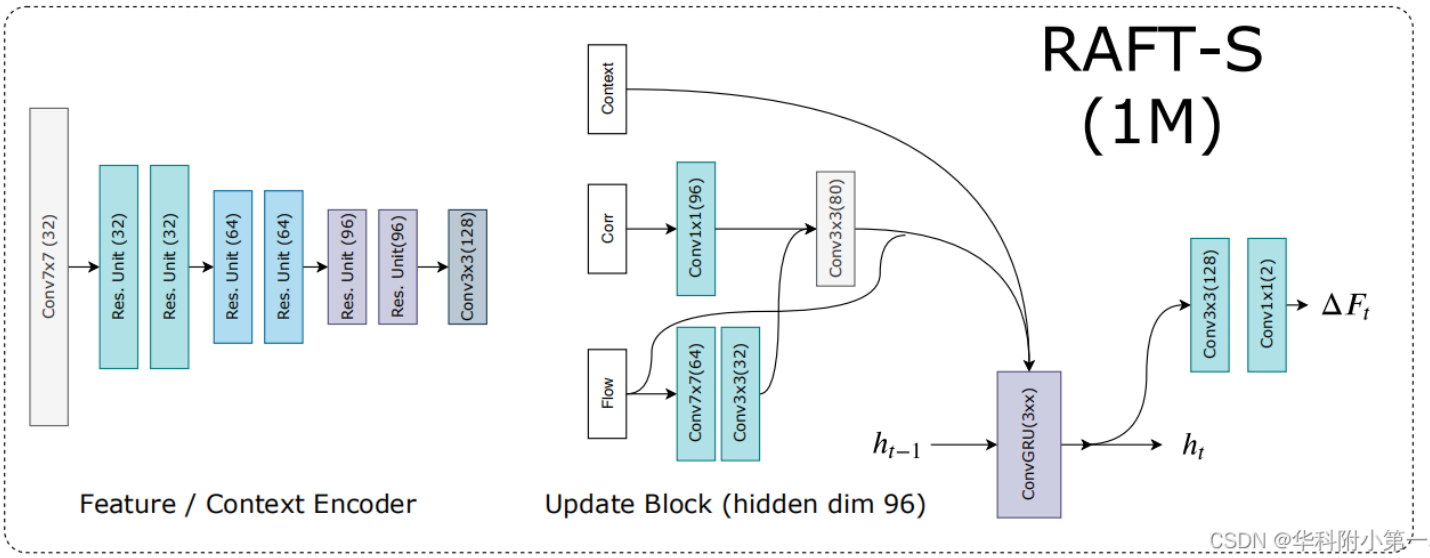


图7：完整4.8M参数模型（5.3M带上采样模块）和1.0M参数的小模型的网络结构细节。上下文和特征编码器具有相同的结构，唯一的区别是特征编码器使用实例归一化，而上下文编码器使用批归一化。在RAFT-S中，我们用瓶颈残差单元替换残差单元。更新块使用上下文特征、相关特征和光流特征来更新潜在隐藏状态。更新后的隐藏状态用于预测光流更新。完整模型使用两个卷积GRU更新块，分别带有1x5过滤器和5x1过滤器，而小模型使用带有3x3单个过滤器的GRU。

B 上采样模块

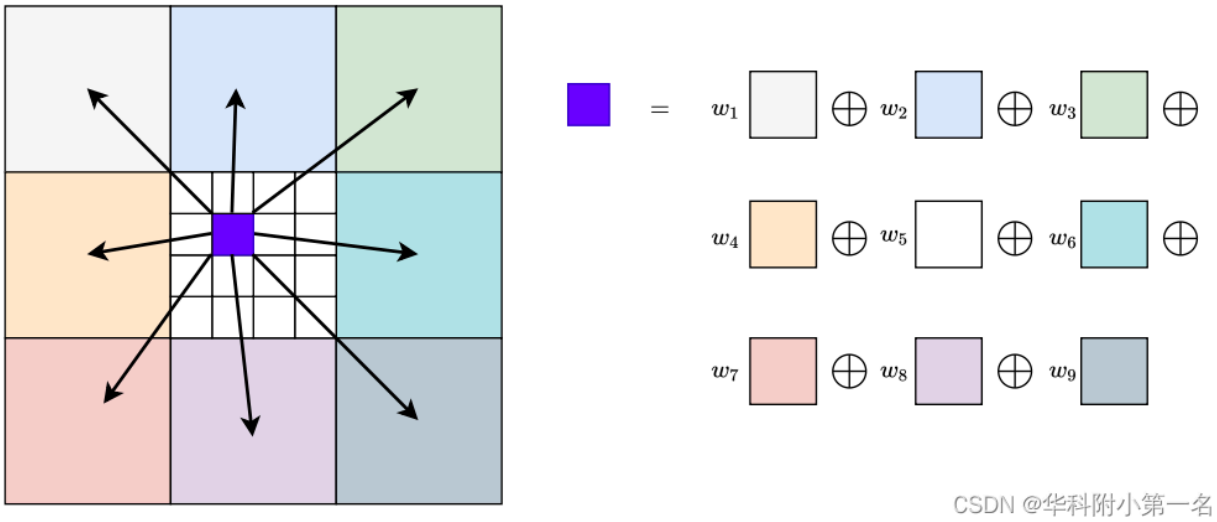


图8：上采样模块的图示。通过使用网络预测的权重，高分辨率光流场（小框）的每个像素都被视为其9个粗分辨率邻居的凸组合。

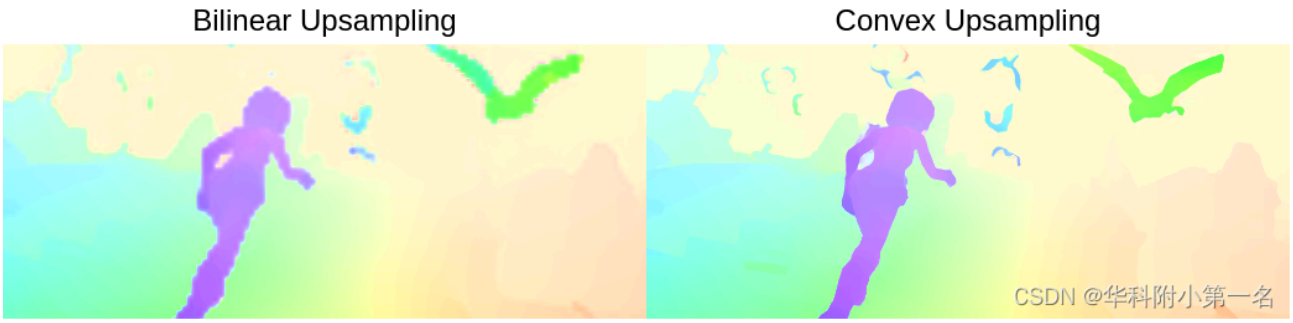


图9：我们的上采样模块提高了运动边界附近的精度，并且还让RAFT恢复小型快速移动物体的光流，例如图中所示的鸟类。

C 训练细节

Stage	Weights	Training Data	Learning Rate	Batch Size (per GPU)	Weight Decay	Crop Size
Chairs	-	C	4e-4	6	1e-4	[368, 496]
Things	Chairs	T	1.2e-4	3	1e-4	[400, 720]
Sintel	Things	S+T+K+H	1.2e-4	3	1e-5	[368, 768]
KITTI	Sintel	K	1e-4	3	1e-5	[288, 960]

表3：训练计划的详细信息。数据集缩写：C：FlyingChairs，T：FlyingThings，S：Sintel，K：KITTI-2015，H：HD1K。在Sintel微调阶段，数据集分布为 S(.71)、T(.135)、K(.135)、H(.02)。

光度增强：我们通过随机扰动亮度、对比度、饱和度和色调来执行光度增强。我们使用 Torchvision ColorJitter，亮度为0.4，对比度为 0.4，饱和度为0.4，色调为0.5/π。在KITTI上，我们将增强程度降低到亮度0.3、对比度0.3、饱和度0.3和色调0.3/π。概率为0.2时，对每个图像独立执行颜色增强。

空间增强：我们通过随机缩放和拉伸图像来执行空间增强。随机缩放的程度取决于数据集。对于FlyingChairs，我们在2[-0.2,1.0]范围内进行空间增强 、其他依次为FlyingThings2[-0.4,0.8]、Sintel 2[-0.2,0.6]和KITTI 2[-0.2,0.4]，空间增强的执行概率为0.8。

遮挡增强：延续HSM-Net，我们还以0.5的概率随机擦除I2中的矩形区域以模拟遮挡。

D 时间、参数和训练迭代次数

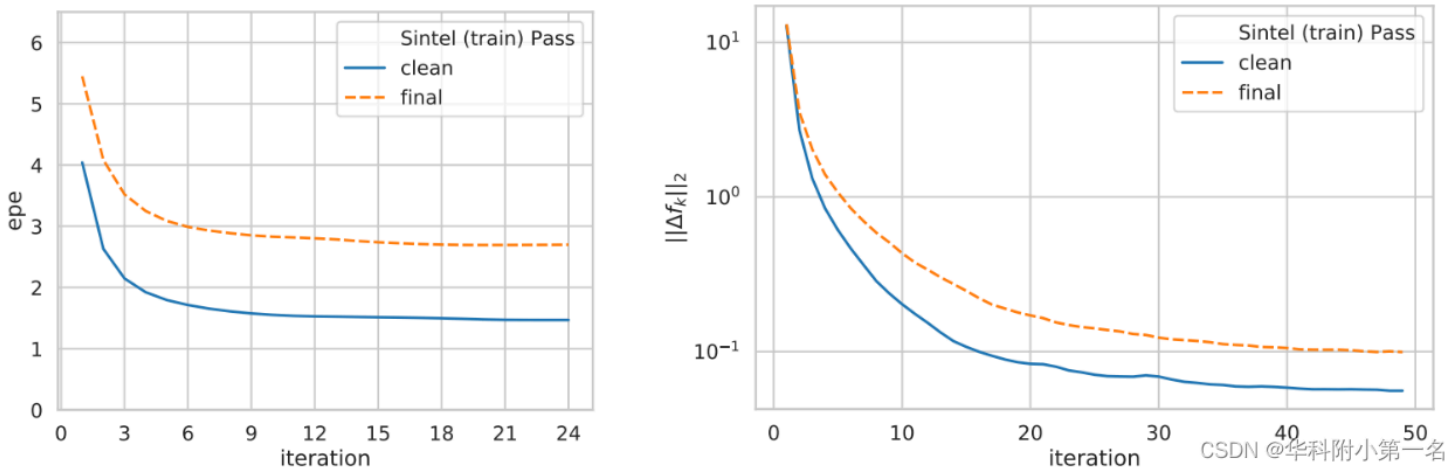


图10：（左）Sintel 集上的EPE作为推断时迭代次数的函数。（右）每次更新的幅度||Δfk||2在所有像素上取平均值，表明收敛到一个固定点fk→f *。

Method	Parameters (M)	Time (Reported)	Time (1080Ti)	Training Iter. (#GPUs)	Accuracy
LiteFlowNetX [22]	0.9M	0.03s	-	2000k	4.79
LiteFlowNet [22]	5.4M	0.09s	0.09s	2000k	4.04
IRR-PWC [24]	6.4M	-	0.20s	850k	3.95
PWCNet+ [41]	9.4M	0.03s	0.04s	1700k	3.93
VCN [49]	6.2M	0.18s	0.26s	220k(4)	3.63
FlowNet2 [25]	162M	0.12s	0.11s	7000k	3.54
Ours (small)	1.0M	-	0.05s	160k(2)	3.37
Ours (mixed)	5.3M	-	0.10s	240k(1)	2.85
Ours	5.3M	-	0.10s	200k(2)	2.83

表4：在Sintel（train）final pass上的参数数量、推断时间、训练迭代和精度。我们使用GTX 1080Ti GPU 在10次更新后公布了我们方法的计时和精度。如果可能，我们下载其他方法的代码并使用我们的机器重新计时。如果模型使用多个GPU进行训练，我们会在括号中公布用于训练的GPU数量。我们还可以使用混合精度训练 Ours(mixed)训练RAFT，并在仅有单个GPU上训练时获得类似的结果。总体而言，与之前的工作相比，RAFT需要更少的训练迭代次数和参数。

三、参考内容

（论文解读）RAFT: Recurrent All-Pairs Field Transforms for Optical Flow
https://blog.csdn.net/qq_43414059/article/details/108842025

ECCV 2020最佳论文讲了啥？作者为ImageNet一作、李飞飞高徒邓嘉
https://zhuanlan.zhihu.com/p/205020999