NeRF-SLAM:Real-Time Dense Monocular SLAM with Neural Radiance Fields

Abstract：

稠密单目SLAM可以提供精确的**位姿估计**和**带有相关不确定性的深度地图，**这些正确的信息可以用来**实时拟合**场景的**神经辐射场**

利用我们提出的**基于不确定性**的**深度损失**，我们不仅实现了良好的**光度精度**，而且实现了良好的**几何精度**

Introduction：

单目相机相较于其他传感器有天然的优势：**低成本、轻量、标定简单**

但是由于缺少**对场景深度的显示测量**，单目的3D重建仍然是一个具有挑战性的问题

近些年，深度学习方法在**光流估计**、**深度估计**上均取得了最佳的表现，也有很多工作在SLAM中使用深度学习模块

但是从一组随意拍摄的的单目视频中实时建立一个几何和光度准确的3D地图在目前仍然无法完成

最近NeRFs实现了具有光度准确的3D地图建立

但是，构建NeRFs所需要的**体积渲染代价高昂**，所以其很难被推断，导致了重建速度很慢；同时NeRFs需要真值位姿估计用于最初的收敛

但是最近的研究表明，通过给定已知位姿的图像是可以实时的拟合辐射场，还有一些研究表明如果给定一个足够好的初始位姿，真值位姿并不是严格需要的 -> 实时无姿态的NeRFs重建可以得到精确的地图

尽管如此，由于没有深度监管，NeRF表示面临一个基本问题：通过密度对曲面进行参数化容易出现“漂浮物”、鬼几何（这是因为初始化不好或者收敛到不好的局部最小值）

研究表明，加入深度监督显著的改善了对于鬼影几何的去除，并且深度信号可以让辐射场更快速的收敛

我们的见解是：构建一个单目稠密SLAM，输出近乎完美的位姿估计，结合稠密深度地图和不确定性估计，为建立场景的神经辐射场提供了正确的信息

主要贡献：

我们提出了首个结合**稠密单目SLAM**和**分层体积的神经辐射场**优点的场景重建框架

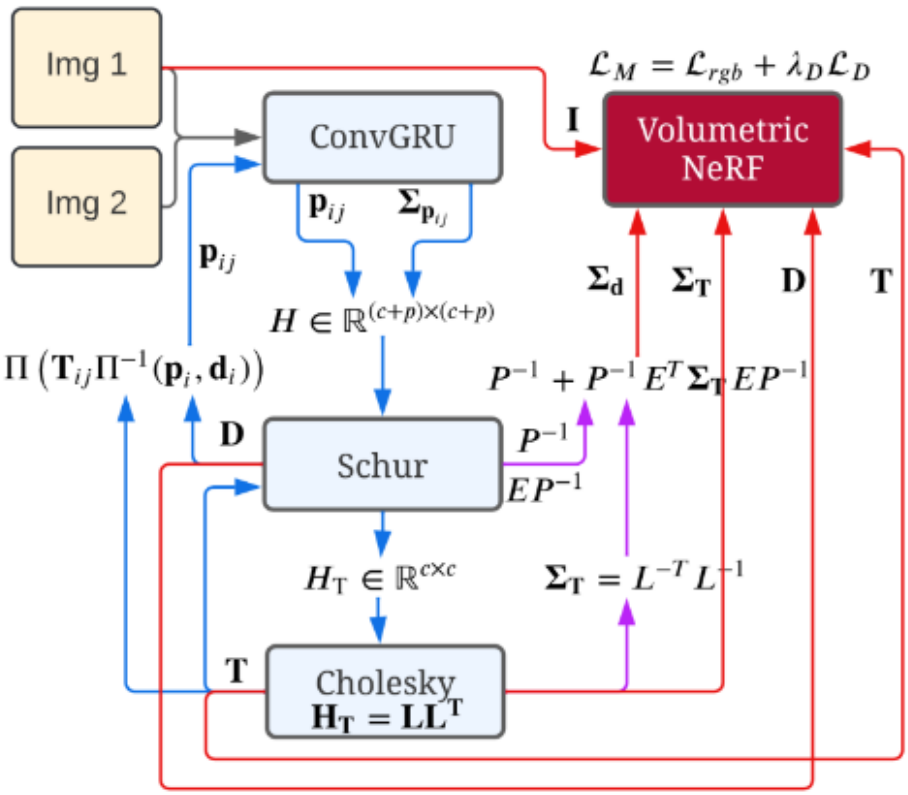
我们的方法从图像流中构建准确的辐射场，不需要姿态或者深度作为输入，并且可以实时运行

我们在Replica数据集上实现了单目方法的SOTA性能

Related Work：

Methodology：

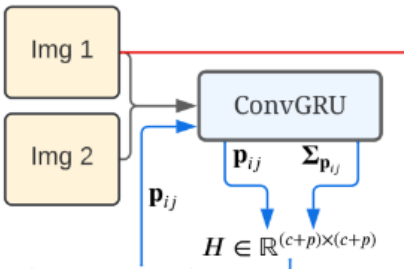
核心思路是利用稠密单目SLAM的输出监管一个NeRF，稠密单目SLAM的作用是提供**稠密深度地图**的估计、**相机位姿**的估计、以及他们的**不确定性估计**。有了这些信息我们可以通过**深度边缘协方差加权**的**稠密深度损失**训练NeRF。



1. Tracking: Dense SLAM with Covariances

我们沿用了Droid-SLAM作为跟踪模块、其可以提供每个关键帧的稠密深度图和位姿。

Droid-SLAM的输入是一个图像序列 $img1, img2...$ ，其核心模块是一个ConvolutionalGRU（**ConvGRU**），这个模块的作用是图像帧 i 和图像帧 j 之间的稠密光流 p_{ij}

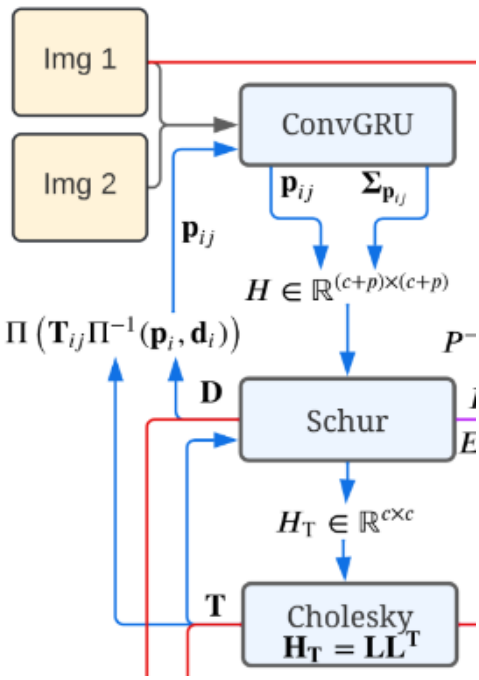


对于**ConvGRU**，输入**两个图像帧之间的相关性**和一个**当前光流的预测** p_{ij} ，输出一个**新的光流** p_{ij} 以及**每个光流测量的权重** $\Sigma_{p_{ij}}$

有了 p_{ij} 和 $\Sigma_{p_{ij}}$ 作为测量值，Droid-SLAM解决了稠密BA问题，其中**三维几何**被参数化为**每个关键帧的一组逆深度图**。这种参数化的结构可以通过构建**线性最小二乘**让求解稠密BA问题变得高效，**稠密BA问题**通过将**系统方程线性化**为**近似的**相机/深度**箭头块状稀疏Hseeian矩阵** $H \in \mathbb{R}^{(c+p)\times(c+p)}$,其中 c 和 p 是相机和点的维度数。

$$\begin{bmatrix} C & E \\ E^T & P \end{bmatrix} \begin{bmatrix} \Delta\boldsymbol{\xi} \\ \Delta\mathbf{d} \end{bmatrix} = \begin{bmatrix} \mathbf{v} \\ \mathbf{w} \end{bmatrix}$$

b 是残差， C 是分块相机矩阵， P 是每个关键帧每个像素的逆深度对应的对角矩阵。 $\Delta\xi$ 表示在 $SE(3)$ 中相机位姿的李代数上的更新， Δd 表示每个像素逆深度的更新， E 是相机/深度非对角Hessian分块矩阵， v 和 w 分别对应姿态残差和深度残差



为了解决线性最小二乘问题，我们利用**Hessian矩阵的Schur补**来计算**缩减的相机矩阵** H_T ，其不依赖于深度信息并且具有更小的维度 $\mathbb{R}^{c\times c}$

$$\begin{bmatrix} I & -EP^{-1} \\ 0 & I \end{bmatrix} \begin{bmatrix} C & E \\ E^T & P \end{bmatrix} \begin{bmatrix} \Delta\boldsymbol{\xi} \\ \Delta\mathbf{d} \end{bmatrix} = \begin{bmatrix} I & -EP^{-1} \\ 0 & I \end{bmatrix} \begin{bmatrix} \mathbf{v} \\ \mathbf{w} \end{bmatrix}$$

$$\begin{bmatrix} C - EP^{-1}E^T & 0 \\ E^T & P \end{bmatrix} \begin{bmatrix} \Delta\boldsymbol{\xi} \\ \Delta\mathbf{d} \end{bmatrix} = \begin{bmatrix} \mathbf{v} - EP^{-1}\mathbf{w} \\ \mathbf{w} \end{bmatrix}$$

$$H_T = C - EP^{-1}E^T$$

由此产生的较小相机位姿问题，可以通过Cholesky分解 $H_T = LL^T$ 求解，其中 L 是下三角Cholesky因子，然后通过向前和向后迭代求解位姿 T

Cholesky分解求解线性方程组$AX = b$

1. 对A作cholesky分解（A需要时对称正定矩阵）

$$A = LL^T$$

2. 原线性方程转换为$LL^TX = b$，令$LY = b$，求Y

$$Y = L^{-1}b$$

3. $LL^TX = LY \longrightarrow L^TX = Y$，求$X$

$$X = L^{-T}Y$$

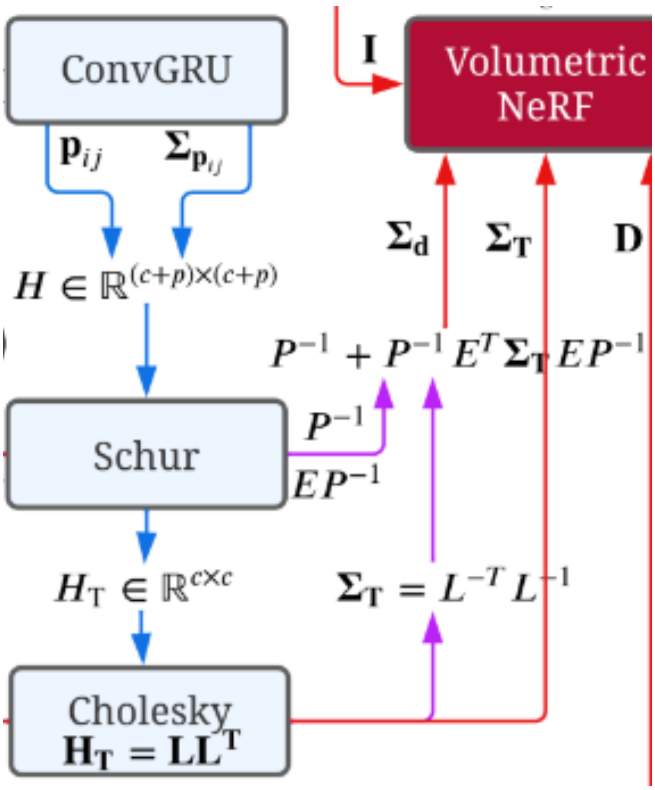
有了相机位姿T，就可以求出深度 D

有了相机位姿T和深度 D ，Dorid-SLAM就可以计算诱导光流，并反馈给ConvGRU网络，其中 Π 和 Π^{-1} 分别是相机投影模型和反向投影模型

我们继续计算Dorid-SLAM生成的稠密深度地图和位姿的边缘协方差，为此我们要利用Hessian矩阵的结构，我们的分块划分如下所示：

$$Hx = b, \ \ i.e. \ \ \begin{bmatrix} C & E \\ E^T & P \end{bmatrix} \begin{bmatrix} \Delta\xi \\ \Delta d \end{bmatrix} = \begin{bmatrix} v \\ w \end{bmatrix}$$

其中 H 是Hessian矩阵， b 是残差， C 是分块相机矩阵， P 是每个关键帧每个像素的逆深度对应的对角矩阵。 $\Delta\xi$ 表示在 $SE(3)$ 中相机位姿的李代数上的更新， Δd 表示每个像素逆深度的更新， E 是相机/深度非对角Hessian分块矩阵， v 和 w 分别对应姿态残差和深度残差



通过这样划分Hessian矩阵，我们可以高效的计算稠密深度和位姿的边缘协方差矩阵

- **Hessian矩阵在最大似然（MLE）问题中被认为约等于信息矩阵**
- **协方差的逆=信息矩阵**

T 和 D 的边际概率（边缘协方差矩阵）分别为协方差矩阵的左上元素和右下元素（对应C和P的位置）

$$\Sigma_d = P^{-1} + P^{-T}E^T\Sigma_T EP^{-1}$$
$$\Sigma_T = (LL^T)^{-1}$$

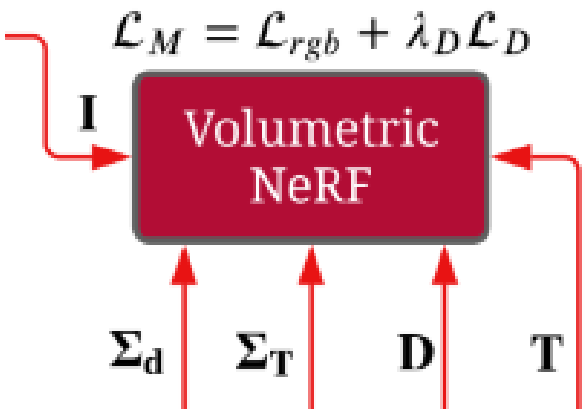
$$H = \begin{bmatrix} I & EP^{-1} \\ 0 & I \end{bmatrix} \begin{bmatrix} C - EP^{-1}E^T & 0 \\ 0 & P \end{bmatrix} \begin{bmatrix} I & 0 \\ P^{-1}E^T & I \end{bmatrix}$$

$$\Sigma = H^{-1} = \begin{bmatrix} I & 0 \\ -P^{-1}E^T & I \end{bmatrix} \begin{bmatrix} (C - EP^{-1}E^T)^{-1} & 0 \\ 0 & P^{-1} \end{bmatrix} \begin{bmatrix} I & -EP^{-1} \\ 0 & I \end{bmatrix}$$

至此，给定Tracking模块计算出的所有信息：**位姿、深度、各自的边缘协方差矩阵**，以及**输入的RGB图像**，我们可以同时优化NeRF参数和相机位姿

---

2. Mapping: Probabilistic Volumetric NeRF 概率体积NeRF



因为已经给定了每个关键帧的稠密深度图，我们可以对神经体积进行深度监管。但是深度图由于其密度而极为嘈杂，因为即使是缺少纹理的区域也被赋予了深度

从上面的图可以看出，稠密单目SLAM得到的点云充满噪声，而且包含较大的离群点。使用这些深度图监管NeRF会导致有偏差的重建

Rosinol等人的工作表明，对于经典的TSDF体积融合，深度估计的不确定性是一个很好的信号来加权深度值。

受到这个工作的启发，我们我们使用深度不确定性估计来加权深度损失，并用于监管神经体积。

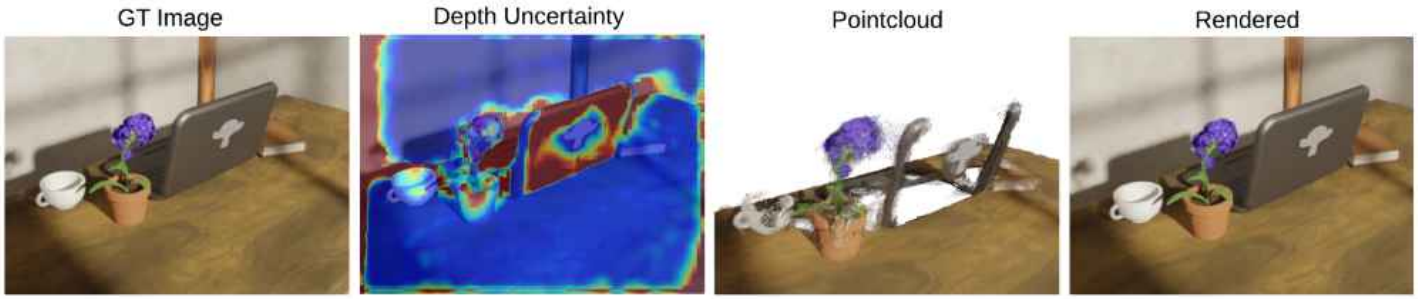


Figure 1. From left to right, input RGB image, estimated depth uncertainty, back-projected depth-maps into a pointcloud, after thresholding the depth by its uncertainty ($\sigma_d \leq 1.0$) for visualization, and our resulting neural radiance field, rendered from the same viewpoint as the input image. Our pipeline is capable of reconstructing neural radiance fields in real-time given only a stream of RGB images.

上图为：输入RGB图像、他的不确定性深度图、得到的点云（按照不确定度 $\sigma_d \leq 1.0$ 作为阈值后进行可视化）、使用不确定性加权的深度损失得到结果

给定不确定性损失，我们把建图损失表示为：

$$\mathcal{L}_M(T,\Theta) = \mathcal{L}_{rgb}(T,\Theta) + \lambda_D \mathcal{L}_D(T,\Theta)$$

给定超参数 λ_D 平衡深度和颜色监管（我们将 λ_D 设置为1.0）的情况下，我们最小化关于姿态 T 和神经参数 Θ 。特别的，我们的深度损失定义为如下：

$$\mathcal{L}_D(T,\Theta) = \|D - D^*(T,\Theta)\|^2_{\Sigma_D}$$

其中， D^* 是渲染深度， D, Σ_D 是由Tracking模块估计的稠密深度和不确定性。我们将深度 D^* 渲染为预测的射线终止距离。每个像素的深度是通过沿着像素光线的3D位置采样。评估样本 i 处的密度 σ_i ，并对得到的密码进行alpha复合计算，于标准的体积渲染类似：

$$d^* = \sum_i \mathcal{T}_i (1 - exp(-\sigma_i \delta_i)) d_i$$

其中， d_i 为沿着光线的采样点 i 的深度， $\delta_i = d_{i+1} - d_i$ 为相邻采样点之间的距离差。 σ_i 是采样点 i 在三维世界坐标系下通过计算MLP生成的体积密度。 $\mathcal{T}_i$ 为沿着光线方向到达采样点的累积投射率：

$$\mathcal{T}_i = exp(-\sum_{j<i} \sigma_i \delta_i)$$

我们的RGB损失与原生NeRF的定义相同：

$$\mathcal{L}_{rgb}(T,\Theta) = \|I - I^*(T,\Theta)\|^2$$

其中， I^* 是通过体积渲染处的彩色图像，与深度图像的渲染类似。每个像素的颜色也是通过沿着像素光线方向进行采样，并将得到的密度和颜色进行alpha复合计算的： $\sum_i \mathcal{T}_i (1 - exp(-\sigma_i \delta_i)) c_i$ ，其中 $\mathcal{T}_i$ 的计算方式和上面一样， c_i 是由MLP得到的颜色。对于给定的样本 i ,密度 σ_i 和颜色 c_i 同时被估计

Mapping进程持续的最小化mapping损失函数 $\mathcal{L}_M(T,\Theta)$

3. Architecture

我们的系统由一个跟踪线程和一个建图线程组成，两者实时并行运行。跟踪线程不断最小化关键帧活动窗口的BA重投影误差。建图线程优化从跟踪线程接收到的所有关键帧，不使用滑动窗口

两个线程之间的唯一通信发生在跟踪线程生成新的关键帧时。在每一个新的关键帧中，跟踪线程将当前关键帧的姿态及其各自的图像和估计深度图以及深度的边缘协方差发送给建图线程。

只有在当前滑窗优化中可用的信息被发送给建图线程。跟踪线程的主动滑动窗口最多由8个关键帧组成。当前一关键帧与当前帧之间的平均光流大于阈值(2.5像素)时，跟踪线程生成新的关键帧。

最后，建图线程还负责渲染重建的交互式可视化。