LightGlue 特征匹配算法研究报告

摘要

本文研究了 LightGlue 算法在特征匹配中的应用及其优势，特别是在即时定位与地图构建（SLAM）系统中的重要性。LightGlue 算法采用深度网络框架，结合自注意力机制和交叉注意力机制，旨在提高特征匹配的准确性、效率和易训练性。通过与 SuperGlue 算法对比，LightGlue 在准确度、效率及训练便捷性方面均表现优越。实验结果表明，LightGlue 在使用不同特征点提取算法时，在 DISK 算法提取特征点的基础上能够提供更高的匹配精度。另外，LightGlue 在处理大量特征点时，运行时间明显低于 SuperGlue，尤其在大规模数据集上的效率优势尤为突出。LightGlue 算法的自适应深度和宽度机制能够根据输入图像的复杂性动态调整网络结构，从而在保持高精度的同时，提升了处理速度。总的来说，LightGlue 在 SLAM 系统和视觉定位等 3D 计算机视觉任务中具有广泛的应用前景，并为机器人、自主驾驶等领域提供了有力的技术支持。

关键词： LightGlue；SLAM；特征匹配；特征提取；

1 引言

即时定位与地图构建（Simultaneous localization and mapping，SLAM）主要用于机器人和无人驾驶车辆等技术，是它们能够在未知的环境中（如 GPS 定位不精准的情况下）构建地图并且能够确定自己的位置。

图像间的特征匹配一直以来是 SLAM 任务的重要一环。特征匹配帮助 SLAM 系统在连续图像帧之间找到对应的特征点对，通过这些匹配对计算相机或机器人的相对运动。准确的特征匹配能够确保运动估计的精度，从而提高定位的准确性。

大多数匹配方法由三个步骤组成：特征点检测、特征描述和特征匹配。给定两幅图像，在检测阶段，首先从两幅图像中检测出角点等显著点作为特征点，然后根据特征点周围的信息提取局部描述符。根据特征点检测和特征描述阶段产生的两组向量进行特征匹配，通过最邻近搜索或更复杂的匹配算法找到。这些特征点使用编码其局部视觉外观的高维表示进行匹配。在表现出对称性、弱纹理或由于视点和光照变化而导致的外观变化的情况下，可靠地描述每个点是具有挑战性的。为了剔除由遮挡和缺失点引起的离群点，这种表示也应该具有判别性。这就产生了两个难以满足的相互冲突的目标，鲁棒性和唯一性。

在《LightGlue: Local Feature Matching at Light Speed》中，作者设计了一个更准确、更高效、更容易训练的深度网络 LightGlue [1] 的特征匹配算法。作者重新审视了以往的设计决策，并结合了许多简单而有效的架构修改。在有限的资源下训练高性能的深度匹配器，在短短几天的 GPU 时间内达到最先进的精度。LightGlue 能更快、更好地匹配稀疏特征。最终优

化的模型在典型的室外条件下，以 8 倍更高的速度提供了更接近密集匹配器 LoFTR [2] 的精度。

2 相关工作

2.1 特征提取算法

在特征提取算法中，许多传统主流算法是基于检测器的方法，其中 SIFT [3] 和 ORB [4] 被认为是最成功的手工设计的局部特征，并被广泛应用于许多三维计算机视觉任务中。ORB 算法分为关键点和描述子两部分，ORB 将 FAST 特征点检测方法与 BRIEF 特征描述子结合起来。FAST 特征点检测方法是一种角点检测算法，主要检测局部像素灰度变化明显的地方，以速度快著称，算法思想是如果一个像素与它邻域的像素差别较大（过亮或过暗），那么它更可能是角点。BRIEF 特征描述子是对前一步提取出特征点的周围图像区域进行描述，BRIEF 是一种二进制描述子，描述向量由许多个 0 和 1 组成，这里的 0 和 1 编码了关键点附近两个像素（比如说 p 和 q）的大小关系：如果 p 比 q 大，则取 1，反之就取 0。

ORB 算法在计算时对图片进行两次操作，先提取特征点再进行描述子的生成，这无疑增大了算法的复杂度。其实在特征点提取的过程中能够同时把描述子提取出来，之后提出了基于深度学习的 SuperPoint [5] 方法，SuperPoint 是全卷积神经网络架构，它在全尺寸图像上运行，并在单次前向传递中产生带有固定长度描述符的特征点检测，能够同时提取特征点的位置以及描述子。

2.2 基于深度学习的局部特征匹配方法

SuperGlue [6] 是一种基于图卷积神经网络的特征匹配算法，SuperGlue 的核心思想是 Attentional Graph Neural Network（注意力图神经网络）中的注意力机制。它的输入是两张图像中特征点以及描述子，输出是图像特征之间的匹配关系。对输入的描述子和坐标通过一个 keypoint encoder 进行聚合，然后送到注意力机制。输出的是 matching-descriptor，这一步也可以看成是描述子增强，接着把匹配问题建模成运输问题进行求解匹配。

SuperGlue 将 Transformers [7] 的表达方式与最优运输相结合来解决部分指派问题。它学习关于场景几何和相机运动的强大先验知识，因此对极端变化具有鲁棒性，并且在数据域中具有良好的泛化性。SuperGlue 继承了早期 Transformer 的局限性，训练难度大，复杂度随关键点数量呈二次方增长，所以 SuperGlue 在计算上是昂贵的，而图像匹配的效率对于需要低延迟 (如跟踪) 或高处理量 (如大规模映射) 的任务是至关重要的。LightGlue 为典型的操作条件带来了巨大的改进，重新审视了 SuperGlue 的多项设计决策，推导出了简单有效的改进，在内存和计算方面更高效、更准去，并且更容易训练，在图像对更容易的情况下，推导更快。

3 本文方法

3.1 Light 框架

Lightglue 架构如图 1所示，给定一对输入局部特征 (d , p)，每个局部特征由一个二维点位置 p=(x , y) 和视觉描述子 d 组成。每一层基于位置编码的自注意力单元和交叉注意力单

元对视觉描述符进行上下文增强。一个置信度分类器 c 帮助决定是否停止推断。如果有少数点是有信心的，则推论继续到下一层，同时将有信心无法匹配的点进行剪枝。一旦达到置信状态，LightGlue 根据点的两两相似度和一元匹配性预测点与点之间的分配。

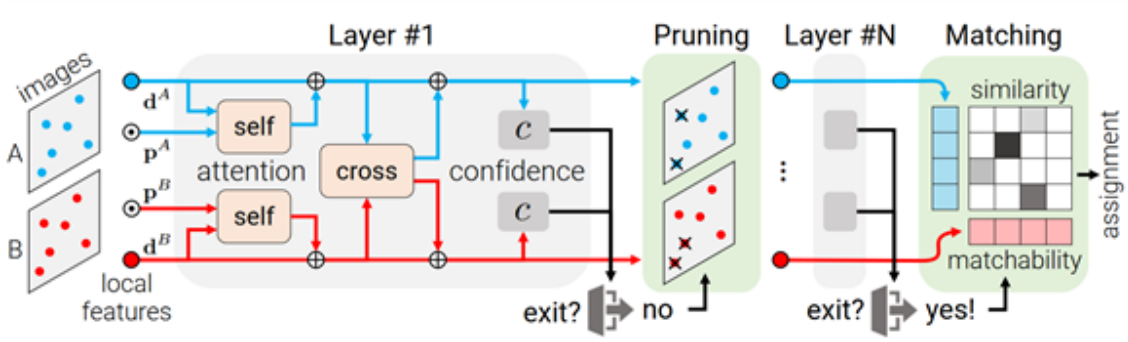

图 1. LightGlue 框架图

3.2 注意力机制单元

初始化：我们将图像 $I \in \{A, B\}$ 中的每个局部特征 i 与一个状态 $E \in \mathbb{R}^d$ 相关联。该状态由相应的视觉描述符 $x_i^l \leftarrow d_i^l$ 初始化，然后由每一层更新。我们将一个层定义为一个自注意力单元和一个交叉注意力单元的序列。在每个注意力单元中，由源图像 $S \in \{A, B\}$ 聚合的消息 $m_i^{I \leftarrow S}$，由多层感知器 (MLP) 更新状态：

$$x_i^l \leftarrow x_i^l + \text{MLP}([x_i^l | m_i^{l \leftarrow S}]) \tag{1}$$

其中 $[\cdot|\cdot]$ 为两个向量的叠加。该消息由一个注意力机制计算为图像 S 的所有状态 j 的加权平均：

$$m_i^{I \leftarrow S} = \sum_{j \in S} \text{Softmax}(a_{ik}^{IS})_j W x_j^S, \quad k \in S \tag{2}$$

其中 W 是一个投影矩阵，a_{ik}^{LS} 是图像 I 和 S 的点 i 和 j 之间的注意力分数。

3.3 对应预测

基于每层更新后的状态，LightGlue 设计了一个轻量级头部来预测分配。计算点对之间的相似性分数和每个点的可匹配性（matchability）分数，然后结合这些分数来预测软赋值矩阵。

分配分数：我们首先在两幅图像的点之间计算一个成对的分数矩阵 S：

$$S_{ij} = \text{Linear}(x_i^A)^T \text{Linear}(x_i^B)^T, \quad \forall (i, j) \in A \times B \tag{3}$$

其中 Linear 是一个带有偏差的学习线性变换，该分数编码了每一对应点，即同一 3D 点的 2D 投影。本文还计算了，对于每一个点，一个匹配性分数 (matchability score) 为 $\sigma_i = \text{Sigmoid}(\text{Linear}(x_i))$, 这个分数编码了 i 有一个对应点的可能性。在另一幅图像中未被检测到的点，例如被遮挡时，是不可匹配的，因此 $\sigma_i \to 0$, 这种类型会成为之后剪枝的点。

对应关系：将相似性和匹配性得分结合成一个软偏赋值矩阵 P：

$$P_{ij} = \sigma_i^A \sigma_i^A \prod_{k \in A} \text{Softmax}(S_{kj})_i \prod_{k \in B} \text{Softmax}(S_{ik})_j \tag{4}$$

当两个点都被预测为可匹配，并且它们的相似度高于两幅图像中的任何其他点时，一对点（i，j）就产生了对应关系。选择 P_{ij} 大于某个阈值 τ 且在行和列上都大于其他元素的对。

$$m_i^{I\leftarrow S} = \sum_{j\in S} \text{Softmax}(a_{ik}^{IS})_j W x_j^S, \quad k \in S \tag{5}$$

其中 W 是一个投影矩阵，a_{ik}^{LS} 是图像 I 和 S 的点 i 和 j 之间的注意力分数。

自注意力：每个点关注同一幅图像的所有点。我们对每幅图像 I 执行相同的后续步骤，因此为了清晰起见，删除了上标 I。对于每个点 i，首先通过不同的线性变换将当前状态 x_i 分解为键向量和查询向量 k_i 和 q_i。然后定义点 i 和 j 之间的注意力分数为 $a_{ij} = q_i^T R(p_j - p_i)k_j$，其中 R 是点与点之间相对位置的旋转编码。位置编码是注意力的一个关键部分，因为它允许根据位置来处理不同的元素。注意到，在射影相机几何学中，视觉观测值的位置是等变的，即相机在像平面内的平移：来自同一平行面上的 3D 点的 2D 点以相同的方式平移，并且它们的相对距离保持不变。这就需要一种只捕获点的相对位置而不捕获点的绝对位置的编码。旋转编码使得模型能够从 i 中检索到位于学习到的相对位置的点 j。位置编码不作用于值 v_j，因此不会溢出到状态 x_i。编码对所有层都是相同的，因此只计算一次并缓存。

交叉注意力:I 中的每个点关注另一幅图像 S 中的所有点，我们为每个元素计算一个键 k_i，但不进行查询。这样可以将得分表示为:

$$a_{ij}^{IS} = k_i^{I^T} k_j^S = a_{ij}^{SI} \tag{6}$$

因此，对于 I←S 和 S←I 消息，我们只需要计算一次相似度，这种技巧在之前被称为双向注意。由于该步骤较为昂贵，复杂度为 O（NMd），因此节省了 2 倍的显著因子。

3.4 自适应深度和宽度

LightGlue 增加了两种机制，避免了不必要的计算，节省了推理时间：

1) 根据输入图像对的难易程度，减少了层数；

2) 提前剪枝出自信被拒绝的点。

置信度分类器：LightGlue 的主干通过上下文增强输入的视觉描述符。如果图像对容易，即视觉重叠度高、外观变化小，这些往往是可靠的。在这种情况下，早期层的预测是自信的，并且与晚期层的预测完全一致。然后我们可以输出这些预测并停止推断。在每一层的最后，LightGlue 推断每个点的预测赋值的置信度：

$$c_i = \text{Sigmoid}(\text{MLP}(x_i)) \in [0, 1] \tag{7}$$

一个较高的值表明 i 的表示是可靠的和最终的，它有信心地匹配或不匹配。在最坏情况下，紧凑型 MLP 仅增加 2% 的推理时间，但通常情况下更节省。

退出准则：对于给定的 L 层，如果 $c_i > \gamma_i$，则认为该点是有信心的。如果所有点的足够比例 α 是有信心的，则停止推断：

$$\text{exit} = \left(\frac{1}{N+M} \sum_{I\in\{A,B\}} \sum_{i\in I} \left[c_i^I > \gamma_i \right] \right) > \alpha \tag{8}$$

分类器本身在早期层的置信度较低。因此，我们根据每个分类器的验证精度在整个层中衰减 y_i。退出阈值 α 直接控制了准确率和推理时间的权衡。

点剪枝：当不满足退出准则时，被预测为既可信又不可匹配的点不太可能在后续层中辅助其他点的匹配。这些点例如在图像中明显不可见的区域。因此，我们在每一层丢弃它们，只将剩下的点馈送到下一层。这在注意力的二次复杂度给定的情况下，显著减少了计算量，并且不影响准确率。

4 复现细节

4.1 与已有开源代码对比

本文参考了 LighGlue 和 SuperGlue 的源代码，主要对比了位置编码、预测匹配等模块的性能，并且使用一对 easy 图像对和一对 difficult 图像对进行 LightGlue 和 SuperGlue 运行时间的比较，LightGlue 受到 SuperGlue 的启发，但在对其准确性、效率和易训练性至关重要的方面有所不同。另外基于不同方法的特征点提取结果，比较了 SIFT + LightGlue 、SuperPoint+ LightGlue 和 DISK [8] + LightGlue 的输出结果以及特征点位置信息。LighGlue 和 SuperGlue 的对比如下：

位置编码：SuperGlue 使用 MLP 对绝对点位置进行编码，并与描述符进行早期融合。我们观察到模型在整个图层中都倾向于忘记这个位置信息。相反，LightGlue 依赖于一个相对编码，它在图像之间具有更好的可比性，并在每个自注意力单元中添加。这使得利用位置变得更加容易，提高了更深层的精度。

预测匹配部分：SuperGlue 通过使用 Sinkhorn 算法求解一个可微的最优运输问题来预测分配。它包括行向和列向归一化的多次迭代，在计算和内存方面都很昂贵。LightGlue 将相似性和匹配性分离开来，从而更有效地进行预测，这也产生了更简单的梯度。

深度监督：由于 Sinkhorn 算法是昂贵的，SuperGlue 在每一层之后都无法做出预测，只能在最后一层进行监督。LightGlue 较轻的头部使得在每一层预测一个任务并对其进行监督成为可能。这加快了收敛速度，并允许在任何层之后退出推理，这是 LightGlue 效率提升的关键。

易训练性：LightGlue 架构极大地提高了合成单应矩阵预训练的收敛速度。

4.2 实验环境搭建

实验在 VS Code 中的 WSL2(windows for linux) 平台进行，安装的系统是 Ubuntu 20.04.4 LTS，运行环境 Python==3.10.0。

4.3 创新点

本文复现的创新点主要体现在使用一对 easy 图像对和一对 difficult 图像对进行 LightGlue 和 SuperGlue 运行时间的比较，还使用采集数据比较了 SIFT + LightGlue 、SuperPoint+ LightGlue 和 DISK + LightGlue 的输出结果以及特征点位置信息。

本次实验的数据采集是从 DJI Phantom 4 无人机采集的 50 张深圳大学科技楼倾斜航空摄影测量的图像中选取了一张与地面方向角度差不多的图片，同时利用手机相机采集了地面

的影像用来进行特征匹配。

5 实验结果分析

对比不同的特征点提取算法在 LightGlue 下的匹配准确度，以及在相同的特征点提取算法下的特征匹配算法的准确度与 LightGlue 算法的准确度进行了对比。

5.1 基于不同方法的特征点提取结果对比

对比 LightGlue 针对不同局部特征生成的特征。通过比较两个最近邻特征点的距离来判断匹配的质量，这个方法是特征匹配中常用的筛选方法，即比值测试（Ratio Test）。比值测试的原理：在特征点检测和描述阶段，每个关键点都会有一个描述符（或特征向量），描述该关键点周围的图像信息。在进行特征匹配时，通常会计算两幅图像中的特征点之间的距离（通常使用欧氏距离或其他相似度度量）。对于每个特征点，找到它在另一幅图像中的最近邻点（最相似的点），然后计算它们之间的距离。例如：对于每个特征点 m，在另一幅图像中找到它的两个最近邻点 n1 和 n2，计算它们的距离。如果 m 到 n1 的距离远远小于 m 到 n2 的距离，即 m.distance<cc*n1.distance，则可以认为 m 和 n1 是一个有效的匹配对。这里的 cc 是一个比值阈值，用来判断距离是否足够小，通常设定为一个小于 1 的数值 [9]，本实验中设置为 0.75。通过比值测试，可以排除那些最近邻点之间差异很小的特征点对，从而选择出更有可能是正确匹配的特征点对。

SIFT + LightGlue 、SuperPoint+ LightGlue 和 DISK + LightGlue 的输出结果以及特征点位置信息如图 2、 3、 4所示：

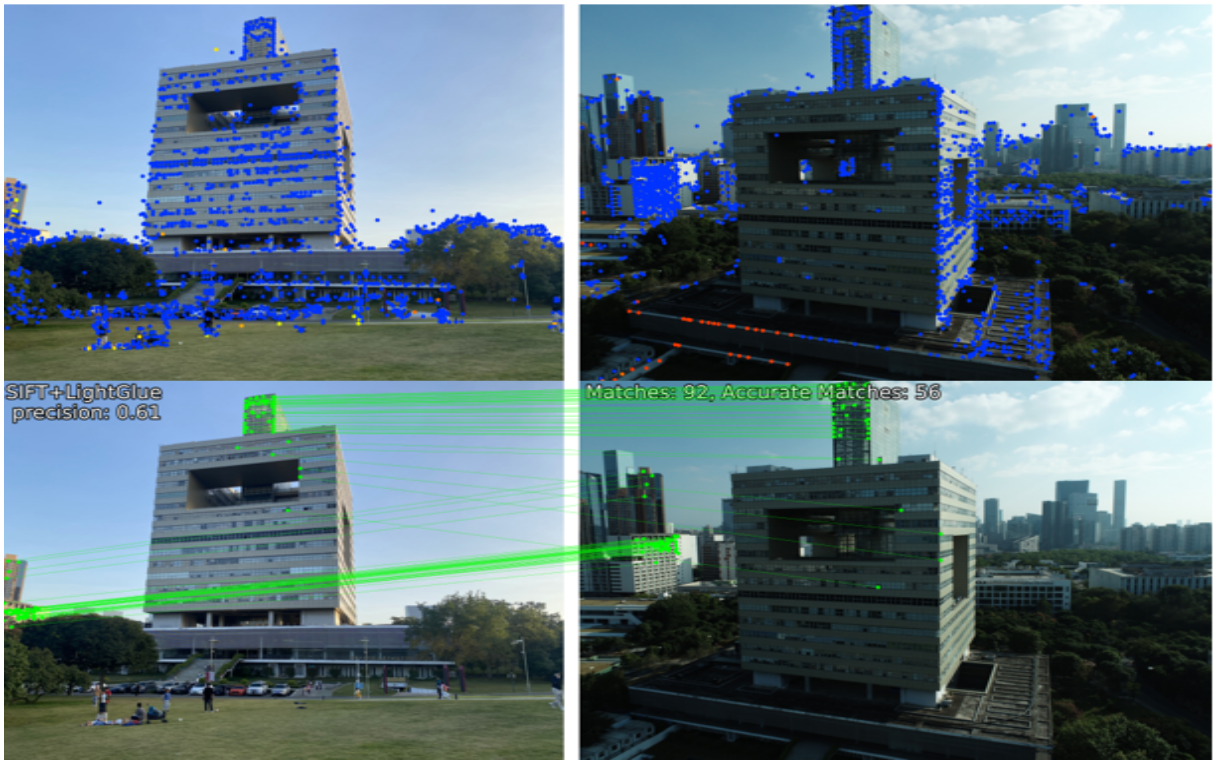

图 2. SIFT + LightGlue

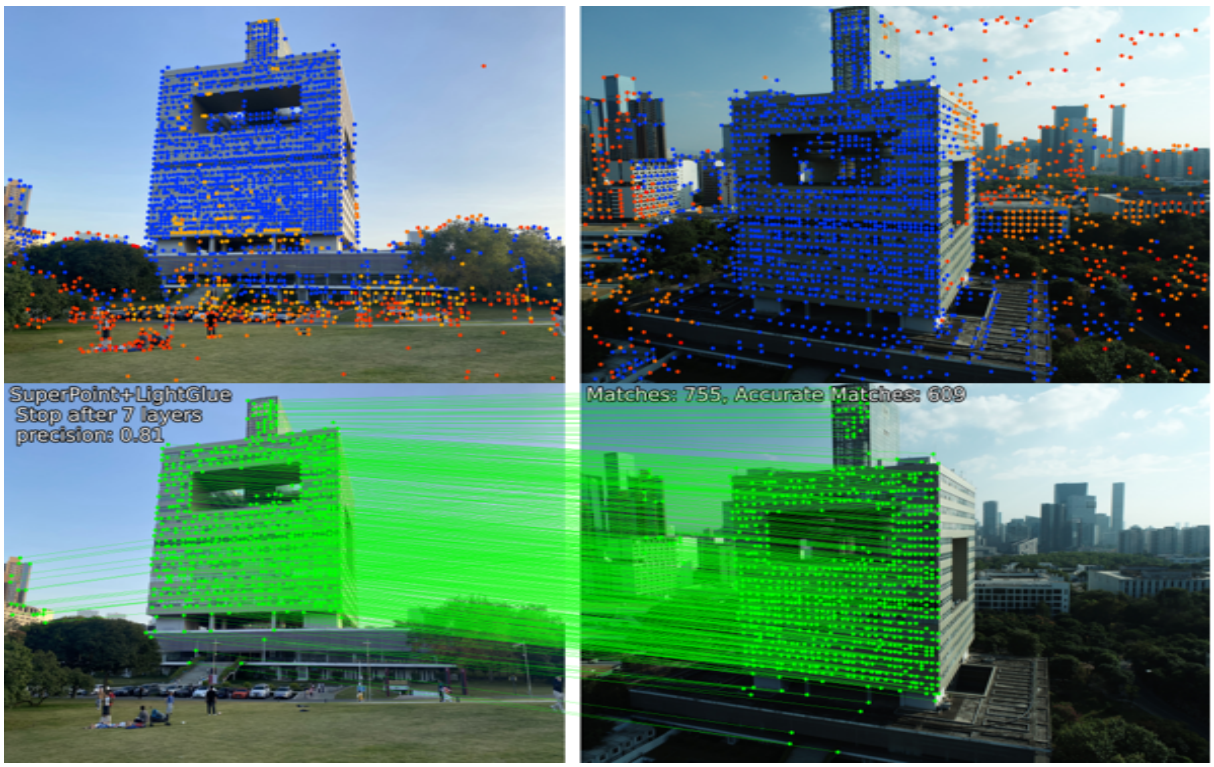

图 3. SuperPoint+ LightGlue

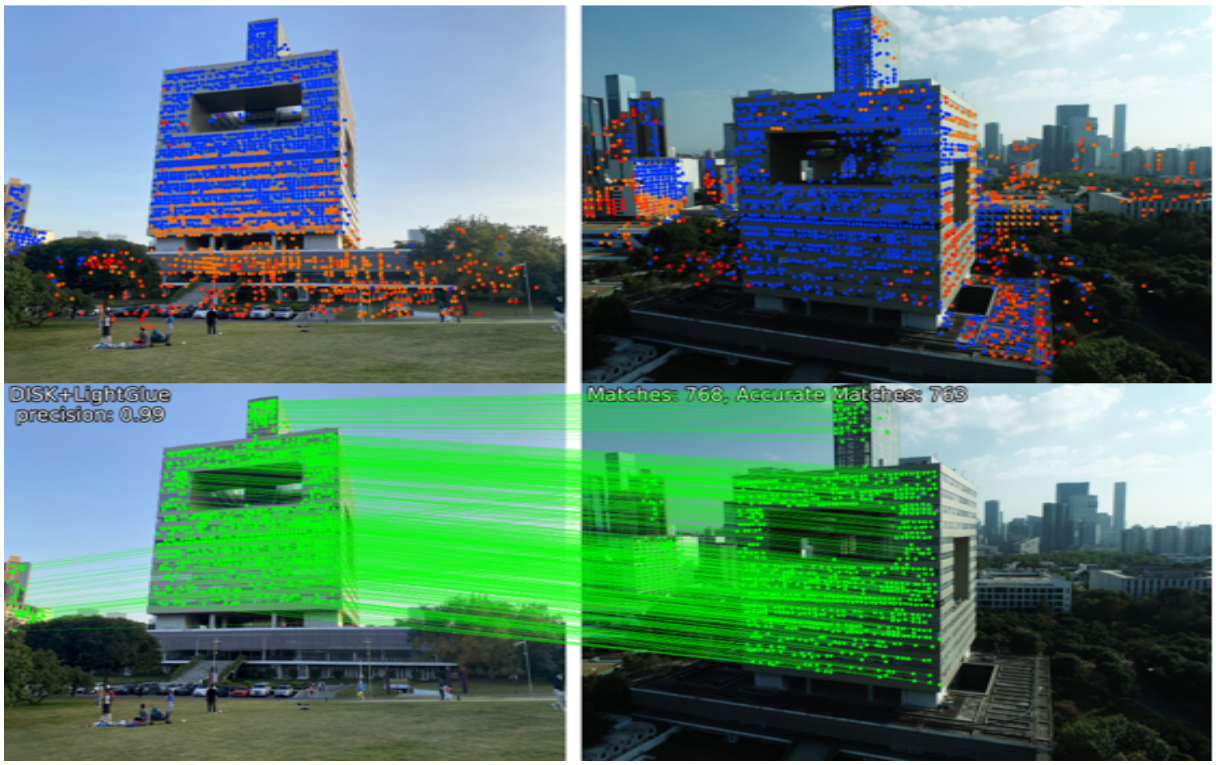

图 4. DISK + LightGlue

综合上述实验结果可以的出结论：SIFT 算法提取的特征点数量最少，只有 92 个，同时匹配的数量相对较少，准确率为 61%。Superpoint 提取的特征点和 DISK 提取的特征点数量差不多，但准确率没有 DISK 高。SIFT 作为传统算法已经落后了，SuperPoint 和 DISK 比传

统算法从各个方面都有优势。SLAM 系统在应用与实时应用要求比较高的情况下，可以使用 Super Point 和 DISK 特征提取算法。

5.2 LightGlue vs SuperGlue

对比所用图像对如图 5所示，其中一对图片是相似度非常高的图片对（上）称为 easy 对，另外一对是建筑在白天和黑夜的对比，相比较特征匹配比较困难，称为困难 difficult 对。

图 5. 对比图像对

对这两对图片进行运行时间的比较，分别在特征点提取的数量在 256，512,1024，2048，4096 上对 SuperGlue 和 Lightglue 进行对比，由于 SuperGlue 使用的 Sinkhorn 算法来预测分配，包括行向和列向归一化的多次迭代，在计算和内存方面比较昂贵。这里实验将迭代的次数设置为 5 次，减少迭代的次数来减少时间的消耗。通过测量重复次数 100 次的实验结果如图 6所示：

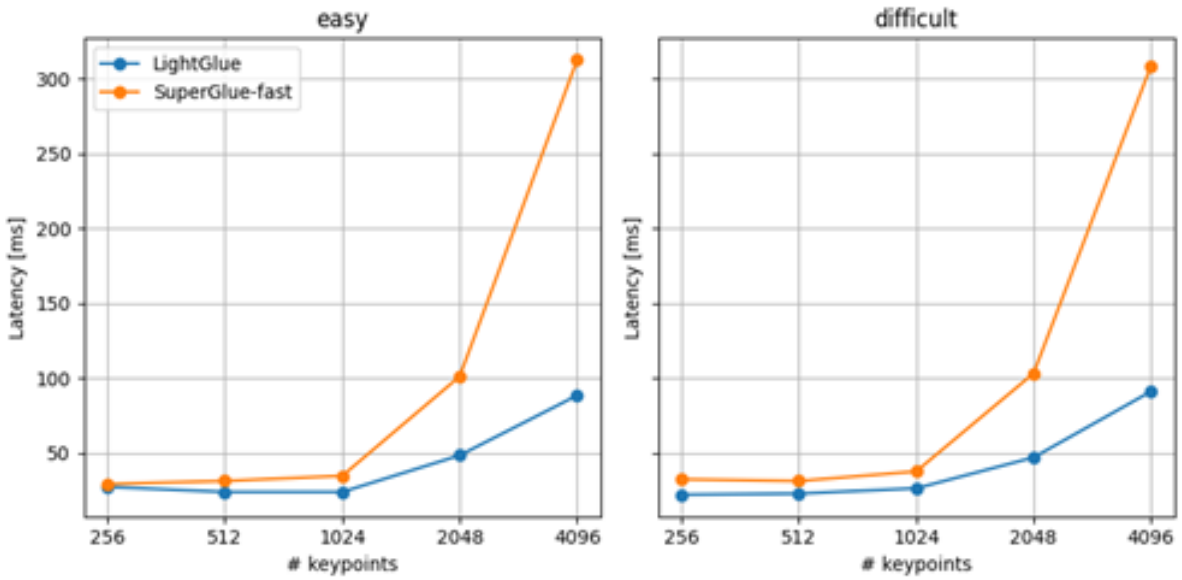

图 6. 运行时间对比

6 总结与展望

在本次复现中，探讨了 LightGlue 算法在特征匹配中的应用与优势。LightGlue 算法针对 SLAM 中的特征匹配问题，提供了高效、准确且易于训练的解决方案。在特征提取方面，传统方法如 SIFT 和 ORB 算法虽然成功，但在计算复杂度和效率上有所不足。基于深度学习的 SuperPoint 和 SuperGlue 算法显著提升了特征提取和匹配的效果，然而 SuperGlue 由于复杂度高、训练难度大、计算资源需求高，限制了其广泛应用。相较之下，LightGlue 通过对 SuperGlue 的多项设计进行改进，在内存和计算效率上取得了显著提升，且更易于训练。实验对比结果表明，LightGlue 在精度和效率方面均超越了 SuperGlue。通过在深圳大学科技楼空中和地面环境的图像数据集上的实验，验证了 LightGlue 在实际应用中的优越性能。LightGlue 不仅能够在短时间内达到最先进的精度，还能在资源有限的情况下进行高效训练，显著降低了应用门槛。

总的来说，LightGlue 算法在特征匹配领域展现了强大的潜力。其高效、准确和易于训练的特性，使其在 SLAM、视觉定位等 3D 计算机视觉任务中具有广泛的应用前景。未来，随着进一步优化和改进，LightGlue 有望在更多实际场景中得到应用和推广，为机器人、自主驾驶等技术的发展提供强有力的支持。

参考文献

[1] Philipp Lindenberger, Paul-Edouard Sarlin, and Marc Pollefeys. Lightglue: Local feature matching at light speed. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 17627–17638, 2023.

[2] Jiaming Sun, Zehong Shen, Yuang Wang, Hujun Bao, and Xiaowei Zhou. Loftr: Detector-free local feature matching with transformers. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8922–8931, 2021.

[3] David G Low. Distinctive image features from scale-invariant keypoints. *Journal of Computer Vision*, 60(2):91–110, 2004.

[4] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. Orb: An efficient alternative to sift or surf. In *2011 International conference on computer vision*, pages 2564–2571. Ieee, 2011.

[5] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superpoint: Self-supervised interest point detection and description. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 224–236, 2018.

[6] Paul-Edouard Sarlin, Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superglue: Learning feature matching with graph neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4938–4947, 2020.

[7] A Vaswani. Attention is all you need. *Advances in Neural Information Processing Systems*, 2017.

[8] Michał Tyszkiewicz, Pascal Fua, and Eduard Trulls. Disk: Learning local features with policy gradient. *Advances in Neural Information Processing Systems*, 33:14254–14265, 2020.