Practical Deep Feature-Based Visual-Inertial Odometry

Charles Hamesse[1,2][a], Michiel Vlaminck[2][b], Hiep Luong[2][c] and Rob Haelterman[1][d]
[1]*Department of Mathematics, Royal Military Academy, Belgium*
[2]*imec, IPI, URC, Ghent University, Belgium*

Keywords: Visual-Inertial Odometry, Visual Features, Deep Features.

Abstract: We present a hybrid visual-inertial odometry system that relies on a state-of-the-art deep feature matching front-end and a traditional visual-inertial optimization back-end. More precisely, we develop a fully-fledged feature tracker based on the recent SuperPoint and LightGlue neural networks, that can be plugged directly to the estimation back-end of VINS-Mono. By default, this feature tracker returns extremely abundant matches. To bound the computational complexity of the back-end optimization, limiting the number of used matches is desirable. Therefore, we explore various methods to filter the matches while maintaining a high visual-inertial odometry performance. We run extensive tests on the EuRoC machine hall and Vicon room datasets, showing that our system achieves state-of-the-art odometry performance according relative pose errors.

1 INTRODUCTION

Visual-inertial odometry (VIO) is a technique for estimating the 6-degree-of-freedom (6DoF) pose of a camera from a sequence of images and inertial measurements. It is widely used in robotics, augmented reality (AR), and virtual reality (VR) applications. If we add a loop detection and closure method to VIO, we have a complete Visual-Inertial Simultaneous Localization and Mapping (VISLAM) system. The increased accuracy makes that such SLAM systems are used in a wide range of large-scale industries such as autonomous vehicles, civil engineering or agriculture.

Traditional VIO and VISLAM methods rely on hand-crafted feature extraction and matching algorithms for their front-ends, i.e. the image processing pipeline between the sensor input and the optimization back-end. Prominent methods include VINS-Mono (Qin et al., 2018), ORB-SLAM3 (Campos et al., 2021) or OpenVINS (Geneva et al., 2020). These methods typically use descriptor or KLT-based feature tracking, both having their respective strengths and weaknesses. KLT tracking is usually faster but less robust to large viewpoint changes (which could be due to fast camera motion), poor image contrast (e.g. difficult visibility conditions) and occlusions. Descriptor-based tracking allows to track features in the longer term, but is more expensive computationally. One of the most studied VIO algorithms is VINS-Mono, which has achieved state-of-the-art performance for several years and whose open-source code is at the core of an important quantity of research works. To name a few, VINS-Fusion (Qin et al., 2019) extends VINS-Mono to multiple sensors, PLI-VINS (Zhao et al., 2022) extends VINS-Mono with line features, R2Live (Lin et al., 2021) uses VINS-Mono's feature tracker in a LiDAR-visual-inertial odometry pipeline.

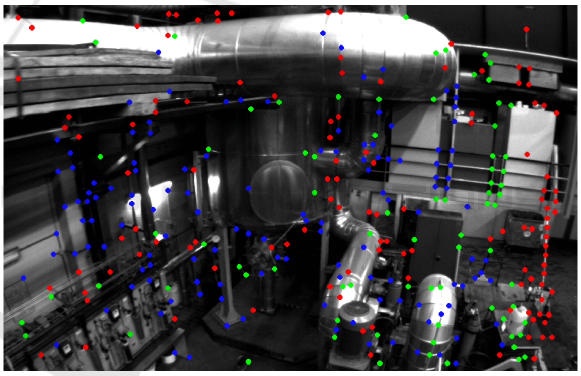
Figure 1: Features tracked with our system in the EuRoC sequence MH_01. The red-green-blue color code indicates features that have been observed in respectively 2 to 8, 8 to 16 and more than 16 frames.

[a] https://orcid.org/0000-0002-2321-0620
[b] https://orcid.org/0000-0003-3986-823X
[c] https://orcid.org/0000-0002-6246-5538
[d] https://orcid.org/0000-0002-1610-2218

Hamesse, C., Vlaminck, M., Luong, H. and Haelterman, R.
Practical Deep Feature-Based Visual-Inertial Odometry.
DOI: 10.5220/0012320200003654
Paper published under CC license (CC BY-NC-ND 4.0)
In *Proceedings of the 13th International Conference on Pattern Recognition Applications and Methods (ICPRAM 2024)*, pages 240-247
ISBN: 978-989-758-684-2; ISSN: 2184-4313
Proceedings Copyright © 2024 by SCITEPRESS – Science and Technology Publications, Lda.

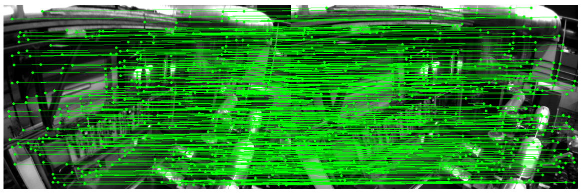
Figure 2: Matches obtained with SuperPoint and LightGlue on a sample frame pair of EuRoC's MH01 sequence, without filtering.

In recent years, there has been a significant progress in image feature extraction and matching techniques. More precisely, deep learning-based feature algorithms based on variants of convolutional neural networks (CNNs) have been shown to achieve state-of-the-art results. The applications of these methods are numerous (e.g., automated object tracking, motion-based segmentation), but they are particularly well suited for visual odometry tasks. The recently introduced LightGlue (Lindenberger et al., 2023) algorithm presents a significant increase of performance of the state-of-the-art of deep feature extraction and matching. One key property is that LightGlue is adaptive to the difficulty of the problem: the inference is much faster on image pairs that are intuitively easy to match, for example because of a larger visual overlap or limited appearance change. This is precisely the case for visual odometry tasks, where it is expected that adjacent frames have a large overlap.

By default, the combination of SuperPoint and LightGlue returns extremely abundant matches. Figure 2 shows the density of matches on a sample frame of the EuRoC's dataset MH01 sequence. For practical purposes, it is important to limit this number of matches that are actually used in VIO estimator backend, in order to bound the computational complexity. Therefore, it becomes relevant to develop ways of limiting the number of matches.

In this paper, we propose a new VIO front-end that uses a deep learning-based feature extraction and matching algorithm. Our method is built upon VINS-Mono and LightGlue and can serve as a direct replacement for VINS-Mono's default feature tracker, making it easy to integrate in existing, practical systems. Through experiments on the EuRoC micro-aerial vehicle dataset (Burri et al., 2016), we show that our feature tracker consistently improves the performance compared to the default VINS-Mono. Our contributions are the following:

1. We develop a fast and accurate deep feature tracker for visual odometry, which we integrate in the VINS-Mono VIO system. With default settings, this feature tracker already achieves better odometry performance than the ordinary VINS-Mono. In some variants, our feature tracker with VINS-Mono's backend achieves an accuracy superior to the current state-of-the-art deep feature-based HFNet-SLAM system (Liu and Aitken, 2023).
2. To bound the computational complexity of the visual-inertial estimator and thus guarantee a certain operational frame rate, we study different ways of filtering feature matches and limiting the number of feature observations returned by our feature tracker. This allows to pick a variant of our system according to the performance / speed trade-off for each application.

Following open science principles and to stimulate further work on our system, we make all of our contributions available open source[1].

2 RELATED WORK

Although odometry and SLAM systems exist with a plethora of sensor setups involving one or multiple cameras, LiDARs, IMUs, etc, the classic visual-inertial setup remains of great interest thanks to its simplicity, small physical dimension and performance. However, since our contribution does not affect the processing of IMU measurements in the VIO system, we start with an overview of the techniques to process visual information for odometry. We cover the main trends that have emerged in the field, starting from the inception of such methods in the 2000s with classical, hand-made features until today's neural network-based learned features.

2.1 Traditional Feature-Based Visual Odometry

Visual SLAM, or Simultaneous Localization and Mapping through visual data, has been a subject of extensive research and development over the years. The inception of monocular SLAM can be traced back to 2007 when Davison introduced MonoSLAM (Davison et al., 2007). This algorithm utilized an extended Kalman filter (EKF) to achieve real-time localization by leveraging large image patch features and sparse prior scene knowledge. However, it had limitations in handling occlusions, motion blur, and scenarios with sparse texture details. Some of these challenges were overcome with the introduction of the multi-state constraint Kalman filter (MSCKF)

[1]https://github.com/charleshamesse/LightGlue-VINS-Mono

in (Mourikis and Roumeliotis, 2007), which proposes a tightly coupled visual-inertial approach that jointly optimizes camera poses and IMU measurements. Although Kalman filter-based odometry systems are relatively simple and elegant, optimization-based methods have been introduced to increase accuracy. Odometry systems based on bundle adjustment optimization have been adopted with a great success, albeit at the cost of increased computational complexity. However, this increase can be controlled with a clever control of the number of variables considered in the optimization problem. This can be achieved by limiting the number of active feature tracks and using keyframe-based or sliding window optimization. VINS-Mono (Qin et al., 2018) is a robust visual-inertial odometry estimation system that maintains recent states of map points and cameras in a sliding window, optimized with local bundle adjustment. Visual features are tracked using KLT and optical flow. ORB-SLAM (Campos et al., 2021) uses ORB features and a descriptor-based approach for matching, which is more expensive computationally but can offer better long-term data association performance. (Note that both VINS-Mono and ORB-SLAM also have full SLAM capabilities, with loop closure detection and optimization methods.)

2.2 Deep Feature-Based Visual Odometry

Nowadays, deep learning techniques for feature extraction and matching significantly outperform their traditional counterparts. In (DeTone et al., 2018), the authors have introduced a self-supervised fully convolutional model called SuperPoint, capable of computing pixel-wise features with excellent matching capabilities. This has shown to be very successful in numerous subsequent works. For example, SuperPointVO (Han et al., 2020) proposes a stereo odometry system based on feature extraction by SuperPoint. The features are fed to a traditional stereo VO backend without loop closing. Experiments on the KITTI dataset show a performance close to other state-of-the-art stereo SLAM system. (Tang et al., 2019) moved from SuperPoint and introduced the GCNv2 network to replace the ORB algorithm for extracting local features in the RGB-D variant of the ORB-SLAM system. DF-SLAM (Kang et al., 2019) integrated learned features with a traditional stereo SLAM system, achieving both high accuracy and real-time performance. In a similar manner, LIFT-SLAM (Bruno and Colombini, 2020) uses the Learned Invariant Feature Transform (LIFT) proposed by (Yi et al., 2016) for feature detection, orientation estimation and description, then uses a traditional stereo ORB-SLAM backend. HFNet-SLAM (Liu and Aitken, 2023) integrates HFNet features (Sarlin et al., 2019) for feature tracking and loop detection on the ORB-SLAM3 framework, achieving state-of-the-art performance for full SLAM in various configurations (monocular and monocular-inertial). Furthermore, with the optimisation of TensorRT technology, the authors show that the entire system can run in real-time at 50 FPS on a standard Nvidia RTX2070 GPU. Unfortunately, HFNet-SLAM only reports full SLAM results with loop closing, making the comparison difficult. DXSLAM (Li et al., 2020) similarly integrates HFNet features, but this time in a traditional RGB-D SLAM backend (also based on ORB-SLAM).

Evaluation of visual odometry methods is commonly done on the EuRoC micro aerial vehicle dataset (Burri et al., 2016), the TUM dataset (Sturm et al., 2012) or KITTI (Geiger et al., 2013). Among all the deep feature-based methods mentioned in the previous paragraph, only HF-Net has an open source monocular-inertial variant against which we can compare our results.

3 METHOD

We start with an overview of the proposed VIO pipeline, then focus on each specific component and variant that we have implemented.

3.1 Overview

We propose a monocular visual-inertial odometry system called LightGlue-VINS-Mono. As the name indicates, this system is built upon the existing SuperPoint-LightGlue deep feature matcher (DeTone et al., 2018) (Lindenberger et al., 2023) and the traditional VINS-Mono visual-odometry system (Qin et al., 2018). To do so, we develop a fully-fledged visual feature tracker based on LightGlue, which we connect to the VINS-Mono VIO back-end. From an input-output perspective, our feature tracker functions exactly the same as the original feature tracker from VINS-Mono: it takes image frames as input, and returns feature observation tuples containing the feature identifier, its coordinates in the current image frame, etc. We implement a strategy to run the feature extraction only once on each image, saving the results to avoid unnecessary re-execution of SuperPoint. Then, for every pair of adjacent frames, we run LightGlue and then filter out potentially bad matches: the match filter component of our pipeline implements various

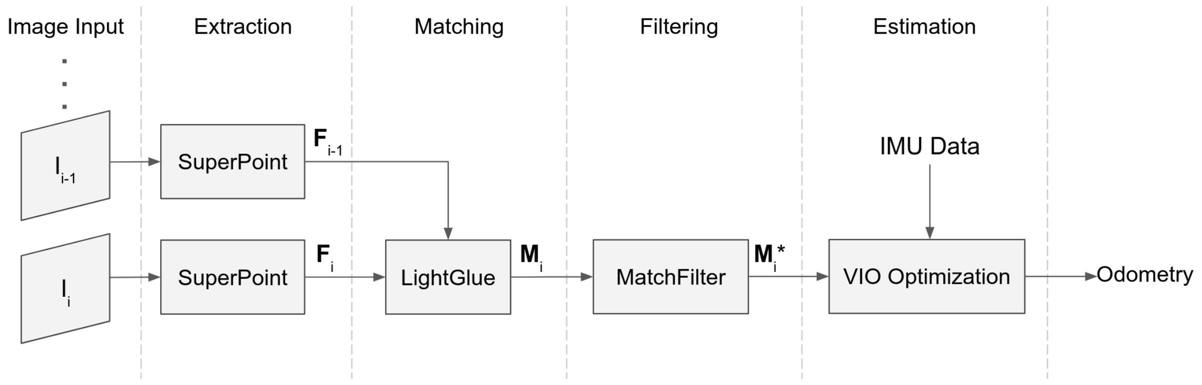

Figure 3: Overview of our visual-inertial odometry pipeline. Each incoming image frame $\mathbf{I}_i$ is fed to SuperPoint, which outputs features $\mathbf{F}_i$. Together with the previous features $\mathbf{F}_{i-1}$, they are fed to LightGlue, which returns the matches $\mathbf{M}_i$. Then, the MatchFilter component selects a relevant subset of all matches $\mathbf{M}_i^*$, which is finally fed to the state estimator back-end.

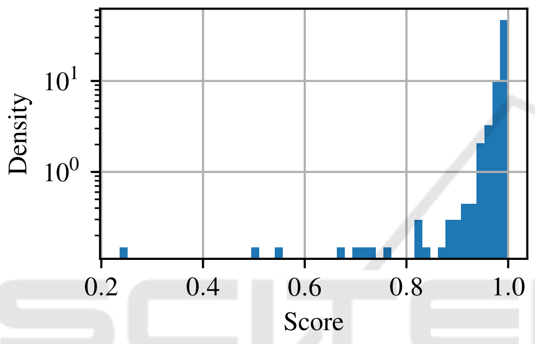

Figure 4: Histogram of the score values of the 443 matches returned by applying SuperPoint+LightGlue on one pair of frames, taken exactly in the middle of EuRoC's MH_01 sequence. Note that the vertical scale of the plot is logarithmic.

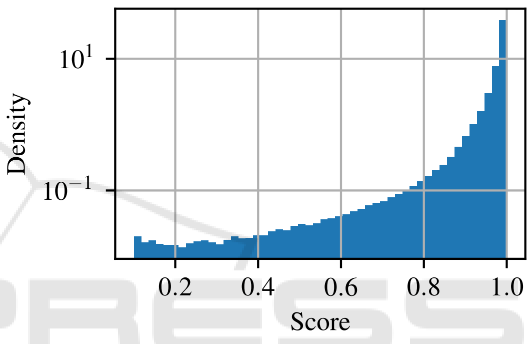

Figure 5: Histogram of the score values of the 1.5M matches returned by applying SuperPoint+LightGlue on all adjacent pairs of frames of EuRoC's MH_01 sequence. Note that the vertical scale of the plot is logarithmic.

strategies to reject matches based on their score output by LightGlue, as explained in the next sections.

A key difference between our LightGlue and VINS-Mono's traditional feature tracker is the abundance of matches that are returned. As shown in the introduction on Figure 2, SuperPoint and LightGlue return significantly more matches than the traditional tracker, even in the presence of large parallax or motion blur. However, the optimization-based VIO back-end has a computational complexity that grows drastically with this number of matches. Therefore, match filtering should be applied to bound the complexity of the optimization problem. To do so, we use an interesting property of LightGlue: its ability to return a confidence score with each match. Based on this score, we implement various match filtering strategies, as we will explain in the next sections.

Further operations in our VIO system follow the default VINS-Mono implementation. For simplicity and to better study the performance contribution of our feature tracker, we do not develop an associated loop closure method. If need be, one could append a loop closure method to our pipeline or use that from VINS-Mono. Figure 3 shows the main components of the LightGlue-VINS-Mono system. The following sections describe each component.

3.1.1 Feature Extraction

The SuperPoint neural network takes the i-th image $\mathbf{I}_i$ as input and returns the set of features:

$$\mathbf{F}_i = \{\mathbf{f}_{i,0}, \mathbf{f}_{i,1}, \ldots, \mathbf{f}_{i,N_{\text{features}}-1}\} \quad (1)$$

$$\text{where } \mathbf{f}_{ij} = \langle x_{ij}, y_{ij}, \mathbf{d}_{ij} \rangle \quad (2)$$

Here x_{ij}, y_{ij} and $\mathbf{d}_{ij}$ are respectively the coordinates of the j-th feature in the i-th image and the associated feature descriptor. N_{features} is the target number of features to find in each image, which is a user-input parameter of SuperPoint.

3.1.2 Feature Matching

Once the features are extracted for $\mathbf{I}_i$, the LightGlue feature matcher takes the current features $\mathbf{F}_i$ and pre-

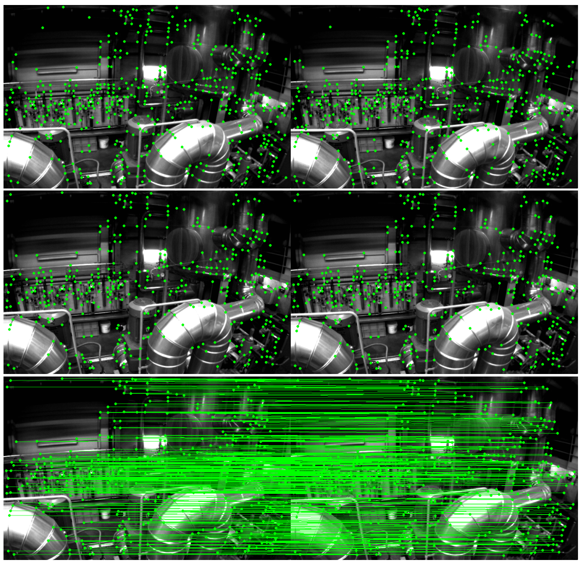
Figure 6: Matches returned for a frame pair in EuRoC's MH01 sequence. Top: all points found by SuperPoint, center: points used in matches found by LightGlue, bottom: points matched with connecting line.

vious features $\mathbf{F}_{i-1}$ and returns the following set of matches:

$$\mathbf{M}_i = \{\mathbf{m}_{i,0},\, \mathbf{m}_{i,1},\, \ldots,\, \mathbf{m}_{i,M-1}\} \tag{3}$$

$$\text{where } \mathbf{m}_k = \langle \mathbf{f}_{i-1,j^-},\, \mathbf{f}_{i,j^+},\, s_k \rangle \tag{4}$$

where M is the number of matches returned in the current frame pair which are indexed by k, j^- is the index of the matched feature in the set $\mathbf{F}_{i-1}$, and similarly for j^+ and $\mathbf{F}_i$. Then, $s_k \in [0,1]$ is the match confidence score. The match scores returned by LightGlue are often very close to the unit. We study their distribution and show the results in Figures 4 and 5. Moreover, LightGlue returns extremely abundant matches, successfully matching a large portion of feature points extracted by SuperPoint (see Figure 6).

Each feature needs a unique identifier number so that the VIO back-end may optimize its location in a window of frames. Therefore, we augment each match such that:

$$\hat{\mathbf{m}}_k = \langle \mathbf{f}_{i-1,j^-},\, \mathbf{f}_{i,j^+},\, s_k,\, n_k \rangle \tag{5}$$

where n_k is the unique feature identifier which can be either propagated from the previously matched frame pair if the feature was already observed, or created by incrementing the current total feature count if it is a newly observed feature.

3.1.3 Match Filtering

Our feature extraction and matching pipeline outputs extremely abundant matches, and we want to limit the number of matches actually used in the optimization back-end. This is important to limit the computational complexity of the visual-inertial optimization and guarantee a certain speed. We use the confidence score to implement two different strategies to discard these potentially wrong matches:

- The first strategy is to discard all matches $\mathbf{m}_k$ with a score s_k below some threshold. As shown on Figures 4 and 5, the scores returned by the feature matcher are mostly concentrated in values close to 1. Therefore, we implement two variants of the filter with the threshold $s_{k,\max}$ set at 0.9 and 0.95.
- The second strategy is to take the $M_{\max}$-best matches at each frame pair. We evaluate this method with the maximum number of matches $M_{\max}$ set to 300, 400 and 500.

The set of filtered matches for the i-th frame is noted $\mathbf{M}_i^*$ and is the complete output of the feature tracker which is fed to the VIO back-end. For its state-of-the-art performance and to make our solution practical, we use VINS-Mono's VIO back-end estimator, without any modification.

4 EVALUATION

We evaluate our system against the baseline VINS-Mono, the state-of-the-art traditional VIO, and HFNet-SLAM, a state-of-the-art deep feature-based VI-SLAM system. In both cases, we deactivate the loop closure methods to perform a fair evaluation, testing the pure visual-inertial odometry capabilities of the algorithms. We test several variants of our proposed system, using the score threshold-based match rejection strategy and the $M_{\max}$-best match retention strategy, and one without any filtering. All of our tests are executed on a laptop computer with an Intel i9-11900H CPU and an Nvidia RTX3080 GPU. All methods are executed on the same device and evaluated with the same method.

4.1 Accuracy

The metrics considered are the Absolute Trajectory Error (ATE) and the Relative Pose Error (RPE, computed for each adjacent frame pair), which we split into the Translation part (RPE-T) and Rotation part (RPE-R). It is common knowledge that the RPEs are more relevant than the ATE for the evaluation of visual odometry, as they target the local quality of the trajectory estimation. Also, the ATE is sensitive to when the errors happen along the trajectory (Zhang and Scaramuzza, 2018). Therefore, the ATE is more

Table 1: Evaluation of our proposed method LightGlue-VINS-Mono with the minimum s_k (set to 0.9 and 0.95), and the $M_{\max}$-best (from 200 to 1000) against state-of-the-art traditional and deep feature-based methods. The RMS Relative Pose Error in Translation (RPE-T) is reported in millimeters. VINS and HFNet represent VINS-Mono and HFNet-SLAM, both executed with the loop closure component deactivated and otherwise default settings. Bold indicates the best performance.

	VINS	HFNet	LightGlue-VINS-Mono (ours)										
Seq.	Def.	Def.	m-0.9	m-0.95	b-200	b-300	b-400	b-500	b-600	b-700	b-800	b-900	b-1000
MH_01	4.07	3.92	3.23	455.21	120.65	2.95	2.39	2.32	2.32	2.12	2.01	1.89	**1.86**
MH_02	3.19	3.40	2.41	6.33	2.67	2.09	1.85	1.72	2.08	1.90	1.76	1.66	**1.49**
MH_03	6.95	5.80	10.73	65.45	4.61	131.92	4.06	3.91	4.17	4.02	3.83	3.88	**3.77**
MH_04	7.46	6.55	58.28	79.54	5.37	4.62	4.51	4.47	4.47	4.39	4.35	4.37	**4.41**
MH_05	6.69	4.43	40.41	39.33	64.33	4.15	3.88	3.78	4.05	3.89	**3.70**	**3.70**	3.86
V1_01	5.27	3.39	3.24	3.43	3.21	2.92	2.80	2.75	2.88	2.82	**2.74**	2.80	2.78
V1_02	5.76	2.86	3.07	74.70	87.36	2.88	2.78	2.78	2.83	2.77	2.78	2.77	**2.75**
V1_03	6.42	3.44	34.43	44.16	3.66	2.98	2.79	3.27	2.78	2.75	2.76	2.77	**2.73**
V2_01	3.10	3.32	2.77	3.31	2.81	1.91	1.62	1.47	1.74	1.61	1.47	**1.39**	1.46
V2_02	4.52	3.19	2.86	3.50	61.61	2.33	2.01	1.97	2.12	1.97	1.91	**1.89**	**1.89**
V2_03	7.39	3.32	75.78	54.58	5.24	3.75	3.62	3.62	3.49	3.57	3.54	**3.45**	**3.45**

Table 2: Evaluation of our proposed method LightGlue with the minimum s_k (set to 0.9 and 0.95), and the $M_{\max}$-best (from 200 to 1000) against state-of-the-art traditional and deep feature-based methods. For readability, the RMS Relative Pose Error in Rotation (RPE-R) is reported in hundredths of degrees. VINS and HFNet represent VINS-Mono and HFNet-SLAM, both executed with the loop closure component deactivated and otherwise default settings. Bold indicates the best performance.

	VINS	HFNet	LightGlue-VINS-Mono (ours)										
Seq.	Def.	Def.	m-0.9	m-0.95	b-200	b-300	b-400	b-500	b-600	b-700	b-800	b-900	b-1000
MH_01	0.41	0.39	0.32	45.52	12.06	0.29	0.24	0.23	0.23	0.21	0.20	0.19	**0.19**
MH_02	0.32	0.34	0.24	0.63	0.27	0.21	0.18	0.17	0.21	0.19	0.18	0.17	**0.15**
MH_03	0.69	0.58	1.07	6.55	0.46	13.19	0.41	0.39	0.42	0.40	0.38	0.39	**0.38**
MH_04	0.75	0.66	5.83	7.95	0.54	0.46	0.45	0.45	0.45	0.44	**0.43**	0.44	0.44
MH_05	0.67	0.44	4.04	3.93	6.43	0.41	0.39	0.38	0.41	0.39	**0.37**	**0.37**	0.39
V1_01	0.53	0.34	0.32	0.34	0.32	0.29	0.28	**0.27**	0.29	0.28	**0.27**	0.28	0.28
V1_02	0.58	0.29	0.31	7.47	8.74	0.29	0.28	0.28	0.28	0.28	0.28	0.28	**0.27**
V1_03	0.64	0.34	3.44	4.42	0.37	0.30	0.28	0.33	0.28	0.28	0.28	0.28	**0.27**
V2_01	0.31	0.33	0.28	0.33	0.28	0.19	0.16	0.15	0.17	0.16	0.15	**0.14**	0.15
V2_02	0.45	0.32	0.29	0.35	6.16	0.23	0.20	0.20	0.21	0.20	**0.19**	**0.19**	**0.19**
V2_03	0.74	0.33	7.58	5.46	0.52	0.38	0.36	0.36	**0.35**	0.36	**0.35**	**0.35**	**0.35**

relevant for full SLAM systems, where the aim is to produce globally consistent trajectories. However, since it remains very often reported, we chose to compute both type of metrics in this paper.

Tables 1 and 2 report the results of the RMS RPE-T and RPE-R, respectively. On both of these tables, we see that LightGlue-VINS-Mono in the higher $M_{\max}$ variants achieves state-of-the-art performance, with an RMS RPE-T significantly lower than VINS-Mono and HFNet-SLAM in most cases. Also, we see that the threshold-based match rejection results in relatively poor performance. This might be due to the fact that in some sequences, the camera motion is such that very few matches can be found in frame pairs, resulting in too few matches being used with such a strong rejection policy. With the $M_{\max}$-best retention strategy, we ensure that some matches are always used, even if they are relatively scarce. An interesting remark is that the accuracy continues to increase as we add more matches (and thus allowing lower quality matches compared to lower $M_{\max}$ variants), which testifies for the general quality of matches returned by SuperPoint and LightGlue.

Table 3 reports the results of the RMS ATE. Here, unlike with the relative errors, HFNet-SLAM shows a consistent superior performance. This could be explained by the fact that it matches features in the current frame against all known features in the local map,

Table 3: Evaluation of our proposed method LightGlue with the minimum s_k (set to 0.9 and 0.95), and the $M_{\max}$-best (from 200 to 1000) against state-of-the-art traditional and deep feature-based methods. The RMS Absolute Translation Error (ATE) is reported in meters. VINS and HFNet represent VINS-Mono and HFNet-SLAM, both executed with the loop closure component deactivated and otherwise default settings. Bold indicates the best performance.

	VINS	HFNet	LightGlue-VINS-Mono (ours)										
Seq.	Def.	Def.	m-0.9	m-0.95	b-200	b-300	b-400	b-500	b-600	b-700	b-800	b-900	b-1000
MH_01	1.49	**1.24**	2.43	26.10	72.04	1.67	1.53	1.59	2.18	2.15	2.04	1.86	1.88
MH_02	2.47	**1.03**	2.65	6.42	3.63	2.67	2.21	2.12	3.09	2.67	2.32	2.50	2.18
MH_03	1.79	**1.15**	2.94	4.07	2.80	161.76	1.66	1.61	1.73	1.49	1.38	1.40	1.28
MH_04	1.03	**1.00**	1.06	1.60	1.65	1.28	1.19	1.31	1.40	1.43	1.35	1.36	1.57
MH_05	0.84	**0.60**	0.87	0.95	6.00	1.11	1.00	0.92	1.10	1.06	1.12	0.98	1.28
V1_01	6.19	5.51	6.29	6.65	6.20	5.87	5.50	5.42	**5.25**	5.26	5.31	5.56	5.34
V1_02	2.62	**1.99**	2.46	34.43	173.23	2.55	2.12	2.23	2.30	2.28	2.27	2.27	2.30
V1_03	5.29	**2.36**	26.82	171.53	5.47	3.91	3.38	3.98	3.18	3.18	3.31	3.34	3.27
V2_01	1.87	**0.78**	3.03	6.21	3.00	1.73	1.03	1.00	1.51	1.33	1.21	1.06	1.21
V2_02	2.68	**0.97**	2.49	3.33	175.51	1.90	1.86	1.88	1.66	1.65	1.68	1.67	1.67
V2_03	3.28	**0.90**	8.21	12.39	9.24	2.88	2.56	2.61	1.95	2.66	2.59	2.12	2.15

allowing to potentially re-use features that are not detected in a given past frame, leading to better global trajectory consistency. This is not the case with our tracker: if a feature was detected in frame $i-1$ and i but not in frame $i+1$, detecting it again in frame $i+2$ will have our matcher treat it as a new feature and the previous feature track is lost. In other words, our matcher only operates on adjacent frame pairs, which could also explain its better performance in relative errors. The combination of both approaches would be an interesting direction for further research.

4.2 Computational Complexity

In terms of computational complexity, the evaluation is harder to perform as the time needed by the optimizer of VINS-Mono to converge depends on numerous factors: the number of matches, which directly impacts the number of feature observations for which we need to compute a residual, and the current state of the system with respect to the optimization landscape, including the minimum to reach.

To give an order of magnitude of the influence of the number of matches used in the optimization, we measure the average iteration time in $M_{\max}$-best variants of our system, as well as the default VINS-Mono system. Table 4 reports these measurements. In practice, we can limit the number of iterations or the total time used by the solver, potentially stopping before reaching the local minimum[2]. Our variants can be slightly or severely slower than the default VINS-Mono, and thus depending on the target application and resources, we may choose a certain match filtering strategy.

Table 4: Evaluation of the average iteration time required by VINS-Mono's estimator back-end, computed on the MH01 sequence. Default means VINS Mono's default feature tracker.

Variant	Iteration time [ms]
Default	1.67
b-300	2.11
b-400	3.23
b-500	3.94
b-600	4.11
b-700	4.78
b-800	5.38
b-900	6.70
b-1000	7.41

5 CONCLUSIONS

In this paper, we have presented a practical visual-inertial odometry system that relies on state-of-the-art deep feature matching and traditional visual-inertial optimization-based state estimation. More precisely, we develop a fully-fledged feature tracker that can be plugged directly to the estimation back-end of VINS-Mono. Our deep feature tracker uses SuperPoint, LightGlue, and custom methods to limit the num-

[2]For example, on the EuRoC dataset, the default max iteration number for VINS-Mono is 8, and the total solver time is 40ms, so that it can achieve real-time at roughly 20 frames per second.

ber of matches to bound the complexity of the VIO optimization while maintaining maximum accuracy. Our extensive tests on the EuRoC datasets show that our system achieves state-of-the-art odometry performance according to relative translation and rotation errors, at the cost of a slight increase of computational complexity. Now, our tests highlight different ways in which this work could be improved:

1. A dedicated, deep feature-based loop closure system could be appended for full SLAM capability.
2. As shown in the tests, the local map-based matching of HFNet-SLAM and ORB-SLAM could potentially improve the global consistency of the trajectories returned by our system.
3. The current implementation of the feature tracker is in Python and PyTorch; important speed improvements could be obtained by switching to a C++ / TensorRT implementation.

Following open science principles and to stimulate work in the directions mentioned hereabove, we open source the code of our system (upon acceptance).

REFERENCES

Bruno, H. M. S. and Colombini, E. (2020). Lift-slam: a deep-learning feature-based monocular visual slam method. *Neurocomputing*, 455:97–110.

Burri, M., Nikolic, J., Gohl, P., Schneider, T., Rehder, J., Omari, S., Achtelik, M. W., and Siegwart, R. (2016). The euroc micro aerial vehicle datasets. *The International Journal of Robotics Research*.

Campos, C., Elvira, R., Gomez, J. J., Montiel, J. M. M., and Tardos, J. D. (2021). ORB-SLAM3: An accurate open-source library for visual, visual-inertial and multi-map SLAM. *IEEE Transactions on Robotics*, 37(6):1874–1890.

Davison, A. J., Reid, I. D., Molton, N., and Stasse, O. (2007). Monoslam: Real-time single camera slam. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29:1052–1067.

DeTone, D., Malisiewicz, T., and Rabinovich, A. (2018). Superpoint: Self-supervised interest point detection and description. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*.

Geiger, A., Lenz, P., Stiller, C., and Urtasun, R. (2013). Vision meets robotics: The kitti dataset. *International Journal of Robotics Research (IJRR)*.

Geneva, P., Eckenhoff, K., Lee, W., Yang, Y., and Huang, G. (2020). OpenVINS: A research platform for visual-inertial estimation. In *Proc. of the IEEE International Conference on Robotics and Automation*, Paris, France.

Han, X., Tao, Y., Li, Z., Cen, R., and Xue, F. (2020). Superpointvo: A lightweight visual odometry based on cnn feature extraction. In *2020 5th International Conference on Automation, Control and Robotics Engineering (CACRE)*, pages 685–691.

Kang, R., Shi, J., Li, X., Liu, Y., and Liu, X. (2019). Df-slam: A deep-learning enhanced visual slam system based on deep local features. *ArXiv*, abs/1901.07223.

Li, D., Shi, X., Long, Q., Liu, S., Yang, W., Wang, F., Wei, Q., and Qiao, F. (2020). DXSLAM: A robust and efficient visual SLAM system with deep features. *arXiv preprint arXiv:2008.05416*.

Lin, J., Zheng, C., Xu, W., and Zhang, F. (2021). R2live: A robust, real-time, lidar-inertial-visual tightly-coupled state estimator and mapping. *IEEE Robotics and Automation Letters*, 6:7469–7476.

Lindenberger, P., Sarlin, P.-E., and Pollefeys, M. (2023). LightGlue: Local Feature Matching at Light Speed. In *ICCV*.

Liu, L. and Aitken, J. M. (2023). Hfnet-slam: An accurate and real-time monocular slam system with deep features. *Sensors*, 23(4).

Mourikis, A. I. and Roumeliotis, S. I. (2007). A multi-state constraint kalman filter for vision-aided inertial navigation. *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pages 3565–3572.

Qin, T., Li, P., and Shen, S. (2018). Vins-mono: A robust and versatile monocular visual-inertial state estimator. *IEEE Transactions on Robotics*, 34(4):1004–1020.

Qin, T., Pan, J., Cao, S., and Shen, S. (2019). A general optimization-based framework for local odometry estimation with multiple sensors.

Sarlin, P.-E., Cadena, C., Siegwart, R., and Dymczyk, M. (2019). From coarse to fine: Robust hierarchical localization at large scale. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12716–12725.

Sturm, J., Engelhard, N., Endres, F., Burgard, W., and Cremers, D. (2012). A benchmark for the evaluation of rgb-d slam systems. In *Proc. of the International Conference on Intelligent Robot Systems (IROS)*.

Tang, J., Ericson, L., Folkesson, J., and Jensfelt, P. (2019). Gcnv2: Efficient correspondence prediction for real-time slam. *IEEE Robotics and Automation Letters*, 4(4):3505–3512.

Yi, K., Trulls, E., Lepetit, V., and Fua, P. (2016). Lift: Learned invariant feature transform. volume 9910, pages 467–483.

Zhang, Z. and Scaramuzza, D. (2018). A tutorial on quantitative trajectory evaluation for visual(-inertial) odometry. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7244–7251.

Zhao, Z., Song, T., Xing, B., Lei, Y., and Wang, Z. (2022). Pli-vins: Visual-inertial slam based on point-line feature fusion in indoor environment. *Sensors*, 22(14).