2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 14-18, 2024. Abu Dhabi, UAE

LF^{2}SLAM: Learning-based Features For visual SLAM

Marco Legittimo[1], Francesco Crocetti[1], Mario Luca Fravolini[1], Giuseppe Mollica[2], Gabriele Costante[1]

***Abstract*— Autonomous robot navigation relies on the robot's ability to understand its environment for localization, typically using a Visual Simultaneous Localization And Mapping (SLAM) algorithm that processes image sequences. While state-of-the-art methods have shown remarkable performance, they still have limitations. Geometric VO algorithms that leverage hand-crafted feature extractors require careful hyperparameter tuning. Conversely, end-to-end data-driven VO algorithms suffer from limited generalization capabilities and require large datasets for their proper optimizations. Recently, promising results have been shown by hybrid approaches that integrate robust data-driven feature extraction with the geometric estimation pipeline. In this work, we follow these intuitions and propose a hybrid VO method, namely Learned Features For SLAM (LF^{2}SLAM), that combines a deep neural network for feature extraction with a standard VO pipeline. The network is trained in a data-driven framework that includes a pose estimation component to learn feature extractors that are tailored for VO tasks. A novel loss function modification is introduced, using a binary mask that considers only the informative features. The experimental evaluation performed shows that our approach has remarkable generalization capabilities in scenarios that differ from those used for training. Furthermore, LF^{2}SLAM exhibits robustness in more challenging scenarios, *i.e.*, characterized by the presence of poor lighting and low amount of texture, with respect to the state-of-the-art ORB-SLAM3 algorithm.**

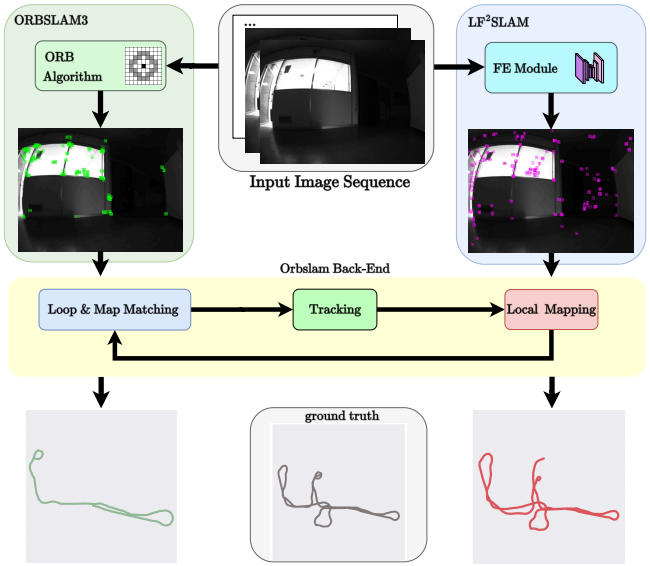

Fig. 1: Comparison between the hybrid LF^{2}SLAM and the state-of-the-art ORBSLAM3 approaches: in challenging scenarios, the proposed data-driven Feature Extraction Module (FE Module) achieves better performances compared to the ORB geometric algorithm.

Supplementary Material

Code: github.com/isarlab-department-engineering/LFFS

I. Introduction

In recent years, vision-based algorithms such as Visual SLAM (VSLAM) and Visual Odometry (VO), have garnered significant interest in robotics [1], [2]. These algorithms take advantage of the continuous reduction of sensors' cost and size. Although depth cameras (Depth-RGB), stereo cameras, and inertial sensors (IMU) are today widely used to enhance the robustness and accuracy of VO algorithms, monocular approaches remain appealing for several robotic applications due to their lightweight setups and the more straightforward calibration procedures [3]. The impressive results achieved by cutting-edge VSLAM algorithms like ORB-SLAM3 and Direct Sparse Mapping (DSM) [4], [5] have clearly shown the effectiveness of the two image-matching paradigms: feature-based and direct methods [6]. These strategies lay their foundation on geometric models that exploit feature matching or tracking to estimate the camera pose transformation. Extracting robust visual features is, therefore, decisive for the overall pose estimation procedure. However, hand-crafted feature extractors have been demonstrated to lack robustness to non-ideal image characteristics, which is a common condition in practical applications.

A new paradigm shift has started to emerge with the advent of Deep Learning (DL) and Machine Learning (ML) methodologies, giving a new thrust in the development of more robust algorithms [7]. At first, end-to-end approaches based on Convolutional Neural Networks (CNN) have been proposed to directly estimate global camera motion and pose from the image stream [8], [9], [10]. Despite the promising results, the primary drawback limiting end-to-end approaches from overtaking traditional geometric approaches is the relevant training effort requested and the difficulty in collecting rich datasets that ensure adequate generalization

[1]The authors are with the Department of Engineering, University of Perugia, 06125 Perugia, Italy. email:{francesco.crocetti, gabriele.costante, marco.legittimo, mario.fravolini}@unipg.it.

[2]The author is with ART Spa, Voc. Pischiello, 20 - 06065 Passignano sul Trasimeno. email:giuseppe.mollica@artgroup-spa.com

This work was in part supported by PNRR-M4C2 – I1.1 – MUR Call for proposals n. 1409 del 14-09-2022 - Bando PRIN 2022PNRR - ERC sector PE6- Project: LiSA - Listen, See and Act: fusing audio-video cues to perceive visible and invisible events and develop perception-to-action solutions for autonomous vehicles - Project Code P2022MEBFM- CUP Code D53D23017510001 (CUP Code of the University of Perugia Unit: J53D23015010001) funded by the European Union – NextGen- erationEU, and in part by the University of Perugia, Fondo di Ricerca di Ateneo 2022, Project "MiRA: Mixed Reality and AI Methodologies for Immersive Robotics"

over different application scenarios.

Recently, a new research direction has emerged, intending to combine the capability of DL models to extract robust features with geometric models to take advantage of the strengths of both paradigms. Hence, hybrid methods started to flourish, integrating data-driven feature extractors [11], [12], [13] with the geometric VO or SLAM pipelines. Alleviating the need for extensive training datasets with task-specific labeling, these methods improve the performance of the overall estimation system.

Although hybrid methods have been successfully applied in numerous applications, their potential has not been thoroughly evaluated in particularly challenging environments such as low light and poorly textured environments.

Within this line of research, the present study introduces the novel LF^{2}SLAM (Learned Features For SLAM) architecture, a CNN-based feature extraction module blended in a standard state-of-the-art VSLAM algorithm.

LF^{2}SLAM is a novel hybrid approach specifically designed to overcome the limitations of the environment-depend feature extraction processes that affect the traditional geometric VSLAM pipelines, Figure 1. Our algorithm leverages the Superpoint algorithm [14], adapting it to extract sparse keypoints and descriptors within an image for pose estimation purposes. Specifically, we combine the Superpoint network with the Monodepth2 [15] framework, and introduce a loss function to learn sparse feature extractors that minimize the pose estimation error. Extensive tests demonstrate that LF^{2}SLAM achieves better performance than its counterpart that employs hand-crafted feature extractors in challenging scenarios, where darkness and poor texture conditions may obstacle the feature extraction process. To better evaluate the performance, we also collected a custom challenging indoor dataset to assess the generalization capabilities and the robustness of the proposed approach.

II. Related Work

The robot's ability to localize itself by using vision sensors is typically achieved by relying on VO or VSLAM [16], [17] techniques. As these approaches rely on feature extraction to compute the pose transformation, several hand-crafted algorithms, such as SIFT [18], FAST [19], and ORB [20] have been with vision-based pose estimation pipelines in the last decades. ORB, in particular, has become popular thanks to its fundamental role in the ORBSLAM framework [21]. This framework has seen further improvements in its subsequent versions [22], [4]. Despite the impressive results achieved, these strategies suffer from different limitations: i) they are, in general, not robust to image non-idealities, and ii) tuning hyper-parameters is non-trivial since it is strongly dependent on environmental aspects, *e.g.*, light conditions and the amount of texture in the scene.

Due to these factors, alternatives to hand-engineered feature extractors have been proposed. by exploiting data-driven approaches [23]. Specifically, deep learning models that leverage CNNs can compute features invariant to geometric and photometric changes, including illumination, background, viewpoint, and scale [24]. The first contribution in this field is [25]. The authors suggest a straightforward technique to pinpoint potentially stable points in training images, along with a regression model that identifies keypoints unaffected by variations in weather and lighting conditions. DetNetm [26], instead, is a neural network that learns to compute local covariant features. This approach is then improved with TCDET [27], which ensures the detection of discriminative image features and guarantees repetitiveness. Finally, in [28], hand-engineered and learned filters are combined to obtain a lightweight and efficient detector with real-time performance.

Differently from these works that treat feature detection and description as separate problems, in the last few years, methods that jointly detect features and compute the associated descriptors have been proposed [29], [14], [30].

The integration of the aforementioned learning-based feature extractors into VO and SLAM systems has been recently considered. DXSLAM [31], for instance, exploits learned features to obtain scene representations that are robust to environmental changes and combines them into the SLAM pipeline. DROID-SLAM [32], instead, is characterized by an end-to-end recurrent architecture that can be easily adapted to different setups (monocular, stereo, and RGB-D). Other approaches, such as [33] and its improved version [34], exploit Graph Convolutional Neural Networks (GCN) [35] and RGB-D sensors to estimate the agent's pose.

In [36], [37] Superpoint feature extractor with standard VO/SLAM pipelines is integrated. Specifically, the former combines it with ORBSLAM2 [22] and with GCNv2 [34], analyzing the performance of both integrations on a single dataset. However, the impact of the different hyper-parameters on the pose estimation accuracy is not analyzed. [37], instead, focuses on the integration of Superpoint with ORBSLAM3. An in-depth analysis is presented, as well as a hyper-parameters exploration to evaluate performances. However, both methods employ the pretrained Superpoint network provided by Magic Leap[1], hence, the feature extractors are not specifically trained for VO or SLAM tasks.

In contrast to previous methods, in this work, we propose a novel approach, namely LF^{2}SLAM, that exploits the Superpoint feature extractor network and the Monodepth2 [15] training strategy. Specifically, we apply the image warping framework from Monodepth2 to the sparse feature extracted from Superpoint and propose an image reconstruction loss that promotes the learning of sparse feature extractors specific to the pose estimation task. After optimization, the learned sparse feature extractor is integrated into ORB-SLAM3 by replacing the ORB algorithm.

III. The LF^{2}SLAM Framework

The novelty of the LF^{2}SLAM framework lies in the development of a deep data-driven network capable of extracting visual features suitable for pose estimation. These features can then be used by a geometric VO/SLAM pipeline.

[1]https://github.com/magicleap/SuperPointPretrainedNetwork

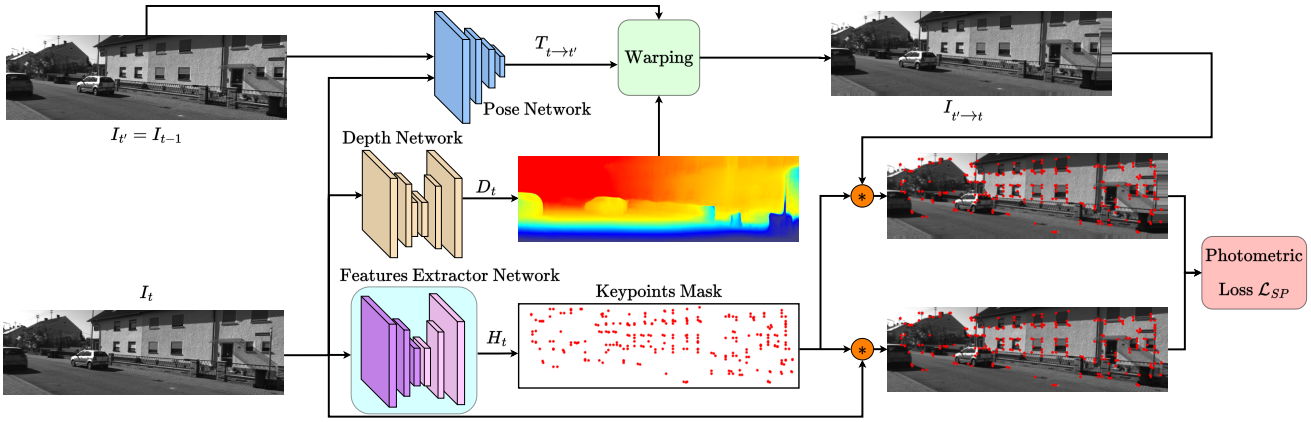

Fig. 2: **LF^{2}SLAM training pipeline.** The reprojected target image $I_{t'\to t}$ is reconstructed through the Warping block, which takes as input the pose $T_{t\to t'}$ between the target I_t and the source image $I_{t'}$, and the depth map D_t of I_t. $T_{t\to t'}$ and D_t are returned by the Pose Network and the Depth Network, respectively. The heatmap H_t, *i.e.*, a keypoints mask, is estimated by the Feature Extractor network. Then, an element-wise multiplication, depicted by the orange asterisk, of H_t with both I_t and $I_{t'\to t}$ is performed. These two masked images are finally compared for computing the Photometric Loss $\mathcal{L}_{SP}$.

To accomplish this goal, we design a specific new training procedure for the Superpoint (SP) network within the Monodepth2 (MD2) data-driven framework. Subsequently, the trained SP model is embedded into the ORBSLAM3 (OS3) system. Figure 3 shows the integration of the trained SP network within the OS3 framework by replacing the existing OS3 feature extraction block, namely ORB, with our novel SP feature extractor.

For a better understanding of the overall scheme, in the following, the description of the methodology behind our architecture is divided into two subsections. In Section III-A, the general description of the SP network and its integration within the MD2 framework are discussed. Section III-B, instead, describes the integration of the trained CNN feature extractor within the OS3 framework.

A. SLAM-specific Superpoint feature extractor

The proposed Superpoint CNN features a novel training loss function that is based on an ad-hoc 2D training mask that modifies the original MD2 loss function. The purpose of this modification is to obtain a finer selection of image keypoints that are particularly robust for VO tasks.

1) The Superpoint Network: SP is a CNN that extracts sparse features in an image. It consists of an Encoder-Decoder architecture in which only the Encoder layers have learnable parameters. SP takes a grayscale image $\mathbb{R}^{H\times W\times 1}$ as input and returns a Heatmap H. The Heatmap has the same size as the input image, *i.e.*, $\mathbb{R}^{H\times W\times 1}$, and each cell has a value $\in [0,1]$ that quantifies the probability that the corresponding pixel in the input image is a feature point. A binary mask of the same image size is then derived by comparing the values of H with a given threshold $0<\alpha<1$ and assigning 1 to the pixels that exceed the threshold and 0 to the remaining ones. When this mask is applied to the input image through element-wise multiplication, it results in the isolation of the pixels having a high probability of being keypoint. Choosing a high threshold thus implies excluding many keypoints, keeping only the more robust ones.

2) Monodepth2-Superpoint joint training: To learn an SP network that produces features suitable for VO purposes, we integrate it within the MD2 training framework. This is achieved by modifying the standard photometric loss function from [15] to include the keypoint mask H computed from the SP network.

Figure 2 shows in detail the integration of the SP network within the MD2 training framework. Specifically, the target image $I_t \in \mathbb{R}^{H\times W\times 1}$ is fed both into the SP network and the MD2 depth and pose estimation networks. SP extracts features within the target image and generates a heatmap H_t that encodes for each pixel the probability of being a keypoint. Following the standard MD2 training process, the source images $I_{t'}$ with $t' \in \{t_{+1}, t_{-1}\}$ along with the target image I_t, are used to reconstruct the warped image $I_{t'\to t}$ leveraging the estimations of the depth map of the target image D_t, and the pose between the target and the source image $T_{t\to t'}$. The existing MD2 training algorithm is indeed based on the computation of the reprojection photometric loss, which is derived by reconstructing the target image from the source one, *i.e.*, $I_{t'\to t}$, by exploiting the outputs provided by the pose and depth networks, *i.e.*, D_t and $T_{t\to t'}$.

In the present study, instead, our goal is to design a modified reprojection loss function that takes into account solely the salient keypoints extracted from the SP network. The rationale of this modification is led by the following idea: to minimize the reprojection error computed on those sparse points, the SP encoder needs to compute features that are consistent with the pose estimation task.

Building on this, our new SP-dependant loss function is defined as follows:

$$\mathcal{L}_{SP} = pe(H * I_t, H * I_{t'\to t}) \tag{1}$$

where $*$ is the element-wise matrix multiplication and pe

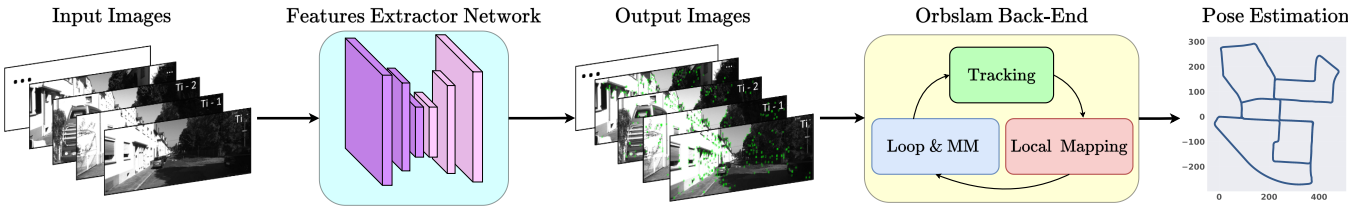

Fig. 3: **Poses estimation with LF^{2}SLAM.** Grayscale images are fed into the Features Extractor Network. The resulting keypoints are subsequently utilized by the ORBSLAM3 backend to generate precise pose estimations .

refers to the photometric error between the original and the reconstructed images. More explicitly, by using SSIM [38] and L1-norm, the expression of our SP loss is:

$$\mathcal{L}_{SP} = H * \left(\frac{\alpha}{2} \left(1 - SSIM(I_t, I_{t' \to t})\right) + (1-\alpha)||I_t - I_{t' \to t}||_1 \right) \tag{2}$$

However, the loss function in Equation (2) can be trivially minimized by setting the heatmap H to zero, *i.e.*, by not extracting keypoints. To prevent this trivial solution we design two regularization terms. The first one is the $\mathcal{L}_{sparse}$ contribution which is defined as follow:

$$\mathcal{L}_{sparse} = \left| \sum_i \sum_j H(i,j) - N_d \right| \tag{3}$$

This regularization term forces the training toward the selection of a number of feature points close to an assigned desired value N_d, preventing the $H = 0$ solution. Nonetheless, this term alone might force the SP network to extract the requested number of points regardless of their saliency.

To avoid this potential problem we introduce an additional regularization term $\mathcal{L}_{SPconst}$ imposing that the extracted keypoints should not excessively deviate from those computed by the SP model from [14]:

$$\mathcal{L}_{SPconst} = crossEntropy(d_{SP}, d_{SPconst}) \tag{4}$$

In this equation, d_{SP} is the output of the last encoder layer of the SP network that we are optimizing, while $d_{SPconst}$ is the equivalent output provided by another SP pre-trained network whose weights are kept constant. Considering Equations (3), (4), the overall loss function is defined as:

$$\mathcal{L} = \alpha \mathcal{L}_{SP} + \beta \mathcal{L}_{sparse} + \gamma \mathcal{L}_{SPconst} \tag{5}$$

After optimization, the trained SP network is integrated into the OS3 pipeline as discussed in the following Section III-B.

B. Integration of our model within ORBSLAM3

In this study, we leverage the OS3 monocular configuration, which is based on three separate threads that run in parallel: i) the Tracking thread that receives the current camera frame and estimates the pose; ii) the Local Mapping thread that performs map optimization through bundle adjustment; iii) the Loop & Map Matching (Loop & MM) thread that identifies loop closings. The Tracking thread is also responsible for feature extraction, performed by the ORB algorithm. We replace the ORB module with the trained SP feature extractor network. At inference time, as clarified in the scheme in Figure 3, we input grayscale images to the trained SP model which extracts features. These are then ranked according to their likelihood values of being a salient point, and the N ones having the highest values are retained. The resulting N features are fed into the standard OS3 pipeline to estimate the camera pose and the sparse map.

IV. Experimental Setup

A. LF^{2}SLAM Training Details

In the training phase, LF^{2}SLAM is initialized with the pretrained versions of the depth and pose MD2 network on the KITTI dataset [39], and with the pretrained SP network on the COCO dataset. The original pretrained version of SP is also used to compute the loss (4) as explained in Section III-A.2. All the networks, *i.e.*, depth, pose, and SP ones, are fine-tuned on the KITTI sequences from 00 to 08. The parameters of the loss function in Equation 5 are set to $\alpha = 1$, $\beta = 5e-3$, and $\gamma = 1$, while N_d in Equation (3) is equals to 2000. We adopt the ADAM [40] optimizer with an initial learning rate of $1e-4$ that we divide by 10 after epoch 15. Training is performed for a total of 20 epochs on a workstation equipped with an Nvidia GPU GTX Titan X with 12 GB of VRAM and requires about 36 hours. At inference time, the trained SP network takes approximately 320 MB of VRAM in the GPU.

B. Experimental Setting

The proposed LF^{2}SLAM is compared with the OS3 standard pipeline that uses the ORB feature extractor. Both algorithms share two hyperparameters that need to be set, namely the number of features (*n_feat*) and the number of pyramid levels (*n_levels*). For OS3 *n_feat* is the maximum number of extracted features, while for LF^{2}SLAM is the number of features with the highest probability. On the other hand, *n_levels* refers to the number of times an image is scaled. For each scaled image, OS3 extracts features by applying the FAST algorithm, while LF^{2}SLAM performs a forward pass. Both algorithms aggregate the extracted features once all the scaled images have been processed. The optimal hyperparameters are found by evaluating the pose estimation accuracy on a validation sequence for each dataset for both algorithms. Then, we test each algorithm with these hyperparameters on the remaining dataset sequences. In the following, we refer to each combination of (*n_feat*, *n_levels*) as (k, p), *i.e.*, keypoints, and pyramid levels. Specifically, k ranges in $[800, 2200]$ with $step = 200$, while p in

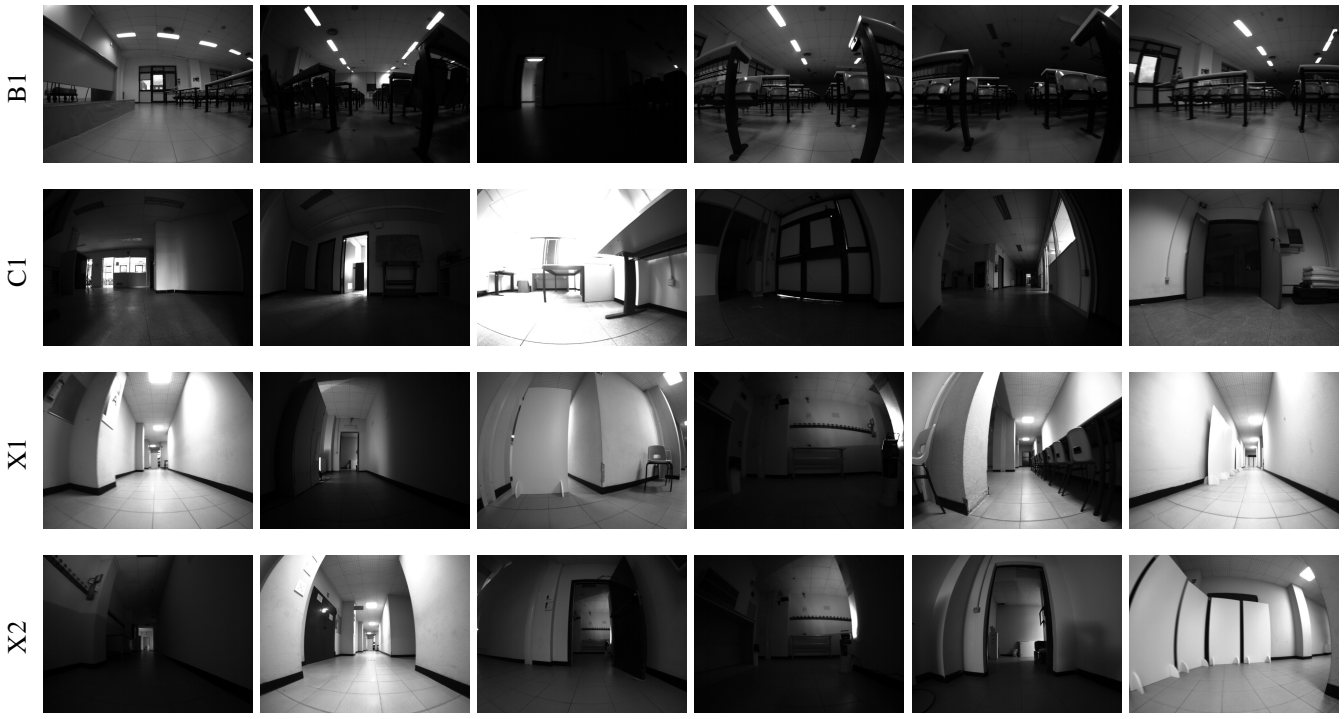

Fig. 4: **UNILAB dataset** is composed of four different challenging sequences acquired with a ground robot in low-light and poor-texture environments. Sequences B1, C1, X2, and X1 are represented from the first to the last line, respectively.

$\{1, 2, 4, 6, 8\}$, resulting in 40 total combinations. To select the best combination, we exploit three different accuracy metrics, as we explain below, evaluated on specific dataset sequences (Section IV-D).

C. Metrics

To assess the accuracy of the estimates of the two algorithms, we employ three metrics that can be computed by utilizing the EVO toolbox [41]. The toolbox provides quantitative metrics to compare the ground truth of a sequence with the estimated trajectory by a VO algorithm. In particular, we rely on the Absolute Pose Error (APE), which includes both translational and rotational components. Defining the estimated trajectory and the ground truth at timestamp i as $P_{gt,i}, P_{est,i} \in SE(3)$ respectively, the error is defined as:

$$E_i = P_{gt,i}^{-1} P_{est,i} \in SE(3) \tag{6}$$

Since we are interested in the separate analysis of translational and rotational error we decompose Equation (6) in its respective components.

1) Absolute Translation Error: The Absolute Translation Error is defined as follows:

$$ATE_{t,i} = ||trans(E_i)|| \tag{7}$$

In the experiments, we consider the Root Mean Square Error (RMSE) of the ATE. To simplify the notation we refer to the $RMSE_{ATE}$ as ATE.

2) Absolute Rotational Error: Analogous considerations are made to isolate the rotational part of Equation (6) leading to the following Absolute Rotational Error:

$$ARE_{t,i} = |angle(log_{SO(3)}(rot(E_i))|, \tag{8}$$

in which $log_{SO(3)}(\cdot)$ is the inverse of $exp_{SO(3)}(\cdot)$. In this case, as well, we take into account the RMSE and we denote $RMSE_{ARE}$ as ARE.

3) Trajectory Ratio: The aforementioned metrics implemented in the EVO toolbox are based on the matched timestamps between the estimated trajectory and the ground truth. Therefore, they do not take into account those cases where the VO algorithm loses track, as the metrics are only computed on the part of the trajectory for which estimates are provided. Thus, we introduce the trajectory ratio R_{traj} coefficient defined as:

$$R_{traj} = \frac{l(t_{est})}{l(t_{gt})}, \tag{9}$$

where t_{est} is the estimated trajectory, t_{gt} is the ground truth, and the function $l(*)$ computes the length of a trajectory. This metric ranges in $[0, +\infty[$. Specifically, $R_{traj} = 1$ means that the VO algorithm maintains its track without any loss. On the other hand, $R_{traj} \approx 0$ means that the VO algorithm failed at the very beginning of the sequence, while if $R_{traj} \gg 1$, it implies that the VO algorithm drifts. Analyzed together with ATE and ARE, the R_{traj} metric provides a more clear perspective on the accuracy of a VO algorithm.

4) Summary Metric: Since our objective is to choose the best hyperparameters on a specific sequence of a dataset, we need to define a single general metric that encapsulates the ones introduced so far. Moreover, we require a unique score to evaluate which approach achieves the best performance on the same sequence. For these reasons, we introduce the

Summary metric S defined as:

$$S = \begin{cases} \frac{ATE+ARE}{R^2_{traj}}, \; if \;\; 0 < R_{traj} < 1 \\ R^2_{traj}(ATE + ARE), \; if \;\; R_{traj} \geq 1 \end{cases} \tag{10}$$

Due to randomness factors present in the algorithm, we perform three runs on each test sequence (see Section IV-D for details) and then select the best result based on the S metric.

D. Datasets

As reference datasets, we select EuRoC [42] and the KITTI [39]: the former contains indoor sequences recorded by a Micro Aerial Vehicle (MAV), while the latter outdoor ones recorded by car. The KITTI sequences from 00 to 08 are used to train the proposed LF^{2}SLAM approach. For this dataset, the hyper-parameter tuning for both LF^{2}SLAM and OS3 is performed on sequence 10, while for the test we consider the sequence 09. On the EuRoC dataset, instead, we consider the sequence *MH_01_easy* for the hyper-parameter selection, while all the others are used for testing.

In addition to these datasets, we decide to evaluate the performance in challenging scenarios with non-ideal lighting conditions. In particular, we collect a novel dataset composed of four sequences, which we refer to as UNILAB, by using a ground robot in three different indoor areas of the University Department. Figure 4 shows some examples extracted from this dataset. In the following, the sequences are referred to as X1, X2, B1, and C1, and the first, *i.e.*, X1, is used to select the best hyperparameters.

V. Results and Discussions

As a first analysis, we consider the sensitivity of each method to hyperparameters variations. For this purpose, in Table I we report the best setting for the hyperparameters (k, p) for each algorithm evaluated in terms of the summary metric S with the associated mean and the standard deviation computed over the 40 configurations. The analysis of the sensitivity highlights the remarkable robustness of our model to hyperparameters variations. LF^{2}SLAM outperforms OS3 in both the KITTI/10 and the UNILAB/X1 sequences while demonstrating comparable performance in EuRoC/MH_01. These results show that when deploying LF^{2}SLAM the selection of the hyper-parameters is less critical, facilitating its practical use.

The comparison results on the test sequences are instead provided in Tables II and III.

In Table II the experiments on the sequences are performed with the best hyperparameter combination found in each dataset. The analysis of the results highlights that LF^{2}SLAM outperforms OS3 in the KITTI/09 sequence and the UNILAB more challenging sequences. Conversely, our approach shows comparable performances on the EuRoC dataset. While the LF^{2}SLAM better performance on the KITTI dataset is expected since our feature extractor is trained with images from sequences of the same scenario, more interesting considerations can be drawn by observing the results on EuRoC. Despite our approach being tested on a scenario different from that used for training, it exhibits comparable performance to OS3, which proves its generalization capabilities. The quantitative performance on the UNILAB dataset, instead, demonstrates that the proposed LF^{2}SLAM achieves considerably higher scores compared to the OS3 baseline. These results are remarkable considering the more challenging sequences of this dataset, characterized by sudden light changes and texture-less scenarios (Figure 4). While our approach exploits the convolutional feature extraction to achieve robustness against these conditions, OS3 frequently loses track due to the poor extracted keypoints.

TABLE I: HYPERPARAMETERS TUNING

Dataset	Sequence	Method	(k, p)	ATE [m]	ARE [deg]	R_{traj}	S
KITTI	10	ORBSLAM3	$(2200, 4)$	5.909	1.216	1.035	7.632
		LF^{2}SLAM	$(2200, 4)$	6.209	1.054	1.029	7.690
EuRoC	MH_01	ORBSLAM3	$(2200, 8)$	0.041	1.828	1.016	1.930
		LF^{2}SLAM	$(2200, 2)$	0.039	2.003	1.008	2.074
UNILAB	X1	ORBSLAM3	$(1000, 6)$	0.549	1.985	0.928	2.942
		LF^{2}SLAM	$(1800, 4)$	0.160	1.347	0.903	1.848

	$[\mu \pm \sigma]$ of S		
Method	KITTI/10	EuRoC/MH_01	UNILAB/X1
ORBSLAM3	$2.767E4 \pm 7.467E4$	$\mathbf{2.110 \pm 0.09}$	85.54 ± 109.92
LF^{2}SLAM	$\mathbf{1.180E2 \pm 3.802E2}$	2.290 ± 0.30	$\mathbf{15.55 \pm 51.69}$

TABLE II: Summary results with best (k, p) tuned on KITTI/10, EuRoC/MH_01, and UNILAB/X1, respectively.

Dataset	Sequence	ORBSLAM3	LF^{2}SLAM
KITTI	09	43.32	**9.62**
EuRoC	MH_02	**1.845**	1.944
	MH_03	**1.729**	1.836
	MH_04	**1.523**	2.369
	MH_05	**1.942**	1.983
	V1_01	**5.923**	7.767
	V1_02	**1.429**	1.904
	V1_03	**2.241**	4.005
	V2_01	13.77	**2.702**
	V2_02	**1.284**	1.702
	V2_03	**3.730**	13.02
UNILAB	B1	5.651	**2.627**
	C1	7.356	**3.989**
	X2	58.18	**4.151**

In Table III, instead, the results with the best (k, p) pair selected on the sequence KITTI/10 are shown. This allows us also to assess the robustness of each method when deployed in a different context without any further refinement of the hyperparameters. Some qualitative results are also shown in Figure 5. As in the previous case, LF^{2}SLAM shows comparable performance to OS3 on the EuRoC sequences, outperforming the latter in three sequences. Conversely, on the UNILAB dataset, our approach provides always better scores, particularly on sequences UNILAB/C1 and UNILAB/X1 (see Figures 5(c) and 5(d)). These findings

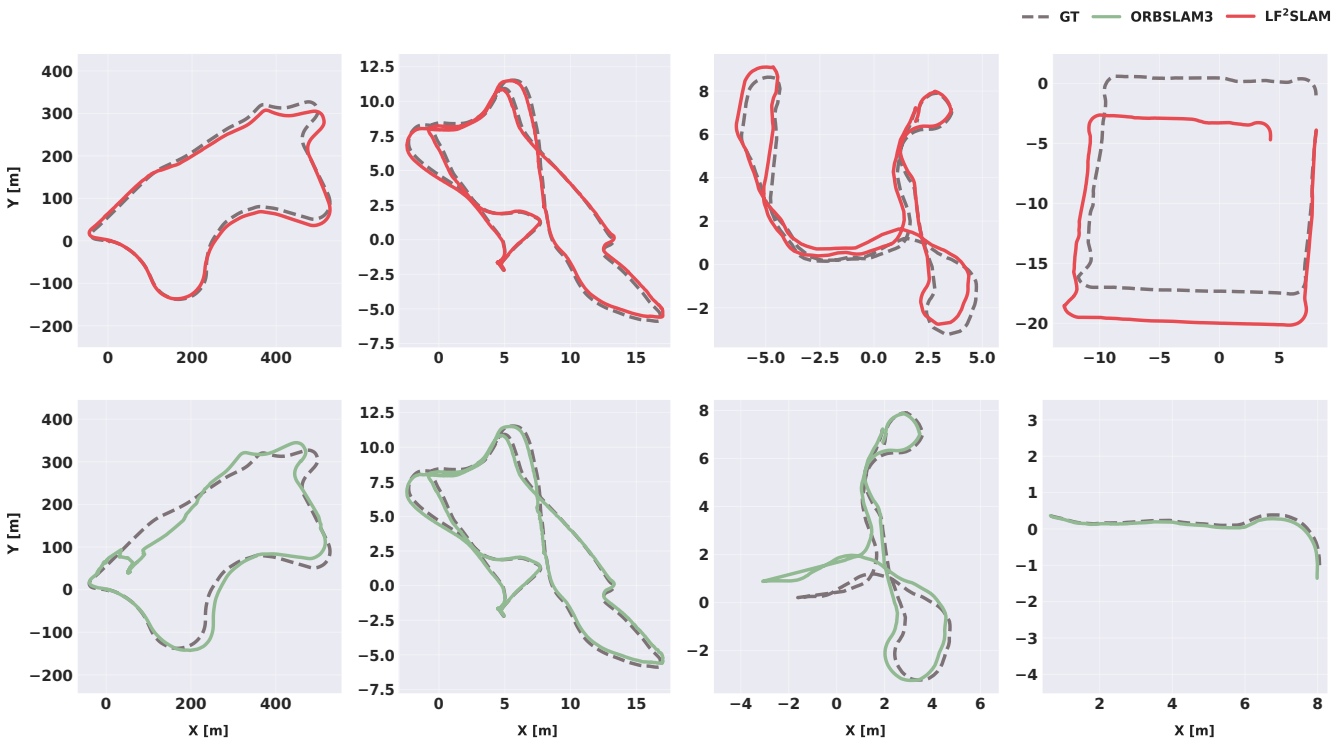

Fig. 5: **Samples of LF^{2}SLAM and ORBSLAM3 output trajectories with KITTI/10 best $(\mathbf{k}, \mathbf{p})$.** (a) KITTI/09: OS3 (green) partially drifts, while LF^{2}SLAM (red) achieves better performance. (b) EuRoC/MH_05: both algorithms achieve excellent performances. (c) UNILAB/C1: OS3 loses its track (the left portion of ground truth is not tracked). (d) UNILAB/X2: OS3 almost completely loses its tracking while LF^{2}SLAM results in a much better trajectory estimation.

confirm the generalization capabilities of our approach and its robustness in challenging scenarios characterized by non-ideal conditions compared to OS3.

TABLE III: Summary results with best (k, p) tuned on KITTI/10.

Dataset	Sequence	ORBSLAM3	LF^{2}SLAM
EuRoC	MH_02	**1.797**	1.822
	MH_03	2.120	**1.780**
	MH_04	**1.555**	2.013
	MH_05	**1.892**	2.632
	V1_01	**5.918**	6.475
	V1_02	**1.418**	1.739
	V1_03	**1.965**	4.713
	V2_01	7.256	**2.318**
	V2_02	**1.169**	2.030
	V2_03	6.195	**3.994**
UNILAB	B1	4.813	**4.719**
	C1	8.031	**1.814**
	X2	136.8	**4.984**

Time Analysis: As our approach replaces the ORB extraction module of OS3 with the proposed LF^{2}SLAM features extractor network, the computational time comparison is performed on this component. The time profiling performed on a reference sequence from the EuRoC dataset indicates an average processing time of 4ms for keypoints extraction on a full-size image (752x480) with ORBSLAM (CPU Intel Core i7-9800X 3.80 GHz), and 17ms for the same processing using our LF^{2}SLAM features extractor network (GPU NVidia RTX 2080 Ti), showing a slightly higher computational cost of the proposed approach.

VI. Conclusions and Future Direction

In this paper, we presented a hybrid framework for VSLAM, namely LF^{2}SLAM. The proposed approach exploits a novel training procedure that allows to extract features that are suitable for image-based pose estimation tasks. Once the DL-based feature extract is trained, we integrated it into a state-of-the-art VO/SLAM pipeline, *i.e.*, OS3. The experimental comparison demonstrated that our solution achieves remarkable generalization capabilities in different environments (indoor, outdoor) and with diverse platforms (aerial, ground, and automotive), and robustness with respect to hyperparameters selection. Moreover, our approach consistently outperforms the state-of-the-art SLAM OS3 baseline in challenging scenarios with non-ideal image conditions.

Future work will explore a novel training process to estimate binary descriptors jointly with the keypoint detection to also replace the BRIEF algorithm in ORBSLAM3. Furthermore, novel optimization procedures to manage ex-

treme conditions, such as almost dark environments, will be explored.

REFERENCES

[1] K. Yousif, A. Bab-Hadiashar, and R. Hoseinnezhad, "An overview to visual odometry and visual slam: Applications to mobile robotics," *Intelligent Industrial Systems*, vol. 1, no. 4, pp. 289–311, 2015.

[2] L. R. Agostinho, N. M. Ricardo, M. I. Pereira, A. Hiolle, and A. M. Pinto, "A practical survey on visual odometry for autonomous driving in challenging scenarios and conditions," *IEEE Access*, vol. 10, pp. 72 182–72 205, 2022.

[3] N. Brasch, A. Bozic, J. Lallemand, and F. Tombari, "Semantic monocular slam for highly dynamic environments," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 393–400.

[4] C. Campos, R. Elvira, J. J. G. Rodríguez, J. M. Montiel, and J. D. Tardós, "Orb-slam3: An accurate open-source library for visual, visual–inertial, and multimap slam," *IEEE Transactions on Robotics*, vol. 37, no. 6, pp. 1874–1890, 2021.

[5] J. Engel, V. Koltun, and D. Cremers, "Direct sparse odometry," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 3, pp. 611–625, 2017.

[6] N. Yang, R. Wang, and D. Cremers, "Feature-based or direct: An evaluation of monocular visual odometry," *arXiv preprint arXiv:1705.04300*, pp. 1–12, 2017.

[7] R. Li, S. Wang, and D. Gu, "Ongoing evolution of visual slam from geometry to deep learning: Challenges and opportunities," *Cognitive Computation*, vol. 10, pp. 875–889, 2018.

[8] E. Parisotto, D. Singh Chaplot, J. Zhang, and R. Salakhutdinov, "Global pose estimation with an attention-based recurrent network," in *Proceedings of the ieee conference on computer vision and pattern recognition workshops*, 2018, pp. 237–246.

[9] S. Milz, G. Arbeiter, C. Witt, B. Abdallah, and S. Yogamani, "Visual slam for automated driving: Exploring the applications of deep learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2018, pp. 247–257.

[10] G. Costante, M. Mancini, P. Valigi, and T. A. Ciarfuglia, "Exploring representation learning with cnns for frame-to-frame ego-motion estimation," *IEEE robotics and automation letters*, vol. 1, no. 1, pp. 18–25, 2015.

[11] M. Ahmadi, A. A. Naeini, M. M. Sheikholeslami, Z. Arjmandi, Y. Zhang, and G. Sohn, "Hdpv-slam: Hybrid depth-augmented panoramic visual slam for mobile mapping system with tilted lidar and panoramic visual camera," in *2023 IEEE 19th International Conference on Automation Science and Engineering (CASE)*. IEEE, 2023, pp. 1–8.

[12] L. Sun, W. Yin, E. Xie, Z. Li, C. Sun, and C. Shen, "Improving monocular visual odometry using learned depth," *IEEE Transactions on Robotics*, vol. 38, no. 5, pp. 3173–3186, 2022.

[13] H. M. S. Bruno and E. L. Colombini, "Lift-slam: A deep-learning feature-based monocular visual slam method," *Neurocomputing*, vol. 455, pp. 97–110, 2021.

[14] D. DeTone, T. Malisiewicz, and A. Rabinovich, "Superpoint: Self-supervised interest point detection and description," in *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, 2018, pp. 224–236.

[15] C. Godard, O. Mac Aodha, M. Firman, and G. J. Brostow, "Digging into self-supervised monocular depth prediction," October 2019.

[16] W. Chen, G. Shang, A. Ji, C. Zhou, X. Wang, C. Xu, Z. Li, and K. Hu, "An overview on visual slam: From tradition to semantic," *Remote Sensing*, vol. 14, no. 13, p. 3010, 2022.

[17] M. Legittimo, S. Felicioni, F. Bagni, A. Tagliavini, A. Dionigi, F. Gatti, M. Verucchi, G. Costante, and M. Bertogna, "A benchmark analysis of data-driven and geometric approaches for robot ego-motion estimation," *Journal of Field Robotics*, vol. 40, no. 3, pp. 626–654, 2023. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/rob.22151

[18] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, vol. 60, no. 2, pp. 91–110, 2004.

[19] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection," in *European conference on computer vision*. Springer, 2006, pp. 430–443.

[20] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "Orb: An efficient alternative to sift or surf," in *2011 International conference on computer vision*. Ieee, 2011, pp. 2564–2571.

[21] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, "Orb-slam: a versatile and accurate monocular slam system," *IEEE transactions on robotics*, vol. 31, no. 5, pp. 1147–1163, 2015.

[22] R. Mur-Artal and J. D. Tardós, "ORB-SLAM2: an open-source SLAM system for monocular, stereo and RGB-D cameras," *IEEE Transactions on Robotics*, vol. 33, no. 5, pp. 1255–1262, 2017.

[23] L. Jing and Y. Tian, "Self-supervised visual feature learning with deep neural networks: A survey," *IEEE transactions on pattern analysis and machine intelligence*, vol. 43, no. 11, pp. 4037–4058, 2020.

[24] K. Wang, S. Ma, J. Chen, F. Ren, and J. Lu, "Approaches challenges and applications for deep visual odometry toward to complicated and emerging areas," *IEEE Transactions on Cognitive and Developmental Systems*, 2020.

[25] Y. Verdie, K. Yi, P. Fua, and V. Lepetit, "Tilde: A temporally invariant learned detector," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 5279–5288.

[26] K. Lenc and A. Vedaldi, "Learning covariant feature detectors," in *European conference on computer vision*. Springer, 2016, pp. 100–117.

[27] X. Zhang, F. X. Yu, S. Karaman, and S.-F. Chang, "Learning discriminative and transformation covariant local feature detectors," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 6818–6826.

[28] A. Barroso-Laguna, E. Riba, D. Ponsa, and K. Mikolajczyk, "Key.net: Keypoint detection by handcrafted and learned cnn filters," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 5836–5844.

[29] K. M. Yi, E. Trulls, V. Lepetit, and P. Fua, "Lift: Learned invariant feature transform," in *European conference on computer vision*. Springer, 2016, pp. 467–483.

[30] Y. Ono, E. Trulls, P. Fua, and K. M. Yi, "Lf-net: Learning local features from images," *Advances in neural information processing systems*, vol. 31, 2018.

[31] D. Li, X. Shi, Q. Long, S. Liu, W. Yang, F. Wang, Q. Wei, and F. Qiao, "Dxslam: A robust and efficient visual slam system with deep features," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 4958–4965.

[32] Z. Teed and J. Deng, "Droid-slam: Deep visual slam for monocular, stereo, and rgb-d cameras," *Advances in Neural Information Processing Systems*, vol. 34, pp. 16 558–16 569, 2021.

[33] J. Tang, J. Folkesson, and P. Jensfelt, "Geometric correspondence network for camera motion estimation," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 1010–1017, 2018.

[34] J. Tang, L. Ericson, J. Folkesson, and P. Jensfelt, "Gcnv2: Efficient correspondence prediction for real-time slam," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3505–3512, 2019.

[35] T. Derr, Y. Ma, and J. Tang, "Signed graph convolutional networks," in *2018 IEEE International Conference on Data Mining (ICDM)*. IEEE, 2018, pp. 929–934.

[36] C. Deng, K. Qiu, R. Xiong, and C. Zhou, "Comparative study of deep learning based features in slam," in *2019 4th Asia-Pacific Conference on Intelligent Robot Systems (ACIRS)*. IEEE, 2019, pp. 250–254.

[37] G. Mollica, M. Legittimo, A. Dionigi, G. Costante, and P. Valigi, "Integrating sparse learning-based feature detectors into simultaneous localization and mapping—a benchmark study," *Sensors*, vol. 23, no. 4, p. 2286, 2023.

[38] Z. Wang, A. Bovik, H. Sheikh, and E. Simoncelli, "Image quality assessment: From error visibility to structural similarity," *Image Processing, IEEE Transactions on*, vol. 13, pp. 600 – 612, 05 2004.

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

[40] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[41] M. Grupp, "evo: Python package for the evaluation of odometry and slam," 2017.

[42] M. Burri, J. Nikolic, P. Gohl, T. Schneider, J. Rehder, S. Omari, M. W. Achtelik, and R. Siegwart, "The euroc micro aerial vehicle datasets," *The International Journal of Robotics Research*, 2016. [Online]. Available: http://ijr.sagepub.com/content/early/2016/01/21/0278364915620033.abstract