LIFT-SLAM: a deep-learning feature-based monocular visual SLAM method*

Hudson M. S. Bruno[1] and Esther L. Colombini[1][0000−0003−0467−3133]

Laboratory of Robotics and Cognitive Systems, Institute of Computing
University of Campinas, Campinas, São Paulo, Brazil
hudson.bruno@ic.unicamp.br,esther@ic.unicamp.br
http://larocs.ic.unicamp.br

Abstract. The Simultaneous Localization and Mapping (SLAM) problem addresses the possibility of a robot to localize itself in an unknown environment and simultaneously build a consistent map of this environment. Recently, cameras have been successfully used to get the environment's features to perform SLAM, which is referred to as visual SLAM (VSLAM). However, classical VSLAM algorithms can be easily induced to fail when the robot motion or the environment is too challenging. Although new approaches based on Deep Neural Networks (DNNs) have achieved promising results in VSLAM, they still are unable to outperform traditional methods. To leverage the robustness of deep learning to enhance traditional VSLAM systems, we propose to combine the potential of deep learning-based feature descriptors with the traditional geometry-based VSLAM, building a new VSLAM system called LIFT-SLAM. Experiments conducted on KITTI and Euroc datasets show that deep learning can be used to improve the performance of traditional VSLAM systems, as the proposed approach was able to achieve results comparable to the state-of-the-art while being robust to sensorial noise. We enhance the proposed VSLAM pipeline by avoiding parameter tuning for specific datasets with an adaptive approach while evaluating how transfer learning can affect the quality of the features extracted.

Keywords: Visual SLAM · Hybrid Mehtods · Deep Learning

M.Sc. Disseration submitted to CTDR 2020 - Defense date: 4/5/2020
Supervisor: Prof. Dr. Esther Luna Colombini

1 Introduction

The ability to know its localization in an environment is an essential task for mobile robots, and it has been a subject of research in robotics for decades. To correctly localize itself, the robot must know its pose (position and orientation) in the environment. In the last decades, the advances in hardware technologies,

* This work was supported by the Brazilian National Council for Scientific and Technological Development (CNPq) and by the company Quinto Andar.

such as embedded GPUs, allowed significant advances on mobile robots pose estimation through camera-based methodologies of odometry and SLAM, which are called Visual Odometry (VO) and Visual Simultaneous Localization and Mapping (VSLAM). Much work has been done to develop accurate and robust VO and VSLAM systems. However, traditional approaches still depend on significant engineering effort on a classic pipeline: Initialization, feature detection, feature matching, outlier rejection, motion estimation, optimization, and relocalization. Furthermore, the traditional approaches tend to fail in challenging environments (inadequate illumination, featureless areas, etc.), when the camera is moving at high speed or if the camera suffers some distortions (rolling shutter effect, unfavorable exposure conditions, etc.). Moreover, if the camera is monocular, these systems have scale uncertainty.

Recent applications of deep learning-based methods in VO and VSLAM have achieved promising results, bringing robustness to the situations as mentioned earlier. Many works have proposed using Deep Neural Networks (DNNs) to estimate camera motion with an end-to-end system. These systems can replace the entire traditional VO pipeline, which depends on significant engineering effort to develop and tune [1–3]. However, these methods are not able to outperform traditional methods yet. Thus, some new works propose to replace only some modules of the VSLAM traditional pipeline with DNNs, creating hybrid methods [4–6]. This approach can leverage the robustness of deep learning to enhance traditional VSLAM systems.

In this work, we propose to use the Learned Invariant Feature Transform (LIFT) [7] to extract features from images and use these features in a traditional VSLAM pipeline based on ORB-SLAM [8] for monocular camera applications. Hence, we explore the potential of deep neural networks to improve the performance of conventional VSLAM systems. Our main contributions are: (1) a novel hybrid VSLAM algorithm based on the LIFT network, (2) an evaluation of how transfer learning and fine-tuning can affect the quality of a Hybrid VSLAM system, (3) we extend the proposed system with an adaptive approach that can enhance its performance while avoiding fine-tuning of parameters that are usually dependable on the dataset, (4) we conduct experiments on public KITTI [9] and Euroc [10] datasets and present a set of experiments to confirm the robustness of algorithms based on learned features under camera distortions.

2 Related Work

Feature-based approaches. The Parallel Tracking and Mapping (PTAM) [11] algorithm was proposed to solve MonoSLAM problems. To reduce computational costs, the authors split tracking and mapping into two separate tasks, processed in parallel. That way, the tracking estimates camera motion in real-time, and the mapping estimates accurate 3D positions of feature points with a computational cost [12]. It is the first real-time method that was able to incorporate BA. They have also created an automatic initialization with a 5-point algorithm. The main ideas of PTAM were used in ORB-SLAM [8], a feature-based monocular VSLAM

system with three threads: Tracking, Local Mapping, and Loop Closing. It relies on ORB features and uses a place recognition system based on Bag-of-Words (BoW). The mapping step adopts graph representations, allowing the system to perform local and global pose-graph optimization. Later, ORB-SLAM was extended to stereo and RGB-D cameras [13].

End-to-end deep learning-based approaches. One of the first end-to-end approaches is DeepVO [3]. In DeepVO, a Recurrent Neural Network (RNN) estimates the camera pose from features learned by a Convolutional Neural Network (CNN). The CNN architecture proposed is based on an architecture used to compute optical flow from a sequence of images called Flownet [14]. Then two stacked Long-Short Term Memory (LSTM) layers are applied to estimate temporal changes from the features predicted by a CNN. Another end-to-end approach, based on unsupervised learning called UnDeepVO, is presented in [1]. The network relies on stereo image pairs to recover the scale during training while using consecutive monocular images for testing. Moreover, the loss function defined for training the networks uses spatial and temporal dense information. The system successfully estimates the pose of a monocular camera and the depth of its view. In [2], an end-to-end system that uses a similar architecture to DeepVO is proposed. However, instead of employing LSTMs, attention phase is included.

Hybrid approaches. Hybrid approaches replace some modules of the traditional VSLAM pipeline. In [6], the authors propose a monocular system called Neural Bundler. It is an unsupervised DNN that estimates motion. Then, it constructs a conventional pose graph, enabling an efficient loop closing procedure based on the pose graph's optimization. A recent hybrid approach called SuperGlue [15] proposed a graph neural network with an attention mechanism to perform the matching between two sets of local features. They use the DNN between feature extraction and pose estimation, a learnable "middle-end," as it lies between the front-end and back-end of a traditional VSLAM system. Recently, some works proposed using learned features to replace local features (ORB, SIFT, etc.) of VSLAM systems. In DF-SLAM [5], the TFeat network [16] is used to create descriptors for features extracted from stereo images with the FAST corner detector applied over ORB-SLAM2 's pipeline[13].

3 LIFT-SLAM

Our proposed method is a deep-learning feature-based monocular VSLAM system called LIFT-SLAM. It reconstructs sparse maps that are graph-based and keyframe-based, which allows us to perform bundle adjustment to optimize the estimated poses of the camera. We use the DNN called LIFT to extract features that are used in a pipeline based on ORB-SLAM [8]. The Learned Invariant Feature Transform (LIFT) is a DNN proposed by Yi et al. [7] that implements local feature detection, orientation estimation, and description in a supervised end-to-end approach. The network architecture comprises three main modules based on CNNs: Detector, Orientation Estimator, and Descriptor. The algorithm works with patches of images. After giving a patch as input, the detector network

provides a score map of this patch. A soft argmax operation [17] is performed over this score map to return the potential feature point location. After this, it performs a crop operation centered on the feature location, used as input to the orientation estimator. The orientation estimator module predicts an orientation to the patch. Thus, a rotation is applied in the patch according to the estimated orientation. Lastly, the descriptor network computes a feature vector from the rotated patch, that is the output.

Originally, LIFT was trained with photo-tourism image sets. They used a Structure from Motion (SfM) algorithm called VisualSFM [18] to reconstruct the scenes from the image sets with SIFT features. Photo-tourism data contains different geometrical aspects when compared to a typical VO dataset. Usually, in VO datasets, the images are sequential, captured with the same camera that progressively changes its position and orientation. On the other hand, the photo-tourism images capture views of the same scene from different perspectives. To address this aspect, we perform a transfer learning in the LIFT network to generate a version of the LIFT that is fine-tuned with VO datasets' features.

3.1 LIFT-SLAM Pipeline

As aforementioned, our pipeline is based on ORB-SLAM's pipeline [8]. Figure 1 shows an overview of our pipeline that is described next.

Tracking. In tracking, for each frame, we extract LIFT keypoints and descriptors. We use these features in all feature matching operations needed in initialization, tracking, mapping, and place recognition. Then, as in ORB-SLAM, the camera pose is predicted with a constant velocity model. Later, we optimize the camera pose by searching for more map point correspondences in the current frame by projecting the local map 3D points into the image. Lastly, the tracking step decides if the current frame should be a keyframe.

Mapping. For each new keyframe, the mapping step is performed. First, it inserts the keyframe into the covisibility graph as a new node, and its edges are computed based on the shared map points with other keyframes. Furthermore, new map points are created by triangulating LIFT features from keyframes connected in the covisibility graph. A local bundle adjustment is responsible for optimizing the covisibility graph. It is applied to all keyframes connected to the current keyframe in the covisibility graph (including the current keyframe) and all map points seen by those keyframes. Finally, in keyframes culling, we discard keyframes that are redundant to improve the covisibility graph's size.

Relocalization and loop closure. To perform place recognition, we have created a visual vocabulary in an offline step with the DBoW2 library[1] [19]. The dictionary was created with LIFT descriptors of approximately 12,000 images collected from outdoors and indoor sequences from the TUM-mono VO dataset [20]. In this way, we can generate a vocabulary that provides good results in both environments. The built vocabulary has six levels and 10 clusters per level. Thus we get 10^6 visual words, as suggested in [21]. If the tracking is lost, we query the

[1] https://github.com/dorian3d/DBoW2

Bag of Words (BoW) of the current frame into the database to find keyframe candidates for global relocalization. The loop closing task runs in a separate thread. It gets the last keyframe processed by the local mapping and tries to detect if it closes a loop. After converting the keyframes to BoW, a similarity score between the current keyframe and its neighbors' covisibility graph is computed. The similarity between two BoW is given by the L2-score, as defined in [22]. The loop candidates are accepted if there are at least three candidates detected in the same covisibility graph. After finding the loop candidates, it computes a rigid-body transformation from the candidate keyframe to the loop keyframe. This transformation, the similarity one, informs about the drift accumulated in the trajectory, and it also works as a geometrical validation of the loop. If a similarity transformation is successfully found, we proceed to correct the loop.

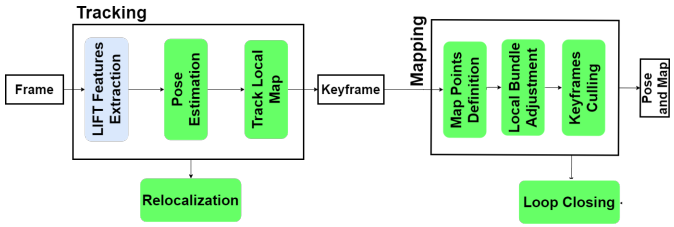


Fig. 1: An overview of LIFT-SLAM pipeline. Tracking and mapping are sequential tasks, relocalization is called when the camera pose tracking is lost, and loop closing is a task running in parallel over the keyframes processed by mapping.

3.2 Versions of LIFT-SLAM

To explore the potential of our approach and to find changes that might lead to an improvement in general results, we developed some different versions of LIFT-SLAM. The next sections describe the decision process to create these versions and how we developed them.

Fine-tuned LIFT-SLAM In this version of LIFT-SLAM, we use these fine-tuned models to perform feature detection and description. To refine the LIFT network, we had to collect the ground-truth data. As proposed in LIFT's paper [7], we generate the ground-truth with SIFT keypoints collected with VisualSFM [18]. We created two sets of ground-truth data. The first one comprises images from sequences 00, 06, 09, and 10 (8434 images) of the KITTI dataset, whereas the second contains images from the sequences MH_04, V1_03, and V2_03 (6104 images) of the EuRoC dataset. After collecting the datasets, we train the network in two versions, one for each dataset. We used the TensorFlow version of LIFT provided by the authors in their *github*[2].

[2] https://github.com/cvlab-epfl/tf-lift

Adaptive LIFT-SLAM A wrong data association in feature matching might affect the quality of the motion estimation. Therefore, to select the best features, a threshold is applied right after feature matching. In this way, the matches with greater distance than this threshold are discarded. On the other hand, if the threshold value is too small, we might reject good matches and loose track of the camera pose in challenging environments. Usually, two thresholds are used to mitigate this problem: the higher threshold (TH_{HIGH}) and the lower threshold (TH_{LOW}). We use TH_{LOW} when we need to be more restrictive about the quality of the matches, as in relocalization or map point triangulation. However, while performing our experiments, we found out that for different datasets the best values for these thresholds changes. Therefore, traditional methods usually fine-tune these thresholds every time there is a change in the dataset. This is not desirable since, in real-world applications, it is not possible to deduce these thresholds' values. Hence, we propose an adaptive method that decides the threshold values online.

To do so, after estimating the pose, we search map correspondences by projecting the map points from the last frame into the current frame. If the number of outliers approaches the number of map points, the number of matches gets too small and, consequently, the tracking is lost. We use this to create our adaptive method. It changes the thresholds values based on the difference between the number of map points and the number of outliers. Therefore, if this difference decrease, we increase the thresholds values.

4 Experiments

4.1 Datasets

Two datasets were chosen to evaluate our algorithms: KITTI [9] and Euroc MAV [10]. The KITTI dataset is a collection of images recorded from a moving car, and Euroc MAV is a set of images collected by Micro Aerial Vehicles indoors. Therefore, the datasets have different environments (e.g., outdoor/indoor, size, illumination, dynamic/static, etc.), and camera motion (e.g., acceleration, velocities, DoF, etc.), which allow us to evaluate the robustness of the algorithms to different situations. Moreover, in LIFT-SLAM fine-tuned approaches, using different datasets allowed us to validate the network's improvement for VO problems in general, without a network bias for a dataset.

4.2 Trajectory Evaluation

We generate a quantitative and qualitative comparison between the estimated trajectories and the ground-truth data for each sequence of the datasets. The quantitative evaluation in KITTI sequences are based on Relative Pose Error (RPE) of translation and rotation, as described in [23], and Absolute Trajectory Error (ATE), detailed in [20]. Due to the stochastic nature of the algorithms, all of the quantitative metrics are an average of 5 executions. The estimates on Euroc sequences were evaluated only by ATE.

ORB-SLAM's results were computed by our executions since, in ORB-SLAM's paper, an evaluation with RPE is not presented and does not provide results in the Euroc dataset. Furthermore, we present qualitative comparisons showing a 2-D plot of the trajectories. Moreover, for LIFT-SLAM versions that are not adaptive we set the matching thresholds values to $TH_{LOW} = 1$ and $TH_{HIGH} = 2$ for Euroc sequences and $TH_{LOW} = 2$ and $TH_{HIGH} = 3$ for KITTI sequences, as these were the best thresholds we found during our experiments.

The quantitative comparison of all algorithms in the KITTI dataset (Table 1) shows that, in general, LIFT-SLAM systems presented a better performance than ORB-SLAM, especially in smaller sequences, such as 03 and 04. Furthermore, we can notice that the proposed versions of LIFT-SLAM achieved a better performance than LIFT-SLAM in most of the sequences. The algorithm performance improved even when we used the Euroc dataset to fine-tune the LIFT network confirming that the network learned important features from VO datasets.

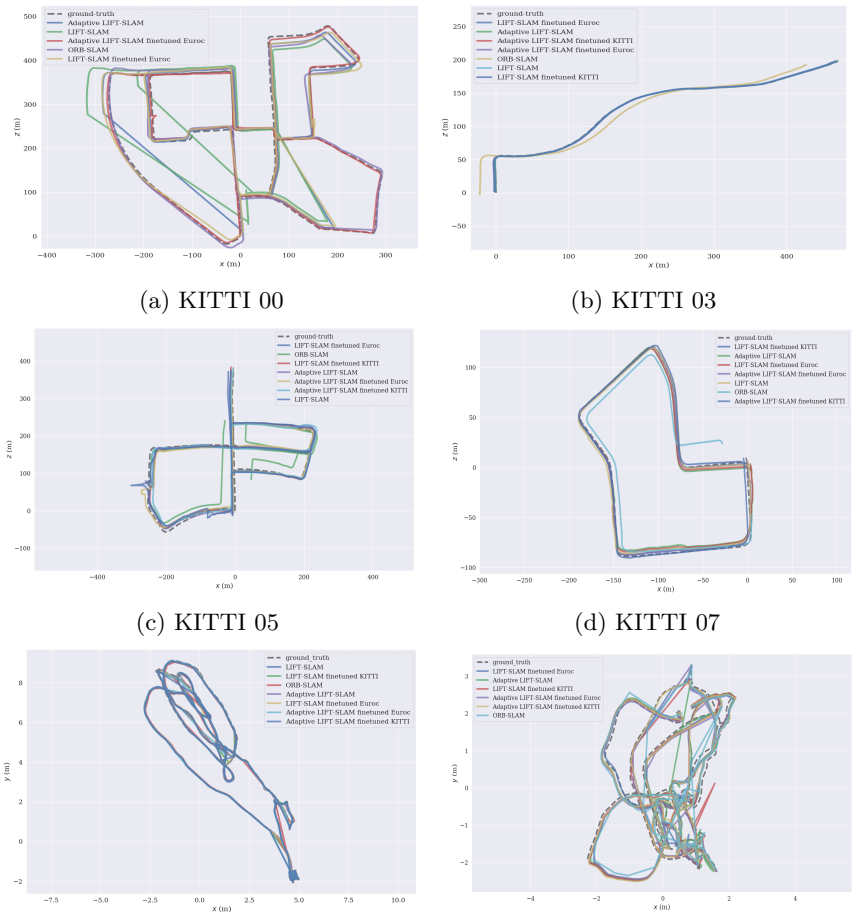
Table 1: Quantitative comparison of ORB-SLAM and LIFT-SLAM versions in the KITTI dataset. "X" are sequences unavailable due to tracking failure, and "-" sequences we did not execute the algorithm to avoid biased results.

Algorithm	Metric	00	01	02	03	04	05	06	07	08	09	10
ORB-SLAM	ATE (m)	11.54	X	X	15.13	4.29	**7.74**	20.26	13.47	**39.51**	**49.67**	19.94
	RPE_{trans} (%)	4.46	X	X	9.75	3.71	**3.35**	8.11	7.43	**12.16**	26.51	8.65
	RPE_{rot} (deg/m)	3.28	X	X	2.78	2.15	3.57	2.88	3.58	3.05	11.13	3.62
LIFT-SLAM	ATE (m)	18.77	X	X	1.10	0.40	8.09	18.47	4.03	80.97	59.88	31.84
	RPE_{trans} (%)	6.71	X	X	0.87	**2.10**	4.46	7.76	2.51	27.63	20.65	10.08
	RPE_{rot} (deg/m)	**2.20**	X	X	**0.34**	0.65	2.58	2.49	3.60	2.10	2.12	2.25
LIFT-SLAM fine-tuned with KITTI	ATE (m)	-	X	**29.83**	1.91	**0.36**	12.47	-	**2.54**	188.51	-	-
	RPE_{trans} (%)	-	X	8.80	1.32	2.16	5.02	-	**1.80**	48.90	-	-
	RPE_{rot} (deg/m)	-	X	**2.11**	**0.34**	0.52	2.43	-	**2.67**	2.11	-	-
LIFT-SLAM fine-tuned with Euroc	ATE (m)	9.84	X	34.23	0.97	0.42	11.50	**16.58**	3.98	82.61	54.91	30.34
	RPE_{trans} (%)	3.49	X	9.84	0.86	2.22	5.35	**7.05**	2.60	28.99	**19.16**	9.81
	RPE_{rot} (deg/m)	2.63	X	2.10	0.46	0.50	**1.91**	**2.36**	3.64	**1.95**	**2.08**	2.20
Adaptive LIFT-SLAM	ATE (m)	13.70	X	40.33	**0.84**	0.47	10.85	17.83	4.09	81.69	57.74	**10.51**
	RPE_{trans} (%)	**2.64**	X	11.54	**0.78**	2.22	5.49	7.50	2.67	28.49	19.28	4.96
	RPE_{rot} (deg/m)	4.95	X	2.22	0.38	0.60	2.97	2.42	3.42	2.05	2.17	**1.57**
Adaptive LIFT-SLAM fine-tuned with KITTI	ATE (m)	-	X	48.09	1.91	0.42	10.35	-	4.10	185.15	-	-
	RPE_{trans} (%)	-	X	9.57	1.29	2.11	4.64	-	2.64	47.20	-	-
	RPE_{rot} (deg/m)	-	X	2.43	**0.34**	0.57	2.93	-	3.51	2.00	-	-
Adaptive LIFT-SLAM fine-tuned with Euroc	ATE (m)	**8.06**	X	40.04	2.23	0.51	13.55	30.38	3.63	184.43	59.62	29.87
	RPE_{trans} (%)	3.18	X	**8.73**	1.46	2.22	6.09	12.24	2.42	47.10	19.91	9.72
	RPE_{rot} (deg/m)	2.99	X	2.49	**0.34**	**0.48**	3.11	2.91	4.02	2.02	2.14	2.24

Figure 2(a-d) shows the qualitative comparison between the algorithms in the KITTI dataset (only some sequences are presented here due to paper length restrictions). In sequence 00 (Fig. 2a) most of the algorithms could not track the entire trajectory, except for Adaptive LIFT-SLAM fine-tuned with Euroc and ORB-SLAM. Figure 2b shows the difference in performance in smaller sequences between ORB-SLAM and all LIFT-SLAM versions. Moreover, in sequences 05 and 07 ORB-SLAM could not detect loop-closure, thus, its trajectories has an accumulated drift as shown in Figures 2c and 2d.

Table 2 shows the quantitative comparison between the algorithms in Euroc dataset. We can notice that ORB-SLAM has the smallest average in 3 sequences. Moreover, the proposed versions of LIFT-SLAM performed better than original LIFT-SLAM in all sequences. The algorithm's performance improved even when

we used the KITTI dataset to fine-tune the LIFT network. Therefore we confirmed that this network version also learned essential features from the dataset. Figure 2(e-f) shows the qualitative comparison between the algorithms in Euroc dataset. LIFT-SLAM and most of its versions could not track some of the sequences completely, except for Adaptive LIFT-SLAM fine-tuned with Euroc. Moreover, it is possible to notice that ORB-SLAM loses track multiple times in sequence V1_01. Therefore, considering the quantitative and qualitative results of all versions of LIFT-SLAM, Adaptive LIFT-SLAM fine-tuned with Euroc sequences is the one with better overall results.



(a) KITTI 00

(b) KITTI 03

(c) KITTI 05

(d) KITTI 07

(e) Euroc MH_01

(f) Euroc V1_01

Fig. 2: Qualitative results in KITTI (a-d) and Euroc (e-f) dataset.

Table 2: ATE (m) comparison between ORB-SLAM and LIFT-SLAM variations in the Euroc dataset. "X" are sequences unavailable due to tracking failure, and "-" sequences we did not execute the algorithm to avoid biased results.

Algorithm	MH_01	MH_02	MH_03	MH_04	V1_01	V1_03
ORB-SLAM	0.048	0.037	**0.040**	0.432	**0.100**	**0.370**
LIFT-SLAM	0.062	0.227	0.144	1.859	X	X
LIFT-SLAM fine-tuned with KITTI	0.115	0.042	0.055	**0.117**	0.117	X
LIFT-SLAM fine-tuned with Euroc	0.117	0.062	0.053	-	0.150	-
Adaptive LIFT-SLAM	0.046	**0.034**	X	X	0.101	X
Adaptive LIFT-SLAM fine-tuned with KITTI	0.455	X	0.116	X	0.194	X
Adaptive LIFT-SLAM fine-tuned with Euroc	**0.044**	0.053	0.049	-	0.157	-

4.3 Robustness tests

To test our system's robustness to camera sensor noise, we created different image distortion in some KITTI and Euroc sequences, simulating camera ill exposure conditions. We did not train our networks for dealing with these distortions. These scenarios were emulated with the application of gamma power transformation and quantile-based truncation, as proposed in [24]. As Adaptive LIFT-SLAM fine-tuned with Euroc sequences obtained the best overall results, we tested it under the described scenarios and compared its performance with ORB-SLAM under the same scenarios. The quantitative results of the tests are shown in table 3. ORB-SLAM could not track the camera's pose with some distortion in sequences KITTI 03, while LIFT-SLAM failed in some cases for Euroc MH_02. However, we can notice that in most of the sequences, LIFT-SLAM improved its performance when we applied the distortions. This occurs because the distortions remove some outliers from the images, which allows the algorithms to select better keypoints. Moreover, the learned features are more robust to ill exposure as the datasets used to fine-tune the network naturally contain varying illumination. Figure 3 shows the comparison of the trajectories obtained with each distortion in KITTI and Euroc. In sequence KITTI 03, LIFT-SLAM's trajectories were not much affected by distortions (Figure 3b). Furthermore, the trajectories of both algorithms were more affected in KITTI 07 (Figures 3c and 3d), where ORB-SLAM could not track a considerable part of the trajectory. In MH_02, the trajectories of both algorithms were less affected, but they lost track of the pose and relocalized in some sequence parts.

4.4 Comparison with literature

The results obtained have shown that LIFT is capable of improving a typical VSLAM algorithm. Moreover, transfer learning proved to be crucial in our system since it improved our algorithms in different VSLAM problems. The main drawback in our system is that the good performance in large environments depends on loop closure detection to correct drift accumulation. To compare our results with those available in the literature, we chose Adaptive LIFT-SLAM fine-tuned with Euroc sequences. Table 4 summarizes this comparison in the KITTI dataset. We chose different monocular VO and VSLAM algorithms to compare with LIFT-SLAM: traditional methods, hybrid methods, and end-to-end methods.

Table 3: Robustness tests results. Adaptive LIFT-SLAM fine-tuned with Euroc sequences is employed."X" are sequences unavailable due to tracking failure and "-" are the sequences we did not execute the algorithms.

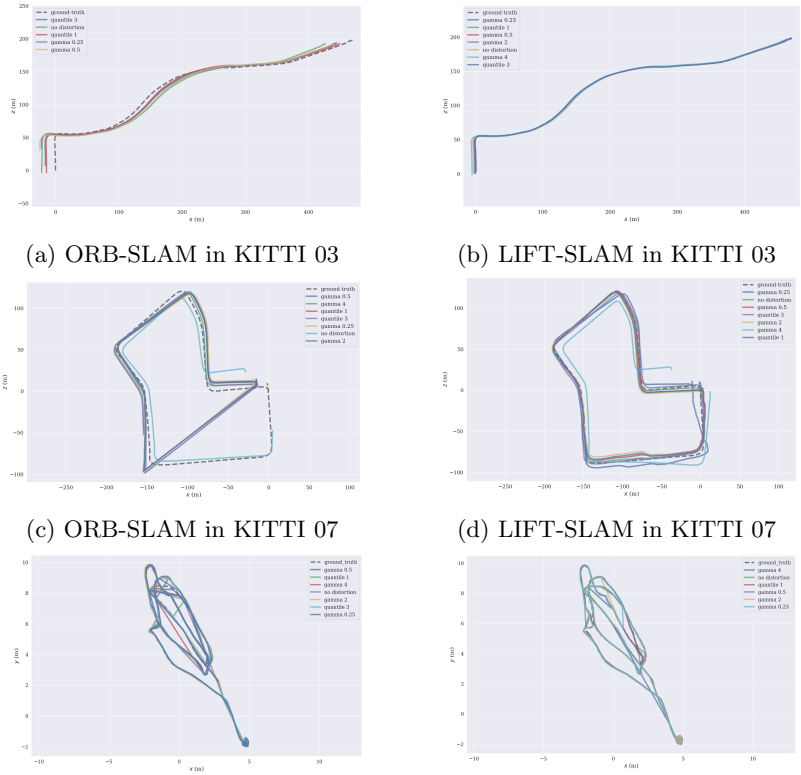
Sequence	Distortion	ORB-SLAM			LIFT-SLAM		
		RPE_{trans} (%)	RPE_{rot} (deg/m)	ATE (m)	RPE_{trans} (%)	RPE_{rot} (deg/m)	ATE (m)
KITTI 03	no distortion	9.75	2.78	15.13	1.46	0.34	2.23
	$\gamma = 0.25$	7.68	1.95	11.72	1.02	0.40	1.23
	$\gamma = 0.5$	8.25	2.24	11.38	1.28	0.36	1.74
	$\gamma = 2$	X	X	X	1.07	0.51	1.47
	$\gamma = 4$	X	X	X	2.82	0.70	5.23
	Truncation in Q_1	8.34	1.41	13.63	1.36	0.45	2.07
	Truncation in Q_3	9.78	2.23	15.66	1.10	0.46	1.27
KITTI 07	no distortion	7.43	3.58	13.47	2.42	4.02	3.63
	$\gamma = 0.25$	7.61	2.42	12.54	2.09	3.17	3.30
	$\gamma = 0.5$	6.37	2.02	9.58	1.99	3.88	2.86
	$\gamma = 2$	5.89	2.11	9.36	3.43	4.32	5.91
	$\gamma = 4$	6.61	7.06	7.84	8.03	7.40	16.13
	Truncation in Q_1	8.50	2.47	10.69	2.45	4.10	3.58
	Truncation in Q_3	7.01	2.40	7.08	2.69	3.77	4.49
Euroc MH_02	no distortion	-	-	0.037	-	-	0.053
	$\gamma = 0.25$	-	-	0.055	-	-	0.035
	$\gamma = 0.5$	-	-	0.040	-	-	0.039
	$\gamma = 2$	-	-	0.061	-	-	0.037
	$\gamma = 4$	-	-	0.010	-	-	0.194
	Truncation in Q_1	-	-	0.039	-	-	0.043
	Truncation in Q_3	-	-	0.043	-	-	X

Table 4: Comparison of LIFT-SLAM with results from monocular VO/VSLAM algorithms available in the literature. We fill with "X" results that are unavailable due to tracking failure and with "-" results not reported by the authors.

Algorithm	Type	Metric	00	01	02	03	04	05	06	07	08	09	10
LIFT-SLAM	Hybrid	ATE (m)	8.06	X	40.04	2.23	**0.51**	13.55	30.38	3.63	184.43	59.62	29.87
		RPE_{trans} (%)	**3.18**	X	8.73	**1.46**	**2.22**	6.09	12.24	**2.42**	47.10	19.91	9.72
		RPE_{rot} (deg/m)	2.99	X	2.49	**0.34**	**0.48**	3.11	2.91	4.02	2.02	2.14	**2.24**
ORB-SLAM	Traditional	ATE (m)	11.54	X	X	15.13	4.29	7.74	20.26	13.47	39.51	49.67	19.94
		RPE_{trans} (%)	4.46	X	X	9.75	3.71	3.35	8.11	7.43	12.16	26.51	8.65
		RPE_{rot} (deg/m)	3.28	X	X	2.78	2.15	3.57	2.88	3.58	3.05	11.13	3.62
DeepVO[3]	End-to-end	ATE (m)	5.33	X	21.28	1.51	1.62	4.85	12.34	2.26	46.68	6.62	8.80
		RPE_{trans} (%)	-	-	-	8.49	7.19	2.62	5.42	3.91	-	-	8.11
		RPE_{rot} (deg/m)	-	-	-	6.89	6.97	3.61	5.82	4.60	-	-	8.83
NeuralBundler [6]	Hybrid	ATE (m)	-	-	-	-	-	-	-	-	-	-	-
		RPE_{trans} (%)	3.24	-	4.85	-	-	1.83	2.74	3.53	-	6.23	-
		RPE_{rot} (deg/m)	1.35	-	1.60	-	-	0.7	2.6	2.02	-	2.11	-

5 Conclusion

In this work, we successfully apply a deep neural network in the front-end of a traditional visual SLAM algorithm. This approach showed that it is possible to improve VSLAM algorithms' performance with learned feature extraction and description. We also showed that transfer learning could be used to fine-tune these networks with VO/VSLAM datasets to improve the performance of the entire system on cross-datasets. Moreover, we successfully created a method to adapt the matching thresholds while executing the VO pipeline, depending on the number of outliers. This method allowed us to eliminate the fixed values of the matching thresholds without requiring dataset fine-tuning. All of these methods allowed us to evaluate five variations of LIFT-SLAM: LIFT-SLAM fine-tuned with KITTI sequences, LIFT-SLAM fine-tuned with Euroc sequences, Adaptive LIFT-SLAM, Adaptive LIFT-SLAM fine-tuned with KITTI sequences and Adaptive LIFT-SLAM fine-tuned with Euroc sequences. The proposed hybrid system can operate in different environments (indoors and outdoors) and



(a) ORB-SLAM in KITTI 03 (b) LIFT-SLAM in KITTI 03

(c) ORB-SLAM in KITTI 07 (d) LIFT-SLAM in KITTI 07

(e) ORB-SLAM in Euroc MH_02 (f) LIFT-SLAM in Euroc MH_02

Fig. 3: Qualitative results of the robustness tests.

improve its results with an artificial distortion applied to the images (gamma power transformation and quantile-based truncation). This showed us that a selection of the learned features could improve the performance of the algorithm. Therefore, in future work, we plan to add an attention-based mechanism to select the best features for VSLAM.

References

1. R. Li, S. Wang, Z. Long, D. Gu, Undeepvo: Monocular visual odometry through unsupervised deep learning, CoRR abs/1709.06841 (2017). arXiv:1709.06841.
2. E. Parisotto, D. S. Chaplot, J. Zhang, R. Salakhutdinov, Global pose estimation with an attention-based recurrent network, IEEE/CVF CVPRW (2018) 350–359.
3. S. Wang, R. Clark, H. Wen, N. Trigoni, Deepvo: Towards end-to-end visual odometry with deep recurrent convolutional neural networks, in: IEEE ICRA, 2017, pp. 2043–2050.
4. D. DeTone, T. Malisiewicz, A. Rabinovich, Self-improving visual odometry, CoRR abs/1812.03245 (2018).

5. R. Kang, J. Shi, X. Li, Y. Liu, X. Liu, DF-SLAM: A deep-learning enhanced visual SLAM system based on deep local features, CoRR abs/1901.07223 (2019). arXiv:1901.07223.
6. Y. Li, Y. Ushiku, T. Harada, Pose graph optimization for unsupervised monocular visual odometry, in: IEEE ICRA, 2019, pp. 5439–5445.
7. K. M. Yi, E. Trulls, V. Lepetit, P. Fua, Lift: Learned invariant feature transform., in: ECCV, Vol. 9910, Springer, 2016, pp. 467–483.
8. R. Mur-Artal, J. M. M. Montiel, J. D. Tardós, ORB-SLAM: a versatile and accurate monocular SLAM system, IEEE Transactions on Robotics 31 (5) (2015) 1147–1163.
9. A. Geiger, P. Lenz, C. Stiller, R. Urtasun, Vision meets robotics: The kitti dataset, INT J ROBOT RES 32 (11) (2013) 1231–1237.
10. M. Burri, J. Nikolic, P. Gohl, T. Schneider, J. Rehder, S. Omari, M. W. Achtelik, R. Siegwart, The euroc mav datasets, INT J ROBOT RES (2016).
11. G. Klein, D. Murray, Parallel tracking and mapping for small ar workspaces, 6th IEEE and ACM International Symposium (2007) 225–234.
12. T. Taketomi, H. Uchiyama, S. Ikeda, Visual slam algorithms: a survey from 2010 to 2016, IPSJ Transactions on Computer Vision and Applications 9:16 (2017).
13. R. Mur-Artal, J. Tardos, Orb-slam2: an open-source slam system for monocular, stereo and rgb-d cameras, IEEE Transactions on Robotics (10 2016).
14. A. Dosovitskiy, P. Fischer, E. Ilg, P. Häusser, C. Hazirbas, V. Golkov, P. v. d. Smagt, D. Cremers, T. Brox, Flownet: Learning optical flow with convolutional networks, in: 2015 IEEE International Conference on Computer Vision (ICCV), 2015, pp. 2758–2766. doi:10.1109/ICCV.2015.316.
15. P.-E. Sarlin, D. DeTone, T. Malisiewicz, A. Rabinovich, Superglue: Learning feature matching with graph neural networks, in: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2020, pp. 4938–4947.
16. V. Balntas, E. Riba, D. Ponsa, K. Mikolajczyk, Learning local feature descriptors with triplets and shallow convolutional neural networks, in: Proceedings of the British Machine Vision Conference (BMVC), BMVA Press, 2016.
17. O. Chapelle, M. Wu, Gradient descent optimization of smoothed information retrieval metrics, Information Retrieval 13 (3) (2010) 216–235.
18. C. Wu, Towards linear-time incremental structure from motion, in: International Conference on 3D Vision - 3DV, 2013, pp. 127–134.
19. D. Galvez-López, J. D. Tardos, Bags of binary words for fast place recognition in image sequences, IEEE Transactions on Robotics 28 (5) (2012) 1188–1197.
20. J. Engel, V. Usenko, D. Cremers, A photometrically calibrated benchmark for monocular visual odometry, Vol. abs/1607.02555, 2016. arXiv:1607.02555.
21. R. Mur-Artal, J. Tardos, Fast relocalisation and loop closing in keyframe-based slam, IEEE ICRA (2014) 846–853.
22. D. Nistér, H. Stewenius, Scalable recognition with a vocabulary tree, in: IEEE CVPR, Vol. 2, IEEE Computer Society, 2006, p. 2161–2168.
23. A. Geiger, P. Lenz, R. Urtasun, Are we ready for autonomous driving? the kitti vision benchmark suite, in: IEEE CVPR, 2012.
24. C. R. Steffens, L. R. V. Messias, P. L. J. Drews, S. S. d. C. Botelho, Can exposure, noise and compression affect image recognition? an assessment of the impacts on state-of-the-art convnets, in: IEEE LARS/SBR, 2019, pp. 61–66.