IV-SLAM: Introspective Vision for Simultaneous Localization and Mapping

Sadegh Rabiee
Department of Computer Science
University of Texas at Austin
Austin, TX 78712
`srabiee@cs.utexas.edu`

Joydeep Biswas
Department of Computer Science
University of Texas at Austin
Austin, TX 78712
`joydeepb@cs.utexas.edu`

Abstract

Existing solutions to visual simultaneous localization and mapping (V-SLAM) assume that errors in feature extraction and matching are independent and identically distributed (i.i.d), but this assumption is known to not be true – features extracted from low-contrast regions of images exhibit wider error distributions than features from sharp corners. Furthermore, V-SLAM algorithms are prone to catastrophic tracking failures when sensed images include challenging conditions such as specular reflections, lens flare, or shadows of dynamic objects. To address such failures, previous work has focused on building more robust visual frontends, to filter out challenging features. In this paper, we present *introspective vision* for SLAM (IV-SLAM), a fundamentally different approach for addressing these challenges. IV-SLAM explicitly models the noise process of reprojection errors from visual features to be context-dependent, and hence non-i.i.d. We introduce an autonomously supervised approach for IV-SLAM to collect training data to learn such a context-aware noise model. Using this learned noise model, IV-SLAM guides feature extraction to select more features from parts of the image that are likely to result in lower noise, and further incorporate the learned noise model into the joint maximum likelihood estimation, thus making it robust to the aforementioned types of errors. We present empirical results to demonstrate that IV-SLAM 1) is able to accurately predict sources of error in input images, 2) reduces tracking error compared to V-SLAM, and 3) increases the mean distance between tracking failures by more than 70% on challenging real robot data compared to V-SLAM.

Visual simultaneous localization and mapping (V-SLAM) extracts features from observed images, and identifies correspondences between features across time-steps. By optimizing the re-projection error of such features, V-SLAM reconstructs the trajectory of a robot along with a sparse 3D map of the locations of the features in the world. To accurately track the location of the robot and build a map of the world, V-SLAM requires selecting features from static objects, and correctly and consistently identifying correspondences between features. Unfortunately, despite extensive work on filtering out bad features [1–3] or rejecting unlikely correspondence matches [4–6], V-SLAM solutions still suffer from errors stemming from incorrect feature matches and features extracted from moving objects. Furthermore, V-SLAM solutions assume that re-projection errors are independent and identically distributed (i.i.d), an assumption that we know to be false: features extracted from low-contrast regions or from regions with repetitive textures exhibit wider error distributions than features from regions with sharp, locally unique corners. As a consequence of such assumptions, and the reliance on robust frontends to filter out bad features, even state of the art V-SLAM solutions suffer from catastrophic failures when encountering challenging scenarios such as specular reflections, lens flare, and shadows of moving objects encountered by robots in the real world. We present *introspective vision* for SLAM (IV-SLAM), a fundamentally different approach for addressing these challenges – instead of relying on a robust frontend to filter out bad features, IV-SLAM builds a context-aware

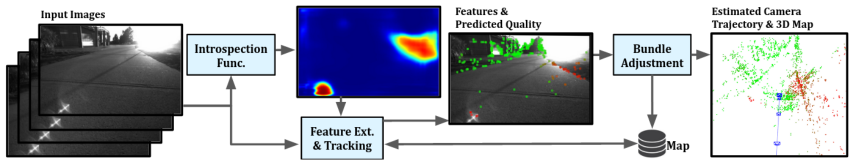

Figure 1: IV-SLAM pipeline during inference.

total noise model [7] that explicitly accounts for heteroscedastic noise, and learns to account for bad correspondences, moving objects, non-rigid objects and other causes of errors. IV-SLAM is capable of learning to identify causes of V-SLAM failures in an autonomously supervised manner, and is subsequently able to leverage the learned model to improve the robustness and accuracy of tracking during actual deployments.

1 Introspective Vision for SLAM

In visual SLAM, the pose of the camera $\mathbf{T}_t^w \in SE(3)$ is estimated and a 3D map $\mathbf{M} = \left\{\mathbf{p}_k^w | \mathbf{p}_k^w \in \mathbb{R}^3, k \in [1, N]\right\}$ of the environment is built by finding correspondences in the image space across frames. For each time-step t, the V-SLAM frontend processes image $\mathbf{I}_t$ to extract features $\mathbf{z}_{t,k} \in \mathbb{P}^2$ associated with 3D map points $\mathbf{p}_k^w$. The reprojection error of $\mathbf{p}_k^w$ onto the image $\mathbf{I}_t$ is defined as $\epsilon_{t,k} = \mathbf{z}_{t,k} - \hat{\mathbf{z}}_{t,k}$, where $\hat{\mathbf{z}}_{t,k}$ is the prediction of the observation $\mathbf{z}_{t,k}$. The solution to SLAM is achieved via a nonlinear optimization problem:

$$\overset{*}{\mathbf{T}}{}^w_{1:t}, \overset{*}{\mathbf{M}} = \underset{\mathbf{T}^w_{1:t}, \mathbf{M}}{\arg\min} \sum_{t,k} L\left(\epsilon_{t,k}^T \Sigma_{t,k}^{-1} \epsilon_{t,k}\right), \tag{1}$$

where $\Sigma_{t,k}$ is the covariance matrix associated to the scale at which a feature has been extracted and L is a robust loss function. IV-SLAM models the observation error distribution to be dependent on the observations, i.e. $\mathbf{z} = \hat{\mathbf{z}}(\mathbf{T^w_{1:t}}, \mathbf{M}) + \tilde{\epsilon}(\mathbf{z})$, where $\tilde{\epsilon} \sim \tilde{\phi}$ and $\tilde{\phi}$ is a heteroscedastic noise distribution. Let $p_{\tilde{\phi}}$ be the probability density function (PDF) of $\tilde{\phi}$, we want $p_{\tilde{\phi}} \propto \exp(-L)$, where $L \in \mathcal{L}$ is a loss function from the space of robust loss functions. In this paper, we choose $L \in \mathcal{H} \subset \mathcal{L}$, where $\mathcal{H}$ is the space of Huber loss functions and specifically

$$L_{\delta(\mathbf{z})}(x) = \begin{cases} x & \text{if } x < \delta(\mathbf{z}) \\ 2\delta(\mathbf{z})(\sqrt{x} - \delta(\mathbf{z})/2) & \text{otherwise} \end{cases} \tag{2}$$

where $x \in [0, \infty)$ is the squared error value and $\delta(\mathbf{z})$ is an observation-dependent parameter of the loss function and is correlated with the reliability of the observations. IV-SLAM learns an empirical estimate of $\delta(z)$ such that the corresponding error distribution $\tilde{\phi}$ better models the observed error values. During the training phase, input images and estimated observation error values are used to learn to predict the reliability of image features at any point on an image. During the inference phase, a context-aware $\delta(z)$ is estimated for each observation using the predicted reliability score, where a smaller value of $\delta(z)$ corresponds to an unreliable observation. The resultant loss function $L_{\delta(\mathbf{z})}$ is then used in Eq. 1 to solve for $\mathbf{T}^w_{1:t}$ and $\mathbf{M}$. IV-SLAM builds on top of a feature-based V-SLAM algorithm and is agnostic of the type of image features or the V-SLAM frontend. Fig. 1 illustrates the IV-SLAM pipeline during inference.

Introspection Function. In order to apply a per-observation loss function $L_{\delta(\mathbf{z})}$, IV-SLAM learns an *introspection function* $\mathbb{I} : \mathbf{I} \times \mathbb{R}^2 \rightarrow \mathbb{R}$ that given an input image $\mathbf{I}_t$ and a location $(i, j) \in \mathbb{R}^2$ on the image, predicts a *cost value* $c_{t\,i,j} \in [0, 1]$ that represents a measure of reliability for image features extracted at $\mathbf{I}_t(i, j)$. Higher values of $c_{t\,i,j}$ indicate a less reliable feature. $\mathbb{I}$ is implemented as a fully convolutional neural network such that given an input image $\mathbf{I}_t$, outputs an image of the same size $\mathbf{I}_{\mathbf{c}t}$, which we refer to as the image feature *costmap* and $c_{t\,i,j} = \mathbf{I}_{\mathbf{c}t}(i, j)$. We use the same architecture as that used by Zhou et al. [8] for the task of image segmentation.

Self-Supervised Training. IV-SLAM requires a set of pairs of input images and their corresponding target image feature costmaps $\{(\mathbf{I}_t, \mathbf{I}_{\mathbf{c}t})\}$ to train the introspection function. The training is performed offline and loose supervision is provided in the form of estimates of the reference pose of the camera $\{\mathbf{T}_t^w\}$ by a 3D lidar-based SLAM solution [9]. $\{\mathbf{T}_t^w\}$ is only used to flag the frames, at which the tracking accuracy of V-SLAM is low, so they are not used for training data generation. To collect the training data, the V-SLAM algorithm is run on the images and at each frame $\mathcal{K}_t$ that has been recognized as reliable for training data labeling, reprojection error values $\epsilon_{t,k}$ are calculated for all matched image features. A normalized cost value $c_{t,k} = \epsilon_{t,k}^T \Sigma_{t,k}^{-1} \epsilon_{t,k}$ is then computed for each image feature, where $\Sigma_{t,k}$ denotes the diagonal covariance matrix associated with the scale at which the feature has been extracted. The set of sparse cost values calculated for each frame is then converted to a costmap $\mathbf{I}_{\mathbf{c}_t}$ the same size as the input image. This is achieved using a Gaussian Process regressor (GP). The generated costmaps along with the input images are then used to train the introspection function using a stochastic gradient descent (SGD) optimizer and a mean squared error loss (MSE) that is only applied to the regions of the image where the uncertainty of the output of the GP is lower than a threshold.

Robust Estimation in IV-SLAM. During inference, input images are run through the introspection function $\mathbb{I}$, which outputs estimated costmaps $\hat{\mathbf{I}}_{\mathbf{c}_t}$. IV-SLAM uses $\hat{\mathbf{I}}_{\mathbf{c}_t}$ to 1) guide the feature extraction process and 2) adjust the contribution of extracted features when solving for $\mathbf{T}_{1:t}^{w}$ and $\mathbf{M}$. First, each input image is divided into equally sized cells and the maximum number of image features to be extracted from each cell C_k is determined to be inversely proportional to $\sum_{(i,j)\in C_k} \hat{\mathbf{I}}_{\mathbf{c}_t}(i,j)$, i.e. the sum of the corresponding costmap image pixels within that cell. Then, extracted features from the input image are matched with map points, and for each matched image feature $\mathbf{z}_{t,k}$ extracted at pixel location (i,j), a specific loss function $L_{\delta(\mathbf{z}_{t,k})}$ is generated as defined in Eq. 2. The loss function parameter $\delta(\mathbf{z}_{t,k}) \in [0, \delta_{\max}]$ is approximated as $\frac{1-\hat{c}_{t,k}}{1+\hat{c}_{t,k}}\delta_{\max}$, where $\hat{c}_{t,k} = \hat{\mathbf{I}}_{\mathbf{c}_t}(i,j) \in [0,1]$ and $\delta_{\max}$ is a positive constant and a hyperparameter that defines the range at which $\delta(\mathbf{z}_{t,k})$ can be adjusted. We pick $\delta_{\max}$ to be the chi-square distribution's 95th percentile. Lastly, the tracked features along with their loss functions are plugged into Eq. 1 and the solution to the bundle adjustment problem are estimated using a nonlinear optimizer.

2 Experimental Results

In this section we evaluate IV-SLAM in terms of its tracking accuracy and the frequency of its tracking failures. We implement it on top of the stereo version of ORB-SLAM [10]. We pick ORB-SLAM as the baseline because it has various levels of feature matching pruning and outlier rejection in place, which indicates that the remaining failure cases that we address with introspective vision cannot be resolved with meticulously engineered outlier rejection methods.

Experimental Setup. State-of-the-art vision-based SLAM algorithms have shown great performance on popular benchmark datasets such as KITTI and EuROC [11]. These datasets, however, do not perfectly reflect the many difficult situations that can happen when the robots are deployed in the wild and over extended periods of time. In order to assess the effectiveness of IV-SLAM on improving visual SLAM performance, in addition to evaluation on the public EuRoC and KITTI datasets, we also put IV-SLAM to test on simulated and real-world datasets that we have collected to expose the algorithm to challenging situations such as reflections, glare, shadows, and dynamic objects. The real-world dataset consists of more than 7 km worth of trajectories traversed by a Clearpath Jackal robot equipped with a stereo pair of RGB cameras as well as a Velodyne VLP-16 3D Lidar. The simulation data is collected in the photo-realistic AirSim simulator and encompasses more than 60 km traversed by a car in different weather conditions.

Evaluation Metrics. We compare our introspection-enabled version of ORB-SLAM with the original algorithm in terms of their camera pose tracking accuracy and the mean distance between tracking failures (MDBF). Both algorithms are run on the test data and their estimated poses of the camera are recorded. If the algorithms loose track due to lack of sufficient feature matches across frames, tracking is reinitialized and continued from after the point of failure along the trajectory and the event is logged as an instance of tracking failure for the corresponding SLAM algorithm. The relative pose error (RPE) is then calculated for both algorithms at consecutive pairs of frames that are d meters apart as defined in [12].

Evaluation on KITTI and EuRoC datasets. We perform leave-one-out cross-validation separately on the KITTI and EuRoC datasets, i.e. to test IV-SLAM on each sequence, we train it on all other sequences in the same dataset. Tables 1 and 2 compare the per trajectory root-mean-square error (RMSE) of the rotation and translation parts of the RPE for IV-SLAM and ORB-SLAM in the KITTI and EuRoC datasets, respectively. The baseline ORB-SLAM does a good job of tracking

Table 1: Tracking Accuracy in the KITTI Dataset

	IV-SLAM		ORB-SLAM	
Sequence	Trans. Err. %	Rot. Err. (deg/100m)	Trans. Err. %	Rot. Err. (deg/100m)
00	0.69	0.25	0.69	0.25
01	**1.43**	0.22	1.47	0.22
02	0.79	**0.22**	**0.76**	0.24
03	0.74	**0.19**	**0.70**	0.23
04	**0.49**	0.13	0.55	0.13
05	0.40	0.16	**0.38**	0.16
06	**0.49**	**0.14**	0.56	0.19
07	0.49	**0.27**	0.49	0.29
08	**1.02**	0.31	1.05	0.31
09	0.85	0.25	**0.82**	0.25
10	**0.61**	**0.26**	0.62	0.29
Average	0.77	**0.24**	0.77	0.25

Table 2: Tracking Accuracy in the EuRoC Dataset

	IV-SLAM		ORB-SLAM	
Sequence	Trans. Err. %	Rot. Err. (deg/m)	Trans. Err. %	Rot. Err. (deg/m)
MH1	**2.26**	**0.19**	2.42	0.21
MH2	1.78	0.18	**1.49**	**0.16**
MH3	3.27	0.18	3.27	**0.17**
MH4	3.85	0.16	**3.49**	**0.15**
MH5	**2.98**	**0.16**	3.32	0.18
V1_1	8.93	**1.05**	**8.85**	1.06
V1_2	**4.38**	0.41	4.46	**0.39**
V1_3	**7.85**	**1.24**	14.86	2.35
V2_1	**2.92**	**0.76**	4.37	1.39
V2_2	2.89	0.62	2.76	**0.59**
V2_3	**11.00**	2.49	12.73	**2.39**
Average	**4.74**	**0.68**	5.64	0.82

Table 3: Aggregate Results for Simulation and Real-world Experiments

	Real-World			Simulation		
Method	Trans. Err. %	Rot. Err. (deg/m)	MDBF (m)	Trans. Err. %	Rot. Err. (deg/m)	MDBF (m)
IV-SLAM	**5.85**	**0.511**	**621.1**	**12.25**	**0.172**	**450.4**
ORB-SLAM	9.20	0.555	357.1	18.20	0.197	312.7

in the KITTI dataset. There exists no tracking failures and the overall relative translational error is less than 1%. Given the lack of challenging scenes in this dataset, IV-SLAM performs similar to ORB-SLAM with only marginal improvement in the overall rotational error. While EuRoC is more challenging than the KITTI given the higher mean angular velocity of the robot, the only tracking failure happens in the V2_3 sequence and similarly for both ORB-SLAM and IV-SLAM due to severe motion blur. IV-SLAM performs similar to ORB-SLAM on the easier trajectories, however, it significantly reduces the tracking error on the more challenging trajectories such as V1_3. Over all the sequences, IV-SLAM improves both the translational and rotational tracking accuracy.

Evaluation on Challenging datasets. We also evaluate IV-SLAM on our simulation and real-world datasets, which include scenes that are representative of the challenging scenarios that can happen in the real-world applications of V-SLAM. Table 3 summarizes the results and shows the RMSE values calculated over all trajectories. The results show that IV-SLAM leads to more than 70% increase in the MDBF and a 35% decrease in the translation error in the real-world dataset. IV-SLAM similarly outperforms the original ORB-SLAM in the simulation dataset by both reducing the tracking error and increasing MDBF. As it can be seen numerous tracking failures happen in both environments and the overall error rates are larger than those corresponding to the KITTI and EuRoC datasets due to the more difficult nature of these datasets. It is noteworthy that the benefit gained from using IV-SLAM is also more pronounced on these datasets with challenging visual settings. Figure 2 demonstrates an example deployment session of the robot from the real-world dataset.

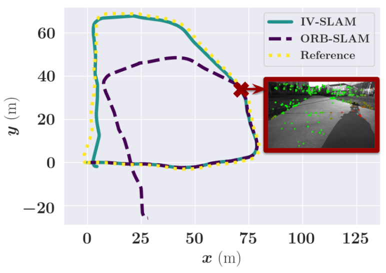

Figure 2: Example deployment session of the robot. IV-SLAM successfully follows the reference camera trajectory while ORB-SLAM leads to severe tracking errors caused by image features extracted on the shadow of the robot.

3 Conclusion

In this paper, we introduced IV-SLAM: a self-supervised approach for learning to predict sources of failure for V-SLAM and to estimate a context-aware noise model for image correspondences. We empirically demonstrated that IV-SLAM improves the accuracy and robustness of a state-of-the-art V-SLAM solution with extensive simulated and real-world data.

References

[1] Nicolas Alt, Stefan Hinterstoisser, and Nassir Navab. Rapid selection of reliable templates for visual tracking. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1355–1362. IEEE, 2010.

[2] Xiang Wang and Hong Zhang. Good image features for bearing-only slam. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2576–2581. IEEE, 2006.

[3] Luca Carlone and Sertac Karaman. Attention and anticipation in fast visual-inertial navigation. *IEEE Transactions on Robotics*, 35(1):1–20, 2018.

[4] Amir R Zamir, Tilman Wekel, Pulkit Agrawal, Colin Wei, Jitendra Malik, and Silvio Savarese. Generic 3d representation via pose estimation and matching. In *European Conference on Computer Vision*, pages 535–553. Springer, 2016.

[5] Yuki Ono, Eduard Trulls, Pascal Fua, and Kwang Moo Yi. Lf-net: learning local features from images. In *Advances in neural information processing systems*, pages 6234–6244, 2018.

[6] Yurun Tian, Xin Yu, Bin Fan, Fuchao Wu, Huub Heijnen, and Vassileios Balntas. Sosnet: Second order similarity regularization for local descriptor learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 11016–11025, 2019.

[7] Bill Triggs, Philip F McLauchlan, Richard I Hartley, and Andrew W Fitzgibbon. Bundle adjustment–a modern synthesis. In *International workshop on vision algorithms*, pages 298–372. Springer, 1999.

[8] Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Scene parsing through ade20k dataset. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 633–641, 2017.

[9] Tixiao Shan and Brendan Englot. Lego-loam: Lightweight and ground-optimized lidar odometry and mapping on variable terrain. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4758–4765. IEEE, 2018.

[10] Raul Mur-Artal and Juan D Tardós. Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. *IEEE Transactions on Robotics*, 33(5):1255–1262, 2017.

[11] Michael Burri, Janosch Nikolic, Pascal Gohl, Thomas Schneider, Joern Rehder, Sammy Omari, Markus W Achtelik, and Roland Siegwart. The euroc micro aerial vehicle datasets. *The International Journal of Robotics Research*, 35(10):1157–1163, 2016.

[12] David Schubert, Thore Goll, Nikolaus Demmel, Vladyslav Usenko, Jörg Stückler, and Daniel Cremers. The tum vi benchmark for evaluating visual-inertial odometry. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1680–1687. IEEE, 2018.