IV-SLAM: Introspective Vision for Simultaneous Localization and Mapping

Sadegh Rabiee
Department of Computer Science
University of Texas at Austin
United States
srabiee@cs.utexas.edu

Joydeep Biswas
Department of Computer Science
University of Texas at Austin
United States
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.

Keywords: SLAM, Introspection, Computer Vision

Visual simultaneous localization and mapping (V-SLAM) extracts features from observed images, and identifies correspondences between features across time-steps. By jointly optimizing the re-projection error of such features along with motion information derived from odometry or inertial measurement units (IMUs), 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, 2, 3] or rejecting unlikely correspondence matches [4, 5, 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 *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

4th Conference on Robot Learning (CoRL 2020), Cambridge MA, USA.

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. Our experimental results demonstrate that IV-SLAM 1) is able to accurately predict sources of error in input images as identified by ground truth in simulation, 2) reduces tracking error on both simulation and real-world data, and 3) significantly increases the mean distance between tracking failures when operating under challenging real-world conditions that frequently lead to catastrophic tracking failures of V-SLAM.

1 Related Work

There exists a plethora of research on designing and learning distinctive image feature descriptors. This includes the classical hand-crafted descriptors such as ORB [8] and SIFT [9], as well as the more recent learned ones [10, 11, 12, 5, 6] that rely on Siamese and triplet network architectures to generate a feature vector given an input image patch. Selecting interest points on the images for extracting these descriptors is traditionally done by convolving the image with specific filters [13] or using first-order approximations such as the Harris detector. More recently, CNN approaches have become popular [14, 15]. Cieslewski et al. [16] train a network that given an input image outputs a score map for selecting interest points. Once features are extracted at selected keypoints, pruning out a subset of them that are predicted to be unreliable is done in different ways. Alt et al. [1] train a classifier to predict good features at the descriptor level, and Wang and Zhang [2] use hand-crafted heuristics to determine good SIFT features. Carlone and Karaman [3] propose a feature pruning method for visual-inertial SLAM that uses the estimated velocity of the camera to reject image features that are predicted to exit the scene in the immediate future frames. A line of work leverages scene semantic information in feature selection. Kaneko et al. [17] run input images through a semantic segmentation network and limit feature extraction to semantic classes that are assumed to be more reliable such as static objects. Ganti and Waslander [18] follow a similar approach while taking into account the uncertainty of the segmentation network in their feature selection process. While these approaches benefit from using the contextual information in the image, they are limited to hand-enumerated lists of sources of error. Moreover, not all potential sources of failure can be categorized in one of the semantic classes, for which well-trained segmentation networks exist. Repetitive textures, shadows, and reflections are examples of such sources. Pruning bad image correspondences once features are matched across frames is also another active area of research. Forward predicting the motion of the robot by means of accurate learned motion models [19] reduces outliers to some extent by limiting the region in the image, where the corresponding match for each feature can lie. The remaining wrong correspondences have traditionally been addressed with RANSAC [20], and more recently deep learning approaches have been developed [21, 22], which use permutation-equivariant network architectures and predict outlier correspondences by processing coordinates of the pairs of matched features. While the goal of these methods is to discard outlier correspondences, not all bad matches are outliers. There is also a grey area of features that for reasons such as specularity, shadows, motion blur, etc., are not located as accurately as other features without clearly being outliers.

Our work is agnostic of the feature descriptor type and the feature matching method at hand. It is similar to the work by Cieslewski et al. [16] in that it learns to predict unreliable regions for feature extraction in a self-supervised manner. However, it goes beyond being a learned keypoint detector and applies to the full V-SLAM solution by exploiting the predicted feature reliability scores to generate a context-aware loss function for the bundle adjustment problem. Unlike available methods for learning good feature correspondences, which require accurate ground truth poses of the camera for training [21, 22, 4], our work only requires rough estimates of the reference pose of the camera. IV-SLAM is inspired by early works on introspective vision [23, 24] and applies the idea to V-SLAM.

2 Visual 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 new input image $\mathbf{I}_t$ (or stereo image pair $(\mathbf{I}_{t,l}, \mathbf{I}_{t,r})$, image features are extracted and matched with those from previous frames. Then, the solution to SLAM is given by

$$\overset{*}{\mathbf{T}}{}_{1:t}^{w}, \overset{*}{\mathbf{M}} = \underset{\mathbf{T}_{1:t}^w, \mathbf{M}}{\arg\max}\, P(\mathbf{T}_{1:t}^w, \mathbf{M} | \mathbf{Z}_{1:t}, \mathbf{u}_{1:t}) = \underset{\mathbf{T}_{1:t}^w, \mathbf{M}}{\arg\max}\, P(\mathbf{Z}_{1:t} | \mathbf{T}_{1:t}^w, \mathbf{M}) P\left(\mathbf{T}_{1:t}^w | \mathbf{u}_{1:t}\right), \tag{1}$$

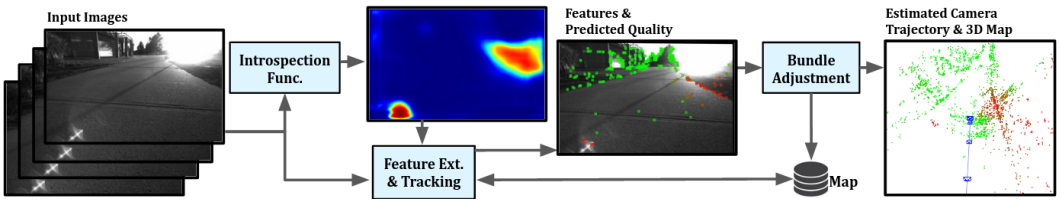


Figure 1: IV-SLAM pipeline during inference.

where $\mathbf{Z_{1:t}}$ represents observations made by the robot and $\mathbf{u}_{1:t}$ are the control commands and/or odometry and IMU measurements. $P(\mathbf{Z}|\mathbf{T}^w_{1:t}, \mathbf{M})$ is the observation likelihood for image feature correspondences, given the estimated poses of the camera and the map $\mathbf{M}$. 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 $\mathbf{p}^w_k$. The observation error here is the reprojection error of $\mathbf{p}^w_k$ onto the image $\mathbf{I}_t$ and is defined as

$$\epsilon_{t,k} = \mathbf{z}_{t,k} - \hat{\mathbf{z}}_{t,k}, \quad \hat{\mathbf{z}}_{t,k} = \mathbf{A}\left[\mathbf{R}^t_w | \mathbf{t}^t_w\right] \mathbf{p}^w_k, \tag{2}$$

where $\hat{\mathbf{z}}_{t,k}$ is the prediction of the observation $\mathbf{z}_{t,k}$, $\mathbf{A}$ is the camera matrix, and $\mathbf{R}^t_w \in SO(3)$ and $\mathbf{t}^t_w \in \mathbb{R}^3$ are the rotation and translation parts of $\mathbf{T}^t_w$, respectively. In the absence of control commands and IMU/odometry measurements, Eq. 1 reduces to the bundle adjustment (BA) problem, which is formulated as a nonlinear optimization:

$$\mathbf{T}^{w\,*}_{1:t}, \mathbf{M}^* = \underset{\mathbf{T}^w_{1:t}, \mathbf{M}}{\arg\min} \sum_{t,k} L\left(\epsilon^T_{t,k} \Sigma^{-1}_{t,k} \epsilon_{t,k}\right), \tag{3}$$

where $\Sigma_{t,k}$ is the covariance matrix associated to the scale at which a feature has been extracted and L is the loss function for $P(\mathbf{Z}|\mathbf{T}^w_{1:t}, \mathbf{M})$. The choice of noise model for the observation error ϵ has a significant effect on the accuracy of the maximum likelihood (ML) solution to SLAM [7]. There exists a body of work on developing robust loss functions [25, 26] that targets improving the performance of vision tasks in the presence of outliers. The reprojection error is known to have a non-Gaussian distribution ϕ in the context of V-SLAM due to the frequent outliers that happen in image correspondences [7]. It is assumed to be drawn from long-tailed distributions such as piecewise densities with a central peaked inlier region and a wide outlier region. While ϕ is usually modeled to be i.i.d., there exist obvious reasons as to why this is not a realistic assumption. Image features extracted from objects with high-frequency surface textures can be located less accurately; whether the underlying object is static or dynamic affects the observation error; the presence of multiple similar-looking instances of an object in the scene can lead to correspondence errors. In the next section, we explain how IV-SLAM leverages the contextual information available in the input images to learn an improved ϕ that better represents the non i.i.d nature of the observation error.

3 Introspective Vision for SLAM

IV-SLAM builds on top of a feature-based V-SLAM algorithm. It is agnostic of the type of image features or the frontend, hence applicable to a wide range of existing V-SLAM algorithms. IV-SLAM 1) equips V-SLAM with an *introspection function* that learns to predict the reliability of image features, 2) modifies the feature extraction step to reduce the number of features extracted from unreliable regions of the input image and extract more features from reliable regions, and 3) modifies the BA optimization to take into account the predicted reliability of extracted features.

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 [26]. In this paper, we choose $L \in \mathcal{H} \subset \mathcal{L}$, where $\mathcal{H}$ is the space of Huber loss functions and specifically

$$L_{\delta(\mathbf{z})}(x) = \begin{cases} x & \text{if } x < \delta(\mathbf{z}) \\ 2\delta(\mathbf{z})(\sqrt{x} - \delta(\mathbf{z})/2) & \text{otherwise} \end{cases} \tag{4}$$

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 uses the introspection function to learn an empirical estimate of $\delta(\mathbf{z})$ 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 $\delta(\mathbf{z})$ is estimated for each observation using the predicted reliability score, where a smaller value of $\delta(\mathbf{z})$ corresponds to an unreliable observation. The resultant loss function $L_{\delta(\mathbf{z})}$ is then used in Eq. 3 to solve for $\mathbf{T}^w_{1:t}$ and $\mathbf{M}$. Fig. 1 illustrates the IV-SLAM pipeline during inference. In the rest of this section, we explain the training and inference phases of IV-SLAM in detail.

3.1 Self-Supervised Training of IV-SLAM

One of the main properties of IV-SLAM as an introspective perception algorithm is its capability to generate labeled data required for its training in a self-supervised and automated manner. This empowers the system to follow a continual learning approach and exploit logged data during robot deployments for improving perception. In the following, we define the introspection function in IV-SLAM and explain the automated procedure for its training data collection.

3.1.1 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. Our implementation of $\mathbb{I}$ uses a deep neural network that given an input image $\mathbf{I}_t$, outputs an image of the same size $\mathbf{I}_{\mathbf{c}t}$. We refer to $\mathbf{I}_{\mathbf{c}t}$ as the image feature *costmap* and $c_{t\,i,j} = \mathbf{I}_{\mathbf{c}t}(i, j)$. We use a fully convolutional network with the same architecture as that used by Zhou et al. [27] for image segmentation. The network is composed of the MobileNetV2 [28] as the encoder and a transposed convolution layer with deep supervision as the decoder.

3.1.2 Collection of Training Data

IV-SLAM requires a set of pairs of input images and their corresponding target image feature costmaps $\mathbf{D} = \{(\mathbf{I}_t, \mathbf{I}_{\mathbf{c}t})\}$ 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 $\{\mathbf{T}^w_t\}$ by a 3D lidar-based SLAM solution [29]. $\{\mathbf{T}^w_t\}$ 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. The automated procedure for generating the dataset $\mathbf{D}$, presented in Algorithm 1, is as follows: The V-SLAM algorithm is run on the images and at each frame $\mathcal{K}_t$, the Mahalanobis distance of the reference and estimated pose of the camera is calculated as

$$d^t = \delta\mathbf{T}_t{}^T \Sigma^{-1}_{\mathbf{T}^w_t} \delta\mathbf{T}_t \tag{5}$$

$$\delta\mathbf{T}_t = \mathrm{Log}(\Delta\mathbf{T}_t) = \mathrm{Log}\left(\hat{\mathbf{T}}^w_t\ \mathbf{T}^t_w\right) \tag{6}$$

where $\delta\mathbf{T}_t \in \mathfrak{se}(3)$ denotes the corresponding element of $\Delta\mathbf{T}_t$ in the Lie algebra of $SE(3)$. $\Sigma_{\mathbf{T}^w_t}$ is the covariance of the reference pose of the camera and is approximated as a constant lower

Algorithm 1 Training Label Generation

Input: Set of matched features and map points $\{(\mathbf{z}_{t,k}, \mathbf{p}_k)\}_{k=1}^N$ for current frame, estimated camera pose $\hat{\mathbf{T}}^w_t$, reference camera pose $\mathbf{T}^w_t$, reference camera pose covariance $\Sigma_{\mathbf{T}^w_t}$
Output: Costmap image $\mathbf{I}_{\mathbf{c}t}$
$l \leftarrow$ costmap computation grid cell size
$(h, w) \leftarrow$ output cost-map size
$\mathbf{I}_\mathbf{g}[\texttt{floor}(\frac{h}{l})][\texttt{floor}(\frac{w}{l})]$
$\Delta\mathbf{T}_t \leftarrow \hat{\mathbf{T}}^w_t\ \mathbf{T}^t_w$
if IsTrackingUnreliable$(\Delta\mathbf{T}_t, \Sigma_{\mathbf{T}^w_t})$ **then**
    return -1.
end if
for $k \leftarrow 1$ to N **do**
    $\epsilon_{t,k} \leftarrow$ CalcReprojectionError$(\mathbf{z}_{t,k}, \mathbf{p}_k)$
    $c_{t,k} \leftarrow \epsilon^T_{t,k} \Sigma^{-1}_{t,k} \epsilon_{t,k}$
end for
for $i \leftarrow 0$ to $\texttt{floor}(\frac{w}{l})$ **do**
    for $j \leftarrow 0$ to $\texttt{floor}(\frac{h}{l})$ **do**
        $\mathbf{y} \leftarrow (il + l/2, jl + l/2)$
        $\mathbf{I}_\mathbf{g}[i][j] \leftarrow$ GaussianProc$(\mathbf{y}, \{(\mathbf{z}_{t,k}, c_{t,k})\}_{k=1}^N)$
    end for
end for
$\mathbf{I}_{\mathbf{c}t} \leftarrow$ ResizeImage$(\mathbf{I}_\mathbf{g}, (h, w))$

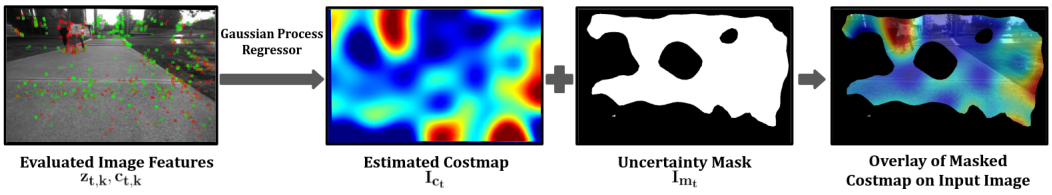


Figure 2: Training label generation for the introspective function.

bound for the covariance of the reference SLAM solution. A χ^2 test with $\alpha = 0.05$ is done for d^t and if it fails, the current frame will be flagged as unreliable and a training label will not be generated for it. At each frame $\mathcal{K}_t$ 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. 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. Figure 2 shows the estimated costmap and the uncertainty mask for an example input image.

The advantage of such a self-supervised training scheme is that it removes the need for highly accurate ground truth poses of the camera, which would have been necessary if image features were to be evaluated by common approaches such as the epipolar error across different frames.

3.2 Robust Estimation in IV-SLAM

During inference, input images are run through the introspection function $\mathbb{I}$ to obtain the estimated costmaps $\hat{\mathbf{I}}_{\mathbf{c_t}}$. IV-SLAM uses $\hat{\mathbf{I}}_{\mathbf{c_t}}$ to both guide the feature extraction process and adjust the contribution of extracted features when solving for $\mathbf{T}_{1:t}^w$ and $\mathbf{M}$.

Guided feature extraction. Each image $\mathbf{I}_t$ 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. This helps IV-SLAM prevent situations where the majority of extracted features in a frame are unreliable.

Reliability-aware bundle adjustment. 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. 4. 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 χ^2 distribution's 95$^{\text{th}}$ percentile, i.e. 7.82 for a stereo observation. In other terms, for each image feature, the Huber loss is adjusted such that the features that are predicted to be less reliable (larger $c_{t,k}$) have a less steep associated loss function (smaller $\delta(\mathbf{z}_{t,k})$). Lastly, the tracked features along with their loss functions are plugged into Eq. 3 and the solution to the bundle adjustment problem, i.e. the current pose of the camera as well as adjustments to the previously estimated poses and the location of map points, are estimated using a nonlinear optimizer.

4 Experimental Results

In this section: 1) We evaluate IV-SLAM on how well it predicts reliability of image features (Section 4.2). 2) We show that IV-SLAM improves tracking accuracy of a state-of-the-art V-SLAM algorithm and reduces frequency of its tracking failures (Section 4.3). 3) We look at samples of sources of failure learned by IV-SLAM to negatively affect V-SLAM. (Section 4.4). To evaluate IV-SLAM, we implement it on top of the stereo version of ORB-SLAM [30]. We pick ORB-SLAM 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.

4.1 Experimental Setup

State-of-the-art vision-based SLAM algorithms have shown great performance on popular benchmark datasets such as [31] and EuRoC [32]. These datasets, however, do not reflect the many difficult situations that can happen when the robots are deployed in the wild and over extended periods of time [33]. 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.

Simulation. In order to evaluate IV-SLAM in a controlled setting, where we have accurate poses of the camera and ground truth depth of the scene, we use AirSim [34], a photo-realistic simulation environment. A car is equipped with a stereo pair of RGB cameras as well as a depth camera that provides ground truth depth readings for every pixel in the left camera frame. The dataset encompasses more than 60 km traversed by the car in different environmental and weather conditions such as clear weather, wet roads, and in the presence of snow and leaves accumulation on the road.

Real-world. We also evaluate IV-SLAM on real-world data that we collect using a Clearpath Jackal robot equipped with a stereo pair of RGB cameras as well as a Velodyne VLP-16 3D Lidar. The dataset consists of more than 7 km worth of trajectories traversed by the robot over the span of a week in a college campus setting in both indoor and outdoor environments and different lighting conditions. The reference pose of the robot is estimated by a 3D Lidar-based SLAM solution, LeGO-LOAM [29]. The algorithm is run on the data offline and with extended optimization rounds for increased accuracy. For both the real-world and simulation datasets the data is split into training and test sets, each composed of separate full deployment sessions(uninterrupted trajectories), such that both training and test sets include data from all environmental conditions.

4.2 Feature Reliability Prediction

In order to evaluate IV-SLAM's introspection function (IF) on predicting reliability of image features, we compare IF's predicted reliability of image features with their corresponding ground truth reprojection error. Since obtaining the ground truth reprojection error requires access to ground truth 3D coordinates of objects associated with each image feature as well as accurate reference poses of the camera, we conduct this experiment in simulation. IV-SLAM is trained on the simulation dataset with the method explained in Section 3. The IF is then run on all images in the test set along with the original ORB-SLAM. For each image feature extracted and matched by ORB-SLAM, we log its predicted cost by the IF, as well as its corresponding ground truth reprojection error. We then sort all image features in ascending order with respect to 1) ground truth reprojection errors and 2) predicted cost values. Figure 3 illustrates the mean reprojection error for the top x% of features in each of these ordered lists for a variable x. The lower the area under a curve in this figure, the more accurate is the corresponding image feature evaluator in sorting features based on their reliability. The curve corresponding to the ground truth reprojection errors indicates the ideal scenario where all image features are sorted correctly. The baseline is also illustrated in the figure as the resultant mean reprojection error when the features are sorted randomly (mean error over 1000 trials) and it corresponds to the case when no introspection is available and all features are treated equally. As can be seen, using the IF significantly improves image feature reliability assessment.

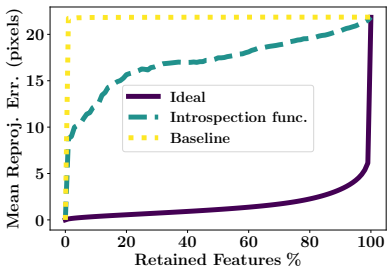


Figure 3: Mean reprojection error for image features sorted 1) randomly (baseline), 2) based on predicted reliability (Introspection func.), 3) based on ground truth reprojection error (ideal)

4.3 Tracking Accuracy and Tracking Failures

We compare our introspection-enabled version of ORB-SLAM with the original algorithm in terms of their camera pose tracking accuracy and robustness. 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 [35].

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 in the KITTI dataset. There exists no tracking failures and the overall relative translational error is less than 1%.

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

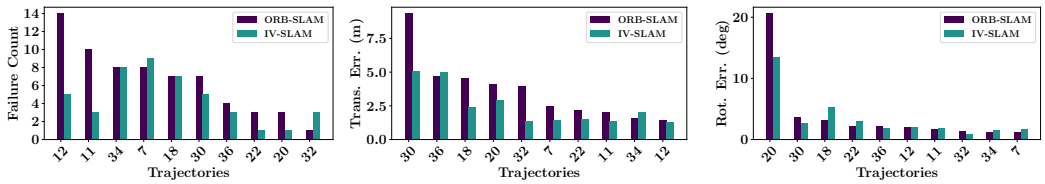
	Real-World			Simulation		
Method	Trans. Err. %	Rot. Err. (deg/m)	MDBF (m)	Trans. Err. %	Rot. Err. (deg/m)	MDBF (m)
IV-SLAM	**5.85**	**0.511**	**621.1**	**12.25**	**0.172**	**450.4**
ORB-SLAM	9.20	0.555	357.1	18.20	0.197	312.7

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.

Challenging datasets. We also evaluate IV-SLAM on the simulation and real-world datasets introduced in Section 4.1, which include scenes that are representative of the challenging scenarios that can happen in the real-world applications of V-SLAM. Given the larger scale of the environment, and faster speed of the robot in the simulation dataset, we pick $d_R = 2$m for the real-world environment and $d_S = 20$m for the simulation. Figures 4 and 5 compare the per trajectory tracking error values as well as the tracking failure count for IV-SLAM and ORB-SLAM in both experimental environments. 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 mean distance between failures (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.

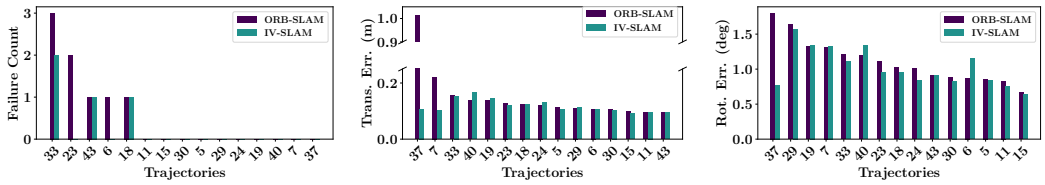
4.4 Qualitative Results

In order to better understand how IV-SLAM improves upon the underlying SLAM algorithm, and what it has learned to achieve the improved performance, we look at sample qualitative results. Figure 7 demonstrates example deployment sessions of the robot from the real-world dataset and compares the reference pose of the camera with the estimated trajectories by both algorithms under test. It shows how image features extracted from the shadow of the robot and surface reflections cause significant tracking errors for ORB-SLAM, while IV-SLAM successfully handles such challenging situations. Figure 6 illustrates further potential sources of failure picked up by IV-SLAM during inference, and demonstrates that the algorithm has learned to down-weight image features extracted from sources such as shadow of the robot, surface reflections, lens flare, and pedestrians in order to achieve improved performance.



(a) (b) (c)

Figure 4: Per trajectory comparison of the performance of IV-SLAM and ORB-SLAM in the simulation experiment. (a) Tracking failure count. (b) RMSE of translational error and (c) RMSE of rotational error over consecutive 20m-long horizons.



(a) (b) (c)

Figure 5: Per trajectory comparison of the performance of IV-SLAM and ORB-SLAM in the real-world experiment. (a) Tracking failure count. (b) RMSE of translational error and (c) RMSE of rotational error over consecutive 2m-long horizons.

5 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. IV-SLAM currently only uses static input images to predict the reliability of features. As future work, we would like to expand the architecture to exploit the temporal information across frames and also leverage the predictions of the introspection function in motion planning to reduce the probability of failures for V-SLAM.

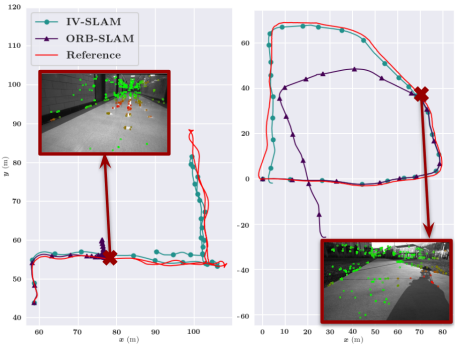


Figure 7: Example deployment sessions of the robot. IV-SLAM successfully follows the reference camera trajectory while ORB-SLAM leads to severe tracking errors caused by image features extracted on the shadow of the robot and surface reflections.

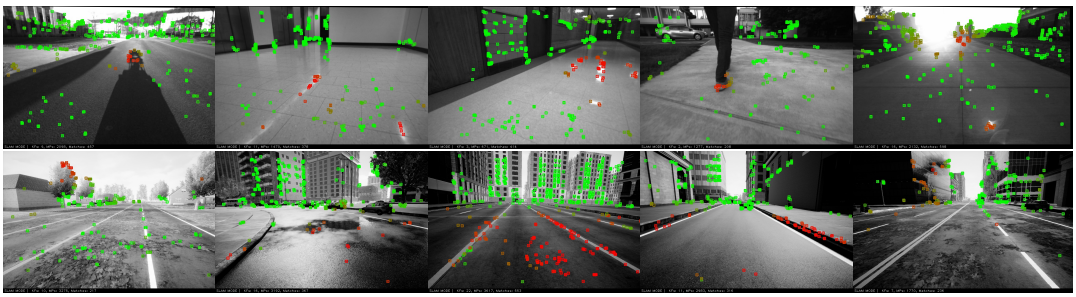


Figure 6: Snapshots of IV-SLAM running on real-world data (top row) and in simulation (bottom row). Green and red points on the image represent the reliable and unreliable tracked image features, respectively, as predicted by the introspection model. Detected sources of failure include shadow of the robot, surface reflections, pedestrians, glare, and ambiguous image features extracted from high-frequency textures such as asphalt or vegetation.

Acknowledgments

This work was partially supported by the Defense Advanced Research Projects Agency (DARPA) under Contract No. HR001120C0031, and by an unrestricted gift from Amazon. In addition, we acknowledge support from Northrop Grumman Mission Systems' University Research Program.

References

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

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

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

[4] A. R. Zamir, T. Wekel, P. Agrawal, C. Wei, J. Malik, and S. Savarese. Generic 3d representation via pose estimation and matching. In *European Conference on Computer Vision*, pages 535–553. Springer, 2016.

[5] Y. Ono, E. Trulls, P. Fua, and K. M. Yi. Lf-net: learning local features from images. In *Advances in neural information processing systems*, pages 6234–6244, 2018.

[6] Y. Tian, X. Yu, B. Fan, F. Wu, H. Heijnen, and V. 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] B. Triggs, P. F. McLauchlan, R. I. Hartley, and A. W. Fitzgibbon. Bundle adjustment–a modern synthesis. In *International workshop on vision algorithms*, pages 298–372. Springer, 1999.

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

[9] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[10] V. Balntas, E. Riba, D. Ponsa, and K. Mikolajczyk. Learning local feature descriptors with triplets and shallow convolutional neural networks. In *Bmvc*, volume 1, page 3, 2016.

[11] A. Mishchuk, D. Mishkin, F. Radenovic, and J. Matas. Working hard to know your neighbor's margins: Local descriptor learning loss. In *Advances in Neural Information Processing Systems*, pages 4826–4837, 2017.

[12] Y. Tian, B. Fan, and F. Wu. L2-net: Deep learning of discriminative patch descriptor in euclidean space. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 661–669, 2017.

[13] H. Bay, T. Tuytelaars, and L. Van Gool. SURF: Speeded up robust features. In *European conference on computer vision*, pages 404–417, 2006.

[14] K. Lenc and A. Vedaldi. Learning covariant feature detectors. In *European conference on computer vision*, pages 100–117, 2016.

[15] N. Savinov, A. Seki, L. Ladicky, T. Sattler, and M. Pollefeys. Quad-networks: unsupervised learning to rank for interest point detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1822–1830, 2017.

[16] T. Cieslewski, K. G. Derpanis, and D. Scaramuzza. Sips: Succinct interest points from unsupervised inlierness probability learning. In *2019 International Conference on 3D Vision (3DV)*, pages 604–613, 2019.

[17] M. Kaneko, K. Iwami, T. Ogawa, T. Yamasaki, and K. Aizawa. Mask-slam: Robust feature-based monocular slam by masking using semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 258–266, 2018.

[18] P. Ganti and S. L. Waslander. Network uncertainty informed semantic feature selection for visual slam. In *2019 16th Conference on Computer and Robot Vision (CRV)*, pages 121–128, 2019.

[19] S. Rabiee and J. Biswas. A friction-based kinematic model for skid-steer wheeled mobile robots. In *International Conference on Robotics and Automation (ICRA)*, pages 8563–8569, 2019.

[20] M. A. Fischler and R. C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24 (6):381–395, 1981.

[21] K. Moo Yi, E. Trulls, Y. Ono, V. Lepetit, M. Salzmann, and P. Fua. Learning to find good correspondences. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2666–2674, 2018.

[22] W. Sun, W. Jiang, E. Trulls, A. Tagliasacchi, and K. M. Yi. Acne: Attentive context normalization for robust permutation-equivariant learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11286–11295, 2020.

[23] S. Rabiee and J. Biswas. IVOA: Introspective Vision for Obstacle Avoidance. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1230–1235, 2019.

[24] S. Daftry, S. Zeng, J. A. Bagnell, and M. Hebert. Introspective perception: Learning to predict failures in vision systems. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1743–1750, 2016.

[25] M. J. Black and A. Rangarajan. On the unification of line processes, outlier rejection, and robust statistics with applications in early vision. *International journal of computer vision*, 19 (1):57–91, 1996.

[26] J. T. Barron. A general and adaptive robust loss function. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4331–4339, 2019.

[27] B. Zhou, H. Zhao, X. Puig, S. Fidler, A. Barriuso, and A. Torralba. Scene parsing through ade20k dataset. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 633–641, 2017.

[28] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4510–4520, 2018.

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

[30] 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*, 33(5):1255–1262, 2017.

[31] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3354–3361, 2012.

[32] 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*, 35 (10):1157–1163, 2016.

[33] X. Shi, D. Li, P. Zhao, Q. Tian, Y. Tian, Q. Long, C. Zhu, J. Song, F. Qiao, L. Song, et al. Are we ready for service robots? the openloris-scene datasets for lifelong slam. *arXiv preprint arXiv:1911.05603*, 2019.

[34] S. Shah, D. Dey, C. Lovett, and A. Kapoor. Airsim: High-fidelity visual and physical simulation for autonomous vehicles. In *Field and service robotics*, pages 621–635, 2018.

[35] D. Schubert, T. Goll, N. Demmel, V. Usenko, J. Stückler, and D. Cremers. The tum vi benchmark for evaluating visual-inertial odometry. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1680–1687, 2018.