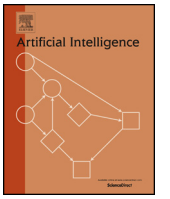
Artificial Intelligence 324 (2023) 103999

Contents lists available at ScienceDirect

Artificial Intelligence

journal homepage: www.elsevier.com/locate/artint



Introspective perception for mobile robots ☆

Sadegh Rabiee *, Joydeep Biswas

The Department of Computer Science, The University of Texas at Austin, Austin TX 78712, United States of America

A R T I C L E I N F O

Article history:
Received 15 April 2022
Received in revised form 29 June 2023
Accepted 26 August 2023
Available online 1 September 2023

Keywords:
Competence-aware perception
Computer vision
Introspection
Mobile robots

A B S T R A C T

Perception algorithms that provide estimates of their uncertainty are crucial to the development of autonomous robots that can operate in challenging and uncontrolled environments. Such perception algorithms provide the means for having risk-aware robots that reason about the probability of successfully completing a task when planning. There exist perception algorithms that come with models of their uncertainty; however, these models are often developed with assumptions, such as perfect data associations, that do not hold in the real world. Hence the resultant estimated uncertainty is a weak lower bound. To tackle this problem we present introspective perception — a novel approach for predicting accurate estimates of the uncertainty of perception algorithms deployed on mobile robots. By exploiting sensing redundancy and consistency constraints naturally present in the data collected by a mobile robot, introspective perception learns an empirical model of the error distribution of perception algorithms in the deployment environment and in an autonomously supervised manner. In this paper, we present the general theory of introspective perception and demonstrate successful implementations for two different perception tasks. We provide empirical results on challenging real-robot data for introspective stereo depth estimation and introspective visual simultaneous localization and mapping and show that they learn to predict their uncertainty with high accuracy and leverage this information to significantly reduce state estimation errors for an autonomous mobile robot.

1. Introduction

As robots become increasingly available, they are deployed for tasks where autonomous navigation in unstructured environments is required. Robots deployed in challenging and previously unseen environments are likely to encounter failures due to situations that were not previously considered by robot developers. One of the significant sources of failure in autonomy is errors in perception, hence, having perception algorithms that are capable of learning to predict the probability of errors in their output is of great importance. Being able to predict the uncertainty in the output of perception empowers a robot to 1) improve state estimation when fusing information from several imperfect perception algorithms, 2) generate plans that reduce the probability of task failures, and 3) request human supervision efficiently and only when it is actually required. Therefore, it is a key step towards having risk-aware robots, which can reason about the probability of successfully completing a task and act accordingly to reduce the probability of task failures.

☆ This paper is part of the Special Issue: Risk-aware Autonomous Systems: Theory and Practice.

* Corresponding author.
E-mail addresses: srabiee@cs.utexas.edu (S. Rabiee), joydeepb@cs.utexas.edu (J. Biswas).

https://doi.org/10.1016/j.artint.2023.103999
0004-3702/

There exists a body of work on deriving analytical solutions to the problem of uncertainty quantification for specific perception algorithms. The resultant solutions, however, often provide weak lower bounds for the uncertainty in the output of perception [1]. Solving perception problems using non-parametric implementations of Bayesian inference such as Gaussian Processes provides an estimate on the full distribution of the output of perception [2]; nevertheless, these approaches are computationally too expensive to solve complex real-world perception problems.

We present introspective perception, a novel approach for predicting accurate estimates of the uncertainty of perception algorithms deployed on mobile robots. Introspective perception leverages the redundancy in the data collected by robots and enforces spatio-temporal consistency constraints across different sensing modalities to evaluate the errors in the output of perception in the deployment environment and without human supervision. It then uses the autonomously labeled data to learn to predict the error distribution of the output of perception as a function of the raw sensory data. Unlike state-of-the-art uncertainty estimation methods, introspective perception continually improves its accuracy in the deployment environment in an autonomously supervised manner.

In this paper we present a general theory for introspective perception and demonstrate successful implementations for two different perception algorithms: visual simultaneous localization and mapping and stereo depth estimation. We present empirical results on real-robot data to demonstrate that introspective perception learns to accurately predict errors in the output of perception without requiring human labeled data and leverages this information to reduce state estimation failures for an autonomous mobile robot. This paper extends our prior works on developing introspection capabilities for specific perception algorithms [3,4]. In this work, we present a novel overarching theory for introspective perception that provides a guideline for how to enable arbitrary perception algorithms with introspection. Moreover, we for the first time implement and evaluate introspective perception for the case where the perception algorithm is an end-to-end machine learning algorithm.

2. Uncertainty in perception

Uncertainties in the output of perception stem from different sources such as shortcomings of the underlying models, partial observability, measurement noise, and etc. These uncertainties are usually classified in two main categories given their source: 1) epistemic uncertainty, and 2) aleatoric uncertainty.

2.1. Epistemic uncertainty

Epistemic uncertainty, also known as model uncertainty, is due to the inability of the model of perception to explain a phenomenon in the general case. This form of uncertainty happens when the sensory input is sampled from part of the observation space, where assumptions made by the perception model do not hold, or in the case of model-free perception, it happens when the input belongs to part of the observation space which is considered out of distribution (OOD) for the training data. Examples for this are the uncertainty of an stereo reconstruction algorithm in estimating the depth of texture-less objects, or the uncertainty of an image segmentation neural network that is only trained on outdoor scenes in segmenting an indoor image.

2.2. Aleatoric uncertainty

The second form of uncertainty, which is due to the noise in the data is known as aleatoric uncertainty. This type of uncertainty happens when there is noise in the sensory input or when there exists partial observability. Examples of this include the uncertainty of an object classifier in discerning different but similar looking types of flowers from each other or the uncertainty of a machine learned depth estimator in predicting the depth of a shadowed part of the scene.

3. Methods for estimating perception uncertainty

Quantifying uncertainty of the output of perception has long been an active area of research due to the great applications that it empowers. A perception algorithm that provides accurate confidence intervals or error bars for its output, enables the autonomous agent to do competence-aware planning [5], leverage this information to improve perception via active learning [6], or even detect adversarial attacks [7].

3.1. Uncertainty estimation for model-based perception

In model-based perception algorithms, it is common to use an unbiased estimator to determine the value of a set of desired parameters from a number of observations. This allows us to use the Cramer-Rao Lower Bound (CRLB) to estimate a lower-bound on the aleatoric uncertainty in the form of the covariance of the parameters of interest. This approach is common in various perception tasks such as simultaneous localization and mapping (SLAM) [1], calibration [8], and tracking [9]. The problem with CRLB is that it is usually a weak bound. Moreover, it is computed conditioned on accurate data association, and assumes accurate mathematical models of the observation likelihood functions. Invalidation of these assumptions in the real-world applications such as robotics reduce the accuracy of CRLB.

In order to also address the epistemic uncertainty due to the shortcomings of the model itself and achieve a tighter bound on the total uncertainty of perception, one approach is to leverage expert knowledge about the internals of a perception algorithm to create heuristic measures of its reliability. Examples of this include using the density of image features or the viewing angle of map points at every frame to calculate a hand-crafted measure of the reliability of visual SLAM algorithms [10,11]. Similar to any hand-crafted heuristic these measures of uncertainty fail to cover all potential sources of failure.

3.2. Uncertainty estimation for model-free perception

Using Bayesian inference for non-parametric and model-free perception is a principled way for quantifying the total uncertainty, i.e. the sum of aleatoric and epistemic uncertainty, of such perception algorithms. There exist exact and approximate Bayesian inference methods.

3.2.1. Bayesian inference

One of the classical examples of Bayesian inference is using Gaussian Processes (GP) for regression or classification tasks [12,2]. GP regression (GPR) is a specific form of Bayesian inference where the prior is a multivariate Gaussian distribution and the goal is to calculate the probability distribution over the space of functions that fit the training data as opposed to the probability distribution of parameters of a specific model. Using the Gaussian process prior and a Gaussian likelihood allows us to analytically derive a closed form solution to the posterior distribution in the Bayes formulation. However, the complexity of computing the posterior is still cubic with respect to the number of training data. In order to improve the scalability, dimensionality reduction techniques are applied to the high dimensional input data and only a subset of the training data is used to compute the predictive posterior distribution [13]. This in turn limits the applicability of such methods for complex perception problems with high dimensional input data.

Deep learning models have shown great success in real-world perception tasks due to their power in automatic feature extraction from high-dimensional data. However, these methods do not come with inherent uncertainty estimation capabilities. Recently Bayesian neural networks (BNN) [14] were introduced as a method to realize Bayesian inference with these models. Compared to the standard NNs where each parameter in the network is a single point estimate, in BNN each parameter is represented as a probability distribution from which we draw samples, and these distributions evolve away from their prior during training. The output of a BNN is a probability distribution that includes information about the total uncertainty. It has been proven that a single-layer infinitely wide BNN is equivalent to a Gaussian process [15]. While promising results have been achieved using BNNs on small problems [16], the main issue with this method, which has prevented it from being used in real-world perception applications is that it is slow to train and difficult to scale.

While obtaining estimates of the full probability distribution of outputs given inputs and model parameters is ideal, implementation of Bayesian inference for complex real-world robotic applications is intractable. Both training and inference on such models are computationally expensive. An alternative approach is using approximation methods for Bayesian inference.

3.2.2. Approximate Bayesian inference

In order to address the computational cost of BNN, approximation methods have been proposed. Gal and Ghahramani [17] showed that using Monte Carlo (MC) dropouts in a NN during inference is a valid approximation of a BNN. In this approach, some layers of a deterministic NN are equipped with dropout units that are used during inference as well as training and their job is to drop a node conditioned on the outcome of sampling from a Bernouli distribution. Each input is passed through the network several times, and the output of all these different trials are used to represent an approximate distribution of the output. Using dropouts during inference can be interpreted as averaging many models with shared weights, where each model is equally weighted. This is different from BNN, where each model is weighted taking into account priors and how well the model fits data.

Averaging the output of several models is also the main idea behind ensemble techniques for estimating the uncertainty of machine learned models. Ensemble methods have been applied to both classical models such as SVMs [18] as well as more recently for deep learning [19]. In the case of deep ensembles the same NN architecture is re-trained several times with different random seeds and the output of the collection of these models for the same input is used to approximate the uncertainty of the output. This technique relies on different initializations of the network parameters as well as the noise in stochastic gradient descent (SGD) for different instances of the same model to converge to different local optima after training. It has been shown that the approach can capture both aleatoric and epistemic uncertainty to some extent [20]. The problem with both MC dropout and ensemble methods is that their run-time and required memory linearly increase with respect to the number of models in the ensemble or the number of drawn samples. This is specifically more pronounced with large perception models that tackle complex real-world problems.

3.3. Out-of-distribution and anomaly detection

There exists a large body of work on out-of-distribution (OOD) detection, where the goal is to detect whether a given input is from the same distribution as the training data. Some of the popular methods for addressing this problem are generative adversarial networks (GANs) [21], autoencoders (AE) [22], and variational autoencoders (VAEs) [23,24]. The common

idea behind these methods is to learn an embedding function that maps the data from the training set to a hyperspace, such that mappings of new OOD data points fall outside of this hyperspace. In the case of AEs, an example input is detected to be OOD if the reconstruction error is high. In the case of VAEs, where the latent space is assumed to be Gaussian, a novelty score is computed as the KL divergence between the latent space distribution of the input and the prior distribution.

While these methods are effective in detecting OOD data, which can potentially lead to high epistemic uncertainty, they cannot capture aleatoric uncertainty of perception. Nevertheless, making accurate predictions of the aleatoric uncertainty of perception and accounting for that in the planning is especially important since this type of uncertainty cannot be reduced by improving the perception function.

3.4. Formal verification techniques for runtime monitoring of perception systems

There is a body of research focused on utilizing formal verification techniques [25,26] to analyze the behavior and ensure the safety of mobile robots at runtime. These techniques use formal specifications, such as temporal logic [27] to define desired properties of the perception algorithms. By monitoring the outputs of the perception algorithms and comparing them with the formal specifications, deviations and anomalies can be detected. Formal verification techniques often rely on high-level abstractions of the perception algorithms [28] and require accurate specifications that align with the specific characteristics of the perception algorithms being tested [29]. Defining these specifications can be difficult and cumbersome in practice, especially for complex perception algorithms.

3.5. Learning to predict perception errors

The available analytical solutions for uncertainty quantification in the case of model-based perception algorithms, e.g. information-theoretic metrics such as the Fisher information metric to compute lower bounds for the uncertainty of perception [1], are conditioned on accurate data association, hence provide overconfident estimates of perception uncertainty in real-world applications. Also, for end-to-end machine learning perception algorithms, the available uncertainty estimation methods are either computationally expensive or are application-specific and require modifications to the architecture of the network if applied to off-the-shelf perception models. A different approach that has recently gained interest includes learning a secondary machine-learned model to predict the probability of failure of the perception algorithm at hand. This technique, referred to as introspective perception, aims to provide a measure of reliability for the perception algorithm. The failure prediction model does not require any information about the underlying details of the perception algorithm. Instead, it relies on raw sensory data to predict the probability of failure in perception. Zhang et al. [30] use labeled data and train a binary classifier, which given an input image predicts the success or failure of a vision system. They test their method for the two different tasks of image classification and image segmentation. Daftry et al. [31] train a convolutional neural network (CNN) that uses both still images and optical flow frames to predict the probability of failure for obstacle detection on an actual UAV. Gurău et al. [32] develop an environment-specific failure prediction model that uses the raw sensory data as well as the location information of the robot to predict the probability of failure of a perception algorithm given examples of its prior failures in the same environment. This body of work solves the binary classification problem of failure prediction as opposed to approximating the distribution of perception errors. Moreover, the aforementioned works require the availability of sufficient labeled data of instances of perception failures for training prior to deployment. Vega-Brown et al. [33] propose a method for learning to predict the covariance of state estimation errors given hand-engineered features extracted from the raw sensory data. However, they do not address the problem of collecting the training data for the covariance estimation model.

4. Introspective perception

We present introspective perception (IPr), a method to empirically learn the error distribution of the output of perception in the deployment environment of a robot. Introspective perception provides the means to accurately estimate the uncertainty in the output of perception algorithms in a computationally efficient manner. IPr applies to arbitrary perception algorithms and learns the distribution of error in their output as a function of the raw sensory data. The key property of IPr that makes the learning of the error distribution scalable is that it is autonomously supervised. Inspired by prior works that leverage the redundancy of information on robots to perform self-calibration [34,35], IPr also exploits this resource, but to learn to predict the uncertainty of perception. It collects the training data required for learning the error distribution by enforcing consistency constraints on the outputs of perception algorithms. This allows IPr to leverage the abundance of data logged by mobile robots in the deployment environment to improve the accuracy of its estimate of the uncertainty of perception. In this section we present the mathematical definition of introspective perception and explain the autonomously supervised training techniques used by IPr.

4.1. Learning to predict errors of perception

In order to motivate our work, we formulate a perception algorithm as a function that produces estimates of a subset of the world states given a set of raw sensory observations as well as prior estimates of a subset of the world states

Table 1
Table of notations.

Notation	Description
$\mathcal{F}(\cdot)$	Perception function
$\mathbf{W}$	Set of world states
$\mathbf{w}_t$	State of the world at time t
$\mathbf{X}^{\mathcal{F}}$	Set of world states estimated by $\mathcal{F}$
$\mathbf{x}_t^{\mathcal{F}}$	Ground truth state at time t of states estimated by $\mathcal{F}$
$\hat{\mathbf{x}}_t^{\mathcal{F}}$	The estimates from $\mathcal{F}$ at time t
$\mathbf{Y}^{\mathcal{F}}$	Set of world states required by $\mathcal{F}$
$\mathbf{y}_t^{\mathcal{F}}$	Ground truth state at time t of states required by $\mathcal{F}$
$\hat{\mathbf{y}}_t^{\mathcal{F}}$	The estimates at time t of states required by $\mathcal{F}$
$\mathbf{Z}$	Set of all observations (sensor readings) by the robot
$\mathbf{Z}^{\mathcal{F}}$	Set of all observations processed by $\mathcal{F}$
$\mathbf{z}_t^{\mathcal{F}}$	Observations at time t processed by $\mathcal{F}$
$\phi^{\mathcal{F}}$	PDF of the error distribution of the output of F
$H_i^j(\cdot)$	Transformation of the world state from time i to time j.

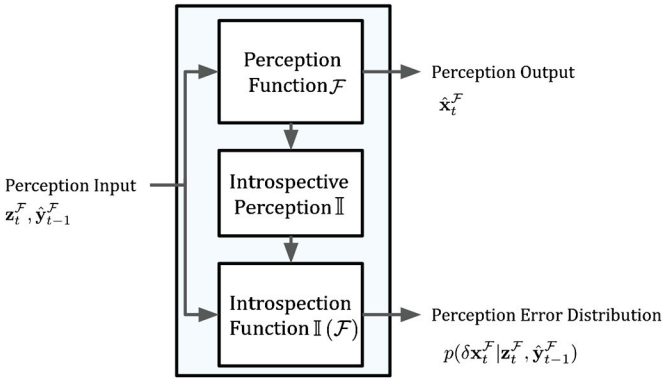


Fig. 1. Architecture of introspective perception. Introspective perception is a higher-order function $\mathbb{I}$ that when applied to a perception function $\mathcal{F}$, produces a function that predicts the error distribution of $\mathcal{F}$ conditioned on the inputs.

$$\mathcal{F}(\cdot) : \mathbf{Z}^{\mathcal{F}} \times \mathbf{Y}^{\mathcal{F}} \rightarrow \mathbf{X}^{\mathcal{F}} \tag{1}$$

$$\hat{\mathbf{x}}_t^{\mathcal{F}} = \mathcal{F}\left(\mathbf{z}_t^{\mathcal{F}}, \hat{\mathbf{y}}_{t-1}^{\mathcal{F}}\right) = \mathcal{F}\left(\mathbf{z}_t^{\mathcal{F}}, \left\langle \hat{\mathbf{x}}_{t-1}^{\mathcal{F}_0}, \hat{\mathbf{x}}_{t-1}^{\mathcal{F}_1}, ..., \hat{\mathbf{x}}_{t-1}^{\mathcal{F}_k} \right\rangle\right) \tag{2}$$

$$\hat{\mathbf{x}}_t^{\mathcal{F}} \in \mathbf{X}^{\mathcal{F}} \subseteq \mathbf{W} \tag{3}$$

where $\mathbf{z}_t^{\mathcal{F}}$ are raw sensor measurements while $\hat{\mathbf{y}}_t^{\mathcal{F}}$ are the intermediate state factors that are the result of processing the raw sensory data by other perception algorithms $\mathcal{F}_0, ..., \mathcal{F}_k$. Table 1 lists the description of all the parameters and variables used in the problem formulation. Consider the example of a human tracking algorithm $\mathcal{F}$. Then at any time-step t, $\mathbf{z}_t$ is the image captured by an RGB camera and $\hat{\mathbf{y}}_{t-1}^{\mathcal{F}}$ is a tuple of the bounding boxes of pedestrians in the image as detected by an object classifier $\mathcal{F}'$ along with the previous time-step estimate of the 3D pose of the pedestrians $\hat{\mathbf{x}}_{t-1}^{\mathcal{F}}$ by $\mathcal{F}$. The human tracking algorithm $\mathcal{F}$ would use $\mathbf{z}_t$ and $\hat{\mathbf{y}}_{t-1}^{\mathcal{F}}$ to estimate the pose of the pedestrians at the current time step $\hat{\mathbf{x}}_t^{\mathcal{F}}$.

The perception error can then be written as

$$\hat{\mathbf{x}}_t^{\mathcal{F}} = \mathbf{x}_t^{\mathcal{F}} \oplus \delta\mathbf{x}_t^{\mathcal{F}}, \qquad \delta\mathbf{x}_t^{\mathcal{F}} \sim \phi^{\mathcal{F}}\left(\mathbf{z}_t^{\mathcal{F}}, \mathbf{y}_{t-1}^{\mathcal{F}}\right). \tag{4}$$

Note that $\phi^{\mathcal{F}}$ is non-stationary and depends on the observations and the state of the agent. The fact that the perception error distribution depends on factors other than only the parameters of the perception function makes it challenging to derive analytical noise models that accurately estimate the uncertainty of the perception output. To address this, introspective perception learns an empirical approximation of the error distribution.

We define introspective perception as a higher-order function $\mathbb{I}$ that when applied to a perception function $\mathcal{F}$, produces a function that predicts the error distribution of $\mathcal{F}$ for given inputs

$$\hat{\phi}^{\mathcal{F}} = \mathbb{I}(\mathcal{F})\left[\mathbf{z}_t^{\mathcal{F}}, \hat{\mathbf{y}}_{t-1}^{\mathcal{F}}\right], \tag{5}$$

where the resultant function $\mathbb{I}(\mathcal{F})[\cdot] : \mathbf{Z}^{\mathcal{F}} \times \mathbf{Y}^{\mathcal{F}} \rightarrow \mathbf{X}^{\mathcal{F}}$ is the *introspection function*. Fig. 1 illustrates the architecture of introspective perception. It should be noted that not all perception functions necessarily include the intermediate state factors $\hat{\mathbf{y}}^{\mathcal{F}}$ as their input. In this work, we adopt a more general definition to demonstrate the applicability of introspective

perception across a broader range of perception functions, but $\hat{\mathbf{y}}^{\mathcal{F}}$ can be omitted from the formulations for the specific case of perception functions that solely rely on raw sensor data $\mathbf{z}_t^{\mathcal{F}}$ as their input.

4.2. Autonomously supervised training of introspective perception

In order to learn the introspection function $\mathbb{I}(\mathcal{F})[\cdot]$, we need training data in the form of recorded errors of the output of $\mathcal{F}$ for various input values of $\mathbf{z}_t^{\mathcal{F}}$ and $\hat{\mathbf{y}}_{t-1}^{\mathcal{F}}$. Manual labeling of the required training data is cumbersome and the resultant learned introspection function will only be accurate in environments for which we have labeled data. Mobile robots, on the other hand, collect a great amount of raw data during deployment. The collected data, although unlabeled, includes redundancy across different sensing modalities and across different time steps. We leverage the naturally available redundancy in data to generate the required labeled data autonomously. Specifically, we propose the following two consistency constraints to autonomously generate the training data for the introspection function: 1) consistency across sensors, and 2) spatio-temporal consistency.

Consistency across sensors. Let $\mathbf{x}_t^{\mathcal{F}} \approx \hat{\mathbf{x}}_t^{\mathcal{F}'}$, where $\mathcal{F}'$ is a supervisory perception function that consumes observations $\mathbf{z}_t^{\mathcal{F}'}$ from high-fidelity supervisory sensors. Then

$$\hat{\mathbf{x}}_t^{\mathcal{F}} \ominus \hat{\mathbf{x}}_t^{\mathcal{F}'} \sim \phi^{\mathcal{F}}\left(\mathbf{z}_t^{\mathcal{F}}, \mathbf{y}_{t-1}^{\mathcal{F}}\right) \quad . \tag{6}$$

In other terms, we can autonomously collect samples of errors of a perception function $\mathcal{F}$ if we have access to a high-fidelity supervisory sensor on our robot that can be used by another perception function $\mathcal{F}'$ to provide accurate estimates of the world state. When deploying a fleet of robots in a real-world environment, a cost-effective way to achieve this is to equip only some of them with additional high-fidelity sensors. For example, most of the robots in the fleet might use stereo RGB cameras for depth estimation while a few of them can additionally be equipped with more expensive long-range and high-resolution Lidars to obtain high-fidelity depth estimations of the environment and use them to provide supervision for passive depth estimation using stereo RGB cameras. Moreover, the supervisory perception function $\mathcal{F}'$ may be computationally expensive and not executable in real time; however, it can be used offline for supervision. For example, $\mathcal{F}'$ can be a structure from motion (SfM) algorithm [36] for dense 3D reconstruction of the environment or a depth estimator that is based on Neural Radiance Fields (NeRF) [37]. These algorithms can be used offline to process the recorded data by the robot and provide supervision for the primary and real-time perception algorithm $\mathcal{F}$.

Spatio-temporal consistency. By leveraging spatio-temporal constraints, we can relate state variables at different time-steps such that

$$H_i^j(\cdot) : \mathbf{X}^{\mathcal{F}} \rightarrow \mathbf{X}^{\mathcal{F}} \tag{7}$$

$$H_i^j\left(\mathbf{x}_i^{\mathcal{F}}\right) = \mathbf{x}_j^{\mathcal{F}} \tag{8}$$

$$\hat{\mathbf{x}}_t^{\mathcal{F}} = H_{t-\delta t}^t\left(\mathbf{x}_{t-\delta t}^{\mathcal{F}}\right) \oplus \delta \mathbf{x}_t^{\mathcal{F}} \quad , \tag{9}$$

where $H_i^j(\cdot)$ is the transformation function converting the world state from time i to time j. For an incremental perception algorithm, it is reasonable to assume that the posterior estimate of a state variable $\hat{\mathbf{x}}_{t-\delta t}^{\mathcal{F}^t}$, i.e. the estimate of $\hat{\mathbf{x}}_{t-\delta t}^{\mathcal{F}}$ made at time t improves in accuracy as δt increases. Hence, assuming $\mathbf{x}_{t-\delta t}^{\mathcal{F}} \approx \hat{\mathbf{x}}_{t-\delta t}^{\mathcal{F}^t}$, we have

$$\hat{\mathbf{x}}_t^{\mathcal{F}} \approx H_{t-\delta t}^t\left(\hat{\mathbf{x}}_{t-\delta t}^{\mathcal{F}^t}\right) \oplus \delta \mathbf{x}_t^{\mathcal{F}} \tag{10}$$

$$\Rightarrow \hat{\mathbf{x}}_t^{\mathcal{F}} \ominus H_{t-\delta t}^t\left(\hat{\mathbf{x}}_{t-\delta t}^{\mathcal{F}^t}\right) \sim \phi^{\mathcal{F}}\left(\mathbf{z}_t^{\mathcal{F}}, \mathbf{y}_{t-1}^{\mathcal{F}}\right) \tag{11}$$

Training. Consistency across sensors and spatio-temporal consistency as shown in Eq. (6) and Eq. (11) provide the means for generating samples of perception errors without the need for manual inspection and labeling of the data. Let $\Psi^{\mathcal{F}} = \{\psi_1^{\mathcal{F}}, \psi_2^{\mathcal{F}}, ..., \psi_N^{\mathcal{F}}\}$ be the empirical distribution of errors of $\mathcal{F}$, where $\psi_i^{\mathcal{F}} \sim \phi^{\mathcal{F}}$ are samples of errors of the output of $\mathcal{F}$ that are obtained using spatio-temporal consistency constraints as well as consistency across sensors. IPr learns an approximation $\mathbb{I}_\theta$ of the introspection function such that

$$\theta^* = \arg\min_\theta \mathrm{KLD}\left(\mathbb{I}_\theta(\mathcal{F})\left[\mathbf{z}_t^{\mathcal{F}}, \hat{\mathbf{y}}^{\mathcal{F}}\right], \Psi^{\mathcal{F}}\right) \quad , \tag{12}$$

where KLD is the Kullback-Leibler divergence.

The introspection function $\mathbb{I}_\theta(\mathcal{F})$ can accurately predict the error distribution of $\mathcal{F}$ when its input data is from the same distribution as the training data of $\mathbb{I}_\theta(\mathcal{F})$, i.e. $\Psi^{\mathcal{F}}$. While similar to any machine learning model, $\mathbb{I}_\theta(\mathcal{F})$ is prone to errors when its input data is out of distribution (OOD) of the training data, the autonomously supervised training of introspective

perception ensures that the occurrence of OOD observations is minimized. IPr collects training data from all deployment environments of the robots at a low cost and without a human in the loop.

It should be noted that the solution to the optimization problem in Eq. (12) changes if modifications are made to the perception function $\mathcal{F}$, e.g. if the perception function is re-trained on new data. As a result, the introspection function needs to be re-trained when the perception function is updated. However, a reasonable assumption is that incremental updates to the perception function do not change the error characteristics of the perception function significantly; therefore, the introspection function can be fine-tuned from its latest state as opposed to re-trained from scratch, which in turn reduces the computational overhead of updating the introspection function.

4.3. Design and implementation of the introspection function

While in theory, the introspection function can be implemented as any function approximator, we use deep learning models due to their strength in learning generalizable representations of their input data. The choice of network architecture can vary for different perception tasks. A good network architecture depends on the type of sensor data consumed by the perception algorithm, the difficulty of the learning task, and the size of the training data set that can be collected using the available consistency constraints. In the design and implementation of the network architecture for each new perception task, we take into account three main points. 1) We use encoder architectures that have been shown to successfully learn representations of the type of sensor data consumed by the perception function. For instance, convolutional neural networks are a great choice for learning representations of image data while recurrent neural networks are a great choice for learning representations of sequential data such as IMU readings. 2) We keep the network architecture simple and its number of parameters small to reduce the chance of overfitting the training data and to ensure that the network can be trained efficiently and can be deployed in real time. 3) We parametrize the error distribution $\phi^{\mathcal{F}}$ which is the output of the introspection function such that it simplifies the learning task. For instance, instead of having the introspection function estimate the distribution of the error vectors, we task the introspection function to estimate the distribution of the magnitude of the error vectors. We introduce example architectures for different perception tasks in sections §7 and §8.

5. Improved robustness in autonomy via introspective perception

Equipping perception algorithms with introspection improves robustness and reliability of an autonomous robot in two main ways. First, it improves accuracy of perception algorithms by producing more accurate uncertainty estimates for intermediate state variables that are used by high-level perception algorithms. Second, uncertainty estimates produced by introspective perception, can be leveraged in planning to prevent task-level failures. In this section we explain each of these two benefits of introspective perception in detail.

5.1. Reducing errors of perception

Perception algorithms in the general case keep a belief over state variables of interest

$$\text{bel}\left(\mathbf{x}_t^{\mathcal{F}}\right) = p(\mathbf{x}_t^{\mathcal{F}}|\mathbf{z}_t^{\mathcal{F}}, \hat{\mathbf{y}}_{t-1}^{\mathcal{F}}) \propto p(\hat{\mathbf{y}}_{t-1}^{\mathcal{F}}|\mathbf{x}_t^{\mathcal{F}}, \mathbf{z}_t^{\mathcal{F}}) p(\mathbf{z}_t^{\mathcal{F}}|\mathbf{x}_t^{\mathcal{F}}) p(\mathbf{x}_t^{\mathcal{F}}) \tag{13}$$

$$\propto p(\hat{\mathbf{y}}_{t-1}^{\mathcal{F}}|\mathbf{x}_t^{\mathcal{F}}) p(\mathbf{z}_t^{\mathcal{F}}|\mathbf{x}_t^{\mathcal{F}}) p(\mathbf{x}_t^{\mathcal{F}}) \quad . \tag{14}$$

A large subset of perception algorithms, however, compute a maximum likelihood estimate (MLE) of the state variables

$$\hat{\mathbf{x}}_t^{\mathcal{F}} = \arg\max_{\mathbf{x}_t^{\mathcal{F}}} p(\hat{\mathbf{y}}_{t-1}^{\mathcal{F}}|\mathbf{x}_t^{\mathcal{F}}) p(\mathbf{z}_t^{\mathcal{F}}|\mathbf{x}_t^{\mathcal{F}}) \quad . \tag{15}$$

The way different perception algorithms solve this problem varies given the task at hand. There are low-level perception algorithms for which the there exist a unique solution to the above problem given the number of available observations and the degree of freedom of the problem. An example for this is depth estimation for associated pairs of image features in stereo images, which can be solved analytically. However, it is often the case that the problem solved by $\mathcal{F}$ is over-determined hence obtaining the output involves solving an optimization problem which minimizes a loss function $\mathcal{L}(\mathbf{x}_t^{\mathcal{F}}) = -\log p(\hat{\mathbf{y}}_{t-1}^{\mathcal{F}}|\mathbf{x}_t^{\mathcal{F}}) - \log p(\mathbf{z}_t^{\mathcal{F}}|\mathbf{x}_t^{\mathcal{F}})$. The term $p(\mathbf{z}_t^{\mathcal{F}}|\mathbf{x}_t^{\mathcal{F}})$ captures the noise distribution of the sensor readings and is obtained via sensor-specific calibration processes. An example for this is the precision of LiDAR range measurements given the surface material of obstacles or the accuracy of sonar range finders given the humidity of the environment. The conditional probability $p(\hat{\mathbf{y}}_{t-1}^{\mathcal{F}}|\mathbf{x}_t^{\mathcal{F}})$ captures both the epistemic uncertainty in the perception algorithm $\mathcal{F}'$ that estimates $\hat{\mathbf{y}}_{t-1}^{\mathcal{F}}$ as well as the aleatoric uncertainty in the data processed by $\mathcal{F}'$. In the absence of an accurate estimate of $p(\hat{\mathbf{y}}_{t-1}^{\mathcal{F}}|\mathbf{x}_t^{\mathcal{F}})$, it is often approximated to be static and from one of the known families of distributions, e.g. Gaussian. However, this is a poor approximation of the true distribution of the error which is often heteroscedastic.

Note that $\hat{\mathbf{y}}^{\mathcal{F}}$ is itself the output of another perception algorithm $\mathcal{F}'$, i.e. $\hat{\mathbf{y}}_t^{\mathcal{F}} = \hat{\mathbf{x}}_t^{\mathcal{F}'}$. Therefore, if we equip $\mathcal{F}'$ with introspection such that $\hat{\phi}^{\mathcal{F}'} = \mathbb{I}\left(\mathcal{F}'\right)\left[\mathbf{z}_t^{\mathcal{F}'}, \hat{\mathbf{y}}_{t-1}^{\mathcal{F}'}\right]$, we can provide $\mathcal{F}$ with the $\hat{\mathbf{y}}^{\mathcal{F}}$ error distribution as predicted by the $\mathcal{F}'$

introspection function. Leveraging the resultant more accurate approximation of $p(\hat{\mathbf{y}}_{t-1}^{\mathcal{F}}|\mathbf{x}_t^{\mathcal{F}})$ in the loss function $\mathcal{L}(\mathbf{x}_t^{\mathcal{F}})$ leads to a more accurate estimate of $\mathbf{x}_t^{\mathcal{F}}$. The same technique also applies to perception algorithms that output a belief over the state variables, e.g. particle filter-based localization or tracking. Using a learned and more accurate approximation of the observation likelihood $p(\hat{\mathbf{y}}_{t-1}^{\mathcal{F}}|\mathbf{x}_t^{\mathcal{F}})$ leads to a better estimate of bel$\left(\mathbf{x}_t^{\mathcal{F}}\right)$.

5.2. Preventing task-level failures

The approximation of distribution of error for the state variables, provided by introspective perception, can also be leveraged in planning to prevent task-level failures that arise from errors in perception, e.g. collision with obstacles due to depth estimation or localization errors.

There exist several planning methods that take into account the uncertainty in the state of the agent when planning. Partially Observable Markov Decision Processes (POMDP) provide the means for computing an optimal policy in the belief space, i.e. when the agent keeps a distribution of states [38]. POMDPs use an observation likelihood function to update their belief over the state of the world given new observations, hence, a more accurate approximation of the observation likelihood function helps improve their performance. The main down-side of POMDPs is their high computational cost. More tractable solutions include set-bounded uncertainty modeling, where it is assumed that a bound on the magnitude of perception and actuation error is known and then planning is limited to a subset of the world states where safety of the plan is guaranteed as long as the error values are within the assumed bounds [39,40]. Chance-constrained optimization methods relax the assumption of requiring bounds on the magnitude of perception and actuation errors, and leverage a known error distribution to solve the planning problem subject to constraints on the probability of failures [41–43]. While the above approaches consume continuous perception error probability distributions, there also exist learning-based risk-aware planning algorithms that require only discrete and sparse histogram representations of the error distributions of low-level perception to predict the probability of failure at the task level, e.g. autonomous navigation [44].

The effectiveness of all above planning methods depends on having accurate models of the uncertainty of perception, and introspective perception provides the means to achieve this in the deployment environment of the robot and in a computationally efficient manner.

6. Introspective perception vs. approximate Bayesian inference

As explained in §3.2, sampling-based approximate Bayesian inference methods provide the means to estimate the uncertainty of a function approximator in a data-driven manner that is scalable in terms of the number of training data samples. In this section, we explain approximate Bayesian inference in detail and discuss how it compares to introspective perception.

For a perception function approximated by $\mathcal{F}_\theta(.)$, where θ is a random vector and the parameters of the function, approximate Bayesian inference generates a set of functions

$$\{\mathcal{F}_{\theta^{(i)}}()\}_{i=1}^{N} : \theta^{(i)} \sim \Theta \quad , \tag{16}$$

where $\theta^{(i)}$ is a sample from the distribution of parameters Θ. Then for each input pair $\left(\mathbf{z}_t^{\mathcal{F}}, \hat{\mathbf{y}}_{t-1}^{\mathcal{F}}\right)$, the output of all $\mathcal{F}_{\theta^{(i)}}$ are computed to generate an empirical distribution of the output of $\mathcal{F}_\theta$ at time t. For instance the expected value of the output is estimated as

$$\hat{\mathbf{x}}_t^{\mathcal{F}} = \frac{1}{N}\sum_{i=1}^{N} \mathcal{F}_{\theta^{(i)}}(\mathbf{z}_t^{\mathcal{F}}, \hat{\mathbf{y}}_{t-1}^{\mathcal{F}}) \quad . \tag{17}$$

Assuming $\mathcal{F}_\theta$ is an unbiased estimator, then $\hat{\phi}^{\mathcal{F}} = \{\mathcal{F}_{\theta^{(i)}}(\mathbf{z}_t^{\mathcal{F}}, \hat{\mathbf{y}}_{t-1}^{\mathcal{F}}) - \hat{\mathbf{x}}_t^{\mathcal{F}}\}_{i=1}^{N}$ forms an empirical distribution approximating the true distribution of error of $\mathcal{F}_\theta$ at time t.

For the case when $\mathcal{F}_\theta()$ is implemented as a neural network, computationally tractable approximate Bayesian inference methods have been proposed. The most popular of these methods are the Monte Carlo Dropout (MCDropout) and deep ensembles. In the case of MCDropout, each $\mathcal{F}_{\theta^{(i)}}$ is a copy of the network with the resultant weights after a random sampling of dropout units in the network. In the ensemble approach, each $\mathcal{F}_{\theta^{(i)}}$ is the result of a different round of training the network with a different initialization of the weights and different sampling of the training data. In both approaches, the ensemble of $\mathcal{F}_{\theta^{(i)}}$ is treated as a uniformly-weighted mixture model when combining the predictions. In the case of the perception function being a classifier, combining the results would be equivalent to averaging the predicted probability values for each class. In the case of regression, it is common to treat the prediction as a mixture of Gaussian distributions such that $\mathcal{F}_{\theta^{(i)}}(\mathbf{z}_t^{\mathcal{F}}, \hat{\mathbf{y}}_{t-1}^{\mathcal{F}}) \sim \mathcal{N}(\mu_{\theta^{(i)}}\left(\mathbf{z}_t^{\mathcal{F}}, \hat{\mathbf{y}}_{t-1}^{\mathcal{F}}\right), \sigma^2)$, where σ^2 is the variance of the output of each model and defined by the user as a hyperparameter. The ensemble prediction is then approximated to be a Gaussian with its mean and variance given by

$$\mu_{\text{ens}}(\mathbf{z}_t^{\mathcal{F}}, \hat{\mathbf{y}}_{t-1}^{\mathcal{F}}) = \frac{1}{N}\sum_{i=1}^{N} \mu_{\theta^{(i)}}\left(\mathbf{z}_t^{\mathcal{F}}, \hat{\mathbf{y}}_{t-1}^{\mathcal{F}}\right) \tag{18}$$

$$\sigma^2_{\text{ens}}(\mathbf{z}_t^{\mathcal{F}}, \hat{\mathbf{y}}_{t-1}^{\mathcal{F}}) = \frac{1}{N}\sum_{i=1}^{N}\left(\sigma^2 + \mu^2_{\hat{\theta}^{(i)}}\left(\mathbf{z}_t^{\mathcal{F}}, \hat{\mathbf{y}}_{t-1}^{\mathcal{F}}\right)\right) - \mu^2_{\text{ens}}(\mathbf{z}_t^{\mathcal{F}}, \hat{\mathbf{y}}_{t-1}^{\mathcal{F}}) \quad . \tag{19}$$

Compared to IPr, approximate Bayesian neural networks (BNN) also use a machine learned function approximator to do uncertainty estimation. For BNNs, the machine learned component used for uncertainty estimation is the same as that used for the perception function. IPr, however, is agnostic of the underlying function approximator used for the perception function. The machine learned component that it uses for uncertainty estimation, i.e. the introspection function, is different from the perception function and specifically trained for estimating the error distribution of perception. Therefore, IPr applies to both model-based and fully-learned perception algorithms. Additionally, the modular architecture of IPr makes it computationally more efficient. While BNNs' usage of memory and inference time scales linearly with the number of Monte Carlo samples or the models in the ensemble, IPr requires a constant amount of memory to run the introspection function. The introspection function can be implemented with a significantly less complex function approximator than the perception function since the task tackled by the introspection function is simpler than the full perception problem. As a result, the computational requirements of IPr are significantly less than those of BNNs.

In §9.5, we discuss the results of using IPr for a fully-learned perception algorithm and compare its performance in terms of perception failure prediction accuracy and also its computational cost with approximate Bayesian inference methods.

7. Introspective visual simultaneous localization and mapping

In this section we present an implementation of introspective perception for Visual SLAM and demonstrate how spatio-temporal consistency constraints can be used to train an introspection function which in turn can be leveraged to reduce failures of V-SLAM.

7.1. Visual SLAM

In visual SLAM, the pose of the camera $\mathbf{T}_t^w \in SE(3)$ is estimated in the world reference frame w 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. The V-SLAM architecture, with the exception of the end-to-end learning approaches [45,46], consists of two main perception modules. The first module is the SLAM front-end $\mathcal{F}_{\text{SF}}$ that is responsible for feature extraction and matching across images and

$$\hat{\mathbf{x}}_t^{\mathcal{F}_{\text{SF}}} = \mathcal{F}_{\text{SF}}\left(\mathbf{z}_t^{\mathcal{F}_{\text{SF}}}, \hat{\mathbf{x}}_{t-1}^{\mathcal{F}_{\text{SF}}}\right), \tag{20}$$

where $\mathbf{z}_t^{\mathcal{F}_{\text{SF}}} = \mathbf{I}_t$ are the input images and $\hat{\mathbf{x}}_t^{\mathcal{F}_{\text{SF}}}$ are extracted and matched image features at time t. The second perception module is the SLAM back-end $\mathcal{F}_{\text{SB}}$ that estimates the pose of the camera given data correspondences provided by the front-end such that

$$\hat{\mathbf{x}}_t^{\mathcal{F}_{\text{SB}}} = \mathcal{F}_{\text{SB}}\left(\mathbf{z}_t^{\mathcal{F}_{\text{SB}}}, \hat{\mathbf{x}}_{1:t}^{\mathcal{F}_{\text{SF}}}\right), \tag{21}$$

where $\mathbf{z}_t^{\mathcal{F}_{\text{SB}}} = u_{1:t}$ are the history of odometry and IMU measurements and $\hat{\mathbf{x}}_t^{\mathcal{F}_{\text{SB}}} = \hat{\mathbf{T}}_{1:t}^w, \hat{\mathbf{M}}$. The SLAM back-end $\mathcal{F}_{\text{SB}}$ solves for the pose of the camera and the map of the environment via MLE such that

$$\begin{aligned}\hat{\mathbf{T}}_{1:t}^w, \hat{\mathbf{M}} &= \underset{\mathbf{T}_{1:t}^w, \mathbf{M}}{\arg\max}\, p\left(\mathbf{T}_{1:t}^w, \mathbf{M} | \hat{\mathbf{x}}_{1:t}^{\mathcal{F}_{\text{SF}}}, \mathbf{z}_t^{\mathcal{F}_{\text{SB}}}\right) \\ &= \underset{\mathbf{T}_{1:t}^w, \mathbf{M}}{\arg\max}\, p\left(\hat{\mathbf{x}}_{1:t}^{\mathcal{F}_{\text{SF}}} | \mathbf{T}_{1:t}^w, \mathbf{M}\right) p\left(\mathbf{T}_{1:t}^w | \mathbf{z}_t^{\mathcal{F}_{\text{SB}}}\right),\end{aligned} \tag{22}$$

where $p(\hat{\mathbf{x}}^{\mathcal{F}_{\text{SF}}} | \mathbf{T}_{1:t}^w, \mathbf{M})$ 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 $\hat{\mathbf{x}}_t^{\mathcal{F}_{\text{SF}}} = \left\{\hat{\mathbf{x}}_{t,k}^{\mathcal{F}_{\text{SF}}} | \hat{\mathbf{x}}_{t,k}^{\mathcal{F}_{\text{SF}}} \in \mathbb{P}^2, k \in [1, N]\right\}$, where $\hat{\mathbf{x}}_{t,k}^{\mathcal{F}_{\text{SF}}}$ is an image feature associated with the 3D map point $\mathbf{p}_k^w$. The observation error here is the reprojection error of $\mathbf{p}_k^w$ onto the image $\mathbf{I}_t$ and is defined as

$$\delta\mathbf{x}_{t,k}^{\mathcal{F}_{\text{SF}}} = \hat{\mathbf{x}}_{t,k}^{\mathcal{F}_{\text{SF}}} - \mathbf{x}_{t,k}^{\mathcal{F}_{\text{SF}}}, \quad \mathbf{x}_{t,k}^{\mathcal{F}_{\text{SF}}} = \mathbf{A}\left[\mathbf{R}_w^t | \mathbf{t}_w^t\right]\mathbf{p}_k^w, \tag{23}$$

where $\mathbf{A}$ is the camera matrix, and $\mathbf{R}_w^t \in SO(3)$ and $\mathbf{t}_w^t \in \mathbb{R}^3$ are the rotation and translation parts of $\mathbf{T}_w^t = \left(\mathbf{T}_t^w\right)^{-1}$, respectively. In the absence of control commands and IMU/odometry measurements, Eq. (22) reduces to the bundle adjustment (BA) problem, which is formulated as a nonlinear optimization:

$$\hat{\mathbf{T}}_{1:t}^w, \hat{\mathbf{M}} = \underset{\mathbf{T}_{1:t}^w, \mathbf{M}}{\arg\min} \sum_{t,k} \mathcal{L}\left(\epsilon_{t,k}^T \Sigma_{t,k}^{-1} \epsilon_{t,k}\right), \tag{24}$$

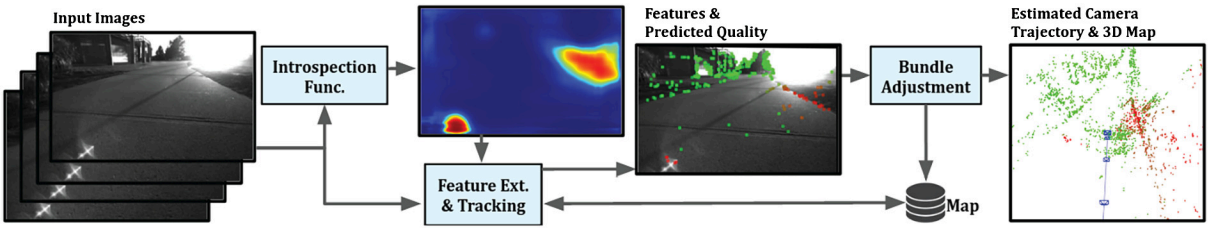


Fig. 2. IV-SLAM pipeline during inference.

where $\Sigma_{t,k}$ is the covariance matrix associated with the scale at which a feature has been extracted, $\epsilon_{t,k} = \delta \mathbf{x}_{t,k}^{\mathcal{F}_{\text{SF}}}$, and $\mathcal{L}$ is the loss function for $p(\hat{\mathbf{x}}^{\mathcal{F}_{\text{SF}}} | \mathbf{T}_{1:t}^{w}, \mathbf{M})$.

The choice of noise model for the observation error $\delta \mathbf{x}_{t,k}^{\mathcal{F}_{\text{SF}}}$ has a significant effect on the accuracy of the maximum likelihood solution to SLAM [47]. There exists a body of work on developing robust loss functions [48,49] that targets improving the performance of vision tasks in the presence of outliers. The reprojection error is known to have a non-Gaussian distribution $\phi^{\mathcal{F}_{\text{SF}}}$ in the context of V-SLAM due to the frequent outliers that happen in image correspondences [47]. 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.

In addition to using robust loss functions, outlier rejection methods are used to detect wrong data associations and remove the corresponding observation error terms from the optimization problem in Eq. (24). Common outlier rejection methods in V-SLAM include RANSAC [50], detecting outliers based on the temporal consistency of data correspondences [51], and learning-based methods [52,53] which use permutation-equivariant network architectures and predict outlier correspondences by processing coordinates of the pairs of matched features. While these methods are effective in removing gross outliers, not all bad image feature matchings are clear outliers; there is an ambiguous 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 [54]. Moreover, outlier rejection methods work best when the majority of data correspondences have small errors, yet most catastrophic failures happen in challenging situations where the majority of data correspondences are unreliable. Furthermore, while the noise distribution $\phi^{\mathcal{F}_{\text{SF}}}$ is usually modeled to be static, there exist obvious reasons why this is not a realistic assumption. Image features extracted from objects with high-frequency surface textures can be located less accurately; whether the 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.

7.2. Improving V-SLAM by learning the re-projection error distribution

In order to equip V-SLAM with introspection, we apply introspective perception to $\mathcal{F}_{\text{SF}}$ as defined in Eq. (5) to learn an approximation of the re-projection error distribution

$$\hat{\phi}^{\mathcal{F}_{\text{SF}}} = \mathbb{I}\left(\mathcal{F}_{\text{SF}}\right)\left[\mathbf{z}_t^{\mathcal{F}_{\text{SF}}}\right] \quad . \tag{25}$$

The resultant introspection function estimates the re-projection error distribution, which we use to define the loss function $\mathcal{L}^{\phi^{\mathcal{F}_{\text{SF}}}} \propto -\log\left(\phi^{\mathcal{F}_{\text{SF}}}\right)$ used by $\mathcal{F}_{\text{SB}}$ to solve for the pose of the camera in Eq. (24). We select $\mathcal{L}^{\phi^{\mathcal{F}_{\text{SF}}}} \in \mathcal{H}$ where $\mathcal{H}$ is the space of Huber loss functions and specifically

$$\mathcal{L}^{\phi^{\mathcal{F}_{\text{SF}}}}_{\theta(\mathbf{x}_{t,k}^{\mathcal{F}_{\text{SF}}})}(x) = \begin{cases} x & \text{if } x < \theta(\mathbf{x}_{t,k}^{\mathcal{F}_{\text{SF}}}) \\ 2\theta(\mathbf{x}_{t,k}^{\mathcal{F}_{\text{SF}}})\left(\sqrt{x} - \theta(\mathbf{x}_{t,k}^{\mathcal{F}_{\text{SF}}})/2\right) & \text{otherwise} \end{cases} \tag{26}$$

where $x = \left\| \delta \mathbf{x}_{t,k}^{\mathcal{F}_{\text{SF}}} \right\|^2 \in [0, \infty)$ is the squared L^2 norm of the error and $\theta(\mathbf{x}_{t,k}^{\mathcal{F}_{\text{SF}}})$ is the parameter of the loss function and depends on the reliability of the image feature $\mathbf{x}_{t,k}^{\mathcal{F}_{\text{SF}}}$. Introspective V-SLAM (IV-SLAM) uses the introspection function to approximate $\theta(\mathbf{x}_{t,k}^{\mathcal{F}_{\text{SF}}})$ such that the corresponding error distribution $\hat{\phi}^{\mathcal{F}_{\text{SF}}}$ better models the observed error values. During the training phase, input images and estimated re-projection 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 $\theta(\mathbf{x}_{t,k}^{\mathcal{F}_{\text{SF}}})$ is estimated for each observed image feature using the predicted reliability score, where a smaller value of $\theta(\mathbf{x}_{t,k}^{\mathcal{F}_{\text{SF}}})$ corresponds to an unreliable image feature. The resultant loss function $\mathcal{L}^{\phi^{\mathcal{F}_{\text{SF}}}}_{\theta(\mathbf{x}_{t,k}^{\mathcal{F}_{\text{SF}}})}$ is then used in Eq. (24) to solve for $\mathbf{T}_{1:t}^{w}$ and $\mathbf{M}$. Fig. 2 illustrates the IV-SLAM pipeline during inference.

7.3. Autonomously supervised training of IV-SLAM

Introspection function architecture. We implement the IV-SLAM introspection function as $\mathbb{I}\left(\mathcal{F}_{\text{SF}}\right)\left[\mathbf{z}_t^{\mathcal{F}_{\text{SF}}}\right] = f \circ g(\mathbf{z}_t^{\mathcal{F}_{\text{SF}}})$, where $g : \mathbf{I} \rightarrow \mathbf{I}$ is a function that given an input image $\mathbf{z}_t^{\mathcal{F}_{\text{SF}}}$, outputs an image $\mathbf{I}_{\mathbf{c}t}$ where $\mathbf{I}_{\mathbf{c}t}(i,j) \in [0,1]$ is the prediction of the normalized reprojection error magnitude for image features extracted at $\mathbf{z}_t^{\mathcal{F}_{\text{SF}}}(i,j)$. Also $f : \mathbf{I} \rightarrow \mathbf{I}$ outputs $\mathbf{I}_{\theta t}$, where $\mathbf{I}_{\theta t}(i,j)$ is the loss function parameter θ in Eq. (26) for an image feature $\mathbf{x}_{t,k}^{\mathcal{F}_{\text{SF}}}$ that is extracted at $\mathbf{z}_t^{\mathcal{F}_{\text{SF}}}(i,j)$. We implement $g(\cdot)$ as a fully convolutional network with the same architecture as that used by Zhou et al. [55] for image segmentation. The network is composed of the MobileNetV2 [56] as the encoder and a transposed convolution layer with deep supervision as the decoder. We train the network using samples of the reprojection error collected in the deployment environment. Moreover, we have $f(\mathbf{I}_{\mathbf{c}t}(i,j)) = \frac{1-\mathbf{I}_{\mathbf{c}t}(i,j)}{1+\mathbf{I}_{\mathbf{c}t}(i,j)}\theta_{\max}$, where $\theta_{\max}$ is a constant positive hyperparameter that defines the range at which $\mathbf{I}_{\theta t}(i,j)$ varies. In this implementation instead of directly learning to predict $\theta(\mathbf{x}_{t,k}^{\mathcal{F}_{\text{SF}}})$, IV-SLAM learns to predict the reprojection error magnitude and then computes $\theta(\mathbf{x}_{t,k}^{\mathcal{F}_{\text{SF}}})$ such that the resultant loss function $\mathcal{L}_{\theta(\mathbf{x}_{t,k}^{\mathcal{F}_{\text{SF}}})}^{\phi^{\mathcal{F}_{\text{SF}}}}$ more accurately represents the corresponding error distribution. For each image feature, the Huber loss is adjusted such that the features that are predicted to be less reliable (larger $\mathbf{I}_{\mathbf{c}t}(i,j)$) have a less steep associated loss function (smaller $\theta(\mathbf{x}_{t,k}^{\mathcal{F}_{\text{SF}}})$).

Training data collection. We use spatio-temporal consistency constraints to collect error samples for $\mathcal{F}_{\text{SF}}$ in the deployment environment and then use the data to train $g(\cdot)$ to predict the reprojection error magnitude for every pixel on an input image. Matched image features are image-space projections of the same object in the world at different time steps. Therefore, they are geometrically related. Using the spatio-temporal consistency constraint definition in Eq. (8), we have

$$\mathbf{x}_{t,k}^{\mathcal{F}_{\text{SF}}} = H_{t-\delta t}^{t}\left(\mathbf{x}_{t-\delta t,k}^{\mathcal{F}_{\text{SF}}}\right) \tag{27}$$

$$= \Pi\left(\mathbf{T}_{t-\delta t}^{t}\Pi^{-1}\left(\mathbf{x}_{t-\delta t,k}^{\mathcal{F}_{\text{SF}}}, d_{t-\delta t,k}\right)\right), \tag{28}$$

where the transformation function $H_{t-\delta t}^{t}(\cdot)$ is composed of the camera projection function and the relative pose of the camera between the two time steps. In the above equation, $\Pi(\cdot) : \mathbb{R}^3 \rightarrow \mathbb{P}^2$ is the camera projection function, $\Pi^{-1}(\cdot) : \mathbb{P}^2 \times \mathbb{R} \rightarrow \mathbb{R}^3$ is the inverse, and $d_{t-\delta t,k}$ is depth of the map point associated with $\mathbf{x}_{t-\delta t,k}^{\mathcal{F}_{\text{SF}}}$ in the reference frame of the camera at time $t-\delta t$. Following Eq. (11), we use the spatio-temporal consistency constraint to compute the re-projection error for the estimated $\mathbf{x}_{t,k}^{\mathcal{F}_{\text{SF}}}$ as

$$\delta\mathbf{x}_{t,k}^{\mathcal{F}_{\text{SF}}} = \hat{\mathbf{x}}_{t,k}^{\mathcal{F}_{\text{SF}}} - \Pi\left(\hat{\mathbf{T}}_{t-\delta t}^{t}\Pi^{-1}\left(\hat{\mathbf{x}}_{t-\delta t,k}^{\mathcal{F}_{\text{SF}}^{t}}, \hat{d}_{t-\delta t,k}^{t}\right)\right), \tag{29}$$

where $\hat{\mathbf{x}}_{t-\delta t,k}^{\mathcal{F}_{\text{SF}}^{t}}$ and $\hat{d}_{t-\delta t,k}^{t}$ are the posterior estimates of $\mathbf{x}_{t-\delta t,k}^{\mathcal{F}_{\text{SF}}}$ and $d_{t-\delta t,k}$ at time t, respectively. $\Pi^{-1}\left(\hat{\mathbf{x}}_{t-\delta t,k}^{\mathcal{F}_{\text{SF}}^{t}}, \hat{d}_{t-\delta t,k}^{t}\right) = \mathbf{p}_k^{t-\delta t}$ is the corresponding map point location in the reference frame of the camera at time $t-\delta t$, and $\hat{\mathbf{T}}_{t-\delta t}^{t}$ is the estimated transformation from the camera frame at time $t-\delta t$ to the camera frame at time t. It should be noted that $\hat{\mathbf{T}}_{t-\delta t}^{t}$ is obtained from solving the bundle adjustment problem (Eq. (24)) by the SLAM back-end $\mathcal{F}_{\text{SB}}$ and to ensure that $\hat{\mathbf{T}}_{t-\delta t}^{t}$ is a good approximation for $\mathbf{T}_{t-\delta t}^{t}$, we verify that it agrees with a 3D lidar-based SLAM solution [57] by performing a χ^2 test with $\alpha = 0.05$ on the Mahalanobis distance between the two. Using Eq. (29), we collect reprojection error samples $\left\{\left\langle \left\| \delta\mathbf{x}_{t,k}^{\mathcal{F}_{\text{SF}}} \right\|^2, \hat{\mathbf{x}}_{t,k}^{\mathcal{F}_{\text{SF}}} \right\rangle\right\}_{k=1:N}$ across different images. In order to prepare the training data for the $g(\cdot)$ component of the introspection function, we post-process the set of computed sparse reprojection error values for every image to interpolate the estimated reprojection error for every pixel on the image using a Gaussian Process (GP) regressor. The outputs of the GP regressor are represented as images $\mathbf{I}_{\mathbf{c}t}$, which we refer to as cost-maps. Then, the pairs of input images and the corresponding cost-maps $\left\{\left\langle \mathbf{I}_t, \mathbf{I}_{\mathbf{c}t} \right\rangle\right\}_{t=1:K}$ are used to train the DNN implementation of $g(\cdot)$ 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. Fig. 3 shows the estimated cost-map and the uncertainty mask for an example input image. The color of each image feature visualized in the figure corresponds to the magnitude of the actual re-projection error of the map point corresponding to that feature. Green features have a small re-projection error and red features have a large re-projection error. The re-projection error of individual features extracted on the same surface can be different due to the stochastic nature of the noise process in the V-SLAM front-end, the different track lengths of the extracted features, the different scales of the extracted features, and the different 3D locations of the corresponding map points. In order to train the introspection function, all of these samples of re-projection errors are used and GP regression is used to obtain a smooth estimate of the magnitude of the re-projection error across the image.

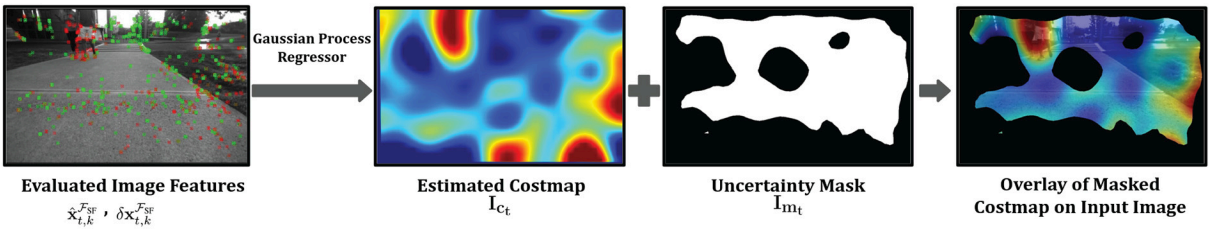


Fig. 3. Autonomous training data labeling for the IV-SLAM introspection function. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

8. Introspective stereo depth estimation

In this section, we present an implementation of introspective perception for a different perception task, stereo depth estimation. We demonstrate how supervisory sensing can be used to learn an introspection function for a depth estimator which in turn is used to predict different types of depth estimate errors.

8.1. Learning distribution of error in depth estimates

A stereo depth estimator $\mathcal{F}_{\mathrm{D}}$ outputs a depth image $\hat{\mathbf{I}}_t^{\mathrm{D}}$ given a stereo pair of captured images as input

$$\hat{\mathbf{I}}_t^{\mathrm{D}} = \mathcal{F}_{\mathrm{D}}\left(\mathbf{z}_t^{\mathcal{F}_{\mathrm{D}}}\right) \tag{30}$$

where $\mathbf{z}_t^{\mathcal{F}_{\mathrm{D}}} = \langle \mathbf{I}_t^l, \mathbf{I}_t^r \rangle$ and $\mathbf{I}_t^l$ and $\mathbf{I}_t^r$ are the images captured by the left and right cameras, respectively. Moreover, $\hat{\mathbf{I}}_t^{\mathrm{D}}(u, v)$ is the estimated depth for the pixel coordinates (u, v) in the left image. Depth estimate error for each pixel location is defined as

$$e_{t,(u,v)}^{\mathrm{D}} = \hat{\mathbf{I}}_t^{\mathrm{D}}(u, v) - \mathbf{I}_t^{\mathrm{D}}(u, v)\,, \tag{31}$$

where $e_{t,(u,v)}^{\mathrm{D}} \sim \phi^{\mathcal{F}_{\mathrm{D}}}\left(\mathbf{z}_t^{\mathcal{F}_{\mathrm{D}}}, \langle u, v \rangle\right)$. The error distribution $\phi^{\mathcal{F}_{\mathrm{D}}}$ is dependent on the context at every point on the image. The texture, shape, and surface material of the objects in the image combined with different lighting conditions can lead to different error distributions across frames and in different regions in the same image.

Following the definition of introspective perception in Eq. (5), we equip a stereo depth estimator with introspection by learning an approximation of the depth error distribution

$$\hat{\phi}^{\mathcal{F}_{\mathrm{D}}} = \mathbb{I}\left(\mathcal{F}_{\mathrm{D}}\right)\left[\mathbf{z}_t^{\mathcal{F}_{\mathrm{D}}}, \langle u, v \rangle\right], \tag{32}$$

where $\hat{\phi}^{\mathcal{F}_{\mathrm{D}}}$ is a piece-wise approximation of $\phi^{\mathcal{F}_{\mathrm{D}}}$ such that

$$\hat{\phi}^{\mathcal{F}_{\mathrm{D}}}_{\theta\left(\mathbf{z}_t^{\mathcal{F}_{\mathrm{D}}}, \langle u, v \rangle\right)}(x) = \begin{cases} \theta_{\mathrm{FP}}\left(\mathbf{z}_t^{\mathcal{F}_{\mathrm{D}}}, \langle u, v \rangle\right) & -R_{\max} < x < -\alpha \\ \theta_{\mathrm{T}}\left(\mathbf{z}_t^{\mathcal{F}_{\mathrm{D}}}, \langle u, v \rangle\right) & -\alpha \leq x \leq \alpha \\ \theta_{\mathrm{FN}}\left(\mathbf{z}_t^{\mathcal{F}_{\mathrm{D}}}, \langle u, v \rangle\right) & \alpha < x < R_{\max} \end{cases} \tag{33}$$

where α is a constant error threshold, $R_{\max}$ is the maximum range of the depth estimator, and θ_{FN}, θ_{T}, and θ_{FP} are the parameters of the error distribution and are learned. The probability density value θ_{FN} corresponds to cases when the estimated depth is larger than the actual depth, which we refer to as instances of false negatives (FN). Moreover, probability density value θ_{FP} corresponds to cases when the estimated depth is smaller than the actual depth, which we refer to as instances of false positives (FP). Using such piece-wise approximation of $\phi^{\mathcal{F}_{\mathrm{D}}}$ rather than a continuous probability density function has the benefit of being easier to learn, while still being informative for predicting failures at the task level.

8.2. Autonomously supervised training of the introspection function

Introspection function architecture. We implement $\mathbb{I}(\mathcal{F}_{\mathrm{D}})$ as a convolutional neural network (CNN) with the same architecture as AlexNet [58]. The input to the network are image patches $\mathbf{I}_{t,\langle u,v \rangle}^l$ extracted from different pixel locations $\langle u, v \rangle$ in the left camera images. The output are predicted probability scores $\langle p_{\mathrm{FP}}, p_{\mathrm{FN}}, p_{\mathrm{TP}}, p_{\mathrm{TN}} \rangle$ for each of the four different classes of false positive (FP), false negative (FN), true positive (TP), and true negative (TN) which make up the parameters of $\hat{\phi}^{\mathcal{F}_{\mathrm{D}}}_{\theta\left(\mathbf{z}_t^{\mathcal{F}_{\mathrm{D}}}, \langle u, v \rangle\right)}$ such that $\theta_{\mathrm{FP}} = \frac{p_{\mathrm{FP}}}{R_{\max} - \alpha}$, $\theta_{\mathrm{FN}} = \frac{p_{\mathrm{FN}}}{R_{\max} - \alpha}$, and $\theta_{\mathrm{T}} = \frac{p_{\mathrm{TN}} + p_{\mathrm{TP}}}{2\alpha}$.

Training data collection. We use consistency across sensors to collect samples of depth estimation error values. Specifically, we use a Kinect depth camera as a supervisory sensor and use its depth estimations $\hat{\mathbf{I}}_t^{\mathrm{D}'} = \mathcal{F}_{\mathrm{D}}{}'(\mathbf{z}_t^{\mathcal{F}_{\mathrm{D}}{}'})$ to obtain the reference depth values for the stereo depth estimator $\mathcal{F}_{\mathrm{D}}$

$$\mathbf{I}_t^{\mathrm{D}} = \Pi^{\mathrm{D}}\big(\Pi^{\mathrm{D}'^{-1}}(\hat{\mathbf{I}}_t^{\mathrm{D}'})\big), \tag{34}$$

where $\Pi^{\mathrm{D}'^{-1}}(\cdot) : \mathbb{P}^2 \times \mathbb{R} \to \mathbb{R}^3$ unprojects pixels from the Kinect depth image into the 3D robot local frame and $\Pi^{\mathrm{D}}(\cdot) : \mathbb{R}^3 \to \mathbb{P}^2 \times \mathbb{R}$ projects 3D points from the robot frame onto the primary camera images and assigns a depth value to each pixel. Following Eq. (6), we use consistency across sensors to compute the depth estimation errors as

$$\delta\mathbf{I}_t^{\mathrm{D}} = \hat{\mathbf{I}}_t^{\mathrm{D}} - \Pi^{\mathrm{D}}\big(\Pi^{\mathrm{D}'^{-1}}(\hat{\mathbf{I}}_t^{\mathrm{D}'})\big), \tag{35}$$

where $e^{\mathrm{D}}_{t,(u,v)} = \delta\mathbf{I}_t^{\mathrm{D}}(u,v)$. The robot collects samples of depth estimate errors during deployment when the supervisory sensor is available, e.g. indoors. This data is then used to train the introspection function.

9. Experimental results

In this section we evaluate our implementations of introspective perception for both V-SLAM and stereo depth estimation. We 1) evaluate the accuracy of the learned introspection functions in predicting errors of perception, 2) assess the effect of introspective perception in reducing failures of autonomy, 3) investigate the use of introspective perception in identifying the causes of robot failures, and 4) evaluate the performance of introspective perception, when the underlying perception algorithm is fully-learned and compare it against SOTA methods for estimating the uncertainty of deep neural networks.

We implement and evaluate IPr for three different perception algorithms. 1) We implement IV-SLAM for the stereo version of ORB-SLAM [59], a popular optimization-based visual SLAM algorithm. 2) We implement introspective depth estimation for JPP-C [60], which is a fast and computationally efficient geometric method for sparse stereo reconstruction. 3) We also implement introspective depth estimation for GA-Net [61], which is a deep-learning-based approach for stereo depth estimation.

9.1. Evaluation datasets

9.1.1. V-SLAM dataset

State-of-the-art vision-based SLAM algorithms have shown great performance on popular benchmark datasets such as KITTI [62] and EuRoC [63]. 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 [64]. 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 [65], 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 the City environment under different 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 LeGO-LOAM [57], a 3D Lidar-based SLAM solution. 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.

9.1.2. Stereo depth estimation dataset

We implement and evaluate introspective perception for two different stereo depth estimation algorithms: JPP-C [60], which is a model-based approach, and GA-Net [61], which is an end-to-end learning approach. For the former, we evaluate the approach on a real-robot dataset. For the latter, we evaluate the approach in simulation, where we can easily collect sufficient data for training the underlying fully-learned perception algorithm.

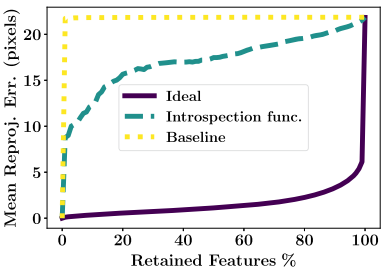


Fig. 4. Mean reprojection error for image features sorted 1) randomly (baseline), 2) based on predicted reliability (Introspection func.), 3) based on ground truth reprojection error (ideal).

Simulation. We use the AirSim simulator with the same sensor configuration as that used for the V-SLAM dataset. For the stereo depth estimation dataset, in addition to the City environment, we also collect data in the Neighborhood environment. This additional environment is held out during the training of the underlying fully-learned perception algorithm and used only for testing to evaluate the performance of IPr on out of distribution (OOD) data.

Real-world. The real-world dataset is collected on a Clearpath Jackal with a similar sensor suite to that used for collecting the V-SLAM dataset. The robot is deployed in both indoor and outdoor settings and the RGB images captured by the stereo pair of Point Grey cameras as well as depth images captured by a Kinect depth sensor are recorded at 30 Hz. The data is processed offline and the Kinect depth images are registered to the RGB images from the left camera to provide reference depth estimates. The indoor dataset spans multiple buildings with different types of tiling and carpet. The outdoor dataset is also collected on different surfaces such as asphalt, concrete, and tile in both dry and wet conditions. The total dataset that is generated from more than 1.4 km robot deployments includes about 120k image frames. The data is split into train and test datasets, each composed of separate full robot deployment sessions, such that both train and test sets include data from all types of terrain.

9.2. Introspection function accuracy

IV-SLAM introspection function. We evaluate the IV-SLAM introspection function based on its performance to correctly predict reliability of image features. We expect image features $\mathbf{x}_{t,k}^{\mathcal{F}_{\text{SF}}}$ for which a smaller loss function parameter $\theta(\mathbf{x}_{t,k}^{\mathcal{F}_{\text{SF}}})$ is predicted, to have larger reprojection errors (lower reliability). 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 §7.3. The introspection function 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 corresponding ground truth reprojection error, as well as the predicted loss function parameter by the introspection function. We then sort all image features in ascending order with respect to 1) ground truth reprojection errors and 2) the negative of the predicted loss function parameter $\theta(\mathbf{x}_{t,k}^{\mathcal{F}_{\text{SF}}})$. Fig. 4 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 introspection function significantly improves image feature reliability assessment. Fig. 5 illustrates the output of the introspection function for image features extracted at different scenes. The function attributes higher uncertainty to features extracted from regions that contain shadow of the robot, surface reflections, lens flare, and pedestrians. It is noteworthy that IV-SLAM has learned to identify these different sources of V-SLAM error autonomously and without any pre-enumeration of the different types of errors. This highlights the advantage of using introspection as opposed to hand-crafted error detection methods. Hand-crafted and heuristic metrics for evaluating the reliability of image features target a specific subset of the sources of image-matching errors by design and do not adapt to predict new sources of image-matching errors. For instance, while spatial entropy can be used to detect instances of textureless or blurry images [66], this metric does not predict the higher reprojection error on the crisp corners of the shadow of a moving object, for which we have observed V-SLAM failures. However, we have shown that such sources of error that are correlated with the contextual and semantic information in the input image can be discovered by introspective perception, and the autonomously supervised training of the introspective perception enables adapting to previously unseen sources of errors, removing the need for pre-enumeration of the error sources.

Depth estimator introspection function. We evaluate the depth estimator introspection function based on its performance in predicting the occurrence of depth estimation errors and their types. Fig. 6 shows the prediction results. The introspection function $\mathbb{I}(\mathcal{F}_{\text{D}})$ can predict the different types of errors with reasonably high accuracy in both indoor and outdoor settings.

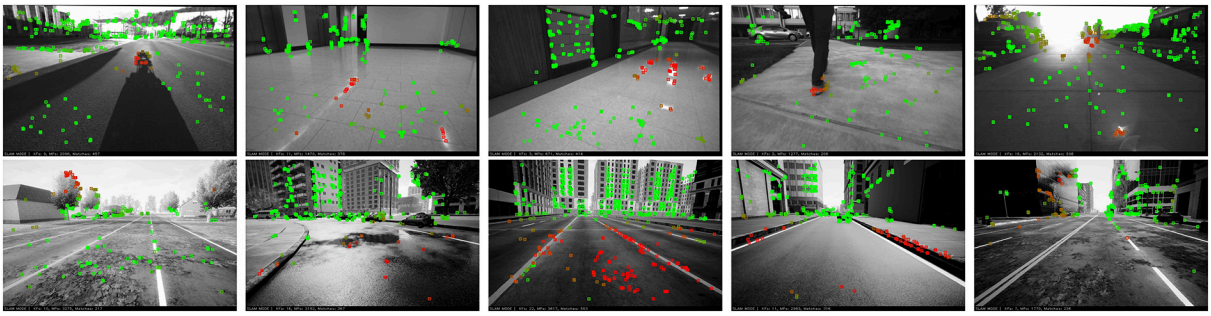


Fig. 5. Snapshots of IV-SLAM running on real-world data (top row) and in simulation (bottom row). Green and red points on the image represent the reliable and unreliable tracked image features, respectively, as predicted by the introspection function. Detected sources of error include shadow of the robot, surface reflections, pedestrians, glare, and ambiguous image features extracted from high-frequency textures such as asphalt or vegetation.

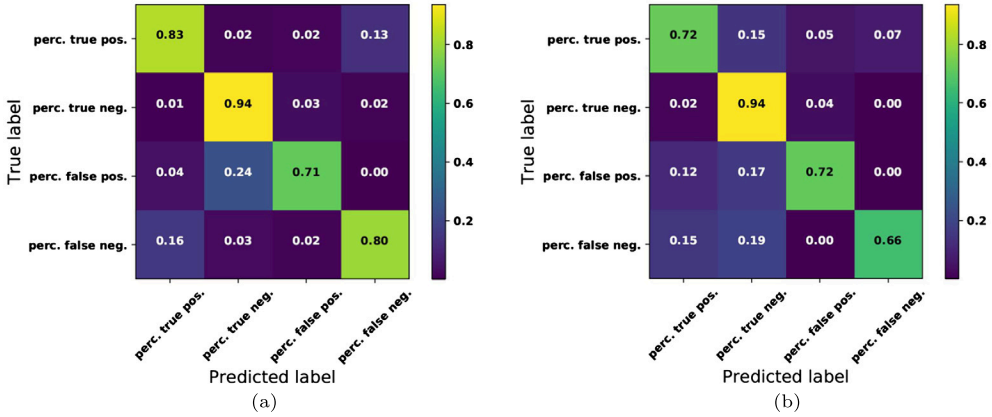


Fig. 6. Prediction results of the depth estimator introspection function on the (a) indoor and (b) outdoor datasets.

Fig. 7 also illustrates the output of $\mathbb{I}(\mathcal{F}_{\mathrm{D}})$ for an example input image, and shows how the introspection function correctly predicts water puddle reflections to lead to depth estimate errors.

9.3. Effect of introspective perception on robustness in autonomy

As explained in §7, IV-SLAM improves the robustness of V-SLAM by using the output of the introspection function, when solving for the pose of the camera in the SLAM back-end. 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 lose 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 [67].

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 2 and 3 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%. 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 §9.1.1, 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. Figs. 8 and 9 compare the per trajectory tracking error values as

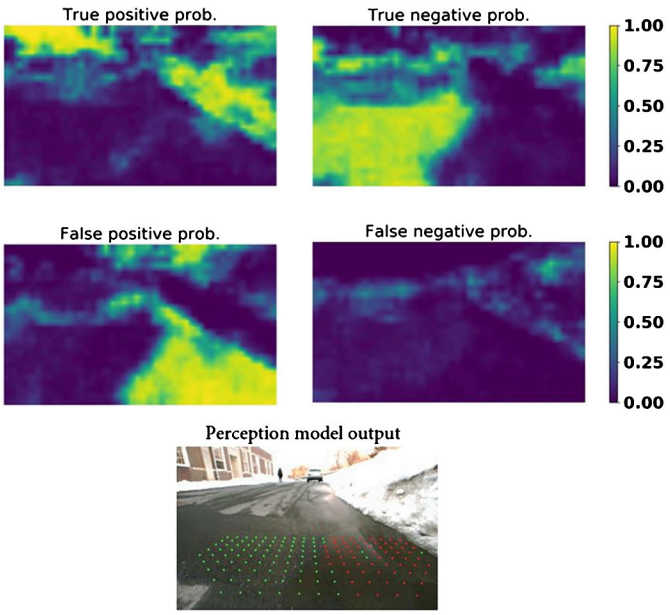


Fig. 7. Example output of the depth estimator introspection function. The bottom image shows the estimated obstacle grid by the depth estimator in front of the robot, where the red and green dots denote the detected occupied and obstacle free cells, respectively. The introspection function correctly predicts water reflections to cause false positives for the perception model (middle left image).

Table 2
Tracking Accuracy in the KITTI Dataset.

	IV-SLAM		ORB-SLAM	
Sequence	Trans. Err. %	Rot. Err. (deg/100 m)	Trans. Err. %	Rot. Err. (deg/100 m)
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 3
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

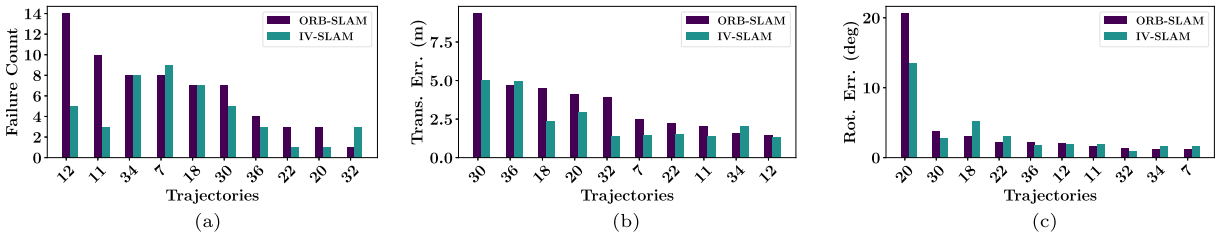


Fig. 8. Per trajectory comparison of the performance of IV-SLAM and ORB-SLAM in the simulation experiment. (a) Tracking failure count. (b) RMSE of translational error and (c) RMSE of rotational error over consecutive 20 m-long horizons.

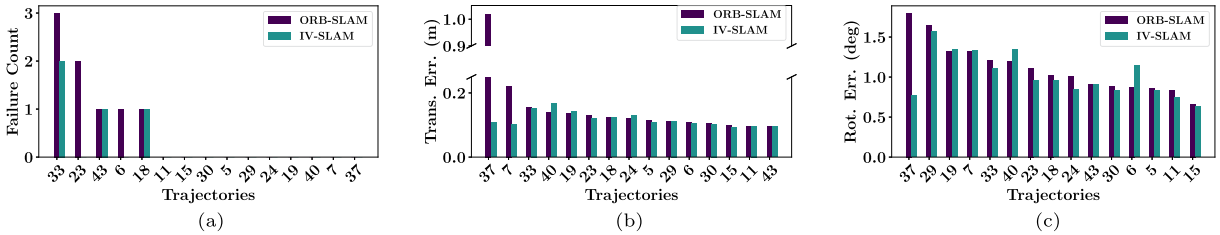


Fig. 9. Per trajectory comparison of the performance of IV-SLAM and ORB-SLAM in the real-world experiment. (a) Tracking failure count. (b) RMSE of translational error and (c) RMSE of rotational error over consecutive 2 m-long horizons.

Table 4
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

well as the tracking failure count for IV-SLAM and ORB-SLAM in both experimental environments. For most of the trajectories, IV-SLAM has a lower number of tracking failures. IV-SLAM also achieves similar or less RPE in most of the trajectories. RPE provides a metric of the local accuracy of the estimated pose of the robot and is calculated for each method only for parts of the trajectory where they have been able to track the pose of the robot. Since the baseline ORB-SLAM had more tracking failures, more parts of the trajectories, including the challenging sections where the failures happen, have been excluded when computing RPE for ORB-SLAM. As a result, the lower number of tracking failures for IV-SLAM while keeping a similar or lower local accuracy level shows that it achieves better global accuracy across the different trajectories. In the simulation experiments, trajectories 7 and 32 exhibit more tracking failures in IV-SLAM. This is due to challenging high-speed turning maneuvers in these datasets, where there are very few matched features across consecutive frames. IV-SLAM rejects unreliable features such as those present on high-frequency asphalt textures near the robot, but it faces a limitation in replacing these rejected features with new matched features due to the significant change in the camera viewpoint during high-speed turns. On the other hand, the baseline algorithm does not remove any of the unreliable features, resulting in continuous tracking but suffering from high RPE, as illustrated in Fig. 8. Table 4 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. Fig. 10 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.

9.4. Categorizing sources of perception errors

Using deep learning for approximating the introspection function has the advantage of learning representations of the sensory data, which capture properties that pose challenges for the perception algorithms and are correlated with occurrence of different types of errors in perception. Such learned representations are useful for debugging and analysis of a perception algorithm and identifying causes of perception errors from logged data. To test this hypothesis, we inspect the

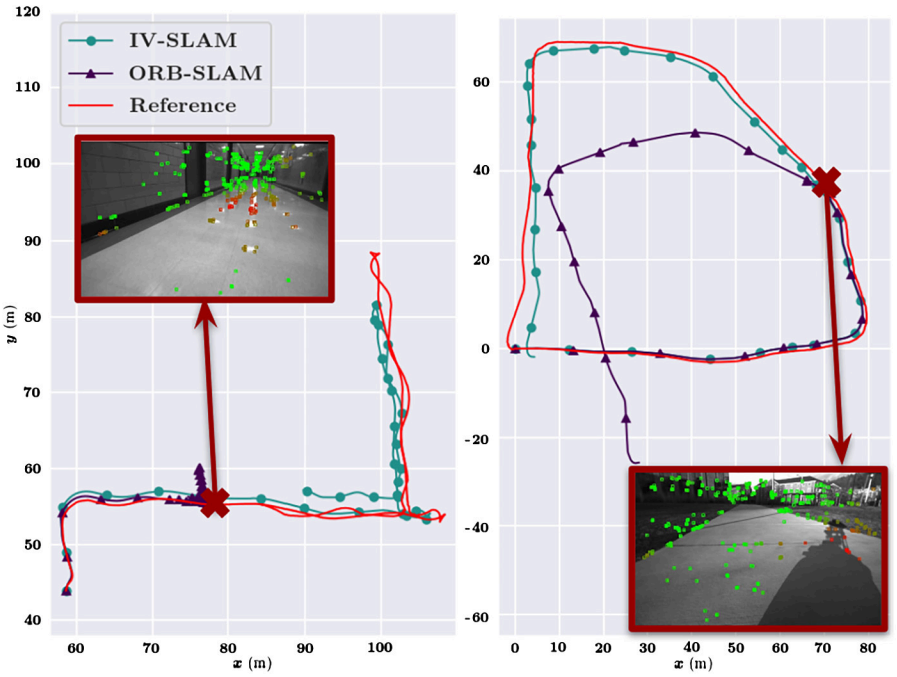


Fig. 10. Example deployment sessions of the robot. IV-SLAM successfully follows the reference camera trajectory while ORB-SLAM leads to severe tracking errors caused by image features extracted on the shadow of the robot and surface reflections.

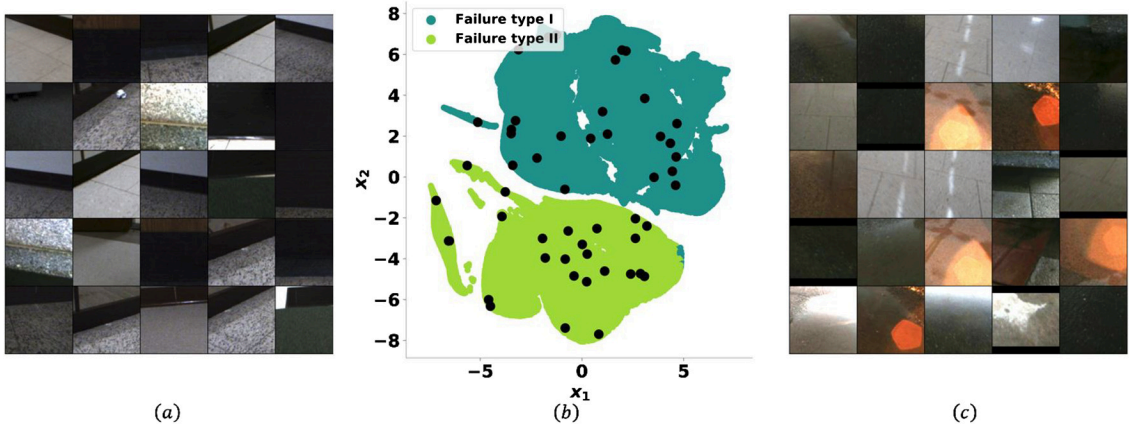


Fig. 11. Clustering predicted sources of error using data embeddings learned by the introspection function of the stereo depth estimator. b) Visualization of the two extracted clusters of perception errors in the embedding space after being projected down to 2*D* via dimensionality reduction. a) Randomly sampled image patches from the failure type II cluster, shown as black dots on (b). c) Image patches randomly sampled from the failure type I cluster. Reflection and lens glare (type I) and the dark stripes at the bottom of the walls (type II) are the most dominant sources of error for the perception algorithm under test.

learned representation by the introspection function of the depth estimator. We run the introspection function on the test dataset and for all data-points predicted to lead to perception errors, we cluster the extracted embedding vectors using the k-means approach. Fig. 11 illustrates the resultant clusters projected down to 2D using t-SNE [68] as well as sampled image patches from each cluster. The result shows that the dark edges at the bottom of the walls and reflection/glare to be the most dominant sources of error for the perception algorithm under test.

9.5. Introspection for fully learned perception

In order to evaluate the effectiveness of introspective perception for fully learned perception algorithms, we implement introspective perception for GA-Net [61], a deep neural network for stereo depth estimation. We compare our approach with SOTA uncertainty estimation methods that exist for deep learning models in terms of 1) failure prediction accuracy for

Table 5
Failure Prediction Results for In-Distribution Data.

Method	Precision	Recall	F-1 Score	NLL
IPr	0.6333	**0.8704**	0.6874	**-0.8717**
Ensemble	0.6642	0.8525	**0.7197**	-0.8182
Ensemble-Uncalib	0.8553	0.5721	0.6156	-0.6191
MCDropout	0.5897	0.7998	0.6259	-0.7853
MCDropout-Uncalib	**0.9875**	0.5000	0.4937	-0.6192

in-distribution data, 2) failure prediction accuracy for out-of-distribution data, and 3) their computational load. We use the simulation dataset described in §9.1.2 for our experiments.

We compare IPr against MCDropout and Ensemble, two common approaches for uncertainty estimation in deep learning. We evaluate them in terms of their ability to predict instances of depth estimation failures. We modify the GA-Net architecture to add dropout layers after ReLU activation units, and then train several instances of the network with different random initializations. For MCDropout, a single instance of the network is used and dropout is turned on during inference. We use a Monte Carlo sample size of 3. For Ensemble, multiple instances of the network are used and dropout is turned off during inference. We use 3 networks in our ensemble. As explained in §6 MCDropout and Ensemble include the variance of the output of each model in the ensemble (or each Monte Carlo samples) as a hyperparameter. We tune this hyperparameter to optimize for the negative log likelihood (NLL) on the training data. We also include in the experiments a version of both MCDropout and Ensemble without the hyperparameter tuning, which we refer to as MCDropout-Uncalib and Ensemble-Uncalib, respectively. The reason for including both calibrated and uncalibrated versions is to investigate whether the approximate BNN approach can be used without any training or calibration after the perception algorithm is trained and hence has the benefit of ease of use.

We train the GA-Net instances and the introspection function and also calibrate MCDropout and Ensemble on the same subset of the data from the City environment. The remaining data collected in the City environment, which is from novel trajectories of the robot navigating the environment under the same weather conditions present in the training dataset, comprises the in-distribution (ID) test dataset. Moreover, the data from the Neighborhood environment comprises the out-of-distribution (OOD) test dataset.

9.5.1. Failure prediction accuracy for in-distribution data

We run the test data through GA-Net equipped with the introspection function, the GA-Net ensemble, as well as the GA-Net MCDropout. For a pair of input stereo images, each method outputs an estimated depth image along with an estimated uncertainty image. For each pixel in a predicted depth image, we label the prediction as a failure if the depth error is greater than a threshold of 1.0 m. We quantify the accuracy of each uncertainty estimation method by comparing the ground truth failure labels with the per-pixel probability of failure predicted by each method. As explained in §6, MCDropout and Ensemble estimate the standard deviation of the predicted depth values, approximating the predictions to be from Gaussian distributions. For each pixel in the predicted depth image, the probability of failure is calculated in terms of the estimated uncertainty σ_{ens} as $p(\text{failure}) = 1 - \text{erf}(\frac{1}{\sqrt{2}\sigma_{\text{ens}}})$.

Table 5 demonstrates the failure prediction results in terms of precision, recall, F-1 score, and negative log likelihood (NLL) for all the methods under test. IPr achieves the best results in terms of recall rate and NLL, and Ensemble achieves the highest F-1 score. The uncalibrated versions of both Ensemble and MCDropout perform poorly and exhibit overconfidence in the accuracy of the depth estimates. The uncalibrated MCDropout predicts almost no predictions to be failures.

It should be noted that while precision, recall, and F1-score evaluate the classification performance of the different methods in predicting failures, the NLL metric also captures how well the predicted probabilities are distributed. In other terms, NLL evaluates how well the predicted error distribution matches the empirical error values. IPr achieves better NLL results since it is specifically trained to predict the error distribution of the perception algorithm under test, whereas the other methods report the empirical variance of the networks predictions as an approximate measure of uncertainty.

9.5.2. Failure prediction accuracy for out-of-distribution data

We separately evaluate the performance of the different methods in predicting failures of perception for data that is out-of-distribution with respect to the environments used for training both the perception algorithm (GA-Net) and the introspection function. In addition to the methods that we tested for the ID dataset, here we also implement and test a combination of IPr and Ensemble. We train an ensemble of 3 introspection functions for GA-Net and use the average output of the functions for predicting probability of failures. Our motivation for this is to investigate whether the ensemble approach can be used to estimate the uncertainty of the introspection function itself for OOD data.

Table 6 shows the failure prediction results summary for the OOD dataset. As expected, the failure prediction accuracy is reduced and NLL increased for all methods compared to the ID dataset. However, similar to the ID dataset, the uncalibrated versions of Ensemble and MCDropout are overconfident, Ensemble outperforms MCDropout, and IPr outperforms both in terms of NLL. The IPr-Ensemble achieves a marginally better NLL compared to IPr. The results show that IPr does a good job of predicting errors of perception also on OOD data, and that leveraging the idea of ensembles in the implementation of the introspection function leads to better or similar performance compared to IPr alone in novel environments.

Table 6
Failure Prediction Results for Out-Of-Distribution Data.

Method	Precision	Recall	F-1 Score	NLL
IPr	0.6162	0.8157	0.6521	-0.8056
IPr- Ensemble	0.6091	**0.8310**	0.6404	**-0.8083**
Ensemble	0.6730	0.8271	**0.7203**	-0.7696
Ensemble-Uncalib	0.9704	0.5000	0.4879	-0.5286
MCDropout	0.6312	0.8121	0.6724	-0.7496
MCDropout-Uncalib	**0.9763**	0.5000	0.4879	-0.5285

Table 7
GPU Inference Time and Memory Usage.

Model	Inference Time (ms)	Memory (MB)
IPr	**16.70 ± 0.11**	**16.97 ± 0.00**
Ensemble	244.85 ± 1.91	222.00 ± 0.00
MCDropout	244.80 ± 1.92	222.00 ± 0.00

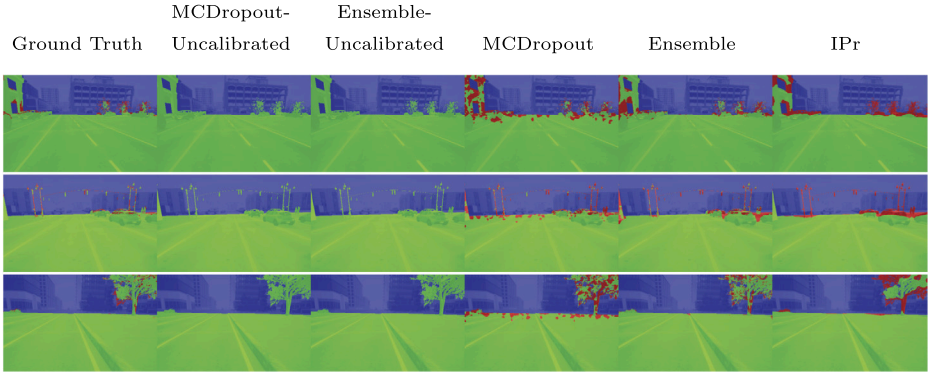


Fig. 12. Visualizations of the failures of an end-to-end depth estimation network (GA-Net) and the predicted failures by different uncertainty prediction methods. The input image to the depth estimator is colorized to highlight pixels, where the estimated depth is predicted to be correct (green), and pixels where the depth error is predicted to be larger than a threshold (red). Pixels that are colored blue are out of range. Both IPr and Ensemble correctly predict foliage, traffic lights, and edges of cars to lead to depth estimate failures.

9.5.3. Computational load and memory consumption

Robots are limited by the computational resources available to them, hence it is important to evaluate the additional computational cost incurred by performing uncertainty estimation. We measure the GPU inference time and memory consumption for each of the uncertainty estimation methods during inference. Table 7 summarizes the results. IPr is more than 15 times faster while consuming significantly less GPU memory compared to MCDropout and Ensemble. Since IPr tackles the problem of predicting the error distribution of the perception algorithm as opposed to solving the full perception problem, it can be implemented with a significantly smaller network architecture, making it ideal for real-time robotic applications, where the computational resources are limited.

9.5.4. Qualitative results

In order to better understand the types of failures of the depth estimator under test, and how IPr and the other methods perform in terms of predicting these failures, we visualize sample qualitative results. Fig. 12 illustrates the ground truth depth estimate failures of GA-Net overlaid on the input images and compares that with predicted failures by the different methods. The uncalibrated versions of MCDropout and Ensemble predict almost no failures. MCDropout has erroneous predictions of failures in regions where the perception algorithm seems to consistently predict the correct depth values. IPr and Ensemble are both able to correctly pick up the failures of the depth estimator without significant false positives. They predict obstacles such as poles, trees, and edges of the cars to lead to depth estimate failures.

10. Discussion and future work

Applying introspective perception to new perception tasks. Introspective perception is intended to capture a wide class of perception algorithms regardless of the functional form. IPr applies to both machine learning and non-machine learning perception algorithms. Given the vast range of perception algorithms, the introspective perception formulation does not provide a single algorithmic solution for all problems. Instead, it provides a general framework with a guideline for en-

abling a mobile robot to learn an empirical approximation of the uncertainty of its perception algorithms in the deployment environment and without the need for human supervision. Applying introspective perception to different perception tasks includes application-specific design choices such as the choice of the function approximator for the introspection function and the parametrization of the error distribution of perception. We introduce two general classes of strategies to formulate self-supervision of introspective perception, and we believe that these two classes can provide insights into how to deploy IPr for new perception problems given their specific settings.

The cost of collecting examples of perception errors. IPr collects samples of errors of perception functions in the deployment environment in order to train a machine learning model that predicts the uncertainty of perception and hence prevents catastrophic failures of the robot. While IPr requires a large amount of data to train an accurate and generalizable model, it does not require examples of a large number of catastrophic failures to train such a model. Catastrophic failures are rare events in well-developed systems, however, small errors that do not always lead to catastrophic failures happen more frequently. Being able to learn to predict instances of smaller errors, allows introspective perception to predict and prevent catastrophic failures that are caused by the accumulation of small errors. For instance, during the training of IV-SLAM the introspection function is provided with frequent instances of large re-projection errors on the moving shadow of the robot, while these errors do not lead to catastrophic tracking failures in the existence of sufficient reliable data correspondences in the same scene. However, learning about the higher uncertainty in the re-projection error of image features extracted on the shadow of the robot and leveraging this information in the V-SLAM back-end prevents catastrophic tracking failures that happen when the majority of image features are extracted from the moving shadow of the robot.

Correlation of errors of the perception function and the introspection function. For cases when the perception function is an end-to-end machine learning algorithm, it is questionable whether the introspection function is susceptible to the same errors as the perception function since the input data to both the perception function and the introspection function is the same and both functions are implemented as machine learning models. However, it should be noted that while the input data to the two functions is the same, the tasks of the two functions are different. The task of the introspection function is to estimate the aleatoric uncertainty of the perception function, i.e. predict the uncertainties in the output of the perception function that can not be reduced by providing more training data to the perception function. A machine learning perception algorithm might provide erroneous results when there is insufficient information in the input, e.g. when the input image is too dark for reliable depth estimation, however, the introspection function tackles the easier learning task of identifying that there is insufficient information in the input.

Fusion of multiple introspection functions. Some perception functions can be written as the composition of smaller perception functions. For instance, as also shown in this work, V-SLAM is composed of a front-end that is responsible for feature detection and matching and a back-end that is responsible for estimating the pose of the camera given data associations provided by the front-end. In this work, we focused on developing an autonomously supervised method for learning an accurate estimate of the error distribution of each of the sub-components of a large perception algorithm that can each be defined as a perception function. To estimate the error distribution of a complex perception function, we also need to reason about the propagation of errors in the output of the sub-components of the perception function. For a complex perception algorithm, modeling how errors in the output of one perception function, as estimated by its introspection function, propagate through a downstream perception function can be non-trivial, especially in the case of nonlinear perception functions. Extending the introspection function to leverage the uncertainty of the intermediate state variables as one of its inputs or using classical filtering methods [69] are potential approaches to tackle this problem.

The role of high-fidelity sensors in training of introspective perception. While the availability of high-fidelity sensors is not a strict requirement for introspective perception, it is one of the viable methods to train the introspection function by leveraging consistency across sensors, as explained in §4.2 and similar to the example of introspective stereo depth estimator presented in §8. In situations where high-fidelity sensors are unavailable, we can use offline supervisory perception functions that consume the same sensory input as the primary perception function but provide higher accuracy outputs at the cost of increased computational requirements. For example, an offline depth estimator based on Neural Radiance Fields (NeRF) [37] can be employed to supervise a stereo depth estimator, although real-time deployment is not feasible. Moreover, the use of spatio-temporal consistency constraints, as presented in §4.2, enables training of introspective perception without the necessity of a supervisory perception function or high-fidelity sensors.

Future work. There are several avenues for future work. First, IPr can be implemented for other safety-critical perception algorithms such as object detection, tracking, and lidar-based localization. Second, the introspection function can be generalized to leverage other informative data such as the robot control commands along with the perception inputs for predicting the errors of perception. Third, in this work we employed a passive approach for collecting the training data for IPr, where all the data logged by the robot during deployment was used for training. However, IPr can be equipped with active-learning techniques [70,71] such that it guides the robot planner to collect more informative data online, and it subsamples a representative set of the logged data offline to train the introspection function more efficiently. Moreover, in §9.4, we showed that representation of the sensor data learned by the introspection function can be used to cluster the perception errors

that are due to the same source. It would be interesting to further investigate the performance of IPr in classifying the different sources of errors. Specifically, labeling and tracking the sources of errors in the test data in order to quantitatively evaluate IPr for identifying the different types of perception errors would be a great next step.

11. Conclusion

In this paper, we defined and presented a general theory for introspective perception. We proposed autonomously supervised methods for training IPr in the deployment environment of a mobile robot. We provided examples of implementing IPr for two different perception algorithms: visual simultaneous localization and mapping and stereo depth estimation. We demonstrated that IPr reduces failures of autonomy that are associated with perception errors. Furthermore, we showed that IPr applies to both model-based and fully-learned perception algorithms, and is computationally significantly less expensive compared to the uncertainty estimation methods for deep learning based perception.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgements

This work is supported in part by NSF (CAREER-2046955, IIS-1954778) and DARPA (HR001120C0031). The views and conclusions contained in this document are those of the authors only.

References

[1] M. Kaess, F. Dellaert, Covariance recovery from a square root information matrix for data association, Robot. Auton. Syst. 57 (2009) 1198–1210.
[2] R. Urtasun, D.J. Fleet, P. Fua, 3D people tracking with Gaussian process dynamical models, in: 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06), vol. 1, IEEE, 2006, pp. 238–245.
[3] S. Rabiee, J. Biswas, IVOA: introspective vision for obstacle avoidance, in: 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2019, pp. 1230–1235.
[4] S. Rabiee, J. Biswas, Iv-slam: introspective vision for simultaneous localization and mapping, in: Conference on Robot Learning (CoRL), 2020.
[5] A.A. Pereira, J. Binney, G.A. Hollinger, G.S. Sukhatme, Risk-aware path planning for autonomous underwater vehicles using predictive ocean models, J. Field Robot. 30 (2013) 741–762.
[6] A. Kapoor, K. Grauman, R. Urtasun, T. Darrell, Active learning with Gaussian processes for object categorization, in: 2007 IEEE 11th International Conference on Computer Vision, IEEE, 2007, pp. 1–8.
[7] H. Liu, R. Ji, J. Li, B. Zhang, Y. Gao, Y. Wu, F. Huang, Universal adversarial perturbation via prior driven uncertainty approximation, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2019, pp. 2941–2949.
[8] G. Pandey, J.R. McBride, S. Savarese, R.M. Eustice, Automatic extrinsic calibration of vision and lidar by maximizing mutual information, J. Field Robot. 32 (2015) 696–722.
[9] X. Zhang, P. Willett, Y. Bar-Shalom, Dynamic Cramer-Rao bound for target tracking in clutter, IEEE Trans. Aerosp. Electron. Syst. 41 (2005) 1154–1167.
[10] S.A. Sadat, K. Chutskoff, D. Jungic, J. Wawerla, R. Vaughan, Feature-rich path planning for robust navigation of MAVs with mono-SLAM, in: 2014 IEEE International Conference on Robotics and Automation (ICRA), IEEE, 2014, pp. 3870–3875.
[11] C. Mostegel, A. Wendel, H. Bischof, Active monocular localization: towards autonomous monocular exploration for multirotor MAVs, in: 2014 IEEE International Conference on Robotics and Automation (ICRA), IEEE, 2014, pp. 3848–3855.
[12] C.E. Rasmussen, Gaussian processes in machine learning, in: Summer School on Machine Learning, Springer, 2003, pp. 63–71.
[13] J. Quinonero-Candela, C.E. Rasmussen, A unifying view of sparse approximate Gaussian process regression, J. Mach. Learn. Res. 6 (2005) 1939–1959.
[14] Z. Ghahramani, Learning dynamic Bayesian networks, in: International School on Neural Networks, Initiated by IIASS and EMFCSC, Springer, 1997, pp. 168–197.
[15] J. Lee, Y. Bahri, R. Novak, S.S. Schoenholz, J. Pennington, J. Sohl-Dickstein, Deep neural networks as Gaussian processes, in: International Conference on Learning Representations, 2018.
[16] M. Dusenberry, G. Jerfel, Y. Wen, Y. Ma, J. Snoek, K. Heller, B. Lakshminarayanan, D. Tran, Efficient and scalable Bayesian neural nets with rank-1 factors, in: International Conference on Machine Learning, PMLR, 2020, pp. 2782–2792.
[17] Y. Gal, Z. Ghahramani, Dropout as a Bayesian approximation: representing model uncertainty in deep learning, in: International Conference on Machine Learning, 2016, pp. 1050–1059.
[18] H. Grimmett, R. Triebel, R. Paul, I. Posner, Introspective classification for robot perception, Int. J. Robot. Res. 35 (2016) 743–762.
[19] B. Lakshminarayanan, A. Pritzel, C. Blundell, Simple and scalable predictive uncertainty estimation using deep ensembles, in: Proceedings of the 31st International Conference on Neural Information Processing Systems, 2017, pp. 6405–6416.
[20] Y. Ovadia, E. Fertig, J. Ren, Z. Nado, D. Sculley, S. Nowozin, J.V. Dillon, B. Lakshminarayanan, J. Snoek, Can you trust your model's uncertainty? Evaluating predictive uncertainty under dataset shift, 2019.
[21] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, Generative adversarial nets, Adv. Neural Inf. Process. Syst. 27 (2014).
[22] P. Vincent, H. Larochelle, Y. Bengio, P.-A. Manzagol, Extracting and composing robust features with denoising autoencoders, in: Proceedings of the 25th International Conference on Machine Learning, 2008, pp. 1096–1103.
[23] D.P. Kingma, M. Welling, Auto-encoding variational Bayes, arXiv preprint, arXiv:1312.6114, 2013.

[24] J. An, S. Cho, Variational autoencoder based anomaly detection using reconstruction probability, in: Special Lecture on IE, vol. 2, 2015, pp. 1–18.
[25] A. Desai, T. Dreossi, S.A. Seshia, Combining model checking and runtime verification for safe robotics, in: Runtime Verification: 17th International Conference, Proceedings, RV 2017, Seattle, WA, USA, September 13-16, 2017, Springer, 2017, pp. 172–189.
[26] S. Ghosh, D. Sadigh, P. Nuzzo, V. Raman, A. Donzé, A.L. Sangiovanni-Vincentelli, S.S. Sastry, S.A. Seshia, Diagnosis and repair for synthesis from signal temporal logic specifications, in: Proceedings of the 19th International Conference on Hybrid Systems: Computation and Control, 2016, pp. 31–40.
[27] A. Balakrishnan, J. Deshmukh, B. Hoxha, T. Yamaguchi, G. Fainekos, Percemon: online monitoring for perception systems, in: Runtime Verification: 21st International Conference, Proceedings 21, RV 2021, Virtual Event, October 11–14, 2021, Springer, 2021, pp. 297–308.
[28] A. Balakrishnan, A.G. Puranic, X. Qin, A. Dokhanchi, J.V. Deshmukh, H.B. Amor, G. Fainekos, Specifying and evaluating quality metrics for vision-based perception systems, in: 2019 Design, Automation & Test in Europe Conference & Exhibition (DATE), IEEE, 2019, pp. 1433–1438.
[29] S. Mitsch, K. Ghorbal, D. Vogelbacher, A. Platzer, Formal verification of obstacle avoidance and navigation of ground robots, Int. J. Robot. Res. 36 (2017) 1312–1340.
[30] P. Zhang, J. Wang, A. Farhadi, M. Hebert, D. Parikh, Predicting failures of vision systems, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2014, pp. 3566–3573.
[31] S. Daftry, S. Zeng, J.A. Bagnell, M. Hebert, Introspective perception: learning to predict failures in vision systems, in: 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE, 2016, pp. 1743–1750.
[32] C. Gurău, D. Rao, C.H. Tong, I. Posner, Learn from experience: probabilistic prediction of perception performance to avoid failure, Int. J. Robot. Res. 37 (2018) 981–995.
[33] W. Vega-Brown, A. Bachrach, A. Bry, J. Kelly, N. Roy, Cello: a fast algorithm for covariance estimation, in: 2013 IEEE International Conference on Robotics and Automation, IEEE, 2013, pp. 3160–3167.
[34] D. Stronger, P. Stone, Towards autonomous sensor and actuator model induction on a mobile robot, Connect. Sci. 18 (2006) 97–119.
[35] J. Holtz, J. Biswas, Automatic extrinsic calibration of depth sensors with ambiguous environments and restricted motion, in: 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2017, pp. 2235–2240.
[36] J.L. Schonberger, J.-M. Frahm, Structure-from-motion revisited, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 4104–4113.
[37] A. Kundu, K. Genova, X. Yin, A. Fathi, C. Pantofaru, L.J. Guibas, A. Tagliasacchi, F. Dellaert, T. Funkhouser, Panoptic neural fields: a semantic object-aware neural scene representation, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022, pp. 12871–12881.
[38] S. Thrun, W. Burgard, D. Fox, Probabilistic Robotics, MIT Press, Cambridge, Mass., 2005.
[39] R. Pepy, A. Lambert, Safe path planning in an uncertain-configuration space using RRT, in: 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems, IEEE, 2006, pp. 5376–5381.
[40] C.J. Ostafew, A.P. Schoellig, T.D. Barfoot, Robust constrained learning-based NMPC enabling reliable mobile robot path tracking, Int. J. Robot. Res. 35 (2016) 1547–1563.
[41] F.S. Barbosa, B. Lacerda, P. Duckworth, J. Tumova, N. Hawes, Risk-aware motion planning in partially known environments, in: IEEE Conference on Decision and Control (CDC), 2021.
[42] L. Blackmore, M. Ono, B.C. Williams, Chance-constrained optimal path planning with obstacles, IEEE Trans. Robot. 27 (2011) 1080–1094.
[43] A.M. Jasour, B.C. Williams, Risk contours map for risk bounded motion planning under perception uncertainties, in: Robotics: Science and Systems, 2019.
[44] S. Rabiee, C. Basich, K.H. Wray, S. Zilberstein, J. Biswas, Competence-aware path planning via introspective perception, IEEE Robot. Autom. Lett. 7 (2022) 3218–3225.
[45] S. Wang, R. Clark, H. Wen, N. Trigoni, DeepVO: towards end-to-end visual odometry with deep recurrent convolutional neural networks, in: 2017 IEEE International Conference on Robotics and Automation (ICRA), IEEE, 2017, pp. 2043–2050.
[46] E. Parisotto, D. Singh Chaplot, J. Zhang, R. Salakhutdinov, Global pose estimation with an attention-based recurrent network, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, 2018, pp. 237–246.
[47] B. Triggs, P.F. McLauchlan, R.I. Hartley, A.W. Fitzgibbon, Bundle adjustment–a modern synthesis, in: International Workshop on Vision Algorithms, Springer, 1999, pp. 298–372.
[48] M.J. Black, A. Rangarajan, On the unification of line processes, outlier rejection, and robust statistics with applications in early vision, Int. J. Comput. Vis. 19 (1996) 57–91.
[49] J.T. Barron, A general and adaptive robust loss function, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019, pp. 4331–4339.
[50] M.A. Fischler, R.C. Bolles, Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography, Commun. ACM 24 (1981) 381–395.
[51] C. Forster, M. Pizzoli, D. Scaramuzza, SVO: fast semi-direct monocular visual odometry, in: 2014 IEEE International Conference on Robotics and Automation (ICRA), IEEE, 2014, pp. 15–22.
[52] K.M. Yi, E. Trulls, Y. Ono, V. Lepetit, M. Salzmann, P. Fua, Learning to find good correspondences, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 2666–2674.
[53] W. Sun, W. Jiang, E. Trulls, A. Tagliasacchi, K.M. Yi, Acne: attentive context normalization for robust permutation-equivariant learning, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2020, pp. 11286–11295.
[54] B. Triggs, P.F. McLauchlan, R.I. Hartley, A.W. Fitzgibbon, Bundle adjustment—a modern synthesis, in: Vision Algorithms: Theory and Practice: International Workshop on Vision Algorithms, Proceedings, Corfu, Greece, September 21–22, 1999, Springer, 2000, pp. 298–372.
[55] B. Zhou, H. Zhao, X. Puig, S. Fidler, A. Barriuso, A. Torralba, Scene parsing through ADE20K dataset, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 633–641.
[56] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, L.-C. Chen, MobileNetV2: inverted residuals and linear bottlenecks, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 4510–4520.
[57] T. Shan, 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), 2018, pp. 4758–4765.
[58] A. Krizhevsky, I. Sutskever, G.E. Hinton, Imagenet classification with deep convolutional neural networks, in: Advances in Neural Information Processing Systems, 2012, pp. 1097–1105.
[59] R. Mur-Artal, J.D. Tardós, ORB-SLAM2: an open-source slam system for monocular, stereo, and RGB-d cameras, IEEE Trans. Robot. 33 (2017) 1255–1262.
[60] S. Ghosh, J. Biswas, Joint perception and planning for efficient obstacle avoidance using stereo vision, in: 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE, 2017, pp. 1026–1031.
[61] F. Zhang, V. Prisacariu, R. Yang, P.H. Torr, GA-Net: guided aggregation net for end-to-end stereo matching, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 185–194.
[62] A. Geiger, P. Lenz, 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, 2012, pp. 3354–3361.
[63] M. Burri, J. Nikolic, P. Gohl, T. Schneider, J. Rehder, S. Omari, M.W. Achtelik, R. Siegwart, The EuRoC micro aerial vehicle datasets, Int. J. Robot. Res. 35 (2016) 1157–1163.

[64] 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.
[65] S. Shah, D. Dey, C. Lovett, A. Kapoor, AirSim: high-fidelity visual and physical simulation for autonomous vehicles, in: Field and Service Robotics, 2018, pp. 621–635.
[66] C. Brunner, T. Peynot, T. Vidal-Calleja, J. Underwood, Selective combination of visual and thermal imaging for resilient localization in adverse conditions: day and night, smoke and fire, J. Field Robot. 30 (2013) 641–666.
[67] D. Schubert, T. Goll, N. Demmel, V. Usenko, J. Stückler, D. Cremers, The TUM VI benchmark for evaluating visual-inertial odometry, in: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2018, pp. 1680–1687.
[68] L.v.d. Maaten, G. Hinton, Visualizing data using t-SNE, J. Mach. Learn. Res. 9 (2008) 2579–2605.
[69] S.J. Julier, J.K. Uhlmann, Unscented filtering and nonlinear estimation, Proc. IEEE 92 (2004) 401–422.
[70] R. Bajcsy, Active perception, Proc. IEEE 76 (1988) 966–1005.
[71] M. Krainin, B. Curless, D. Fox, Autonomous generation of complete 3D object models using next best view manipulation planning, in: International Conference on Robotics and Automation (ICRA), 2011.