Université de Montréal

Calibrated Uncertainty Estimation for SLAM

par

Dishank Bansal

Département de mathématiques et de statistique
Faculté des arts et des sciences

Mémoire présenté en vue de l'obtention du grade de
Maître ès sciences (M.Sc.)
en Discipline

April 29, 2023

Université de Montréal

Faculté des arts et des sciences

Ce mémoire intitulé

Calibrated Uncertainty Estimation for SLAM

présenté par

Dishank Bansal

a été évalué par un jury composé des personnes suivantes :

Aishwarya Agrawal
(président-rapporteur)

Liam Paull
(directeur de recherche)

Glen Berseth
(membre du jury)

Résumé

La focus de cette thèse de maîtrise est l'analyse de l'étalonnage de l'incertitude pour la localisation et la cartographie simultanées (SLAM) en utilisant des modèles de mesure basés sur les réseaux de neurones. SLAM sont un problème fondamental en robotique et en vision par ordinateur, avec de nombreuses applications allant des voitures autonomes aux réalités augmentées. Au cœur de SLAM, il s'agit d'estimer la pose (c'est-à-dire la position et l'orientation) d'un robot ou d'une caméra lorsqu'elle se déplace dans un environnement inconnu et de construire simultanément une carte de l'environnement environnant. Le SLAM visuel, qui utilise des images en entrée, est un cadre de SLAM couramment utilisé. Cependant, les méthodes traditionnelles de SLAM visuel sont basées sur des caractéristiques fabriquées à la main et peuvent être vulnérables à des défis tels que la mauvaise luminosité et l'occultation. L'apprentissage profond est devenu une approche plus évolutive et robuste, avec les réseaux de neurones convolutionnels (CNN) devenant le système de perception de facto en robotique.

Pour intégrer les méthodes basées sur les CNN aux systèmes de SLAM, il est nécessaire d'estimer l'incertitude ou le bruit dans les mesures de perception. L'apprentissage profond bayésien a fourni diverses méthodes pour estimer l'incertitude dans les réseaux de neurones, notamment les ensembles, la distribution sur les paramètres du réseau et l'ajout de têtes de prédiction pour les paramètres de distribution de la sortie. Cependant, il est également important de s'assurer que ces estimations d'incertitude sont bien étalonnées, c'est-à-dire qu'elles reflètent fidèlement l'erreur de prédiction.

Dans cette thèse de maîtrise, nous abordons ce défi en développant un système de SLAM qui intègre un réseau de neurones en tant que modèle de mesure et des estimations d'incertitude étalonnées. Nous montrons que ce système fonctionne mieux que les approches qui utilisent la méthode traditionnelle d'estimation de l'incertitude, où les estimations de l'incertitude sont simplement considérées comme des hyperparamètres qui sont réglés manuellement. Nos résultats démontrent l'importance de tenir compte de manière précise de l'incertitude dans le problème de SLAM, en particulier lors de l'utilisation d'un réseau de neur

Mots clés: SLAM, estimation de l'incertitude, Étalonnage, Les réseaux de neurones, l'apprentissage en profondeur, Robotique

Abstract

The focus of this Masters thesis is the analysis of uncertainty calibration for Simultaneous Localization and Mapping (SLAM) using neural network-based measurement models. SLAM is a fundamental problem in robotics and computer vision, with numerous applications ranging from self-driving cars to augmented reality. At its core, SLAM involves estimating the pose (i.e., position and orientation) of a robot or camera as it moves through an unknown environment and constructing a map of the surrounding environment simultaneously. Visual SLAM, which uses images as input, is a commonly used SLAM framework. However, traditional Visual SLAM methods rely on handcrafted features and can be vulnerable to challenges such as poor lighting and occlusion. Deep learning has emerged as a more scalable and robust approach, with Convolutional Neural Networks (CNNs) becoming the de facto perception system in robotics.

To integrate CNN-based methods with SLAM systems, it is necessary to estimate the uncertainty or noise in the perception measurements. Bayesian deep learning has provided various methods for estimating uncertainty in neural networks, including ensembles, distributions over network parameters, and adding variance heads for direct uncertainty prediction. However, it is also essential to ensure that these uncertainty estimates are well-calibrated, i.e they accurately reflect the error in the prediction.

In this Master's thesis, we address this challenge by developing a system for SLAM that incorporates a neural network as the measurement model and calibrated uncertainty estimates. We show that this system performs better than the approaches which uses traditional uncertainty estimation method, where uncertainty estimates are just considered hyperparameters which are tuned manually. Our results demonstrate the importance of accurately accounting for uncertainty in the SLAM problem, particularly when using a neural network as the measurement model, in order to achieve reliable and robust localization and mapping.
Keywords: SLAM, Uncertainty Estimation, Calibration, Neural Networks, Deep Learning, Robotics

Contents

List of tables

List of figures

Liste des sigles et des abréviations

SLAM	Localisation et cartographie simultanées, de l'anglais *Simultaneous Localization and Mapping*
CNN	Réseau neuronal à convolution, de l'anglais *Convolutional Neural Network*
MCMC	Monte Carlo par chaînes de Markov, de l'anglais *Markov Chain Monte Carlo*
ECE	Erreur de Calibration Attendue, de l'anglais *Expected Calibration Error*
MSE	Erreur quadratique moyenne, de l'anglais *Mean Square Error*
GPS	Système de Positionnement Global, de l'anglais *Global Positioning System*
IMU	Unité de mesure inertielle, de l'anglais *Inertial Measurement unit*
CDF	Fonction de répartition, de l'anglais *Cummulative Density Function*

DNNS	Réseau neuronal profond en tant que capteur, de l'anglais *Deep Neural Network as a Sensor*

Remerciements

Participating in this Master's program has been an incredible journey that I will cherish forever. This program not only expanded my research skills but has also helped me grow as an individual. Working alongside my amazing supervisor, Professor Liam Paull, and other esteemed researchers like Krishna Murthy and Steven Parkison has been a real honour and a privilege. I would like to take this opportunity to express my heartfelt appreciation to Liam, who has been an exceptional supervisor throughout my master's program. Liam has not only been an excellent mentor in terms of research guidance, but he has also been really helpful when it comes to personal well-being.

I am grateful to have crossed paths with amazing people like Sangnie, Fernanda, Gaurav, Ankit, Nandita, Disha, Ishaan, and many others who have supported me throughout this journey. Especially during the pandemic, their support has been priceless, and our board game nights were a much-needed respite from the frustrations of coding.

I can't express enough how much my family means to me. My parents and brother have been my biggest cheerleaders, and I owe all my success to their unwavering support and encouragement. This thesis is a tribute to them, and I am so proud to be able to share this accomplishment with them.

In summary, I feel incredibly grateful for this incredible opportunity to pursue my master's degree, and I am deeply thankful to everyone who has helped me get here. This experience has been truly transformative.

Chapter 1

Introduction

Simultaneous Localization and Mapping (SLAM) comprises the estimation of a robot/embodied-agent pose along with creating a map of the unknown surrounding environment [**4, 52, 20**]. The map is a representation of aspects of interest such as landmarks, obstacles, etc. SLAM is a core building block of any embodied agent such as self-driving cars, mining robots, etc. as it enables the robot to localize itself in the environment and the estimated map can be used for other important tasks such as path planning, manipulation, etc.

One of the most used SLAM frameworks is Visual SLAM - images are the input to the SLAM system - as the camera is a very common sensor modality in robots. The traditional Visual SLAM frameworks work on handcrafted feature extractors for point landmarks which are tracked across frames, odometry is predicted by solving for geometric constraints of tracked points [**39, 36, 12, 11**]. Since the methods depend on handcrafted feature extractors, the SLAM system might not be robust in challenging scenarios such as poor lighting conditions, occlusion, etc.

With the advent of deep learning, owing to the scalability and robustness relative to traditional computer vision methods, CNNs became the de-facto perception system in robotics. Modern embodied agents heavily rely on learning systems (deep neural networks). Various tasks like detecting and localizing objects, processing medical scans and segmenting out anomalies, or predicting control actions of self-driving vehicles from image inputs employ neural network models as a core component [**53, 3**]. CNNs enable to capture an abstract feature representation of a scene in an image, due to which many SLAM methods emerged using CNNs as the perception component of the SLAM system, replacing traditional handcrafted feature extraction methods [**50, 6, 59, 10**].

Pose graph optimization approaches for tackling SLAM problems such as factor graph optimizaion [**7, 26**] are probabilistic approaches, as these methods allow for probabilistic inference over the state of the map and the robot. Probabilistic methods also give the ability

to handle the different sources of noise in the measurements which are used as input to SLAM systems. Hence to allow for probabilistic reasoning and being able to handle noises in SLAM, the perception measurement models should have a probability distribution over the output. Traditionally, different Visual SLAM used different ways to estimate noise/uncertainty in the perception measurement model. Similarly, to be able to integrate the deep learning based method with SLAM, the deep learning-based sensor model needs to be equipped with noise/uncertainty estimation.

Bayesian deep learning has provided different methods for providing probabilistic output in neural networks [**16, 2, 30, 28**]. There are different types of approaches to estimate uncertainty such as ensembles, distribution over network parameters, and adding prediction head for distribution parameters of output. These approaches have been successfully used in CNNs for perception deep learning models in tasks such as object detection, segmentation, depth estimation, etc. Hence these probabilistic approaches allow to integrate deep neural networks with probabilistic SLAM systems.

The quality of uncertainty estimates is indicated by calibration. Calibration evaluates the predicted uncertainty estimate by comparing it to the actual data distribution. A well-calibrated model has uncertainty prediction that is an indicator of error in the prediction. Hence, only having uncertainty estimates in deep neural networks is not enough, they also need to be calibrated. Traditionally, probabilistic robotic systems used to have calibrated noise estimates i.e, calibration experiments can be performed by doing multiple measurements of known ground truth. Variance across those multiple measurements of the same ground truth quantity gives a calibrated uncertainty estimate.

A large amount of research on the calibration of uncertainty estimation in deep learning is limited to classification setup whereas SLAM mostly deals with regression setup [**18, 42**]. Hence, one of the contributions of this work is calibrated uncertainty estimation in neural networks for regression.[1] There have been few efforts to translate the calibration techniques used for small-scale methods such as linear regression to deep neural networks [**13, 29**] but they don't exploit the learning algorithm of the deep networks and don't scale well. To overcome this, we propose a loss function f-Cal based on distribution matching to enforce the calibration of uncertainty estimates in a deep neural network. f-Cal is a neural network equipped with calibrated uncertainty estimation over the output.

Subsequently, there hasn't been much research done on assessing the effectiveness of uncertainty calibration on downstream tasks i.e extent of the influence of calibration on leading decision-making tasks. So combining f-Cal based uncertainty estimation in deep learning with SLAM allows evaluating the effect of calibration on downstream tasks. To

[1]This is shared work with a paper [**1**]. The work was published in ICRA 2022. My contribution to the paper included the theoretical derivation of the method, an initial set of toy experiments, and one set of large-scale experiments

our knowledge, we are one of the first work to assess the effectiveness of calibration for a downstream task.

We chose QuadricSLAM[**41**] as a baseline to build upon, as it works with perceptual input such as object detection. Hence f-Cal based calibrated object detector prefixes the QuadricSLAM seamlessly. Along with that, QuadricSLAM provides a semantic map of the environment, unlike other featured-based SLAM systems such as ORB. Semantic SLAM provides maps of objects, hence calibrated SLAM should allow for object maps with calibrated uncertainty estimates, this can further allow for better belief space planning, but we leave that as possible future work.

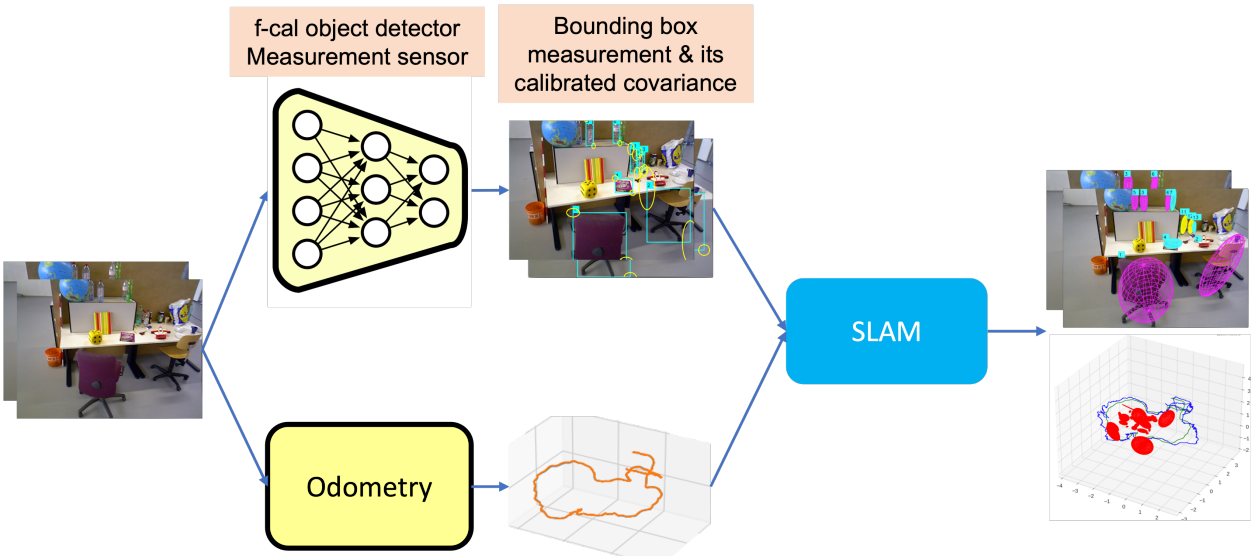


Fig. 1.1. Pipeline for CalibSLAM. CalibSLAM takes calibrated uncertainty estimates of measurement bounding boxes. CalibSLAM provides trajectory and semantic information using quadrics. Odometry can be estimated using sensors on robot or using Visual odometry methods.

Here is a summary of the contributions made in this thesis:

- We propose a distribution matching based loss function to enforce calibration of uncertainty estimates in deep learning - f-Cal which scales well with input dimension, unlike traditional calibration methods such as isotopic regression or temperature scaling.
- We engineer a deep learning based SLAM pipeline with the integration of learned uncertainty estimates, which we call CalibSLAM. See Figure 1.1.
- We evaluate the effectiveness of calibrated uncertainty estimates in deep learning on a downstream task.

Chapter 2

Background

In this chapter we will talk about the core components of this work: Uncertainty Estimation in deep learning (section 2.1) and SLAM (section 2.2). In section 2.1 we describe sources of uncertainty, different methods of uncertainty estimation in deep learning and calibration of uncertainty estimates. In section 2.2 we describe the SLAM formulation using probability and factor graphs. Followed by that, we describe baseline work in section 2.3 in detail.

2.1. Uncertainty Estimation

Uncertainty is a broad and general term used to describe an imperfect state of knowledge or a variability resulting from a variety of factors including, but not limited to, lack of knowledge, physical variation, randomness or stochastic behaviour of the process, and approximation. Depending upon the source or behaviour, uncertainty has been broadly categorized into the following two categories:

- Aleatoric Uncertainty: This uncertainty arises due to inherent stochasticity in the process. An example of this would be coin flip or rolling of dice, as the result of these events depends on stochastic events which are considered out of our control. Aleatoric uncertainty is also known as irreducible.
- Epistemic Uncertainty: This uncertainty arises due to the lack of knowledge about the process. Also known as model uncertainty. Epistemic uncertainty is considered reducible as you can collect more data to increase knowledge about the process. For example, uncertainty about model parameters is epistemic uncertainty because we can collect more data to increase confidence in the parameters.

Uncertainty estimation is the process of predicting/estimating the uncertainty of model predictions. It combines uncertainties in a model, or data and of quantifying their effect on prediction. Uncertainty estimation is an essential step in the evaluation of the robustness of statistical models, especially when applied in risk-sensitive areas.

Probability as a measure of uncertainty

In mathematics, probability theory is the tool to model, represent or work with uncertainties. Hence, we will represent uncertainty in terms of a probability distribution. For example, let's say we have dice represented by a multinomial distribution with known parameters, this distribution represents pure aleatoric uncertainty as multiple outcomes are possible depending on the stochasticity in the process. Now, if the parameters of the multinomial distribution are not known or if the parameters themselves have some distribution, then we have epistemic uncertainty as well because there is a lack of knowledge about the value of the parameters of the dice.
To predict uncertainty in classification is to predict output labels with its confidence and in regression, it's to predict the output with its variance.

2.1.1. Uncertainty Estimation in Deep Learning

In deep learning, we train neural networks parameterized by $\boldsymbol{\theta}$ using dataset $\mathcal{D} = \{(\mathbf{x}_n, \mathbf{y}_n)\}_{n=1}^{N} \sim p^*(\mathbf{y}|\mathbf{x})$, where $p^*(\mathbf{y}|\mathbf{x})$ is the ground truth data-generating distribution, to predict target $\mathbf{y}$ given $\mathbf{x}$ as the input to the network.

In Bayesian deep learning [**55, 17**], instead of learning a point estimate of $\boldsymbol{\theta}$ and predicting a point estimate of $\mathbf{y}$, we try to learn a distribution over $\boldsymbol{\theta}$ and predict a distribution over $\mathbf{y}$. Given dataset $\mathcal{D}$ and some prior $p(\boldsymbol{\theta})$, equations below dictate the bayesian deep learning paradigm:

Learning:

$$p(\boldsymbol{\theta}|\mathcal{D}) = \frac{p(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta})}{p(\mathcal{D})} \propto p(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta}) \tag{2.1.1}$$

So the learning of a neural network requires learning a posterior $p(\boldsymbol{\theta}|\mathcal{D})$over $\boldsymbol{\theta}$ unlike just finding MAP or MLE estimate in vanilla deep learning.

Inference:
Inference means predicting an output y^* for an input x^* given a learned model with parameter $\boldsymbol{\theta}$ on dataset $\mathcal{D}$.

$$p(\mathbf{y}^*|\mathbf{x}^*, \mathcal{D}) = \int_{\boldsymbol{\theta}} \underbrace{p(\mathbf{y}^*|\mathbf{x}^*, \boldsymbol{\theta})}_{\text{Data}} \underbrace{p(\boldsymbol{\theta}|\mathcal{D})}_{\text{Model}} d\boldsymbol{\theta} \tag{2.1.2}$$

The first term in the integral denotes data uncertainty i.e aleatoric uncertainty and the second term denotes model uncertainty i.e epistemic uncertainty. The distribution

$p(\mathbf{y}^*|\mathbf{x}^*, \mathcal{D})$ is called the predictive posterior and denotes predictive uncertainty can be quantified using it.

Below we enumerate different classes of methods in bayesian deep learning to capture different kinds of uncertainty:

- **Bayesian Neural Network**: Model parameters are treated as random variables with an associated probability distribution [**2**]. Techniques based on variational inference, MCMC are used to learn the parameters of the model. Weights are sampled from the distribution to run a forward pass of the network, hence allowing for stochasticity in the network predictions.
- **Ensembles**: Multiple models are trained with different seeds, initialization, etc. The predictions of individual models are aggregated to produce the final prediction. MC-Dropout [**16**] and Deep-Ensembles [**30**] are two major examples of this class of methods.
- **Variance Networks**: This class of methods predict the aleatoric or epistemic uncertainty as the output of the network [**28, 24**]. They are modelled as to predict the parameters of the distribution of the output variable. For example, output modelled as gaussian will have the network predict its mean and variance. The predicted variance is a measure of uncertainty.

2.1.2. Calibration

Calibration is the metric to evaluate the quality of an uncertainty predictor. A predictor is called well-calibrated if the derived predictive confidence represents a good approximation of the actual probability of correctness [**42**]. Calibration error is the metric to evaluate the quality of the predictor, it is the deviation of predicted probabilities from the actual empirical observation.

$$\text{Calibration Error} = | \underbrace{\text{Confidence}}_{\substack{\text{predicted probability} \\ \text{of correctness}}} - \underbrace{\text{Accuracy}}_{\substack{\text{observed frequency} \\ \text{of correctness}}} | \tag{2.1.3}$$

For example, if multiple predictions are made with confidence p, then the predictor is calibrated if exactly p% of predictions are accurate where $p \in [0,1]$ i.e. if multiple predictions have 0.8 confidence then exactly 80% of those predictions should be true and the rest false for a perfectly calibrated classifier/predictor. For regressive predictors, quantiles are used to calculate the confidence.

Expected Calibration Error (ECE) [**42, 18**] is the most used metric used to quantify the calibration of a model/predictor. It is defined as the expected difference between confidence and accuracy i.e.:

$$\text{ECE} = \mathbb{E}_{\hat{P}}[|\mathbb{P}(\hat{Y} = Y|\hat{P} = p) - p|] \tag{2.1.4}$$

To calculate ECE, predictions are partitioned into N equally-spaced bins (size $\frac{1}{N}$) and taking a weighted average of the bin's accuracy/confidence difference. More precisely,

$$ECE = \sum_{n=1}^{N} \frac{|B_n|}{n} |\text{acc}(B_n) - \text{conf}(B_n)| \tag{2.1.5}$$

where B_n is the set of indices of samples whose prediction confidence falls into the interval $I_n = (\frac{n-1}{N}, n/N]$. Accuracy of B_n, $\text{acc}(B_n) = \frac{1}{|B_n|} \sum_{i \in B_n} \mathbf{1}(\hat{y}_i = y_i)$ and average confidence within bin B_n is $\text{conf}(B_n) = \frac{1}{|B_n|} \sum_{i \in B_n} \hat{p}_i$

2.1.3. Related Works

The rapidly growing field of Bayesian deep learning has resulted in the development of models that estimate a distribution over the output space [**15, 16, 28, 30, 2**]. There is a distinction between uncertainty that is due to the stochasticity of the underlying process (aleatoric) versus uncertainty that is due to the model being insufficiently trained (epistemic) [**28**]. Epistemic uncertainty is often estimated by either using ensembles of neural networks or by stochastic regularization at inference time (Monte-Carlo dropout) [**16, 30, 48**]. In this work, epistemic uncertainty is low (i.e., in-distribution setting with reasonably well-trained models such as those common in robot perception), and focuses specifically on calibrating aleatoric uncertainty estimates in regression problems. Such challenging settings have received far less attention in terms of uncertainty estimation [**29, 32, 46, 13**]. Existing calibration techniques are post-hoc and either require a large held-out calibration dataset [**13**] and/or add parameters to the model after training [**58, 13**]. Quantile regression methods [**5, 48, 23, 45, 49**] quantify uncertainty by the fraction of predictions in each quantile. Other methods, such as isotonic regression and temperature scaling, have also been extended to be the regression setting [**29, 13**]. Recently there has been a focus on the calibration of uncertainty estimates in deep neural networks by enhancing the loss function used for training the network which enforces calibration [**19, 1, 13**]. Methods trained with surrogate/enhanced loss functions for calibration are more computationally and data-efficient as they don't need a held-out calibration dataset nor extra parameters.

2.2. SLAM

Simultaneous Localization and Mapping (SLAM) is considered one of the hardest problems in robotics due to its chicken-or-egg paradigm. SLAM is a process by which robots build a map of their unknown environment and simultaneously localize themselves in the environment. The robot is equipped with different kinds of sensors to make a map and localize itself.

Two of the most important modalities of the sensor for SLAM are the odometry sensor and the environment-perceiving sensor. Odometry sensors such as encoder and IMU give relative pose transformation of the robot between two time-steps. An environment-perceiving sensor such as Lidar, RADAR, and camera gives information about landmarks and features in the environment relative to the current pose of the robot, which helps build the map.

2.2.1. Formulation

We will formulate one of the most used formulations of SLAM: Landmark SLAM [**8, 52**]. In this, we have to estimate the robot's trajectory as well as the position of landmarks. Landmarks can be any fixed features in the environment i.e unique objects, identifiers, etc.

Let the robot pose be x_t for $t \in 0, 1 \ldots, T$, and there be N landmarks with pose l_j for $j \in 0, 1 \ldots, N$, robot will also be doing measurements at each pose for different landmarks, measurements are denoted by z_k for $k \in 0,1, \ldots, K$. Now the goal of SLAM is to estimate $X = x_{0:T}$ and $L = l_{0:N}$ given the measurements $Z = z_{0:K}$, mathematically:

$$X^*, L^* = \arg\max_{X,L} P(X, L|Z) \tag{2.2.1}$$

$$\propto \arg\max_{X,L} P(Z|X, L)P(X,L) \tag{2.2.2}$$

$$\propto \arg\max_{X,L} P(x_0) \prod_{t=1}^{T} P(x_t|x_{t-1}) \prod_{k=0}^{K} P(z_k|x_{t_k}, l_{j_k}) \tag{2.2.3}$$

where $P(x_0)$ is a prior on the initial state, $P(x_t|x_{t-1})$ is the motion model (given by odometry sensor), and $P(z|x, l)$ is the landmark measurement model. The above assumes a uniform prior over the landmarks l. Furthermore, it assumes that the data-association problem has been solved, i.e., that the indices t_k and j_k corresponding to each measurement z_k are known.

2.2.2. Factor Graph

A factor graph [**8**] is a graphical model used in probabilistic modelling. It's a bipartite graph where nodes represent unknown variables or factors. Unknown variables in our case can be robot pose or landmark pose and the factor is a function over the subset of these unknown variables, constraining these variables. Edges in the factor graph are always between factors and variables since it's a bipartite graph, and indicate that a particular factor depends on a particular variable.

The node with names is an unknown variable to be estimated. The black dots are factor nodes which connect variables and denote measurement between variables. The factor node between two robot pose variables denotes an odometry measurement and the factor node between robot pose and landmark denotes a landmark measurement. The factor node connecting to the only single variable is a prior factor. So in the Fig. 2.1 we have three

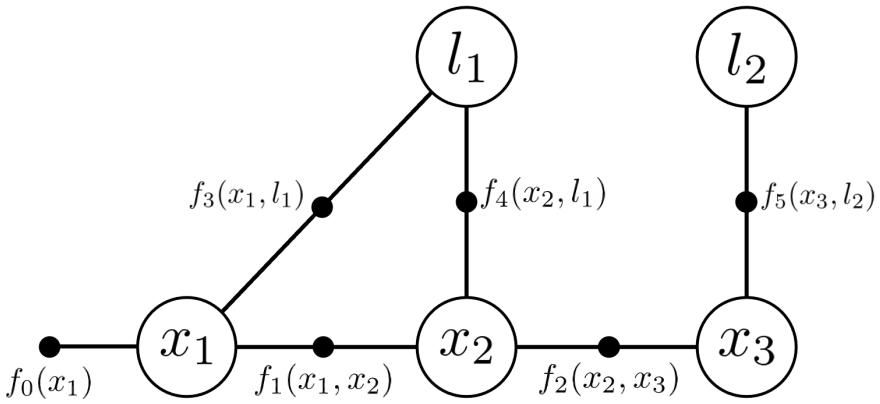


Fig. 2.1. Factor graph representing Landmark-SLAM

types of factor: f_0 is a prior factor connection to only one variable, it generally denotes fixed measurement i.e constraining the variable to some constant value, f_1, f_2 are odometry measurement, giving information about the relative movement between two consecutive poses, f_3, f_4, f_5 denotes landmark measurements which are the relative position of landmarks with respect to robot poses.

A factor graph is just the specification of the probability density $P(X,L|Z)$ using factors which denote independent conditional likelihood of measurements. Formally, the joint density of a factor graph with factor nodes φ_i and variable nodes θ_j is given by:

$$P(\Theta) = \prod_{i=1}^{N} \varphi_i(\Theta_i) \tag{2.2.4}$$

where Θ is the set of all unknown variables θ_j in the graph and Θ_i is set of variables θ_j adjacent to φ_i. Each factor φ_i is a function of variables in Θ_i, determining the measurement or constraint between variables in Θ_i.

Extending the general factor graph to formulate the SLAM problem, we see that Equation 2.2.4 is equivalent to Equation 2.2.1. We have three types of factors: $\varphi_{(t-1)t}(x_{t-1}, x_t)$ denoting odometry measurement, $\varphi_{t_k j_k}(x_{t_k}, l_{j_k})$ denoting landmark measurement and $\varphi_i(\theta_i)$ denoting prior on single variable θ_i where θ_i can be x_t or l_j. Using this Equation 2.2.4 becomes

$$P(\Theta) = \varphi_0(x_0) \prod_{t=1}^{T} \varphi_{(t-1)t}(x_{t-1}, x_t) \prod_{k=1}^{K} \varphi_{t_k j_k}(x_{t_k}, l_{j_k}) \tag{2.2.5}$$

The equivalence between equations 2.2.1 and 2.2.5 can be established by taking:

$$\varphi_0(x_0) \propto P(x_0) \tag{2.2.6}$$

$$\varphi_{(t-1)t}(x_{t-1}, x_t) \propto P(x_t|x_{t-1}) \tag{2.2.7}$$

$$\varphi_{t_k j_k}(x_{t_k}, l_{j_k}) \propto P(z_k|x_{t_k}, l_{j_k}) \tag{2.2.8}$$

SLAM as a Non-Linear Least Squares Problem

Here we assume that the odometry and landmark measurement models follow the Gaussian noise model, defined by:

$$x_t = f(x_{t-1}) + w_t \;\; \Leftrightarrow \;\; P(x_t \mid x_{t-1}) = \frac{1}{\sqrt{2\pi|\Lambda_t|}} \exp -\frac{1}{2}||f(x_{t-1}) - x_t)||^2_{\Lambda_t} \tag{2.2.9}$$

where $f(\cdot)$ is the motion model and w_t is normally distributed noise mean zero and covariance Λ_t.

$$z_k = h(x_{t_k}, l_{j_k}) + v_k \;\; \Leftrightarrow \;\; P(z_k \mid x_{t_k}, l_{j_k}) = \frac{1}{\sqrt{2\pi|\Sigma_k|}} \exp -\frac{1}{2} \|h(x_{t_k}, l_{j_k}) - z_k\|^2_{\Sigma_k} \tag{2.2.10}$$

where $h(\cdot)$ is the measurement model, and v_k is the normally distributed measurement noise with mean zero and covariance Σ_k. Above $||e||^2_\Sigma \triangleq e^T\Sigma^{-1}e$ is the Mahalanobis distance.

Following the above, we derive the factors:

$$\varphi_{(t-1)t}(x_{t-1}, x_t) = \exp -\frac{1}{2}||f(x_{t-1}) - x_t)||^2_{\Lambda_t} \tag{2.2.11}$$

$$\varphi_{t_k j_k}(x_{t_k}, l_{j_k}) = \exp -\frac{1}{2} \|h(x_{t_k}, l_{j_k}) - z_k\|^2_{\Sigma_k} \tag{2.2.12}$$

Now we can show that maximum a posteriori inference in factor graphs is equivalent to non-linear least square minimization. Using Equation 2.2.5

$$\begin{aligned} X^*, L^* &= \arg\max_{X,L} \prod_{t=1}^{T} \varphi_{(t-1)t}(x_{t-1}, x_t) \prod_{k=1}^{K} \varphi_{t_k j_k}(x_{t_k}, l_{j_k}) \\ &= \arg\min_{X,L} -(\sum_{t=1}^{T} \log \varphi_{(t-1)t}(x_{t-1}, x_t) + \sum_{k=1}^{K} \log \varphi_{t_k j_k}(x_{t_k}, l_{j_k})) \\ &= \arg\min_{X,L} \sum_{t=1}^{T} ||f(x_{t-1}) - x_t)||^2_{\Lambda_t} + \sum_{k=1}^{K} \|h(x_{t_k}, l_{j_k}) - z_k\|^2_{\Sigma_k} \end{aligned} \tag{2.2.13}$$

Regarding the prior $P(x_0)$, we will assume that x_0 is given and hence it is treated as a constant. Since the origin of a coordinate system is arbitrary we can fix x_0 at the origin.

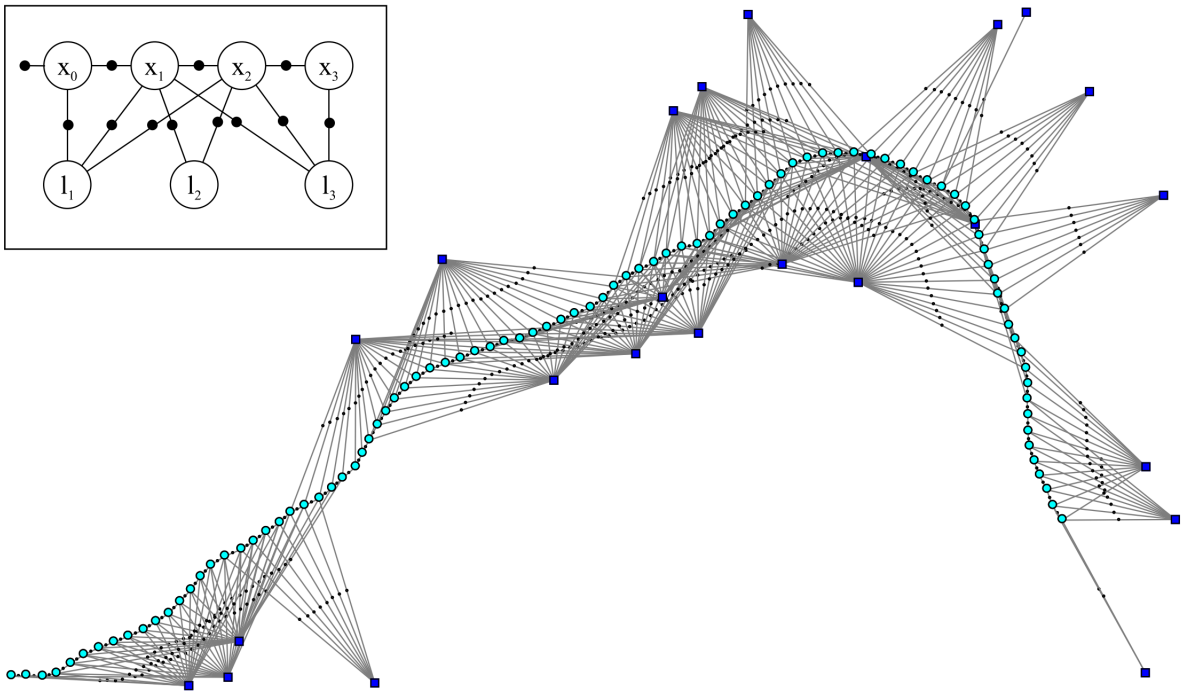


Fig. 2.2. Spatial representation of factor graph for Landmark SLAM. Blue nodes represent landmark positions and cyan nodes represents robot position.

Hence, the above formulation is **non-linear least square minimization**.

Covariance Tuning

As shown in Equation 2.2.13, the minimization problem depends upon the motion and measurement noise covariance, which are design variable and needs to be given or estimated. For optimal solutions, covariances need to be estimated accurately. With traditional sensors such as LIDAR, IMU, etc., there are procedures for sensor calibration which can be used to estimate sensor noise covariances. But with modern deep learning-based sensors such as object detection, depth regression, etc., calibrated covariance estimation is a difficult task. And since covariance can be seen as weighting the different error terms, accurate estimation of covariances such that they reflect the error in the measurement and motion model accurately is a very important task. Hence, the optimization is highly sensitive to the covariance values.

2.2.3. Related Works

Initial works in Visual SLAM used key points/features extracted from images [**39, 14**]. These methods largely deal with dense feature-based map i.e., the predicted map after optimization is key points in the environment. Recently there has been a lot of work on semantic SLAM which optimizes for the semantic map of the environment i.e objects in the environment [**57, 41, 38**].

With the advances of deep learning, there has been a lot of work incorporating deep learning with SLAM [**53, 25, 51, 6**]. SLAM system has different components such as feature detector and descriptor, optimization, etc. Initial incorporation of deep learning into SLAM dealt with better feature extraction and matching leveraging the better representation capabilities of CNNs [**35, 22**] and for visual odometry [**53, 54**]. Deep learning enables estimating the semantics of a scene such as location/shape/size of the objects in a scene using methods such as object detection, pose estimation, etc. Hence, deep learning can aid significantly semantic SLAM which is a useful paradigm for downstream tasks such as task planning, manipulation, etc. Recently work has been done to leverage deep learning methods such as 2D/3D object detection and pose estimation for better semantic SLAM [**41, 57, 37**]. One of the important works which we use as a baseline in our work uses a deep learning-based object detector as a backbone and combine the perceptual output of the object detector with semantic SLAM pipeline [**41**].

2.3. QuadricSLAM

Semantic SLAM is a SLAM framework which enables encoding rich semantic information about the environment such as objects, etc. Traditionally, semantic slam has been limited because of a lack of semantic measurement models, but with the advance of deep learning, semantic information about the environment such as different objects, their location, etc is easily available. QuadricSLAM [**41**] is a semantic SLAM framework which takes 2D object detections as the measurement model and jointly estimates camera poses and quadric parameters of the objects. QuadricSLAM uses the quadric representation to represent 3D landmarks. A quadric is a 3D ellipsoid, which can encode information such as the size, position, and orientation of an object in its representation. The work exploits the fact that quadrics can be estimated using bounding boxes from multiple views. In SLAM, since each landmark is viewed multiple times along the trajectory, it allows objects to be represented and computed as quadrics.

2.3.1. Quadric Parametrization

Quadrics are represented by a 4×4 matrix which represents 3D surfaces. A quadric can also be defined by a set of tangential planes such that all the planes form an envelope around the quadric, this is called a dual form of quadric. A quadric has 9 degrees of freedom which corresponds to 10 independent elements of the matrix representation minus one for the scale. So we will denote a quadric in matrix form by $\mathbf{Q}^*$ and in a parametric vector form by $\hat{\mathbf{q}} = (\hat{q}_1, \ldots, \hat{q}_{10})$.

Since quadrics can represent any sphere, ellipsoid, hyperboloid, cones, etc and we want to restrict a quadric to represent either ellipsoids or spheres, they show constrained quadric parametrization as below:

$$\mathbf{Q}^* = \mathbf{Z}\bar{\mathbf{Q}}^*\mathbf{Z}^\mathbf{T} \tag{2.3.1}$$

where $\bar{Q}^*$ is an ellipsoid at the origin with given radii and Z determines the homogenous transformation which accounts for the translation and rotation of the quadric. i.e

$$\mathbf{Z} = \begin{pmatrix} \mathbf{R}(\theta) & \mathbf{t} \\ \mathbf{0}_3^T & 1 \end{pmatrix} \quad \text{and} \quad \bar{\mathbf{Q}}^* = \begin{bmatrix} s_1^2 & 0 & 0 & 0 \\ 0 & s_2^2 & 0 & 0 \\ 0 & 0 & s_3^2 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix} \tag{2.3.2}$$

where $\mathbf{t} = (t_1, t_2, t_3)$ is quadric translation vector, $\mathbf{R}(\theta)$ is rotation matrix defined by angles $\theta = (\theta_1, \theta_2, \theta_3)$ and $\mathbf{s} = (s_1, s_2, s_3)$ determined the length of semi-axes of ellipsoid. Combining all these parameters, a quadric represented by $\mathbf{q} = (\theta_1, \theta_2, \theta_3, t_1, t_2, t_3, s_1, s_2, s_3)$.

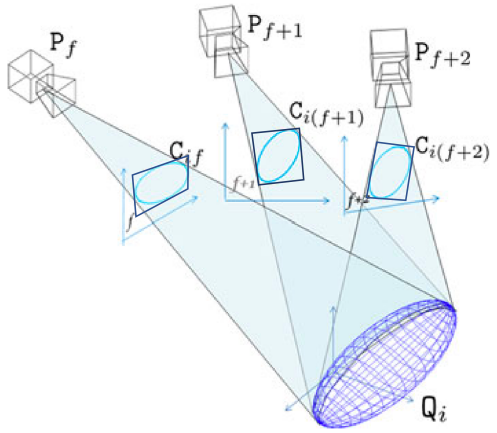


Fig. 2.3. Estimating quadric using multi-view bounding boxes

2.3.2. Quadric Estimation

A quadric of an object can be estimated with multiple views given the bounding box of the object in multiple views, see Figure 2.3. In this section, we detail how to estimate a quadric given bounding boxes in images and their poses. Quadrics are surfaces in 3D space that are represented by a 4×4 symmetric matrix Q. In dual form, a quadric is defined by a set of tangential planes such that the planes form an envelope around the quadric. This dual quadric Q^* is defined so that all planes π fulfill $\pi^T Q^* \pi = 0$. Given poses x_i, and enclosing bounding box b_i we will use the mentioned equation to find parametric vector q of matrix Q.

We can form the homogeneous vectors defining the planes π_{ik} using the landmark bounding box observations b_i and resulting lines l_{ik} by projecting them according to $\pi_{ik} = P_i^T l_{ik} = [\pi_1, \pi_2, \pi_3, \pi_4]$, where P_i is the projection matrix of pose x_i. Using all this we can find the parametric vector q which defines dual quadric Q^* by solving the equations below:

$$(\pi_1^2, 2\pi_1\pi_2, 2\pi_1\pi_3, 2\pi_1\pi_4, \pi_2^2, 2\pi_2\pi_3, 2\pi_2\pi_4, \pi_3^2, 2\pi_3\pi_4, \pi_4^2) \cdot (q_1, \ldots, q_{10})^T = 0 \tag{2.3.3}$$

We can get multiple equations using different poses and bounding boxes i and solve the system of equations using the least squares.

2.3.3. Object Detector Measurement Model

In SLAM, odometry and measurement factors constitute the non-linear optimization formulation. For landmark measurement optimization, we have to derive the measurement model $h(x_t, l_j)$ as shown in Equation 2.2.10. Since the measurement model is a function of the current pose and quadric, we have to derive a formulation to get the bounding box as a measurement model from the pose and quadric. So in this section, we will derive a bounding box measurement model $h(x_t, q_j)$. Given a quadric q_j and camera pose x_t, you can project the quadric on the image plane which will be a dual conic $\mathbf{C}^*$ denoted as $\mathbf{C}^* = \mathbf{P_t Q^*_{q_j} P_t^T}$. $\mathbf{P_t}$ is the projection matrix computed as $\mathbf{P_t} = \mathbf{K}[\mathbf{R}|\mathbf{t}]$ using intrinsic matrix $\mathbf{K}$ and camera pose parameters $\mathbf{R}, \mathbf{t}$ given by x_t. We can take the adjugate of the dual conic $\mathbf{C}^*$ to get its primal $\mathbf{C}$. Once we have a conic C, finding the enclosing bounding box of the conic C, gives us the bounding box b_{tj}. Hence the bounding box measurement model is derived as:

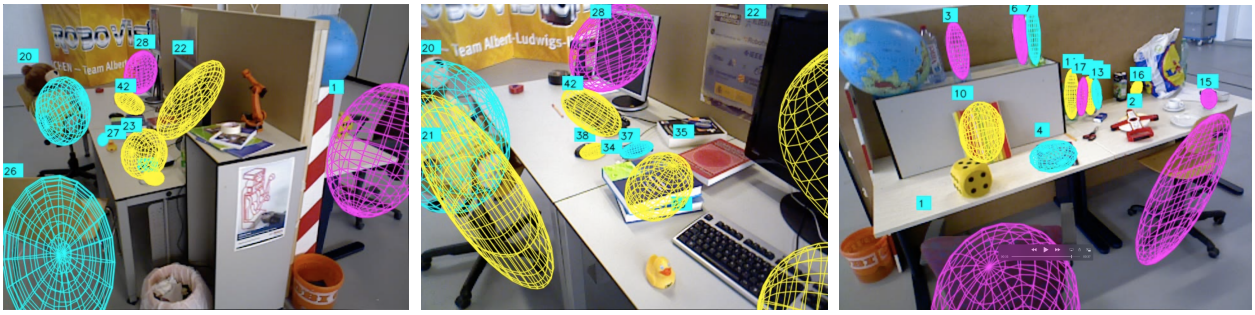
$$h(\mathbf{x_t}, \mathbf{q_j}) = \text{BBox}(\text{adjugate}(\mathbf{P_t Q^*_{q_j} P_t^T})) = b_{tj} \tag{2.3.4}$$

2.3.4. Optimization

Using the optimization framework derived in Equation 2.2.13, in the QuadricSLAM framework and replacing general landmark notation $\mathbf{L}$ with quadric notation $\mathbf{Q}$, we will get following QuadricSLAM optimization:

$$X^*, Q^* = \underset{X,Q}{\text{argmin}} \underbrace{\sum_{t=1}^{T} ||f(x_{t-1}) \ominus x_t)||^2_{\Lambda_t}}_{\text{Odometry factors}} + \underbrace{\sum_{tj} \|h(x_t, q_j) - b_{tj}\|^2_{\Sigma_{tj}}}_{\text{Quadric Landmarks factors}} \tag{2.3.5}$$

We solve the above non-linear least square minimization problems using non-linear optimizers such as Levenberg-Marquardt or Gauss-Newton solvers. Noise parameters such as Σ_t, Λ_{tj} are considered hyper-parameters and have to be tuned manually or using some heuristic.



(a) (b) (c)

Fig. 2.4. Quadric Representation of 3D object Landmarks

Chapter 3

Uncertainty Calibration for Optimality

In this chapter, we aim to theoretically motivate that uncertainty estimates to need to be calibrated for factor graph optimization to reach the true optimal solution. To do so, we will first demonstrate that calibration is necessary for Kalman filters [**27**] to optimally solve as they are simpler methods for state estimation and have been studied extensively in regard to uncertainty estimation. Kalman filters can be thought of as an instance of a factor graph under certain equivalence conditions. By showing the importance of calibration for Kalman filters, we can then generalize this conclusion to factor graphs in general. We use Kalman filters as a starting point for our theoretical analysis of uncertainty calibration because they are a simple and well-studied class of probabilistic inference methods, and have been applied to solve SLAM problems. Through this approach, we aim to theoretically demonstrate the need for calibrated uncertainty estimates in factor graph optimization in order to ensure that the true optimal solution is reached.

3.1. Kalman Filter and Covariance Optimality

In this section, we theoretically show under some assumptions that calibrated uncertainty estimates are a requirement for optimization to give a truly optimal solution. We show how calibrated uncertainty estimates/covariance is required for optimization to work in Kalman Filters.

Kalman Filtering [**27**] is a widely studied technique for state estimation in a wide range of dynamical systems, it is one of the basic approaches towards SLAM as well. Similar to Factor Graphs, Kalman Filter also has a motion model/prediction model denoted by $x_{t+1} = f(x_t)$ which predicts the next state given the current state and a measurement model $z = h(x)$. For a standard Kalman Filter, these functions are assumed linear, with a more general method called extended Kalman filter that considers non-linear motion and measurement model as well.

The Kalman filter works by iteratively applying two steps, predict and update. Optimality of the Kalman filter assumes additive Gaussian noise with zero mean and covariances on both motion and measurement models, which need to be given to the filter. Kalman Filter equations are as follows:

$$\begin{aligned} \mathbf{x}_t &= \mathbf{F}\mathbf{x}_{t-1} + \mathbf{w}_t \\ \mathbf{z}_t &= \mathbf{H}\mathbf{x_t} + \mathbf{v}_t \end{aligned} \tag{3.1.1}$$

where $\mathbf{F}$ is the state transition model, $\mathbf{w}_t$ is the process noise which is assumed white noise with covariance $\mathbf{Q}$, $\mathbf{w}_t \sim \mathcal{N}(0, \mathbf{Q})$. $\mathbf{H}$ is the measurement model with $\mathbf{v}_t$ as the measurement noise which is assumed white noise with covariance $\mathbf{R}$, $\mathbf{v}_t \sim \mathcal{N}(0, \mathbf{R})$.

Kalman filter uses a two-step methodology to estimate the current state. The first step is to predict the next state using the transition/motion model given the current state, hence getting the predicted state estimate, then next is the update/correction step which combines the predicted state and state estimates using the measurement taken. Kalman filter also keeps track of the state covariance matrix $\mathbf{P_t}$ which represents uncertainty in the state estimate.

Predict :

We get a prior estimate from the previous state estimate using the transition model.

$$\begin{aligned} \hat{\mathbf{x}}'_t &= \mathbf{F}\hat{\mathbf{x}}_{t-1} \\ \mathbf{P'}_t &= \mathbf{F}\mathbf{P'}_{t-1}\mathbf{F}^T + \mathbf{Q} \end{aligned} \tag{3.1.2}$$

Update :

In the update equation, the prior estimate is then updated using the current measurement $\mathbf{z_t}$ to obtain the posterior estimate:

$$\begin{aligned} \hat{\mathbf{x}}_t &= \hat{\mathbf{x}}'_t + \mathbf{K}_t\left(\mathbf{z}_t - \mathbf{H}\hat{\mathbf{x}}'_t\right) \\ \text{and} \quad \mathbf{P}_t &= \left(\mathbf{I} - \mathbf{K}_t\mathbf{H}\right)\mathbf{P'}_t \\ \text{with } \mathbf{K}_t = \mathbf{P'}_t\mathbf{H}^T\left(\mathbf{H}\mathbf{P'}_t\mathbf{H}^T + \mathbf{R}\right)^{-1} &= (\mathbf{F}\mathbf{P'}_{t-1}\mathbf{F}^T + \mathbf{Q})\mathbf{H}^T\left(\mathbf{H}(\mathbf{F}\mathbf{P'}_{t-1}\mathbf{F}^T + \mathbf{Q})\mathbf{H}^T + \mathbf{R}\right)^{-1} \end{aligned} \tag{3.1.3}$$

where $\mathbf{I}$ denotes the identity matrix. The matrix $\mathbf{K}_t$ is referred to as the Kalman gain. The Kalman gain is optimal to minimize the mean square error of the state estimate.

Here the quantity of interest for us is the covariance matrix $\mathbf{Q}$ and $\mathbf{R}$ on which the Kalman gain depends. We need $\mathbf{Q}$ and $\mathbf{R}$ to be calibrated i.e should be an accurate variance of error of prediction and measurement model for optimal solution i.e

solution corresponding to the minimum mean squared error. For example, in a system, let actual noise covariance be $\mathbf{Q}_a$ and $\mathbf{R}_a$, but durign solving the state estimation using kalman filter, if we chose some other covariance matrices $\mathbf{Q}$ and $\mathbf{R}$ such that $\mathbf{Q}_a \neq \mathbf{Q}, \mathbf{R}_a \neq \mathbf{R}$, then we will estimate the Kalman gain on which the prediction depends we will get $\mathbf{K}_t = (\mathbf{F}\mathbf{P'}_{t-1}\mathbf{F}^T + \mathbf{Q})\mathbf{H}^T \left(\mathbf{H}(\mathbf{F}\mathbf{P'}_{t-1}\mathbf{F}^T + \mathbf{Q})\mathbf{H}^T + \mathbf{R}\right)^{-1} \neq (\mathbf{F}\mathbf{P'}_{t-1}\mathbf{F}^T + \mathbf{Q}_a)\mathbf{H}^T \left(\mathbf{H}(\mathbf{F}\mathbf{P'}_{t-1}\mathbf{F}^T + \mathbf{Q})\mathbf{H}^T + \mathbf{R}_a\right)^{-1} \neq \mathbf{K}_a$, hence the calculated kalman gain will be wrong and leading to wrong prediction.

Following this, we will show how the Kalman filter can be seen as an instantiation of the factor graph under certain specific conditions, showing that both methods have a similar derivation of optimization cost and behave the same from the perspective of uncertainty calibration.

3.2. Kalman Filter as an instantiation of Factor graph

The Kalman filter can be seen as an instance of a factor graph under the following conditions:

- The motion model and measurement models are linear.
- The random noise in the motion and measurement models is Gaussian.

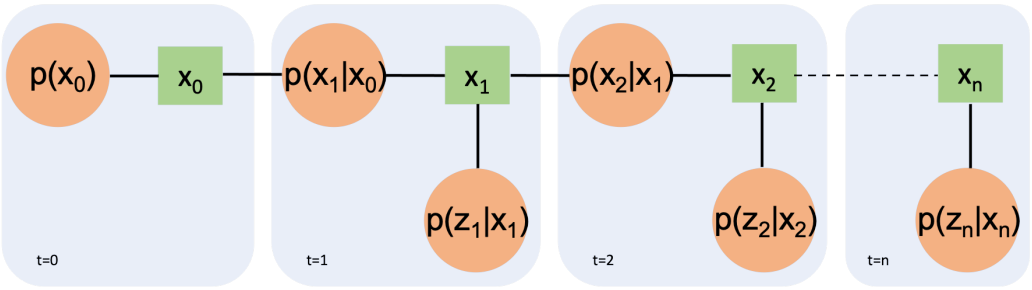


Fig. 3.1. Kalman Filter as a factor graph. In factor graph optimization, all the state variables are jointly optimized, whereas Kalman Filter optimizes the state variables recursively.

3.2.1. Kalman Filter Optimization as Factor graph Optimization

In this section we will derive the Kalman Filter optimization from the perspective of factor graph optimization, showing that the Kalman filter and factor graph optimization are equivalent under the conditions mentioned above.

Kalman filters can be derived as a minimum least squares optimization problem as follows:

Let $\mathbf{x}_t$ be the state of the system at time t, and $\mathbf{z}_t$ be the measurement of the system at time t. The goal of the Kalman filter is to estimate the state $\mathbf{x}_t$ based on the measurement $\mathbf{z}_t$ and the previous state estimate $\mathbf{x}_{t-1}$. The relationship between state transitions and measurement is expressed in Equation 3.1.1.

The Kalman filter seeks to find the state estimate $\mathbf{x}_t$ that minimizes the error between the estimated state and the measurement, subject to the constraints of the state transition model and the measurement model. This can be expressed as a least squares optimization problem:

$$\mathbf{x}_t = \arg\min_{\mathbf{x}} ||\mathbf{z}_t - H\mathbf{x}||^2_{\mathbf{R}} + ||\mathbf{x} - F\mathbf{x}_{t-1}||^2_{\mathbf{Q}+\mathbf{F}\mathbf{P}_{\mathbf{t}-\mathbf{1}}\mathbf{F}^{\mathbf{T}}} \tag{3.2.1}$$

This optimization problem can be solved using the Kalman filter algorithm, which iteratively updates for each time step t, the state estimate $\mathbf{x}_t$ based on the measurement $\mathbf{z}_t$ and the previous state estimate $\mathbf{x}_{t-1}$. Solving the above least square problem will give the same solution for $\mathbf{x}_t$ as shown in Equation 3.1.3

We can see the equivalence between the cost function for optimization of Kalman filter as shown in Equation 3.2.1 and factor graph optimization as shown in eq. 2.2.4. Both of these equations have similar terms which are the transition term error weighted by its uncertainty and the measurement prediction error weighted by its uncertainty. Both of these algorithms solve the same problem from the perspective of state estimation and how the uncertainty optimality affects them, the only difference being factor graph solves the problem at the global scale i.e solve for all the $\mathbf{x}_t \forall t$ jointly whereas the Kalman filter solves recursively i.e one-time step at a time then use a solution of the previous time step for the current time step. But both of these algorithms have the same cost function from the perspective of covariance matrices. Since the cost function is similar and we have shown in Section 3.1 that Covariance matrices need to be calibrated for the Kalman filter to be optimal, we can generalize the similar argument about factor graphs.

Chapter 4

Calibrated Uncertainty for SLAM

In this chapter, we will develop a method to calibrate predictive uncertainties in deep neural networks. In section 4.1 we picture deep neural networks as a sensor modality from the perspective of robotics. Following, in section 4.2 we discuss the lack of calibration in neural networks predicting uncertainty. We use distributional calibration to motivate the derivation of a loss function which we call f-cal that can enforce calibration of uncertainty estimates in deep neural networks. In section 4.3 we then integrate the f-cal based neural network measurement model with the QuadricSLAM framework to have CalibSLAM. CalibSLAM uses calibrated distribution prediction of bounding boxes from the network and uses it as an input to solve the Object SLAM problem using factor graphs.

4.1. Deep neural network as a sensor

In this section, we motivate deep neural networks from the perspective of a sensor modality that may directly be used in probabilistic planning or sensor fusion.

Deep Learning allows us to make use of high-dimensional modalities of data as measurements such as images, audio, tactile etc which gives information-rich output. This has caused a shift towards using deep learning based methods more in robotics. Traditionally, a measurement sensor gives information about the state of the robot, such as LiDAR, IMU, GPS, and RADAR. It's denoted by $z = h(x, \nu)$, i.e sensor output is a measurement z which is a function of the current state of robot x with some noise measurement noise denoted by ν. Deep learning models such as object detection networks, depth estimation networks and segmentation networks are an instantiation of deep neural networks as sensors, where object detections, and depth estimates are sensor outputs and the robot state is defined by image as input.

While modern deep neural networks are performant perception modules, performance (accuracy) alone is insufficient, particularly for safety-critical robotic applications such as self-driving vehicles. Robot autonomy stacks also require these otherwise black box models

to produce reliable and calibrated measures of uncertainties on their predictions, denoted by ν in the measurement model equation. When using sensors like LiDAR, IMU, and GPS it was easy to find the measurement noise ν by sensor calibration methods. In sensor calibration, multiple trials of the same measurement are supposed to be done, allowing for computation of the sensor noise parameter also known as the sensor covariance matrix, usually, it's a constant property of the sensor for fixed settings/environment. Since training data in a deep learning has single ground truth label, calibration of deep neural networks using similar methods as for traditional sensors is not feasible. Existing approaches estimate uncertainty from these neural network perception stacks by modifying network architectures, inference procedures, or loss functions. However, in general, these methods lack calibration, meaning that the predictive uncertainties do not faithfully represent the true underlying uncertainties (process noise). To this end, we develop a loss based on distribution matching which enforces that predicted distribution matches with data distribution which lead to calibrated uncertainty estimates in variance networks.

4.2. f-cal: Calibrated uncertainty estimation for regression tasks

In this section, we explain the method developed for calibrated uncertainty prediction in deep neural networks in regression settings which allows neural network models to be used as replacements for measurement sensors.[1]

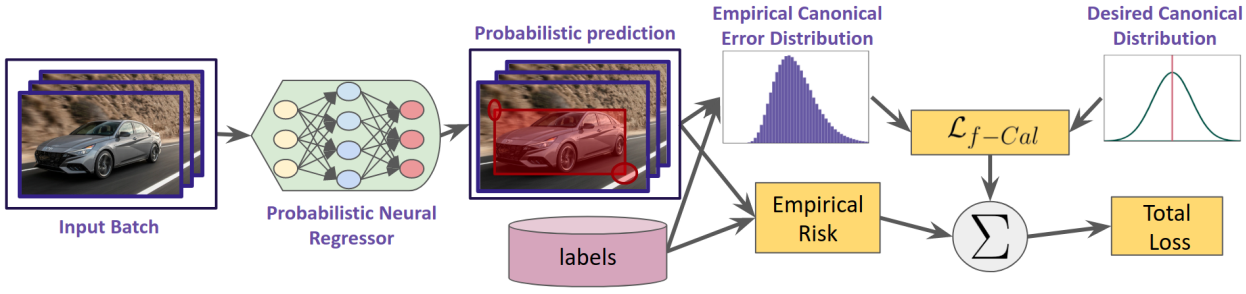


Fig. 4.1. f-cal pipeline

4.2.1. Uncertainty Calibration

Calibrated uncertainty estimates are those where the output uncertainties can be exactly interpreted as confidence intervals of the underlying target label distribution. This allows uncertainty estimates across multiple samples (and models) to be compared. Intuitively, we

[1]This section contains my contribution done in the work [**1**] which is published in ICRA 2022. I contributed to the mathematical derivation of calibration as distributional matching and sets of initial toy experiments to confirm the hypothesis

understand the notion of uncertainty calibration to mean that if we repeated a stochastic experiment a large number of times, for example by asking many different people to label the same image, the "label generating distribution" matches the predictive distribution of the model:

$$y_i \sim f_p(x_i) \tag{4.2.1}$$

However, in practice, it is impractical to label every piece of data multiple times. Instead, we wish to aggregate the labels across many different inputs to produce calibrated predictive distributions. Using our definitions from Sec. 2.1.2 and adapting from [**46**], we can define what we desire in terms of calibration in the case of a deep neural regressor as follows:

Definition 4.2.1 (**Uncertainty Calibration**). *A neural regressor f_p is calibrated if and only if:*

$$p(Y \leq y | s(y)) = \int_{-\infty}^{y} s(y')dy' \quad \forall y \in Y \tag{4.2.2}$$

In the above definition, Y is an instantiation of the random variable y. If we can assume that the noise is sampled from a parametric distribution $s(y; \phi)$, then the probabilistic model needs only output the parameters associated with each sample. In this case, we can consider the model to be calibrated if and only if the aggregated error statistics over multiple outputs of a model align with the parameters predicted by the model.

4.2.2. Calibration as Distribution Matching

Following the definition of distributional calibration (Def. 4.2.1), f-Cal formulates a variational minimization objective to calibrate the uncertainty estimates from a deep neural network regression model.

In the case of a traditional (non-deep learning based) sensor, we would calibrate the noise distribution with the procedure:

(1) Choose a distribution family for the noise

(2) Hold the input fixed at some known value and sample the output many times

(3) Fit the output samples to the distribution family

In the deep neural network as a sensor (DNNS) case, we only have one sample for any given input and we have no knowledge of the ground truth (noise free) label. We can similarly choose a distribution family for our model, but we cannot assume that any of the parameters are fixed across samples. Our approach to overcome this problem will be to assume that there is some canonical element of the distribution family that we can transform each predictive distribution to. Specifically, we seek to approximate the empirical posterior over some canonical transformation of the target variables Y by a simpler (tractable) target

distribution Q (modelling choice). This enables us to leverage an abundant class of distribution matching metrics, f-divergences, to formulate a loss function enforcing distributional calibration.

We assume that we can transform each training sample output distribution to some canonical element of the distribution family. For instance, Gaussian random variables are canonicalized by centering the distribution (subtracting the output label), followed by normalization (scaling the result by the inverse variance). These canonical elements are used (in conjunction with the labels) to determine the *empirical* error distribution. f-Cal then performs distribution matching across this empirical and target distribution.

Motivation

In this section, we show how to form a loss function to enforce the cumulative density function (CDF) of the random variable. For motivation, we take an example of a Gaussian variable as many regression problems have this assumption but this method can be generalized to any distribution of random variables as shown in Theorem 1 in [**46**].

Eq. 4.2.1 for a Gaussian variable $Y \sim \mathcal{N}(\mu, \sigma^2)$ can be be written as

$$\begin{aligned} P(\mu - \sqrt{2}\text{erf}^{-1}(p)\sigma \leq Y \leq \mu + \sqrt{2}\text{erf}^{-1}(p)\sigma) = p, \forall p \in [0,1] \\ P(-n \leq \frac{Y-\mu}{\sigma} \leq n) = p \\ P(-n \leq Z \leq n) = p \end{aligned} \tag{4.2.3}$$

where $\text{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt$ is the special function, $p = erf(\frac{n}{\sqrt{2}})$ and $Z = \frac{Y-\mu}{\sigma}$. The Eq. 4.2.3 constrains the CDF over variable Z which comes from the condition of calibration.

Since Eq. 4.2.3 is the standard CDF equation for standard normal variable we can easily show that for Eq. 4.2.1 to hold true we need:

$$Z \sim \mathcal{N}(0,1) \tag{4.2.4}$$

Hence, we can conclude that enforcing $Z \sim \mathcal{N}(0,1)$ will enforce the calibration of the model.

4.2.3. f-cal algorithm

Given a mini-batch containing N inputs x_i, a probabilistic regressor predicts N sets of distributional parameters $f_p(x_i) = \phi_i (\phi_i \in \Phi)$ to the corresponding probability distribution $s(y_i; \phi_i)$. Define $g : Y \times \Phi \mapsto Z$ as the function that maps the target random variable y_i to a random variable z_i which follows a known canonical distribution. Since these residuals $\{z_1, z_2, \ldots, z_N\}$ must ideally follow a chosen calibrating (target) distribution Q:

$$z_i = g(y_i, \phi_i) \sim Q \tag{4.2.5}$$

The key difference between (4.2.1) and (4.2.5) is that (4.2.5) now applies **for all samples** in the dataset, as opposed to just a single sample. As a result, we can now follow the similar procedure that we would with a traditional sensor and compute the empirical statistics of the residuals of the z_i variables across the entire set (or in practice across a mini-batch) to fit a proposal distribution P_z, and minimize the distributional distance from the canonical distribution Q. This minimization can be performed with a variational loss function that minimizes an f-divergence, $D_f(P_z||Q)$, between these two distributions.

In summary, we propose a distribution matching loss function that augments typical supervised regression losses, and results in the neural regressor being calibrated to the target distribution:

$$\begin{aligned}\mathcal{L} &= (1-\lambda)R_{emp}(f_p) + \lambda\mathcal{L}_{f\text{-Cal}} \\ &= (1-\lambda)R_{emp}(f_p) + \lambda D_f(P_z||Q)\end{aligned} \tag{4.2.6}$$

where λ is a hyper-parameter to balance the two loss terms.

We experiment with a number of f-divergence choices and identify KL-divergence and Wasserstein distance as viable choices.

Importantly, f-Cal is agnostic to the choice of probabilistic deep neural regression model or task. In practice, it is a straightforward modification to the training loss function that can also be applied as a fine-tuning step to a previously partially trained model.

The f-Cal framework is generic and can be applied to arbitrary distributions. In this section we consider the case when the distribution $s(y_i;\phi_i)$ is Gaussian with $\phi_i \triangleq (\mu_i,\sigma_i)$. The variance σ_i^2 denotes the aleatoric uncertainty in this case. The error residuals are computed as $z_i = \frac{y_i-\mu_i}{\sigma_i}$, where μ_i and σ_i are predicted mean and the standard deviation of the ith Gaussian output from the neural network for each input x_i. So, $y_i \sim \mathcal{N}(\mu_i,\sigma_i^2)$, then $z_i \sim \mathcal{N}(0,1)$.

Optionally, one may apply several transforms to the random variables y_i and impose distributional *hyper*-constraints over the transformed variables. In practice, we find that this can improve the stability of the training process and enforces more stable calibration. In this case, we compute the sum-of-squared error residuals $q = \sum_{i=1}^{K} z_i^2$, and enforce the resulting distribution to be Chi-squared with parameter K i.e $q \sim \chi_K^2$, so, in this case, target distribution $Q = \chi_K^2$. Subsequently, we note that as the degrees of freedom K of a Chi-squared distribution increase, it can be approximated by a Gaussian of mean K and variance $2K$ through the application of the central limit theorem:

$$\lim_{K\to\infty} \frac{\chi_K^2 - K}{\sqrt{2K}} \to \mathcal{N}(0,1) \implies \lim_{K\to\infty} \chi_K^2 \to \mathcal{N}(K,2K)$$

In practice, this variation of the central limit theorem for Chi-squared random variables holds for moderate values of K (i.e., $K > 50$). This is practical to ensure, particularly in dense regression tasks such as bounding box object detection (where hundreds of proposals have to be scored per image) and per-pixel regression. We summarize the process for generating the calibration loss in Alg. 4.2.2. This loss is then combined with the typical empirical risk as given by (4.2.6).

Algorithm 4.2.2. f-cal for Gaussian uncertainties

Input: Dataset $\{x_i, y_i\}_{i=1}^{|D|}$, probabilistic neural regressor f_p,
degrees of freedom K, batch size N,
number of samples for hyper-constraint H

for $i = 1 \dots N$ **do**
  $(\mu_i, \sigma_i) \leftarrow f_p(x_i)$
  $z_i \leftarrow \frac{y_i - \mu_i}{\sigma_i}$
end
$\mathcal{C} = \emptyset$ // Samples for chi-squared distribution
for $i = 1 \dots H$ **do**
  $q_i \leftarrow \sum_{j=1}^{K} z_{ij}^2, z_{ij} \sim \{z_1, z_2, \dots z_N\}$
  $\mathcal{C}$.append(q_i)
end
$P_z \leftarrow$ Fit-Chi-Squared-Distribution($\mathcal{C}$)
$\mathcal{L}_{f\text{-Cal}} \leftarrow D_f(P_z || \chi_K^2)$
return $\mathcal{L}$;

4.3. CalibSLAM: Calibrated probabilistic object detector for SLAM

In this section, we integrate the object detector trained with the f-cal algorithm with the QuadricSLAM framework to show the effectiveness of calibrated uncertainty estimates in SLAM.

4.3.1. Backbone: QuadricSLAM

We build CalibSALM on top of QuadricSLAM, as it uses bounding box detection for measurement and treats objects as landmarks in the scene. QuadricSLAM uses a quadric to parametrize an object in the scene which can be estimated using multiple bounding boxes of the object observed in different frames. QuadricSLAM uses YOLO to obtain bounding boxes and some heuristics to obtain the covariance for the bounding boxes of object. Note that in

QuadricSLAM, all the bounding boxes of an object have the same covariance throughout all the frames in the scene. Hence even frames where an object is clearly visible will have the same covariance as frames with an occluded object, hence making the method agnostic to the bounding box's actual error in the frame. QuadricSLAM optimization estimates camera pose and quadrics by solving following optimization:

$$X^*, Q^* = \underset{X,Q}{\operatorname{argmin}} \underbrace{\sum_{t=1}^{T} ||f\left(x_{t-1}\right) \ominus x_t)||^2_{\Lambda_t}}_{\text{Odometry factors}} + \underbrace{\sum_{tj} \|h\left(x_t, q_j\right) - b_{tj}\|^2_{\Sigma_{tj}}}_{\text{Quadric Landmarks factors}} \qquad \text{(2.3.5: revisited)}$$

The core difference between QaudricSLAM and CalibSLAM lies in the estimation of the measurement covariance. Quadric SLAM calculates uncertainty as follows, for the bounding box belonging to Quadric j, the standard deviation for the bounding box in frame t is calculated as

$$\sigma_j = \sqrt{\frac{1}{T}\sum_{t=1}^{T}(w_j^t - \mu_j^w)^2} + \sqrt{\frac{1}{T}\sum_{t=1}^{T}(h_j^t - \mu_j^h)^2} = \sigma_j^w + \sigma_j^h \qquad (4.3.1)$$
$$\Sigma_{tj} = \sigma_j \mathbf{I}_{4\times 4}$$

where w_j^t and h_j^t are the width and height of the bounding box belonging to quadric k in the image at time t. With $\mu_j^w = \frac{1}{T}\sum_{t=1}^{T} w_j^t$ and $\mu_j^h = \frac{1}{T}\sum_{t=1}^{T} h_j^t$. The standard deviation σ_j for each bounding box measurement factor is the sum of the standard deviation of the width and height of bounding boxes over all the frames.

4.3.2. CalibSLAM

We denote the f-cal trained object detector by $h_{\text{cal}} : \Re \mapsto \Phi^k$, where $\Re$ is the domain of images and Φ^k is the domain of k distribution parameters predicted for k bounding boxes in the image I_t. So we have $h_{\text{cal}}(I_t) = \{\phi_i\}_{i=1}^k$, where $I_t \in \Re$ and $\phi_i \in \Phi$, ϕ_i is the distribution parameter for the i^{th} predicted bounding box $\mathbf{b_i}$ i.e $\mathbf{b_i} \sim s(\mathbf{b}; \phi_i)$ where $s(\mathbf{b})$ is the probability distribution over $\mathbf{b}$.

In this work we assume that the bounding box follows a Gaussian distribution i.e. $\phi_i = (\mu_i, \mathbf{\Sigma}_i)$, hence $\mathbf{b}_i \sim \mathcal{N}(\mu_i, \mathbf{\Sigma}_i)$.

The novelty here is the use of calibrated heteroscedastic noise estimation for each bounding box factor for SLAM, unlike the constant noise hyper-parameter in QuadricSLAM. Intuitively this should down-weight the bounding box measurements of quadrics which are occluded in some frames more compared to other measurements because occluded objects should have high measurement uncertainty.

At each frame, we associate the detected bounding box with already added quadrics in the factor graph from previous frames. A bounding box measurement at time t corresponding to

quadric j is denoted as $\mathbf{b}_t^j$ and the distribution parameters as $\mu_t^j, \mathbf{\Sigma}_t^j$. Hence for CalibSLAM, we have:

$$\mu_j^t, \Sigma_j^t = h_{\text{cal}}(I_t) \tag{4.3.2}$$

Note that not all the quadrics will have the measurement at all time steps as not all quadrics will be visible in all frames. Using the above notations incorporating the f-cal object detector in the SLAM framework defined in Eq. 2.3.5, we derive the following optimization:

$$X^*, Q^* = \underset{X,Q}{\text{argmin}} \sum_{t=1}^{T} ||f(\mathbf{x}_{t-1}) \ominus \mathbf{x}_t)||_{\Lambda_t}^2 + \sum_{tj} \left\| h(\mathbf{x}_t, \mathbf{q}_j) - \mu_t^j \right\|_{\mathbf{\Sigma_t^j}}^2 \tag{4.3.3}$$

where $X \triangleq \{\mathbf{x}_t\}_{t=1}^T$ and $Q \triangleq \{\mathbf{q}_j\}_{j=1}^J$

We describe the complete CalibSLAM algorithm in Algo. 4.3.1.

Algorithm 4.3.1. CalibSLAM

Input: Images $\{I\}_{t=1}^T$, calibrated neural regressor h_{cal}, camera odometry between poses $\{\mathbf{x_{t-1}^t}\}_{t=1}^T$

for $t = 1 \dots T$ **do**

  $\{(\mu_t, \Sigma_t)^j\}_{j=1}^k \leftarrow h_{\text{cal}}(I_t)$ // Predicting bounding box mean and covariance

  $f(x_t), \Lambda_t \leftarrow$ odometry

end

$x_1 \leftarrow$ [0,0,0] // put prior on first pose

for $t = 1 \dots T$ **do** **// Initializing poses**

  $x_t = f(x_{t-1})$ // Initializing poses using odometry only

for $j = 1 \dots K$ **do // Initializing quadrics**

  $q_j \leftarrow$ Solving using Equation 2.3.3 given initialized poses x_t

// factor graph optimization

$X^*, Q^* = \text{argmin}_{X,Q} \sum_{t=1}^{T} ||f_t(\mathbf{x_{t-1}}) \ominus \mathbf{x_t})||_{\Lambda_t}^2 + \sum_{tj} \left\| h(\mathbf{x_t}, \mathbf{q_j}) - \mu_t^j \right\|_{\mathbf{\Sigma_t^j}}^2$

Output: X^*, Q^*

Chapter 5

Experiments

In this chapter, we show experiments to empirically verify the effects of uncertainty calibration on SLAM. We employ experiments with different complexity - toy experiments to enable controlled experiments for thorough analysis and TUM-RBGD for analysis on the real-scale dataset.

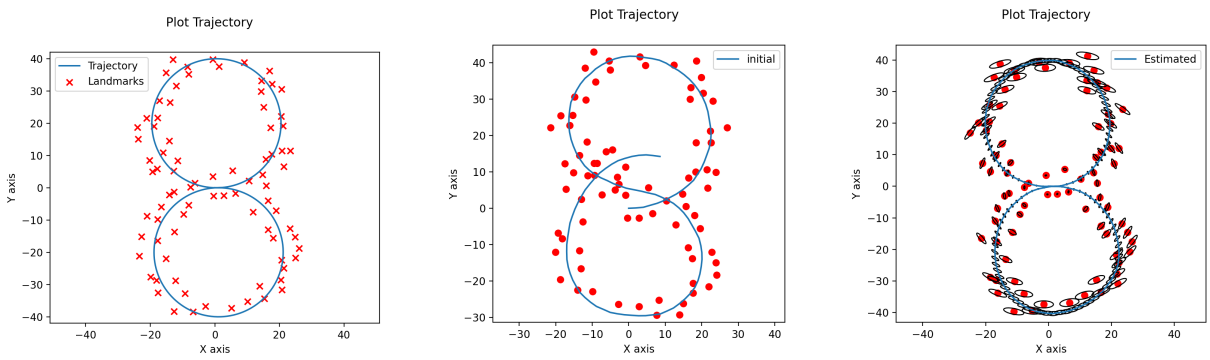
5.1. Toy experiments

We simulate toy experiments to understand the effects of uncertainty calibration on SLAM optimization in small-scale controlled settings. We imitate a robot moving on the 2D plane along the trajectory which makes a figure "8" and the robot is equipped with a laser rangefinder sensor (i.e., it can detect landmarks up to some maximum range and output their distance from the robot and angle). Landmarks are denoted as points in the 2D plane. Figure 5.1 is an example of the mentioned toy experiments.

5.1.1. Setup

We generate the ground truth trajectory with a robot moving in figure "8" by making it move in a circle with velocity $v = 1\text{m/s}$ and angular velocity $\omega = 0.05\text{rad/sec}$. We store points from the generated trajectory every $\text{dt} = 1\text{s}$. This is how we generate the ground truth odometry. After this, we uniformly place landmarks in the area encompassing the trajectory (Note: In the figure, you only see the landmarks in the range of the robot sensor). After generating the ground truth trajectory and landmarks, we generate the noisy odometry and noisy measurements by adding noise to the ground truth values to make a factor graph of the SLAM problem.

Now to test the hypothesis of calibration leading to better SLAM results, we test with two settings:



(a) (b) (c)

Fig. 5.1. (a) Ground truth trajectory (b) Noisy trajectory after adding noise to ground truth trajectory used for initialization (c) Estimated trajectory after solving the factor graph for SLAM.

(1) Giving process and measurement noise hyper-parameter to the factor graph equal to the actual noise used to generate odometry and measurement factors. This simulates the experiment with calibrated uncertainty estimates scenario.
(2) Giving different process and measurement noise hyper-parameter to the factor graph than the actual noise used to generate odometry and measurement factors. This simulates the uncalibrated uncertainty estimates scenario.

5.1.2. Optimization

The two key components for optimization of non-linear least square cost of factor graph optimization as shown eq. 2.2.4 is the initialization of variables and the method of optimization.

We initialize the robot pose variable using noisy odometry measurements, to initialize the landmarks we use the first measurement of the landmark from a noisy robot pose and compose the measurement with robot pose initialization.

For optimization, we choose Levenberg–Marquardt (LM) method [**31**]. LM method is an iterative algorithm to solve non-linear least square problems. It interpolates between gradient descent and the Gauss-Newton method using the damping factor λ.

5.1.3. Evaluation

To evaluate SLAM results we use the metrics ATE (Absolute Trajectory Error) & RPE (Relative Pose Error). ATE is the average squared error between estimated and ground truth poses and RPE gives the squared relative pose (pose at time $t+1$ with respect to pose at time t) error between estimated and ground truth trajectories.

Uncertainty Estimation	ATE	RPE	Landmark RMSE
Underconfident	1.187 ± 0.386	0.111 ± 0.004	1.181 ± 0.384
Calibrated	0.9878 ± 0.356	0.0996 ± 0.004	0.992 ± 0.351
Overconfident	1.034 ± 0.492	0.114 ± 0.006	1.047 ± 0.493

Table 5.1. Effect of Uncertainty Calibration in SLAM. We can see the calibrated uncertainty estimation provides the best results for SLAM. The lower the number the better, we show mean and standard deviation across 20 seeds of experiments

5.1.4. Results

We do three sets of experiments to evaluate the effect of uncertainty calibration on SLAM optimization. We generate the odometry and measurement factors for SLAM optimization by adding noise to known ground truth odometry and landmark measurements. Odometry is the relative pose of the next robot pose with respect to the current robot pose, which is given as an observation to the factor graph. We add odomtery noise $\sigma^a_{\text{odom}} = [0.1, 0.1, \frac{2\pi}{180}]$, which is noise in x, y and θ i.e orientation of the pose. The σ^a stands for actual/ground-truth noise/uncertainty used to generate the odometry factors. The measurement is defined by distance d and bearing (angle) θ of the landmark with respect to the robot pose. Similarly for odometry, we add measurement noise $\sigma^a_{\text{measurement}} = [0.1, \frac{2\pi}{180}]$ which is noise in d and bearing θ. Ideally, this is the noise to be given to the factor graph as a covariance parameter in the least square terms in Eq. 2.2.4 i.e $\Lambda = \text{diag}(\sigma^a_{\text{odom}})$ and $\Sigma = \text{diag}(\sigma^a_{\text{measurement}})$ where $\text{diag}(\cdot)$ indicates a diagonal matrix using the vector as diagonal elements. Now to replicate the real settings we are addressing in this thesis, which are about addressing uncertainty calibration in the measurement sensor from the perspective of neural networks, we ablate over measurement uncertainty/noise in these experiments.

To verify the hypothesis of uncertainty calibration for SLAM, we do the following three experiments:

(1) **Calibrated**: In this, we give the optimization the exact noise/uncertainty parameter value that is the actual noise in the measurements. i.e $\Sigma = \text{diag}(\sigma^a_{\text{measurement}})$.

(2) **Underconfident**: This is the more conservative setting and we consider that the measurement is noisier than the actual noise. For underconfident uncertainty estimates, we used $\Sigma = \text{diag}(2 \times \sigma^a_{\text{measurement}})$ i.e covariance is twice the actual noise.

(3) **Overconfident**: This is the setting which is more optimistic and we consider that the measurements have less noise than the actual noise. This shows more confidence in the measurements. For overconfident uncertainty estimates, we used $\Sigma = \text{diag}(\frac{\sigma^a_{\text{measurement}}}{2})$ i.e covariance is half the actual noise.

Note that we kept the odometry calibrated (i.e $\Lambda = \text{diag}(\sigma^a_{\text{odom}})$) in all the experiments to replicate realistic experiments and have more controlled ablation. We run these experiments over 20 seeds for statistically significant results.

Now based on the results as shown in table 5.1, we see that optimization with calibrated uncertainty estimates gives the optimal results. Hence confirming our hypothesis that calibrating uncertainty estimates gives a better optimization result for SLAM.

5.2. CalibSLAM

5.2.1. Implementation

We test the CalibSLAM on the same benchmark as QuadricSLAM [**41**] which is TUM RBGD [**47**]. We evaluate the two sequences available in the TUM dataset which are semantically rich: `freiburg3_long_office_household` and `freiburg2_desk`. The sequences are stream of images along with their ground truth camera pose. For each image in the sequence there is correponding ground truth robot pose available. Images are used by the f-Cal as input, whereas GT robot pose are used for calculating the accuracy of SLAM robot pose estimates. We implement the experiments using GTSAM [**9**] and the GTSAM Quadrics library.

5.2.2. f-Cal

We use the popular Faster R-CNN [**44**] with a feature pyramid network [**33**] and a Resnet-101 backbone [**21**]. We use the publicly available Detectron2 [**56**] and PyTorch [**43**] implementation and we the model with uncertainty head to regress uncertainty estimates. We train the model with COCO dataset [**34**]. We train this model with f-Cal loss from Equation 4.2.6. The f-Cal object detection model also provides class labels of the objects, which is not used in SLAM optimization directly but is used in data association of objects in different frames.

5.2.3. Odometry using ORB SLAM

We use ORB SLAM [**40**] to provide odometry. The ORB-SLAM generated trajectory plus added noise is used to initialize the trajectory for CalibSLAM. Note that we only use the visual odometry part of ORB-SLAM i.e no global information was used to estimate the trajectory, this is analogous to using only relative odometry between the frames. Any other

image-based method can also be used to generate the trajectory. We initialize the quadrics using the method detailed in section subsection 2.3.2. To initialize a quadric we need at least 3 frames with bounding boxes. We use initialized camera poses and the detected bounding boxes over all the frames in the dataset to get initializations for the quadrics.

We use QuadricSLAM as the starting point and build upon it. We replaced their YOLO-based object detector for bounding box prediction with our f-Cal trained object detector which provides uncertainty of the bounding box as well.

5.2.4. Results

Uncertainty Estimation	Trajectory ATE	Trajectory RPE
Fixed Noise = 5px	0.429 ± 0.117	0.027 ± 0.0033
QuadricSLAM	0.484 ± 0.197	$\mathbf{0.018 \pm 0.0006}$
CalibSLAM	$\mathbf{0.377 \pm 0.081}$	0.033

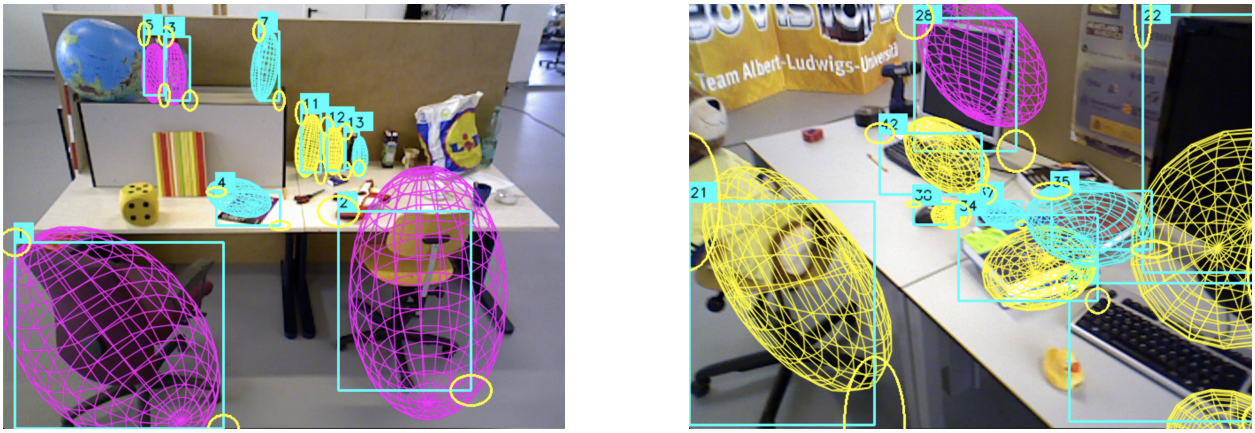
Table 5.2. Results of experiments on TUM RBG-D `freiburg3_long_office_household` SLAM sequences. The results are aggregated over 10 runs with different random initialization of robot pose.

Uncertainty Estimation	Trajectory ATE	Trajectory RPE
Fixed Noise = 5px	1.461 ± 0.071	0.042 ± 0.005
QuadricSLAM	1.407 ± 0.160	$\mathbf{0.033 \pm 0.0005}$
CalibSLAM	$\mathbf{1.394 \pm 0.117}$	0.053 ± 0.010

Table 5.3. Results of experiments on TUM RBG-D `freiburg2_desk` SLAM sequences.The results are aggregated over 10 runs with different random initialization of robot pose.

In the fixed noise experiment, we add fixed noise η to the bounding box measurement as follows:

$$\sigma_k = \eta \tag{5.2.1}$$



(a) (b)

Fig. 5.2. Estimated Quadrics with predicted bounding boxes with uncertainty estimation in CalibSLAM

Quadric SLAM calculates uncertainty as follows, for the bounding box belonging to quadric j, the standard deviation for the bounding box in frame t is calculated as

$$\sigma_j = \sqrt{\frac{1}{T}\sum_{t=1}^{T}(w_j^t - \mu_j^w)^2} + \sqrt{\frac{1}{T}\sum_{t=1}^{T}(h_j^t - \mu_j^h)^2} = \sigma_j^w + \sigma_j^h \qquad \text{(4.3.1: revisited)}$$

$$\Sigma_{tj} = \sigma_j \mathbf{I}_{4\times 4}$$

Whereas for CalibSLAM, the uncertainty is provided by f-Cal network with image I at time t as input as follows:

$$\mu_j^t, \Sigma_j^t = h_{\text{cal}}(I_t) \qquad \text{(4.3.2: revisited)}$$

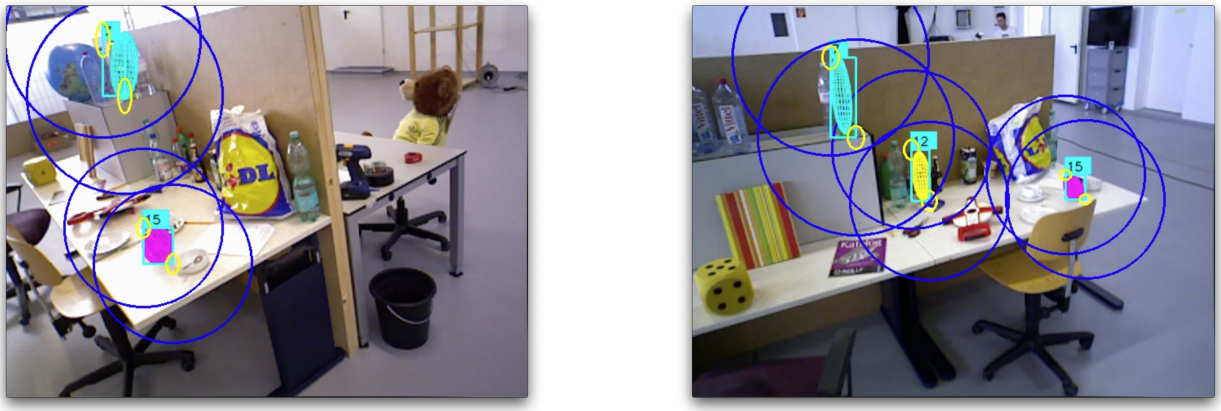
We would like to mention that QuadricSLAM is a really strong baseline since it calculates the standard deviation of actual bounding boxes over the whole dataset and hence the standard deviation is true standard deviation though the downside is that the calculated std dev is fixed over all the frames. Another downside is that calculating standard deviation over all the frames is infeasible for online applications. Whereas CalibSLAM uses f-Cal predicted standard deviation which is trained to be calibrated on the COCO dataset, hence there might be an out-of-distribution error in the prediction of f-Cal uncertainty, whereas QuadricSLAM uncertainty is calculated over the actual data.

As shown in table 5.3, CalibSLAM performs better on Trajectory ATE compared to such a stronger baseline, because of calibration guarantees and uncertainty conditioned on each image rather than fixed hyper-parameter calculated over all the dataset.

5.2.5. Qualitative Results

In this section, we show some qualitative analysis and results of different components of CalibSLAM.

In Figure 5.2, we can see different objects in the scene represented as quadrics with their associated bounding box detections and corresponding uncertainty estimates. The grid structure around the objects denotes their quadric representation, which is a bounded ellipsoid. The rectangle denotes the bounding box measurement of the object that we get from the f-Cal object detector, along with associated uncertainty estimates shown as ellipses at the top-left and bottom-right corners. The radius of the ellipse along two axes denotes 2σ standard deviation in the pixel space, where σ^2 is the covariance of the predicted Gaussian. Note that the covariance of the predicted bounding box of the object 21 in Figure 5.2 (b) is really high compared to other detection, which is a desired behaviour as the object is truncated in the image.



(a) (b)

Fig. 5.3. Comparison of bounding box measurement uncertainty estimates. Yellow denotes uncertainty predicted from f-Cal, Blue denotes uncertainty estimated in QuadricSLAM

Figure 5.3 shows the comparison of uncertainty estimates used in QuadricSLAM vs CalibSLAM. We can clearly see that bounding box uncertainty estimates via heuristics which are used in QuadricSLAM are highly inflated/underconfident. This shows underconfidence in the bounding box measurements, leading to under-utilization of the information being provided from bounding box measurements compared to the odometry, since the measurement term in the optimization will be down-weighted more. Whereas the uncertainty estimates predicted via f-Cal are tightly bounded/calibrated leading to more accurate information utilization of bounding box measurement. Figure 5.4 shows the ground truth trajectory vs estimated trajectories & quadrics using CalibSLAM. We can see that the estimated trajectory roughly matches the ground-truth trajectory.

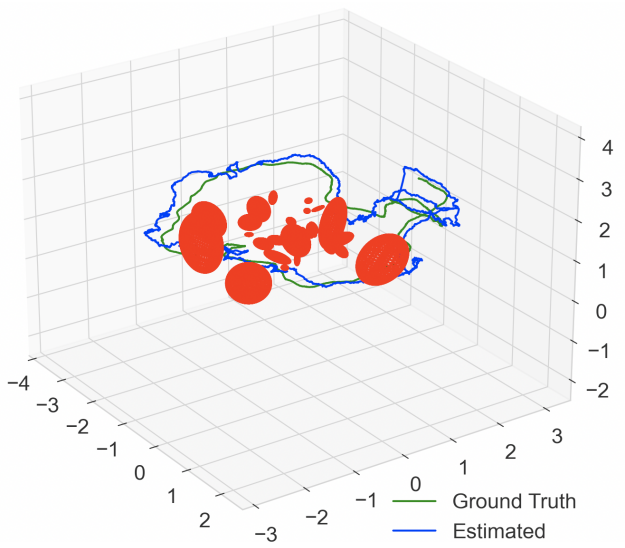


Fig. 5.4. CalibSLAM estimated 3D trajectory with quadrics

Chapter 6

Conclusion

This work was focused on showing that calibrated uncertainty estimates provide better SLAM results compared to other heuristic ways of estimating uncertainty. We introduce the idea of a deep neural network as a measurement sensor and showed f-Cal as calibrated measurement sensor for regression tasks and use it as an object detector for our work. Further, we show the effectiveness of calibrated uncertainty estimation for SLAM by combining QuadricSLAM with f-Cal. We showed improved results over previous semantic SLAM QuadricSLAM work. Hence, we conclude that calibrated uncertainty estimates in SLAM optimization give better results than fixed noise hyper-parameters.

In the sections below we indicate some of the limitations of the work and possible future extensions:

6.0.1. Limitations

- **Out-of-domain operation**: There is no guarantee in the calibration performance of the object detector as the measurement model when operated outside the training data distribution. Hence, uncertainty estimates may not be reliable in this situation, so SLAM might not work optimally in an environment different from than training environment.
- **Lack of odometry calibration**: The current method doesn't provide any uncertainty estimation for the odometry measurement model. So, even though bounding box measurements are calibrated, errors in uncertainty estimation for odometry might still lead to suboptimal SLAM results.

6.0.2. Future Work

- **End-to-End Training**: We can train the neural network based measurement model end-to-end with the SLAM optimization. This should train the measurement model

to increase the SLAM performance overall. This will also allow the possibility to fine-tune the measurement model in a new environment so that it can provide calibrated uncertainty estimates for optimal SLAM performance in a new environment.

- **Belief Space Planning**: Probabilistic estimates of quadrics also allow for belief space planning, where quadrics can be treated as an obstacle with probabilistic boundaries instead of fixed boundaries. This allows for probabilistic planning and manipulation.
- **Data-Association**: Data-association is a major part of any SLAM pipeline, as measurements belonging to the same object need to be associated across the frame. Calibrated uncertainty estimation over the measurement could allow for probabilistic data association.

References

[1] Dhaivat Bhatt, Kaustubh Mani, Dishank Bansal, Krishna Murthy, Hanju Lee et Liam Paull : f-cal: Aleatoric uncertainty quantification for robot perception via calibrated neural regression. *In 2022 International Conference on Robotics and Automation (ICRA)*, pages 6533–6539, 2022.

[2] C Blundell, J Cornebise, K Kavukcuoglu et D Wierstra : Weight uncertainty in neural networks. *arXiv preprint arXiv:1505.05424*, 2015.

[3] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang *et al.* : End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016.

[4] Cesar Cadena, Luca Carlone, Henry Carrillo, Yasir Latif, Davide Scaramuzza, José Neira, Ian Reid et John J Leonard : Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. *IEEE Transactions on robotics*, 32(6):1309–1332, 2016.

[5] Youngseog Chung, Willie Neiswanger, Ian Char et Jeff Schneider : Beyond pinball loss: Quantile methods for calibrated uncertainty quantification. *arXiv preprint arXiv:2011.09588*, 2020.

[6] Jan Czarnowski, Tristan Laidlow, Ronald Clark et Andrew J Davison : Deepfactors: Real-time probabilistic dense monocular slam. *IEEE Robotics and Automation Letters*, 5(2):721–728, 2020.

[7] Frank Dellaert et Michael Kaess : Square root sam: Simultaneous localization and mapping via square root information smoothing. *The International Journal of Robotics Research*, 25(12):1181–1203, 2006.

[8] Frank Dellaert, Michael Kaess *et al.* : Factor graphs for robot perception. *Foundations and Trends® in Robotics*, 6(1-2):1–139, 2017.

[9] Frank Dellaert, Richard Roberts, Varun Agrawal, Alex Cunningham, Chris Beall, Duy-Nguyen Ta, Fan Jiang, lucacarlone, nikai, Jose Luis Blanco-Claraco, Stephen Williams, ydjian, John Lambert, Andy Melim, Zhaoyang Lv, Akshay Krishnan, Jing Dong, Gerry Chen, Krunal Chande, balderdash devil, DiffDecisionTrees, Sungtae An, mpaluri, Ellon Paiva Mendes, Mike Bosse, Akash Patel, Ayush Baid, Paul Furgale, matthewbroadwaynavenio et roderick koehle : borglab/gtsam, mai 2022.

[10] Chengqi Deng, Kaitao Qiu, Rong Xiong et Chunlin Zhou : Comparative study of deep learning based features in slam. *In 2019 4th Asia-Pacific Conference on Intelligent Robot Systems (ACIRS)*, pages 250–254. IEEE, 2019.

[11] Jakob Engel, Vladlen Koltun et Daniel Cremers : Direct sparse odometry. *IEEE transactions on pattern analysis and machine intelligence*, 40(3):611–625, 2017.

[12] Jakob Engel, Thomas Schöps et Daniel Cremers : Lsd-slam: Large-scale direct monocular slam. *In European conference on computer vision*, pages 834–849. Springer, 2014.

[13] Di Feng, Lars Rosenbaum, Claudius Glaeser, Fabian Timm et Klaus Dietmayer : Can we trust you? on calibration of a probabilistic object detector for autonomous driving. 2, 2019.

[14] Christian Forster, Zichao Zhang, Michael Gassner, Manuel Werlberger et Davide Scaramuzza : Svo: Semidirect visual odometry for monocular and multicamera systems. *IEEE Transactions on Robotics*, 33(2):249–265, 2016.

[15] Yarin Gal : Uncertainty in deep learning. *University of Cambridge*, 1(3), 2016.

[16] Yarin Gal et Zoubin Ghahramani : Dropout as a bayesian approximation: Representing model uncertainty in deep learning. *ArXiv*, abs/1506.02142, 2016.

[17] Ethan Goan et Clinton Fookes : Bayesian neural networks: An introduction and survey. *Case Studies in Applied Bayesian Data Science: CIRM Jean-Morlet Chair, Fall 2018*, pages 45–87, 2020.

[18] Chuan Guo, Geoff Pleiss, Yu Sun et Kilian Q Weinberger : On calibration of modern neural networks. *arXiv preprint arXiv:1706.04599*, 2017.

[19] Ali Harakeh et Steven L. Waslander : Estimating and evaluating regression predictive uncertainty in deep object detectors. *CoRR*, abs/2101.05036, 2021.

[20] R. I. Hartley et A. Zisserman : *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second édition, 2004.

[21] Kaiming He, Xiangyu Zhang, Shaoqing Ren et Jian Sun : Deep residual learning for image recognition. *In Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[22] Yisheng He, Wei Sun, Haibin Huang, Jianran Liu, Haoqiang Fan et Jian Sun : Pvn3d: A deep point-wise 3d keypoints voting network for 6dof pose estimation. *In IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[23] Yvonne HS Ho et Stephen MS Lee : Calibrated interpolated confidence intervals for population quantiles. *Biometrika*, 92(1):234–241, 2005.

[24] Moksh Jain, Salem Lahlou, Hadi Nekoei, Victor Butoi, Paul Bertin, Jarrid Rector-Brooks, Maksym Korablyov et Yoshua Bengio : Deup: Direct epistemic uncertainty prediction. *arXiv preprint arXiv:2102.08501*, 2021.

[25] Krishna Murthy Jatavallabhula, Soroush Saryazdi, Ganesh Iyer et Liam Paull : gradslam: Automagically differentiable slam. *arXiv preprint arXiv:1910.10672*, 2019.

[26] Michael Kaess, Ananth Ranganathan et Frank Dellaert : isam: Incremental smoothing and mapping. *IEEE Transactions on Robotics*, 24(6):1365–1378, 2008.

[27] Rudolph Emil Kalman : A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering*, 82(Series D):35–45, 1960.

[28] Alex Kendall et Yarin Gal : What uncertainties do we need in bayesian deep learning for computer vision? *In Advances in neural information processing systems*, pages 5574–5584, 2017.

[29] Volodymyr Kuleshov, Nathan Fenner et Stefano Ermon : Accurate uncertainties for deep learning using calibrated regression. *arXiv preprint arXiv:1807.00263*, 2018.

[30] Balaji Lakshminarayanan, Alexander Pritzel et Charles Blundell : Simple and scalable predictive uncertainty estimation using deep ensembles, 2017.

[31] Kenneth Levenberg : A method for the solution of certain non-linear problems in least squares. *Quarterly of applied mathematics*, 2(2):164–168, 1944.

[32] Dan Levi, Liran Gispan, Niv Giladi et Ethan Fetaya : Evaluating and calibrating uncertainty prediction in regression tasks. *arXiv preprint arXiv:1905.11659*, 2019.

[33] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan et Serge Belongie : Feature pyramid networks for object detection. *In Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2117–2125, 2017.

[34] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár et C Lawrence Zitnick : Microsoft coco: Common objects in context. *In European conference on computer vision*, pages 740–755. Springer, 2014.

[35] Shing Yan Loo, Ali Jahani Amiri, Syamsiah Mashohor, Sai Hong Tang et Hong Zhang : Cnn-svo: Improving the mapping in semi-direct visual odometry using single-image depth prediction. *In 2019 International conference on robotics and automation (ICRA)*, pages 5218–5223. IEEE, 2019.

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

[37] Ziqi Lu, Yihao Zhang, Kevin Doherty, Odin A Severinsen, Ethan Yang et John Leonard : Slam-supported self-training for 6d object pose estimation. *arXiv preprint arXiv:2203.04424*, 2022.

[38] John McCormac, Ronald Clark, Michael Bloesch, Andrew Davison et Stefan Leutenegger : Fusion++: Volumetric object-level slam. *In 2018 international conference on 3D vision (3DV)*, pages 32–41. IEEE, 2018.

[39] Montiel J. M. M. Mur-Artal, Raúl et Juan D. Tardós : ORB-SLAM: a versatile and accurate monocular SLAM system. *IEEE Transactions on Robotics*, 31(5):1147–1163, 2015.

[40] Raúl Mur-Artal et Juan D. Tardós : ORB-SLAM2: an open-source SLAM system for monocular, stereo and RGB-D cameras. *IEEE Transactions on Robotics*, 33(5):1255–1262, 2017.

[41] Lachlan Nicholson, Michael Milford et Niko Sünderhauf : Quadricslam: Dual quadrics from object detections as landmarks in object-oriented slam. *IEEE Robotics and Automation Letters*, 4(1):1–8, 2018.

[42] Jeremy Nixon, Michael W Dusenberry, Linchuan Zhang, Ghassen Jerfel et Dustin Tran : Measuring calibration in deep learning. *In CVPR Workshops*, pages 38–41, 2019.

[43] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga *et al.* : Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

[44] Shaoqing Ren, Kaiming He, Ross Girshick et Jian Sun : Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28, 2015.

[45] M Rueda, S Martínez-Puertas, H Martínez-Puertas et A Arcos : Calibration methods for estimating quantiles. *Metrika*, 66(3):355–371, 2007.

[46] Hao Song, Tom Diethe, Meelis Kull et Peter Flach : Distribution calibration for regression. *arXiv preprint arXiv:1905.06023*, 2019.

[47] J. Sturm, N. Engelhard, F. Endres, W. Burgard et D. Cremers : A benchmark for the evaluation of rgb-d slam systems. *In Proc. of the International Conference on Intelligent Robot Systems (IROS)*, Oct. 2012.

[48] Natasa Tagasovska et David Lopez-Paz : Single-model uncertainties for deep learning. *In Advances in Neural Information Processing Systems*, pages 6417–6428, 2019.

[49] Maxime Taillardat, Olivier Mestre, Michaël Zamo et Philippe Naveau : Calibrated ensemble forecasts using quantile regression forests and ensemble model output statistics. *Monthly Weather Review*, 144(6):2375–2393, 2016.

[50] Keisuke Tateno, Federico Tombari, Iro Laina et Nassir Navab : Cnn-slam: Real-time dense monocular slam with learned depth prediction. *In Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6243–6252, 2017.

[51] Zachary Teed et Jia Deng : Droid-slam: Deep visual slam for monocular, stereo, and rgb-d cameras. *Advances in Neural Information Processing Systems*, 34:16558–16569, 2021.

[52] Sebastian Thrun : Probabilistic robotics. *Communications of the ACM*, 45(3):52–57, 2002.

[53] Sen Wang, Ronald Clark, Hongkai Wen et Niki Trigoni : Deepvo: Towards end-to-end visual odometry with deep recurrent convolutional neural networks. *In 2017 IEEE international conference on robotics and automation (ICRA)*, pages 2043–2050. IEEE, 2017.

[54] P. Wei, G. Hua, W. Huang, F. Meng et H. Liu : Unsupervised monocular visual-inertial odometry network. *In Twenty-Ninth International Joint Conference on Artificial Intelligence and Seventeenth Pacific Rim International Conference on Artificial Intelligence IJCAI-PRICAI-20*, 2020.

[55] Andrew G Wilson et Pavel Izmailov : Bayesian deep learning and a probabilistic perspective of generalization. *Advances in neural information processing systems*, 33:4697–4708, 2020.

[56] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo et Ross Girshick : Detectron2. `https://github.com/facebookresearch/detectron2`, 2019.

[57] Shichao Yang et Sebastian Scherer : Cubeslam: Monocular 3-d object slam. *IEEE Transactions on Robotics*, 35(4):925–938, 2019.

[58] Bianca Zadrozny et Charles Elkan : Transforming classifier scores into accurate multiclass probability estimates. *In Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 694–699, 2002.

[59] Huizhong Zhou, Benjamin Ummenhofer et Thomas Brox : Deeptam: Deep tracking and mapping with convolutional neural networks. *International Journal of Computer Vision*, 128(3):756–769, 2020.