Multi-Robot Autonomous Classification Under Uncertainty

Vladimir Tchuiev

Multi-Robot Autonomous Classification Under Uncertainty

Research Thesis

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy

Vladimir Tchuiev

Submitted to the Senate
of the Technion — Israel Institute of Technology
Iyar 5781 Haifa April 2021

This research was carried out under the supervision of Assoc. Prof. Vadim Indelman, in the Faculty of Aerospace Engineering.

Some results in this thesis have been published as articles by the author and research collaborators in conferences and journals during the course of the author's doctoral research period, the most up-to-date versions of which being:

V. Tchuiev and V. Indelman. Inference Over Distribution of Posterior Class Probabilities For Reliable Bayesian Classification and Object-Level Perception. *IEEE Robotics and Automation Letters (RA-L)*, 3(4):4329–4336, 2018.

V. Tchuiev, Y. Feldman, V. Indelman. Data Association Aware Semantic Mapping and Localization via a Viewpoint-Dependent Classifier Model. classificators. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*. 7742–7749, IEEE, October 2019.

V. Tchuiev and V. Indelman. Distributed Consistent Multi-Robot Semantic Localization and Mapping. *IEEE Robotics and Automation Letters (RA-L)*, 5(3):4649–4656, 2020.

V. Tchuiev and V. Indelman. Epistemic Uncertainty Aware Localization And Mapping For Inference and Belief Space Planning. *Submitted to Artificial Inteligence Journal (AIJ)*, 2021.

Acknowledgements

I would like to thank my advisor Assoc. Prof. Vadim Indelman for his guidance and support in my 4 years of PhD research with his knowledge and insights, Yuri Feldman for his contributions and insights for the paper presented in IROS 2019 conference, Technion's Aerospace Engineering faculty for it's financial support during all this time, the ANPL team for their part with insightful discussion that contributed to the research. Finally, I would like to thank my parents Olga and Ivgeny Tchuiev for continual moral support during my PhD work.

The Technion's funding of this research is hereby acknowledged.

Contents

List of Figures

Abstract

Object classification is an important task in aerospace and robotics. Although in recent years the field saw many advances with the introduction of deep-learning-based approaches, reliable classification remains a challenge. In this thesis, we address two challenges in object classification: classification aliasing for certain relative viewpoints between object & camera, and the epistemic uncertainty of the classifier. We address both of those challenges in four works. The first introduces a semantic SLAM approach that maintains a hybrid belief over objects and classes, and leverages the coupling between them to assist in data association disambiguation and improve classification and localization performance. In the second work we maintain a hybrid belief in a multi-robot distributed setting while addressing double-counting for both continuous and discrete states. The first two works model the coupling between poses and classes via a viewpoint dependent classifier model. The third work proposes a sequential classification approach that reasons about posterior epistemic, or model uncertainty, to improve classification performance. The fourth work proposes two methods that both address classification aliasing and posterior epistemic uncertainty: Multi-Hybrid (MH) and the more computationally efficient Join Lambda Pose (JLP). Eventually, we extend those methods to an active belief-space-planning setting, while utilizing an information-theoretic cost for the posterior class probability. We used a viewpoint dependent classifier uncertainty model that predicts epistemic uncertainty from different viewpoints both for inference and planning. The methods are evaluated both in simulation and experiments and demonstrated increased performance in classification and localization.

Chapter 1

Introduction

1.1 Motivation

Classification and object recognition is a fundamental problem in robotics and computer vision, which plays a significant role in numerous problem domains and applications, including semantic mapping, object-level simultaneous localization and mapping (SLAM), active perception and autonomous driving. Yet, reliable and robust classification in uncertain and ambiguous scenarios is challenging, as object classification is often viewpoint dependent, influenced by environmental visibility conditions such as lighting, clutter and occlusions, and limited by the classifier's training set. In recent years, with modern implementation of deep learning methods, classification performance has vastly improved from previously used methods. In closed testing environments the classifiers perform even better than humans.

Despite the advancements, reliable classification still remains a challenge and the use in real-life applications still remains unreliable. Once the classifier leaves the training data it performs worse, especially when encountering unknown scenes and objects that the training set did not account for. Quantifying this uncertainty opens the possibility of safer classification, and eventually make decision making based of this classification more reliably.

In this work, we focus on two key challenges for object classification. The first is classification aliasing: object often appear different from certain viewpoints, with varying shapes and colors. Moreover, distinguishing between different object classes may prove impossible for certain types of objects from particular relative viewpoints. In addition, object and camera pose may be uncertain as well. A possible solution we consider is modeling the classification scores given the object's candidate class and relative viewpoints between camera and object, where the poses are inferred probabilistically via semantic SLAM. Another solution is integrating both classification and localization data from multiple images into posterior data. We model this viewpoint dependency via a viewpoint dependent classifier model; This model, in addition to assisting in accurate classification, also assists in improving SLAM performance. Previous works

either used most-likely class semantic measurements, or utilized a viewpoint dependent classifier model in a setting with solved localization. In contrast, we consider semantic measurements of probability vectors in a setting with unknown poses, and infer both the poses and the classes in an object based SLAM setting. We utilize this classifier model both in a single robot scenario to assist in data association disambiguation, and in a distributed multi-robot setting. By leveraging the coupling between poses and classes via a hybrid belief, we improve classification and localization performance.

The second challenge is reasoning about test data that is not present in the classifier's training. As such, the classifier, in the presence of such data, provides unreliable results that may induce catastrophic failures in systems that rely on accurate classification, such as autonomous vehicles. Identifying when the classifier provides reliable classification scores is crucial, and reasoning about the accumulated epistemic uncertainty proves beneficial both for classification and subsequent decision making. While existing works that reason about epistemic uncertainty in classification, are doing so on a single image bases, while works that address sequential classification do not consider epistemic uncertainty. In this work we bridge the gap and propose epistemic-uncertainty-aware sequential classification methods; First for a focused case where we do not reason about localization and do so for a single object, and then expand it to a case with unknown localization, integrating epistemic uncertainty inference within a semantic SLAM framework.

As we established, integrating information from multiple sources is highly beneficial. Furthermore, autonomous vehicles are often not alone within the environment they are operating with. Multiple agents within the environment may provide coverage that a single vehicle is not able to. As such, integrating information collected from all the vehicles proves beneficial and provides superior performance compared to using a single vehicle. There are three common frameworks for multi-agent systems: centralized, decentralized, and distributed. With centralized frameworks, all agent communicate with a central processing unit that processes the information sent by every agent. Decentralized systems spread the processing over multiple separate processing units. In distributed systems each agent is independent and autonomously integrates and processes the information it receives from other robots. While planning distributed systems is a the most challenging, such systems provide greater robustness, as the agents within require sharing the information they gathered with others only when the opportunity arises. As robots communicate between each other in distributed systems, it must not count measurements more than once. Otherwise, they risk double counting information which may lead to erroneous and overconfident estimations. Most works about distributed semantic SLAM consider only reason about continuous variables, subsequently addressing double counting only for continuous variables. Other works that do reason about semantic measurements in distributed SLAM do so with most-like-class measurements. In this work, we maintain a hybrid belief over poses (continuous) and classes (discrete) in a distributed setting while utilizing a class probability vector

semantic measurements. In addition, we address double counting for both continuous and discrete random variables.

Up to this point, we discussed addressing uncertainties in classification in a passive setting, where the robot actions are determined by an outside source. The autonomous setting, where the robot determines its actions. Autonomous navigation is a widely researched topic, extending many fields such as autonomous vehicles. In particular, object classification is key for application that require exploration of the surrounding environment. To be fully autonomous, it must be able to make decisions based on its measurement history, while accounting for different sources of uncertainty. The planning under uncertainty problem can be formulated as a partially observed Markov decision process (POMDP). One common approach for solving the POMDP problem is belief space planning (BSP), where a probability density function is maintained over the states; Each candidate action is evaluated by the induced evolution of the belief, and the best action is chosen such it optimizes an objective function. BSP utilizes known models to infer the next best action without the need of prior experience, unlike e.g. reinforcement learning. Existing active classification approaches do not use BSP, instead using reinforcement learning or other methods. In addition, those methods to not consider epistemic uncertainty of the classifier. We, on the other hand, consider the epistemic-uncertainty-aware semantic SLAM methods we developed, and extend those to an active setting, while also considering an epistemic-uncertainty-aware reward function.

1.2 Thesis Contributions

In this thesis we investigate approaches for passive and active classification, while considering viewpoint dependency and epistemic uncertainty. The main contributions of this thesis are formulated as follows:

1. We tackle classification aliasing and localization uncertainty via a semantic SLAM approach that maintains a hybrid belief over poses and classes. We utilize semantic measurements of class probability vectors, and leverage the coupling between poses and classes via a viewpoints dependent classifier model. For a single robot, we use the classifier model to assist in data association disambiguation, and afterwards extend the approach to a distributed multi-robot setting, while addressing double counting both for continuous and discrete random variables, poses and classes respectively.

2. We address classifier epistemic uncertainty in classification. We present methods for sequential classification that reason about posterior epistemic uncertainty in the passive case. First, we address sequential classification in a focused case without localization, then present two methods for addressing classification un-

certainty via both classification aliasing and epistemic uncertainty in a semantic SLAM setting.

3. We address classifier epistemic uncertainty and classification aliasing in a BSP setting; By planning over a novel epistemic-uncertainty-aware reward function, we improve classification scores over existing approaches that do not reason about classifier epistemic uncertainty. We utilize a viewpoint dependent classifier uncertainty model for generating predicted epistemic-uncertainty-aware semantic measurements for planning.

1.3 Thesis Outline

First, we survey the relevant literature in Chapter 2. The main content of the thesis is split into four chapters:

1. Chapter 3: Data association aware SLAM using a viewpoint dependent classifier model.
2. Chapter 4: Distributed semantic SLAM with viewpoint dependent classifier model.
3. Chapter 5: Model uncertainty aware sequential inference of posterior class probability
4. Chapter 6: Posterior classifier epistemic uncertainty aware semantic inference and belief space planning.

The first two chapters address the viewpoint dependency of classifier scores. In Chapter 3 we present an approach for using a viewpoint dependent classifier model to assist in data association disambiguation and improve classification and localization performance. In Chapter 4 we expand the approach presented in the prior chapter to a distributed multi-robot system framework. The next two chapters address classifier epistemic uncertainty. In Chapter 5 we present an approach for sequential classification of a single object that reasons about epistemic uncertainty without considering localization. In Chapter 6 we present two approaches that address viewpoint dependency and epistemic uncertainty simultaneously, first for inference, and then expanding to a BSP setting. In Chapter 7 we conclude and present possible directions for future work.

Chapter 2

Literature Survey

2.1 Classification and Deep Learning

In recent years deep learning based convolutional neural networks (CNNs) since introduction produced vastly superior performance than anything came before that. A modern CNN was introduced first by Krizhevsky et al. [1], AlexNet, that outperformed every other classification algorithm (such as SVM) before that. Since then, many variations of CNNs were introduced that improved classification performance even further. Notable architectures include VGG by Simonyan et al. [2], GoogLeNet by Szegedy et al. [3], ResNet by He et al. [4], and CapsNet by Sabour et al. [5]. Amini et al. [6] presents a dropout-based method that uses spatial for inferring epistemic uncertainty. Malinin and Gales. [7] presented Prior Networks (PN), an approach for estimating predictive uncertainty: both epistemic and aleatoric. SCOD by Sharma et al. [8] utilizes local curvature of deep neural networks to produce epistemic uncertainty estimations. These algorithms classify objects on a single image basis, but for more accurate classification in real environment multiple images may be required.

Several sequential classification algorithms were developed in recent years. Coates and Y. Ng [9] proposed a method that updates a posterior class probability by multiplying prior class probability with a new classification measurement, considering an object detection problem. Omidshafiei et al. [10] mentions the Static State Bayes Filter (SSBF) algorithm that extends the work by Coats and Y. Ng for multiple possible classes. It assumes the prior, posterior and likelihood all categorically distributed. Patten et al. [11] used a method similar to SSBF in the context of active classification. Atanasov et al. [12] proposed a method that updates categorically distributed posterior pairs of candidate object classes and orientations. This approach utilizes a viewpoint dependent observation model. Pillai and Leonard [13] proposed a monocular SLAM-aware object classification system. Omidshafiei et al. [10] developed the hierarchical Bayesian noise inference (HBNI) algorithm. At each time step, the algorithm updates class probabilities with the likelihood of the soft-max classifier output modeled by Dirichlet distribution, with a noise parameter for each possible class. Mu

et al. [14] utilizes a Dirichlet prior and most likely classifier observations, while addressing the challenging problem of data association. Teacy et al. [15], and Feldman & Indelman [16] utilized a Gaussian process viewpoint dependent classifier model to assist in classification tasks. Kopitkov and Indelman [17] utilized a viewpoint dependent classifier model, learned offline via deep learning, for probabilistic inference over robot trajectory. While the above works address sequential classification, none of them reason about uncertainty of classification results and viewpoint dependency between poses and classification scores.

Recently, several works developed methods for computing epistemic uncertainty for deep learning applications. Paass et al. [18] proposed Bootstrapping, where multiple classifiers are trained on the same training set. Grimmett et al. [19] suggested using normalized entropy of class probability as classification uncertainty. However this approach does not consider how certain the class probability itself is. Gal and Ghahramani [20] [21] proposed utilizing neural network dropout to estimate epistemic uncertainty for an input of a single image. Kendal and Cipolla [22] build upon these works, utilizing dropout to compute uncertainty in CNN-based camera relocalization. Both infer epistemic uncertainty from a single image input only. Mishkov and Julier [23] compare between multiple methods to predict uncertainty in classifier output, using Hybrid Monte Carlo (HMS) as a baseline. They found that Stochastic Gradient Langevin Dynamics (SGLD) and Dropout (see [21]) methods performed the best in terms of accuracy. Furthermore, Kendal and Gal [24] analyze the major two types of uncertainty: epistemic uncertainty or model uncertainty, and aleatoric uncertainty that captures noise inherent in observations. Yet, these works consider classification given a single image frame, as opposed to Bayesian sequential classification given multiple images that we consider herein.

A related problem to object classification is object detection - indeed, first, one has to detect there is an object, e.g. in the captured image(s), and only then attempt to infer its class. Viola and Jones [25] presented a machine learning approach for object detection via a series of classifiers to detect and use critical features within the image. This algorithm was tested for face detection. Felszenszwalb et al. [26] detected objects via a mixture of multiscale deformable part models. In the last few years Regional Convolutional Neural Network (RCNN) were on the rise for object detection. Girshick et al. [27] first proposed the R-CNN, and later works [28–30] improved upon it by making it computationally faster.

2.2 Single and Multi-Robot SLAM

Simultaneous localization and mapping (SLAM) is the computational problem of inferring a robot's location in an unknown environment, while mapping it (referred to as Full SLAM), or only utilizing landmarks to produce relative pose constraints (Pose SLAM, see [31]). State of the art SLAM approaches resort to a smoothing formulation,

often expressing the problem in terms of graphical models, such as a factor graph [32]. In the graph based SLAM paradigm the problem is usually divided into two parts: graph construction (front-end, see for example [33]) and graph optimization (back-end, see for example [34]). The front-end is entrusted with processing raw measurements (e.g. images), detecting features and tracking them across different frames, i.e. solving the data association problem. The back-end process formulates appropriate constraints given the tracked features, constructs the corresponding factor graph and performs probabilistic inference to recover the SLAM solution. Several (back-end) approaches for computationally efficient optimization were developed in recent years, such as incremental smoothing and mapping (iSAM) [35] and iSAM2 [36]. A typical assumption in SLAM approaches is that data association, e.g. feature correspondences, is given and outlier-free. Such an assumption is less realistic in many real-world scenarios that exhibit some level of perceptual aliasing. Indeed, incorporating incorrect constraints (due to outliers) within the optimization can lead to catastrophic results. Optimization approaches that attempt to be resilient to outliers overlooked by the front-end, or alternatively, that reason about data association within inference, are under active development by the research community [37].

SLAM approaches that reason about objects as landmarks, instead of traditionally used 3D points, are referred to as object-based SLAM (see [38]). This approach requires the robot to detect objects, generate measurements, and associate these measurements to unique object identifiers that can be acquired for example by classification. As such, object based SLAM is a problem that is coupled with the object classification problem (see [14]). Compared to traditional SLAM solutions that use low level primitives (such as image point features that correspond to 3D landmarks), object-based SLAM is much less memory intensive, as it tracks significantly less landmarks. Moreover, it provides a richer and more useful map of the perceived environment, enabling for example high-level task specification and planning. Kostavelis and Gasteratos [39] provided a recent survey of semantic mapping. Recently Gawel et al. [40] proposed the X-View algorithm, a multi-view semantic localization system, with a graph-based approach for semantic topologies. Nakajima et al. [41] proposed an efficient semantic mapping method with geometric based incremental segmentation. Josifovski et al. [42] presented an approach for pose estimation using a classifier model that utilizes 3D models. Wu et al. [43] presented an approach for relative pose estimation from images of objects using a neural network that is trained on synthetic data.

Data association of features or objects between images is a complex and important problem for a variety of autonomous system applications (e.g. SLAM). Naturally, many approaches were proposed to deal with this problem. An important early work is the joint probability data association (JPDA) approach by Fortmann et al. [44], which considers all possible options, therefore it is computationally slow and not practical. Sunderhauf and Protzel [45] proposed an approach to detect faulty loop closures that lead to erroneous data association in back-end optimization. Wong et al. [46] presented

a Dirichlet Process Mixture Model (DPMM) for data association for a partially observed environment. Olson and Agarwal [47] proposed a robust approach that uses max-mixture models. [48] classified measurements as coherent or not, thus predicting if they will result in erroneous data association. Pathak et al. [49, 50] proposed a data association aware belief space planning algorithm that models the belief as a Gaussian Mixture Model (GMM). Recently few works addressed data association problem with deep neural networks, specifically Long Short Term Memory (LSTM). Milan et al. [51] presented a method based on LSTM neural network for data association, training it on the MOTChallenge dataset. Farazi and Behnke [52] expended on the above work to visually track and associate between identical robots using an LSTM based approach.

More recent works about data association include: Doherty et al. [53] presented a semantic SLAM approach that represents data association hypotheses as multiple nodes in a non-Gaussian sensor model. Berneiter et al. [54] proposed a semantic SLAM approach for solving data association using multiple hypothesis trees. MHJCBB by Wang and Englot [55] is a multi-hypothesis approach for solving data association in SLAM that uses a modified joint compatibility branch and bound approach. Localization uncertainty will be also considered, thus leading to a hybrid belief over continuous (localization) and discrete (classification) variables. Hsiao et al. [56] presented MH-iSAM2: a modification of iSAM2 that addresses multiple data association hypotheses. Ok et al. [57] proposed ROSHAN, a object-based SLAM approach that is specifically designed for high-speed navigation, and represents objects as ellipsoids. We aim to create a data association scheme for classification for multiple objects, including differentiating between them even if their underlying class is the same. Previous works utilize only most likely class classifier output to address data association; in contrast, we consider richer classifier output, that can lead to more reliable data association (e.g. address cases of multiple objects with the same class in the environment).

The following are the most relevant works that present approaches for hybrid belief inference. Segal and Reid [58] proposed a message passing algorithm to optimize hybrid factor graphs for inference. The discrete-continuous graphical model (DC-GM) approach by Lajoie et al. [59] performed inference on a hybrid factor graph that produces near-optimal estimates. Mu et al. [14] proposed a sampling based approach that uses most likely class semantic measurements; this approach performs batch inference using expectation maximization (EM). Bowman et al. [60] utilizes most likely class and bounding box measurements, in addition to geometric measurements, to perform SLAM and DA disambiguation using EM as well. The above approaches consider only the most likely class and do not reason about viewpoint dependency of classification results. In contrast, we utilize richer classifier output in conjunction with a viewpoint dependent model to perform object level SLAM, while maintaining classification and DA hypotheses.

Generally, more information allows better SLAM accuracy. Multi-agent systems can provide significant performance boost over utilizing a single robot. There are mul-

tiple works addressing multi-robot SLAM, both for centralized and distributed systems. Choudhary et al. [61] proposed passive multi-robot object based SLAM by using RGBD cameras and a CNN object classifier. Cunningham et al. [62] proposed DDF-SAM (distributed data fusion smoothing and mapping) to avoid double counting by keeping two maps for each robot: one is self created, other is constructed from neighbouring robots' data. Later Cunningham et al. [63] proposed DDF-SAM 2.0 in which each robot keeps a single map and avoids double counting by information downdating. Indelman et al. [64] proposed a graph based method for distributed consistency and double counting avoidance. This approach was then used in [65] in the context of three-view geometry based distributed localization in unknown environments. While many works about SLAM assume known data association, Indelman et al. [66] proposed an approach for distributed multi-robot SLAM with unknown data association by choosing the most likely hypothesis from multiple options. Walls et al. [67] proposed a distributed geometric SLAM approach that communicates factors between robots. Other approaches for geometric SLAM include Extended Kalman Filter (such as [68]) or Particle Filter based methods (such as [69]). Choudhary et al. [70] presented an approach for distributed semantic SLAM which communicates relative poses between robots and uses object class information for data association.

Consistent estimation is a key issue in a distributed setup, with multiple approaches proposed to address it. Bahr et al. [71] proposed a distributed algorithm for under-water vehicles, with an approach for using all measurements without information loss. Indelman et al. [64] proposed a graph based method that calculated cross-covariance terms that represent the correlation between measurements from different robots, utilizing it for consistent estimation. Cunningham et al. [62] presented the DDF-SAM distributed SLAM algorithm that avoided double counting by creating two maps for each robot: local and global. The global map is updated with condensed local maps. A later work by Cunningham et al. [63] introduced DDF-SAM2, where each robot maintains only the global map. To avoid double counting, the old information during communication is filtered out via down-dating by each robot. Brooks et al. [72] address the problem of distributed information fusion from sensors about classifying and tracking a target. These approaches consider continuous random variables. In contrast, we reason about discrete variables as well.

2.3 Active Classification Under Uncertainty

The active classification problem was studied extensively in the computer vision community. Earlier works include, for example, Aloimonos et al. [73] for active vision, and Connolly [74] for next-best view planner. Wilkers and Tsotsos [75] utilized a decision tree based approach with line segmentation to move a robotic arm around an object. Gao and Koller [76] proposed an active classification approach that utilizes multiple classifiers. Holinger et al. [77] proposed a Bayesian active classification scheme for

underwater applications. Singh et al. [78] proposed the eSIP planning algorithm for multi-robot active classification. Works by Teacy et al. [15] and Velez et al. [79] propose approaches that utilize a viewpoint dependent classifier model to update posterior class probability. Guruau et al. [80] proposed an approach that predicts the classification failure probability for active planning. Other works incorporated probabilistic models into active vision systems [81–83]. Liu et al. [84] proposed an approach that utilizes both geometric and semantic measurements for planning using a sampling based method. Wandzel et al. [85] proposed OO-POMDP for object-oriented planning, which factorizes the robot's belief into independent distributions, allowing for more efficient planning. We note, however, that the above approaches do not consider localization and epistemic uncertainty.

Several planning approaches that do reason about epistemic uncertainty were proposed; Faddoul et al. [86] reasoned about epistemic uncertainty in MDP and POMDP transition matrices, creating a framework for decision making. Hayashi et al. [87] proposed an approach that actively trains uncertain dynamic models via neural network priors. These works do not consider epistemic uncertainty in the context of classification. Lutjens et al. [88] presented a reinforcement learning approach that reasons about epistemic uncertainty for obstacle avoidance with known object poses. The approach utilized both MC dropout and bootstrapping for extracting epistemic uncertainty from measurements. On the other hand, we deal with classifier epistemic uncertainty with unknown poses.

The problem of planning under uncertainty, and in particular in uncertain/unknown environments, is an instantiation of a Partially Observable Markov Decision Process (POMDP). Calculating an optimal global solution, however, is computationally intractable [89]; thus, approaches that trade-off reduced computational complexity and sub-optimal performance must be considered. A common approach is BSP, that typically considers state transition and observation models to be known. The environment in general is only partially observable, thus due to stochasticity we must reason about the "belief space", i.e. the space of all possible distributions over the state space. BSP typically considers state transition and observation models to be given. Van Der Berg et al. [90] proposed a locally optimal motion planning algorithm in the belief space. While typically the belief states are assumed Gaussian, Platt et al. [91] presented an approach that relaxes this assumption. Platt et al. [92] proposed a BSP algorithm for an underactuated agent. Using the belief space, we can plan a trajectory that minimizes a cost function, usually aiming for the most informative measurements. Recently BSP was considered in the context of active SLAM, where the environment is unknown or uncertainty a-priori. While previous approaches assumed a discrete action space (e.g. [93]), maximum likelihood measurements, and a-priory known environment, Indelman et al. [94] proposed a BSP scheme for continuous action space, measurements modeled as random variables, and unknown environment. Atanasov et al. [12] and Patten et al. [95] presented approaches for active classification using a viewpoint

dependent classifier mode using a sampling based method. In the former, the robot and object poses are known, while in the later they are part of the state. Chaves et al. [96] proposed a formulation that reasons whether measurements were actually acquired. Another commonly used assumption in BSP is known data association; Pathak et al. [49] proposed Data Association BSP (DA-BSP) that relaxes this assumption, reasoning about data association within planning. This work enables autonomous disambiguations of data association hypotheses, that can relate to active classification. Burks et al. [97] uses continuous state POMDP with hybrid beliefs over continuous and discrete states.

Deep reinforcement learning (DRL) was first proposed by Mnih et al. [98], utilizing a neural network to learn an action-value function (also known as Q-function) that dictates the control policy. DRL provided promising results in several fields of study, those include robot navigation. Zhelo et al. [99] proposed a curiosity driven reinforcement learning method. Tai and Liu [100] presented a DRL based approach to navigate a robot via depth measurements only in an unknown environment. Tai et al. [101] showed that a robot can be trained via DRL in a virtual environment and perform well in a real one.

There are several works about distributed multi-agent systems that utilize deep reinforcement learning. Chen et al. [102] addressed multi-agent distributed collision avoidance using deep reinforcement learning. Chen et al. [103] proposed a socially aware DRL method for multi-agent navigation in an environment with many humans. Foerster et al. [104] used DRL to learn a communication strategy between agents in a distributed framework. Omidshafiei et al. [105] addressed the problem of multi-task multi-robot DRL with partial observability.

Chapter 3

DA Aware SLAM Using a Viewpoint Dependent Classifier Model

In the approach presented in this chapter the robot aims to localize itself and map geometrically and semantically the observed environment while reasoning about ambiguous data association. This kind of inference requires maintaining a hybrid belief and efficiently updating it with incoming information captured online by the robot's sensors. As our main contribution of this chapter, we utilize a viewpoint dependent classifier model for DA disambiguation by leveraging the coupling between relative viewpoint and classifier outputs. We rigorously incorporate this viewpoint-dependent model within a recursive probabilistic formulation, building upon the DA-BSP framework by Pathak et al. [50], which however, considered only geometric observations. In addition, the proposed approach aids in SLAM, leading to a more accurate inference. Further, while DA-BSP assumes a single scene is observed per time step, we deal with multiple object detections. We demonstrate the strength of utilizing a viewpoint dependent classifier model for DA disambiguation in simulation considering a highly ambiguous environment.

3.1 Preliminaries

Consider a robot operating in a partially known environment containing different, possibly perceptually similar or identical, objects. The robot aims to localize itself, and map the environment geometrically and semantically while reasoning about ambiguous data association (DA). We consider a closed-set setting where each object is assumed to be one of M classes. Moreover, in this work we consider the number of objects in the environment is known. The objects are assumed to be stationary.

Let x_k denote the robot's camera pose at time k; let x_n^o and c_n represent the n-th object pose and class, respectively. We denote the set of all object poses and classes by

$\mathcal{X}^o \doteq \{x_1^o, ..., x_N^o\}$ and $C \doteq \{c_1, ..., c_n\}$. To shorten notations, denote $\mathcal{X}_k \doteq \{x_{0:k}, \mathcal{X}^o\}$.

Further, we denote the data association realization at time k as β_k: given n_k object observations at time k, $\beta_k \in \mathbb{R}^{n_k}$; each element in β_k corresponds to an object observation, and is equal to an object's identity label. For example, if at time k the camera observes 2 objects with hypothesized identity labels 4 in observation 1 and 6 in observation 2, then $\beta_k = [\beta_{k,1}, \beta_{k,2}]^T \in \mathbb{R}^2$, and $\beta_{k,1} = 4$ and $\beta_{k,2} = 6$. Denote $\mathcal{Z}_k \doteq \{z_{k,1}, ..., z_{k,n_k}\}$ as the set of n_k measurements at time k, and a_k as the robot's action at time k.

Each measurement $z_{k,i} \in \mathcal{Z}_k$ consists of two parts: a geometric part $z_{k,i}^{geo}$, e.g. range or bearing measurements to an object, and a semantic part $z_{k,i}^{sem}$. The set of all geometric measurements for time k is denoted $\mathcal{Z}_k^{geo}$, and similarly for semantic measurements $\mathcal{Z}_k^{sem}$, such that $\mathcal{Z}_k = \mathcal{Z}_k^{geo} \cup \mathcal{Z}_k^{sem}$. We assume the geometric and semantic measurements are independent from each other. In addition, we assume independence between measurements at different time steps.

We consider standard motion and geometric observation Gaussian models, such that $\mathbb{P}(x_{k+1}|x_k, a_k) = \mathcal{N}(f(x_k, a_k), \Sigma_w)$ and $\mathbb{P}(z_k^{geo}|x_k, x^o) = \mathcal{N}(h^{geo}(x_k, x^o), \Sigma_v^{geo})$. The process and geometric measurement covariance matrices, Σ_w and Σ_v^{geo}, as well as the functions $f(.), h^{geo}(.)$ are assumed to be known.

For the semantic measurements, we utilize a (deep learning) classifier that provides a vector of class probabilities where $z_{k,i}^{sem} \doteq \mathbb{P}(c_i|I_{k,i})$ given sensor raw observation $I_{k,i}$, e.g. an image cropped from a bounding box of a larger image taken by the camera of object i at time k. To simplify notations we drop index i, as the measurements, both semantic and geometric, apply to each bounding box. Thus, $z_k^{sem} \in \mathbb{R}^M$ with

$$z_k^{sem} \doteq \begin{bmatrix} \mathbb{P}(c = 1|I_k) & \cdots & \mathbb{P}(c = M|I_k) \end{bmatrix}^T. \tag{3.1}$$

A crucial observation, following [16], is that z_k^{sem} is dependent on the camera's pose relative to the object (see Fig. 3.1). In this work we contribute an approach that leverages this coupling to assist in inference and data association disambiguation.

Specifically, we model this dependency via a classifier model $\mathbb{P}(z_k^{sem}|c = m, x_k, x^o)$. The classifier model represents the distribution over classifier output, i.e. class probability vector z_k^{sem}, when an object with a class hypothesis m is observed from relative pose $x^o \ominus x_k$. Note that for M classes we require M classifier models, one for each class. The model can be represented with a Gaussian Process (see [15, 16]) or a deep neural network (see [17]). In this work, we use a Gaussian classifier model, given by

$$\mathbb{P}(z_k^{sem} \mid c, x_k, x^o) = \mathcal{N}(h_c(x_k, x^o), \Sigma_c(x_k, x^o)), \tag{3.2}$$

where the viewpoint-dependent functions $h_c(x_k, x^o)$ and $\Sigma_c(x_k, x^o)$ are learned offline. Note that unlike [15, 16] we do not model correlations in classifier scores among viewpoints. Conversely, we do not assume data association is known.

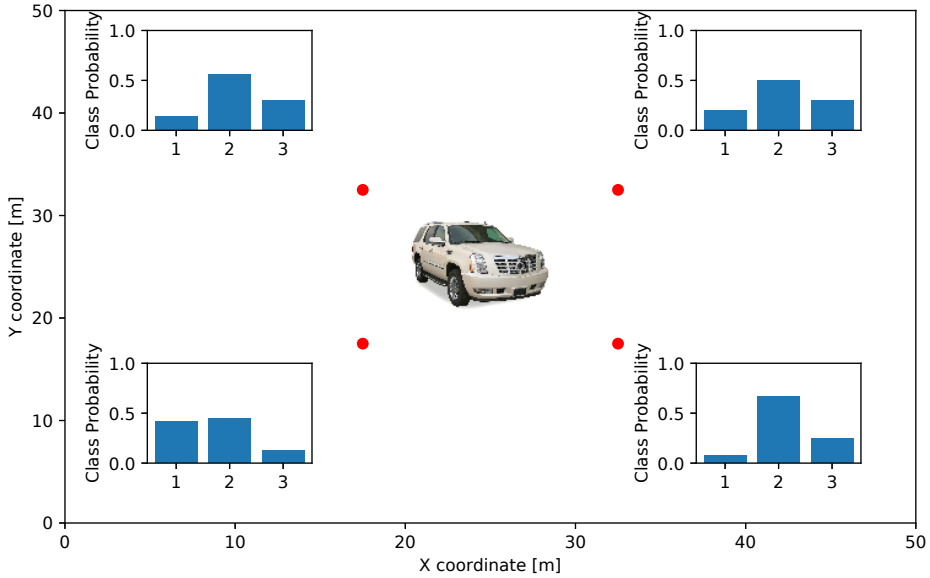


Figure 3.1: A classifier observing an object from multiple viewpoints will produce different classification scores for each viewpoint.

We assume a prior on initial camera and object poses, x_0 and X^o respectively, and class realization probability $\mathbb{P}(C)$. For simplicity, we assume independent variable priors (although this assumption is not required by our approach, and is not true in general, as e.g. some objects are more likely to appear together than others), thus we can write the prior as follows:

$$\mathbb{P}(x_0, \mathcal{X}^o, C) = \mathbb{P}(x_0) \prod_{i=1}^{N} \mathbb{P}(x_i^o)\mathbb{P}(c_i). \tag{3.3}$$

In this paper we use a Gaussian prior for the continuous variables, and uninformative (uniform) prior for the object classes.

Problem formulation: We aim to efficiently maintain the following hybrid belief

$$\mathbb{P}(\mathcal{X}_k, C, \beta_{1:k} \mid \mathcal{H}_k), \tag{3.4}$$

with history $\mathcal{H}_k \doteq \{\mathcal{Z}_{1:k}, a_{0:k-1}\}$. The belief (3.4) is both over continuous variables, i.e. robot and object poses $\mathcal{X}_k$ (continuous variables), and over discrete variables, i.e. object classes C and data association hypotheses thus far, $\beta_{1:k}$. In the following, we incorporate a viewpoint-dependent classifier model and develop a recursive formulation to update that hybrid belief with incoming information captured by the robot as it moves in the environment.

3.2 Approach

In this section we develop a recursive scheme to compute and maintain the hybrid belief from Eq. (3.4). We start by factorizing using the chain rule as

$$\mathbb{P}(\mathcal{X}_k, C, \beta_{1:k}|\mathcal{H}_k) = \underbrace{\mathbb{P}(\mathcal{X}_k|C, \beta_{1:k}, \mathcal{H}_k)}_{b[\mathcal{X}_k]^C_{\beta_{1:k}}} \underbrace{\mathbb{P}(C, \beta_{1:k}|\mathcal{H}_k)}_{w^C_{\beta_{1:k}}}, \tag{3.5}$$

where $b[\mathcal{X}_k]^C_{\beta_{1:k}} \doteq \mathbb{P}(\mathcal{X}_k \mid C, \beta_{1:k}, \mathcal{H}_k)$ is the conditional belief over the continuous variables, and $w^C_{\beta_{1:k}} \doteq \mathbb{P}(C, \beta_{1:k} \mid \mathcal{H}_k)$ is the marginal belief over the discrete variables, and can be considered as the conditional belief weight. Thus, each realization of the discrete variables, i.e. data association and class hypotheses, has its own probability (weight) and gives rise to a different belief over the continuous variables.

Moreover, the factorization (3.5) facilitates computation of marginal distributions that are of interest in practice. In particular, the posterior over robot and object poses can be calculated via

$$\mathbb{P}(\mathcal{X}_k \mid \mathcal{H}_k) = \sum_{\beta_{1:k}} \sum_{C} w^C_{\beta_{1:k}} b[\mathcal{X}_k]^C_{\beta_{1:k}}, \tag{3.6}$$

while the marginal distributions over object classes and data association hypotheses are given by

$$\mathbb{P}(C \mid \mathcal{H}_k) = \sum_{\beta_{1:k}} w^C_{\beta_{1:k}}, \tag{3.7}$$

$$\mathbb{P}(\beta_{1:k} \mid \mathcal{H}_k) = \sum_{C} w^C_{\beta_{1:k}}. \tag{3.8}$$

The posterior $\mathbb{P}(\mathcal{X}_k \mid \mathcal{H}_k)$ in Eq. (3.6) is a mixture belief that accounts for all hypotheses regarding data association and classification. Without semantic observations, our approach degenerates to passive DA-BSP. The term $\mathbb{P}(C \mid \mathcal{H}_k)$ is the distribution over classes of all objects while accounting for both localization uncertainty and ambiguous data association. As such, it is important for robust semantic perception. Finally, the posterior over data association hypotheses, $\mathbb{P}(\beta_{1:k} \mid \mathcal{H}_k)$ accounts for all class realizations for all objects.

Next, we derive a recursive formulation for calculating the continuous and marginal distributions in the factorization (3.5). As will be seen, semantic observations along with the viewpoint-dependent classifier model (3.2) impact both of the terms in the factorization (3.5), and as a result assist in inference of robot and objects poses (via Eq. (3.6)) and helps in disambiguation between data association realizations (via Eq. (3.8)). Furthermore, as discussed in Sec. 2.D, while the number of objects' classes and data association hypotheses (number of weights $w^C_{\beta_{1:k}}$) is intractable, in practice many of

these are negligible and can be pruned.

3.2.1 Conditional Belief Over Continuous Variables: $b[\mathcal{X}_k]^C_{\beta_{1:k}}$

Using Bayes law we get the following expression:

$$\begin{aligned} b[\mathcal{X}_k]^C_{\beta_{1:k}} \equiv \mathbb{P}(\mathcal{X}_k \mid C, \beta_{1:k}, \mathcal{H}_k) \propto \\ \mathbb{P}(\mathcal{Z}_k \mid \mathcal{X}_k, C, \beta_k) \cdot \mathbb{P}(\mathcal{X}_k \mid C, \beta_{1:k-1}, \mathcal{H}_k^-), \end{aligned} \tag{3.9}$$

where $\mathcal{H}_k^- \doteq \{\mathcal{Z}_{1:k-1}, a_{0:k-1}\}$, the normalization constant is omitted as it does not depend on $\mathcal{X}_k$, and β_k is dropped in the second term because it refers to association of $\mathcal{Z}_k$ which is not present.

The expression $\mathbb{P}(\mathcal{Z}_k \mid \mathcal{X}_k, C, \beta_k)$ in Eq. (3.9) is the joint measurement likelihood for all geometric and semantic observations obtained at time k. Given classifications, associations and robot pose at time k, history $\mathcal{H}_k^-$ and past associations $\beta_{1:k-1}$ can be omitted. The joint measurement likelihood can be explicitly written as

$$\begin{aligned} \mathbb{P}(\mathcal{Z}_k \mid \mathcal{X}_k, C, \beta_k) = \\ \prod_{i=1}^{n_k} \mathbb{P}(z^{geo}_{k,i} \mid x_k, x^o_{\beta_{k,i}}) \cdot \mathbb{P}(z^{sem}_{k,i} \mid x_k, x^o_{\beta_{k,i}}, c_{\beta_{k,i}}), \end{aligned} \tag{3.10}$$

where $x^o_{\beta_{k,i}}$ and $c_{\beta_{k,i}}$ are the object pose and class corresponding to the measurement respectively, given DA realization β_k and n_k the number of measurements obtained at time k as before. Note that the viewpoint-dependent semantic measurement term above $\mathbb{P}(z^{sem}_{k,i} \mid x_k, x^o_{\beta_{k,i}}, c_{\beta_{k,i}})$ couples between semantic measurement and robot pose relative to object, making it useful for inference of both.

The term $b^-[\mathcal{X}_k]^C_{\beta_{1:k-1}} \doteq \mathbb{P}(\mathcal{X}_k \mid C, \beta_{1:k-1}, \mathcal{H}_k^-)$ in Eq. (3.9) is the propagated belief over continuous variables, which, using chain rule, can be written as

$$b^-[\mathcal{X}_k]^C_{\beta_{1:k-1}} = \mathbb{P}(x_k | x_{k-1}, a_{k-1}) b[\mathcal{X}_{k-1}]^C_{\beta_{1:k-1}}. \tag{3.11}$$

Overall, the conditional belief (3.9) can be represented as a factor graph (Kschischang et al. [106]). Note that each realization of $\beta_{1:k}$ has a different factor graph topology (observation factors are affected, motion model factors are not). For a fixed $\beta_{1:k}$ with different class assignments C the corresponding conditional belief factor graph topology remains the same (geometric and semantic observation factors connect the same nodes), but semantic factors change, according to class models.

Fig. 3.2 presents an example for 2 factor graphs, in which $k = 2$, $N = 2$, and the DA hypothesis is that at time $k = 1$ the camera observes object 1 for the first graph and 2 for the second, at time $k = 2$ the camera observes objects 1 and 2 for both graphs. To efficiently infer $\mathcal{X}_k$ for every realization of C and $\beta_{1:k}$, state of the art incremental inference approaches, such as iSAM2 [36] can be used. The joint posterior from Eq. (3.4) can thus be maintained following Eq. (3.5) as a set of continuous beliefs

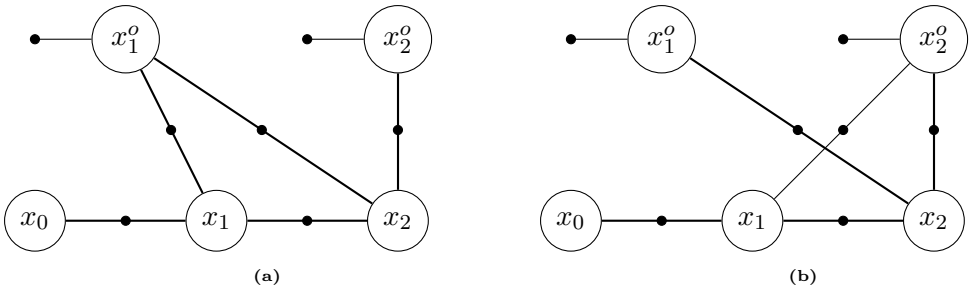


Figure 3.2: A toy example for two factor graphs in our approach, each for a different data association realization. The edges that connect between camera poses correspond to motion model $\mathbb{P}(x_k|x_{k-1}, a_{k-1})$. The edges that connect directly between camera and object poses correspond to the measurement model (3.10) for both semantic and geometric measurements. Thus a viewpoint-dependent semantic measurement model results in geometric constraints on robot-to-object relative pose.

$b[\mathcal{X}_k]^C_{\beta_{1:k}}$ conditioned on the discrete variables $\beta_{1:k}$ and C each represented with a factor graph, along with their corresponding component weights $w^C_{\beta_{1:k}}$, describing the marginal belief over discrete variables. In the next section, we describe how the latter can be calculated.

3.2.2 Marginal Belief Over Discrete Variables: $w^C_{\beta_{1:k}}$

To compute the DA and class realization weight $w^C_{\beta_{1:k}}$ we marginalize over all continuous variables:

$$w^C_{\beta_{1:k}} \equiv \mathbb{P}(C, \beta_{1:k} \mid \mathcal{H}_k) = \int_{\mathcal{X}_k} \mathbb{P}(\mathcal{X}_k, C, \beta_{1:k}|\mathcal{H}_k) d\mathcal{X}_k. \tag{3.12}$$

Using Bayes law, we can expand the above as follows:

$$\begin{aligned} &\mathbb{P}(\mathcal{X}_k, C, \beta_{1:k} \mid \mathcal{H}_k) = \\ &\eta \cdot \mathbb{P}(\mathcal{Z}_k \mid \mathcal{X}_k, C, \beta_{1:k}) \cdot \mathbb{P}(\mathcal{X}_k, C, \beta_{1:k}|\mathcal{H}_k^-) \end{aligned} \tag{3.13}$$

where $\eta = \mathbb{P}(\mathcal{Z}_k \mid \mathcal{H}_k^-)^{-1}$ is a normalization constant and the joint measurement likelihood $\mathbb{P}(\mathcal{Z}_k \mid \mathcal{X}_k, C, \beta_{1:k})$ can be explicitly written as in Eq. (3.10).

We further expand $\mathbb{P}(\mathcal{X}_k, C, \beta_{1:k}|\mathcal{H}_k^-)$ using chain rule:

$$\begin{aligned} &\mathbb{P}(\mathcal{X}_k, C, \beta_{1:k}|\mathcal{H}_k^-) = \mathbb{P}(\beta_k|\beta_{1:k-1}, \mathcal{X}_k, C, \mathcal{H}_k^-) \cdot \\ &\cdot \mathbb{P}(x_k|x_{k-1}, a_{k-1}) \cdot \mathbb{P}(\mathcal{X}_{k-1}, C, \beta_{1:k-1} \mid \mathcal{H}_{k-1}), \end{aligned} \tag{3.14}$$

where:

$$\mathbb{P}(\mathcal{X}_{k-1}, C, \beta_{1:k-1}|\mathcal{H}_{k-1}) = w^C_{\beta_{1:k-1}}\ b[\mathcal{X}_{k-1}]^C_{\beta_{1:k-1}}. \tag{3.15}$$

Eq. (3.15) is the prior belief calculated at time $k-1$ and represented as a continuous belief component along with corresponding weight as described above. The term $\mathbb{P}(\beta_k|\beta_{1:k-1}, \mathcal{X}_k, C, H_k^-)$ from Eq. (3.14) is the object observation model that represents the probability of observing a scene given a hypothesis of camera and object poses. In this paper we use a simple model that depends only on camera and object poses at current time step, thus it can be written as $\mathbb{P}(\beta_k \mid x_k, \mathcal{X}^o_{\beta_k})$, where $\mathcal{X}^o_{\beta_k} \doteq \{x^o_{\beta_{k,i}}\}_{i=1}^{n_k}$. If the model predicts observation of all objects corresponding to β_k then $\mathbb{P}(\beta_k \mid x_k, \mathcal{X}^o_{\beta_k})$

is equal to a constant, otherwise it is zero.

Plugging the above into Eq. (3.12) yields a recursive rule for calculating component weights at time k

$$w^C_{\beta_{1:k}} = \eta \cdot \int_{\mathcal{X}_k} \mathbb{P}(\mathcal{Z}_k|\mathcal{X}_k, C, \beta_k) \cdot \mathbb{P}(\beta_k|x_k, \mathcal{X}^o_{\beta_k}) \cdot \\ \cdot b^-[\mathcal{X}_k]^C_{\beta_{1:k-1}} w^C_{1:k-1} \, d\mathcal{X}_k. \tag{3.16}$$

The normalization constant η (from Eq. (3.13)) does not depend on variables and cancels out when weights are normalized to sum to 1. It is therefore dropped out in subsequent calculations. Note that the realization weight from the previous time step $w^C_{\beta_{1:k-1}}$ is independent from $\mathcal{X}_k$, and thus can be taken out of the integral. Recalling Eq. (3.10), the continuous variables participating in $\mathbb{P}(\mathcal{Z}_k \mid \mathcal{X}_k, C, \beta_{1:k})$ are x_k and $\mathcal{X}^o_{\beta_k}$. Those variables are participating also in $\mathbb{P}(\beta_k \mid x_k, x^o_k)$. As $b^-[\mathcal{X}_k]^C_{\beta_{1:k-1}}$ is Gaussian, all other continuous variables can be marginalized easily. On the other hand, x_k and $\mathcal{X}^o_{\beta_k}$ must be sampled because of the object observation model $\mathbb{P}(\beta_k \mid x_k, \mathcal{X}^o_k)$, which is commonly not Gaussian. If the observation model predicts that the objects will not be observed for most of the samples, then $w^C_{\beta_{1:k}}$ will be small and likely to be pruned. We can express the realization weight as follows:

$$w^C_{\beta_{1:k}} \propto w^C_{\beta_{1:k-1}} \iint_{x_k, \mathcal{X}^o_{\beta_k}} \mathbb{P}(\mathcal{Z}_k \mid x_k, \mathcal{X}^o_{\beta_k}, C, \beta_{1:k}) \cdot \\ \cdot \, \mathbb{P}(\beta_k \mid x_k, \mathcal{X}^o_{\beta_k}) \; b^-[x_k, \mathcal{X}^o_{\beta_k}]^C_{\beta_{1:k-1}} \; dx_k \; d\mathcal{X}^o_{\beta_k}, \tag{3.17}$$

where:

$$b^-[x_k, \mathcal{X}^o_{\beta_k}]^C_{\beta_{1:k-1}} \doteq \mathbb{P}(x_k, \mathcal{X}^o_{\beta_k}|C, \beta_{1:k-1}, \mathcal{H}^-_k) = \\ \int_{\mathcal{X}^-_k \setminus \{x_k, \mathcal{X}^o_{\beta_k}\}} b^-[\mathcal{X}_k]^C_{\beta_{1:k-1}} \; d\left\{\mathcal{X}^-_k \setminus \{x_k, \mathcal{X}^o_{\beta_k}\}\right\}. \tag{3.18}$$

The viewpoint dependent classifier model contributes to data association disambiguation by acting as reinforcement or contradiction to the geometric model. If both 'agree' on the poses' hypothesis, $w^C_{\beta_{1:k}}$ will be large relative to cases where both 'disagree'.

Next, we provide an overview of the inference scheme, then address computational aspects.

3.2.3 Overall Algorithm

The proposed scheme is outlined in Alg. 3.1. For every time step we are input the prior belief $\mathbb{P}(\mathcal{X}_{k-1}, C, \beta_{1:k-1} \mid \mathcal{H}_{k-1})$ represented following Eq. (3.5) as a set of weights $w^C_{\beta_{1:k-1}} \doteq \mathbb{P}(C, \beta_{1:k-1} \mid \mathcal{H}_{k-1})$ and corresponding continuous (Gaussian) belief components $b[\mathcal{X}_{k-1}]^C_{\beta_{1:k-1}} \doteq \mathbb{P}(\mathcal{X}_{k-1} \mid C, \beta_{1:k-1}, \mathcal{H}_{k-1})$. In our implementation we maintain a separate factor graph for each such component. We also obtain an action a_{k-1} and observations $\mathcal{Z}_k$, separated into geometric $\mathcal{Z}^{geo}_k$, and semantic $\mathcal{Z}^{sem}_k$. We propagate each

prior belief component using the motion model $\mathbb{P}(x_k \mid x_{k-1}, a_{k-1})$ (step 3). Each component then splits to a number of subcomponents, one for each possible assignments of data associations β_k at current time (generally a vector of length n_k). Procedure PropWeights at step 5 computes the normalized weight of each subcomponent via Eq. (3.17) as a product of the component (prior) weight $w^C_{\beta_{1:k-1}}$ with an update term comprising the measurement likelihood $\mathbb{P}(\mathcal{Z}_k \mid x_k, \mathcal{X}^o_{\beta_k}, C, \beta_{1:k})$ (both geometric and semantic, see Eq. (3.10)) and object observation model $\mathbb{P}(\beta_k \mid x_k, \mathcal{X}^o_{\beta_k})$, averaged over the propagated belief $b^-[x_k, \mathcal{X}^o_{\beta_k}]^C_{\beta_{1:k-1}}$ from Eq. (3.18). In step 7 we prune low-weight subcomponents by setting their weights to 0 and re-normalizing remaining weights to 1, in an approximation to true posterior (other pruning strategies are equally possible). In step 11 we update the posterior for non-zero weight subcomponents using current measurements. Finally, we return posterior as a set of Gaussian components and corresponding weights.

We next address aspects of computational tractability of the scheme.

2.D Computational Complexity and Tractability

With M candidate classes, and N objects, the number of possible class realizations, and consequently initial number of belief components, is M^N. At time step k each prior component splits into up to N^{n_k} subcomponents as each measurement can in general be associated to any scene object. The maximum number of components at time k is thus $M^N \cdot \prod_{j=1}^k N^{n_j} = O\left(M^N \cdot N^{\psi \cdot k}\right)$ if ψ is an upper bound on n_k, making the approach computationally intractable in theory without pruning. In practice, as observed by [50], the number of components that need to be accounted for is limited by the belief, and is much smaller than the theoretical maximum, with the rest getting negligible weights that can be safely pruned under any scheme. Further, our empirical results suggest that semantic information added through the viewpoint-dependent factors leads to even stronger disambiguation than observed in DA-BSP (which uses only geometric information), both in data association and localization, resulting in smaller number of non-negligible weights.

Additionally, we hypothesize that in practice classification uncertainty is usually limited to only a few classes, and thus would not cause a computational bottleneck even with numerous candidate classes. Further, we avoid explicitly maintaining the initially exponential number of components (M^N) by noting that the classes of objects that were not observed yet under an association hypothesis $\beta_{1:k}$ do not participate in the inference process for that belief component, and thus do not need to be maintained separately. That is, for two class realizations C and C', if $C_{\beta_{1:k}} = C'_{\beta_{1:k}}$ with $C_{\beta_{1:k}} \doteq \{\forall 1 \leq j \leq k, i \;\; c_{\beta_{j,i}}\}$ (i.e. classifications for all associated objects are the same) and $C_{\neg\beta_{1:k}} \neq C'_{\neg\beta_{1:k}}$ (i.e. realizations differ on classifications for objects that do not participate in $\beta_{1:k}$), then $w^{C'}_{\beta_{1:k}} = w^C_{\beta_{1:k}}$ (assuming uninformative prior on classes) and $b[\mathcal{X}_k]^C_{\beta_{1:k}} = b[\mathcal{X}_k]^{C'}_{\beta_{1:k}}$ (always), without need to compute or maintain those separately.

Finally we note that parts of Alg. 3.1 can be readily parallelized ('embarrassingly

parallel'), thanks to computations being independent across components and wide availability of massively parallel processors (e.g. GPUs), contributing to its practical applicability.

3.3 Experiments

In this section we evaluate the performance of our approach in a 2D simulation and demonstrate the advantage of using a viewpoint dependent classifier model for disambiguating between DA realizations and improving inference accuracy. Our implementation uses the GTSAM library [107] with a Python wrapper; all experiments were run on an Intel i7-7700 CPU running at 2800 GHz and with 16GB RAM.

We consider a scenario where the robot navigates in an uncertain perceptually aliased environment represented by a set of scattered objects of the same class, i.e. objects differ in their position and orientation. In this scenario $M = 2$ and $N = 6$, thus the number of possible class realizations is $M^N = 64$. Fig. 3.3a shows the ground truth object poses and robot trajectory.

The prior (3.3) comprises a highly uncertain initial robot pose, and an uninformative prior on object classes. Object poses are assumed to be known up to a certain accuracy (i.e. uncertain map). The prior covariance of the objects is $\Sigma_o = diag(0.05, 0.05, 0.5 \cdot 10^{-3})$, and initial robot pose is $\Sigma_p = diag(100, 100, 0.04)$. The process and geometric measurement covariance matrices are $\Sigma_w = diag(0.75 \cdot 10^{-3}, 0.75 \cdot 10^{-3}, 0.25 \cdot 10^{-3})$ (corresponds to spatial coordinates and orientation), and $\Sigma_v^{geo} = diag(0.1, 0.05)$ (corresponds to range and bearing).

The semantic measurement model (3.2) is defined as:

$$h_c(c = 1, \theta) = \begin{bmatrix} \alpha \sin^2(\theta/2) + (1 - \alpha) \\ \alpha - \alpha \sin^2(\theta/2) \end{bmatrix} \tag{3.19}$$

where θ is the relative angle from the object to camera, calculated from the relative pose $x_k^{rel} \doteq x_k \ominus x^o$. This chosen model represents a mirror symmetrical object (e.g. a car) with a parameter α that corresponds to the viewpoint dependency 'strength', i.e. $\frac{\partial h_c}{\partial \theta}$ values are larger when α increases (for computation details, see [17]). We assume the classifier scores are independent from camera-to-object range as the observations are cropped from bounding boxes, and unless the camera is very close to the object the perspective distortion is negligible. In practice, the classifier model can be learned from images of an object from different viewpoints with corresponding classifier outputs via a neural network or GP for example. The measurement covariance matrix $\Sigma_c \doteq (R^T R)^{-1}$ is defined as $R = K \begin{bmatrix} 1 & -0.5 \\ 0 & 1 \end{bmatrix}$. We note that in general, also Σ_c can be viewpoint-dependent [16,17]. The parameters α and K are constants and take the values $\alpha = 0.25$ and $K = 15$ by default. We sample measurements from our motion, geometric, and semantic models.

Further, we sample 1000 sets of x_k and $x^o_{\beta_k}$ for each computation of $w^C_{\beta_{1:k}}$, see procedure PROPWEIGHTS in Alg. 3.1, and compute them as shown in Eq. (3.17). At each time k we prune components with weight w below threshold $\frac{\{w_k\}_{max}}{150} \leq w$, where $\{w_k\}_{max}$ is the highest weight component at time k.

We compare performance of our approach that utilizes semantic observations along with a viewpoint-dependent classifier model against an alternative that does not use this information, with the latter roughly corresponding to the passive instance of DA-BSP [50]. To quantify performance as a function of α we compare between the following metrics:

1. Entropy over data association weights: for N_k non-pruned weights $\{w_i\}_{i=1}^{N_k}$ we compute the entropy $H(w)$ with $H(w) \doteq -\sum_{i=1}^{N_k} w_i \log(w_i)$.

2. Determinant of position covariance $det(\Sigma)$ of x_k for the highest weight realization at each time k.

3. Estimation error $\tilde{x}^{w_{max}}$, which is the Euclidean distance from ground truth to highest weight estimation for the last pose.

4. Estimation error $\tilde{x}^{w-avg}$, which is the weighted average of all estimation errors for the last pose.

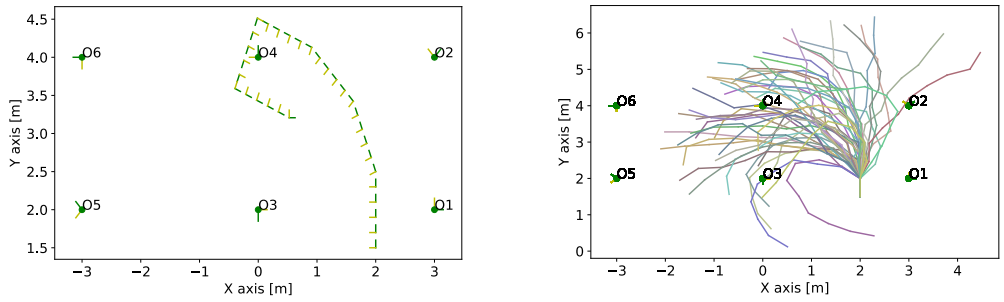


Figure 3.3: **(a)** An example scenario with ground truth camera trajectory, represented in terms of camera poses (green line is the camera heading) and objects $O1$ to $O6$ (green dots indicate position, orientation is indicated by green lines from the dots). **(b)** Multiple sampled paths for the statistical study, each path realization is presented with different color.

Fig. 3.4 shows results of an example scenario for different time steps, comparing between using the viewpoint-dependent classifier model (middle row) and without semantic information (upper row), essentially utilizing passive DA-BSP [50]. At each time k, the plots show the mixture posterior $\mathbb{P}(x_k|\mathcal{H}_k)$ over camera pose x_k, calculated from (3.6), where each component is a Gaussian, thus represented by mean and covariance. Estimated camera poses are shown in red and blue lines, where the blue line represents the camera orientation. Components with higher weight are shown with thicker covariance ellipse lines. To reduce clutter, the posterior over the rest of the continuous variables, i.e. object poses and past robot poses, is not shown. Additionally,

the plots show the ground truth trajectory (from Fig. 3.3a) of the robot. The bottom row reports the probabilities of DA hypotheses from (3.8) for different time instances for both compared cases. The correct association is marked with a green circle.

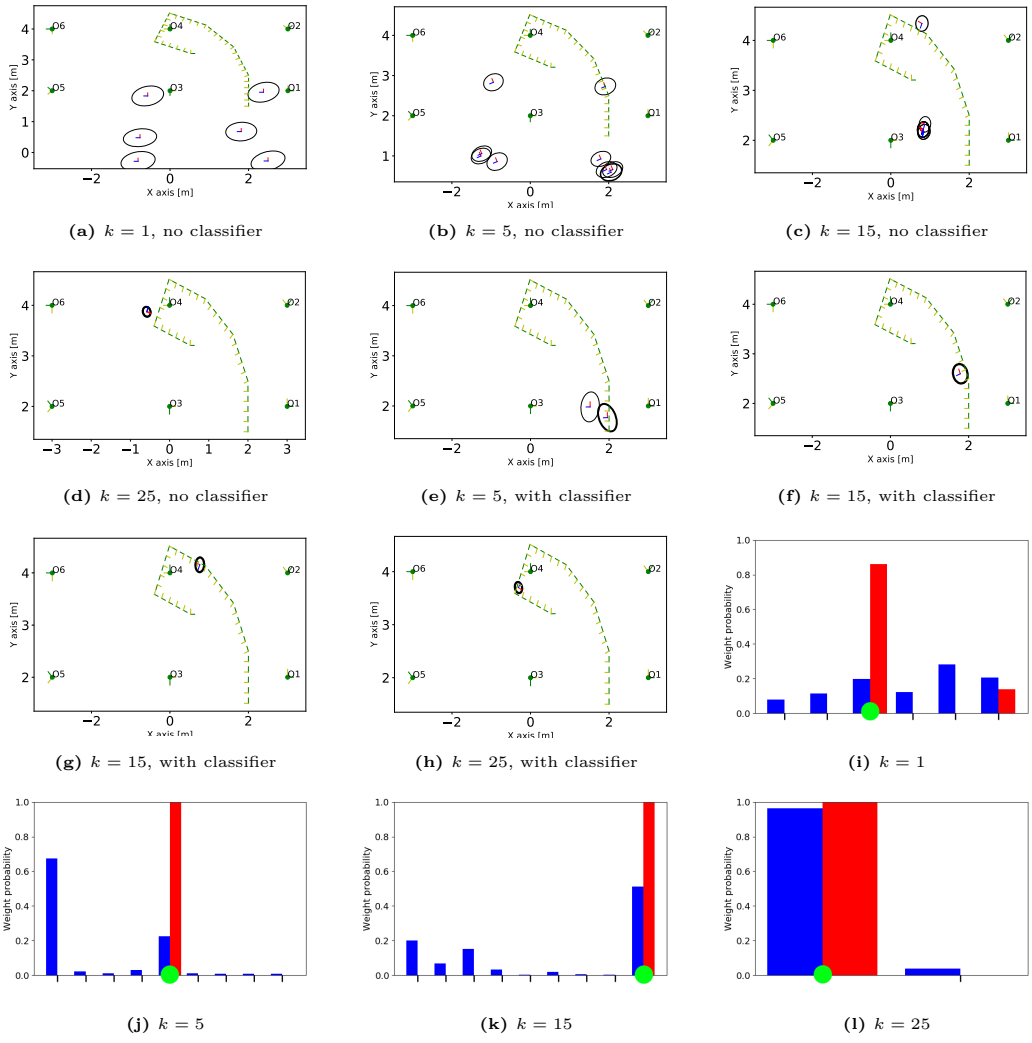
As seen from the upper row of Fig. 3.4, inference without incorporating viewpoint-dependent semantic information results in the first time steps in multiple DA realizations with similar weights. The reason is that given only geometric range and bearing measurements without observing all the objects, inference results can be interpreted in multiple ways, i.e. perceptually aliased (see Fig. 3.4i). Only at time $k = 25$ the DA was disambiguated once the camera observed objects $O1$ and $O2$.

In contrast, utilizing a viewpoint-dependent classifier model admits faster DA disambiguation, as shown in the bottom row of Fig. 3.4. In particular, already at time $k = 1$ the posterior $\mathbb{P}(x_k|\mathcal{H}_k)$ has only two non-negligible components, while at time $k = 5$ there is a single DA realization with significant weight. This shows an improvement over Fig. 3.4b where there are multiple DA realizations with significant weight when not using the classifier model.

The bottom row in Fig. 3.4 presents the realization weights for the times $k = 1, 5, 15, 25$, and compare between weights without and with classifier model. For each realization $\beta_{1:k}$, we present $\mathbb{P}(\beta_{1:k} \mid \mathcal{H}_k)$ after pruning without classifier model as a blue bar, and with as a red bar. If the bar is missing, then $\mathbb{P}(\beta_{1:k} \mid \mathcal{H}_k) = 0$. In all sub-figures the classifier model reduces the number of non pruned DA realizations, and for time $k = 15$ and $k = 25$ the DA is disambiguated with the classifier model. We observe more DA realizations when the classifier model is not used, and at time $k = 15$ the DA with a classifier model fully disambiguated.

Further, we quantify the performance improvement due to the viewpoint-dependent classifier model in a statistical study by sampling multiple ground truth tracks in the scenario, while keeping the same landmarks. The sampled tracks are shown in Fig. 3.3b. For this study, we sampled 50 different tracks with 10 time steps of path length, and performed a statistical analysis on the performance parameters. In all paths, the starting position is identical.

The results of this study are shown in Fig. 3.5, which shows average over each of the mentioned metrics ($H(w)$, $\det(\Sigma)$, $\tilde{x}^{w_{max}}$, $\tilde{x}^{w\text{-avrg}}$). In that figure we also study sensitivity to α, which controls the level of viewpoint-dependency in the considered classifier model (3.19). The plots show a significant improvement of utilizing a classifier model, both for DA disambiguation and inference where the estimation error (Fig. 3.5c, 3.5d) and uncertainty (Fig. 3.5b) are lower when the model is utilized. From all the plots, the most notable performance increase occurs for DA disambiguation (Fig. 3.5a), where stronger viewpoint dependence assists more significantly; Overall, Fig. 3.5 presents a strong advantage for utilizing a viewpoint classifier model in the presented scenario.



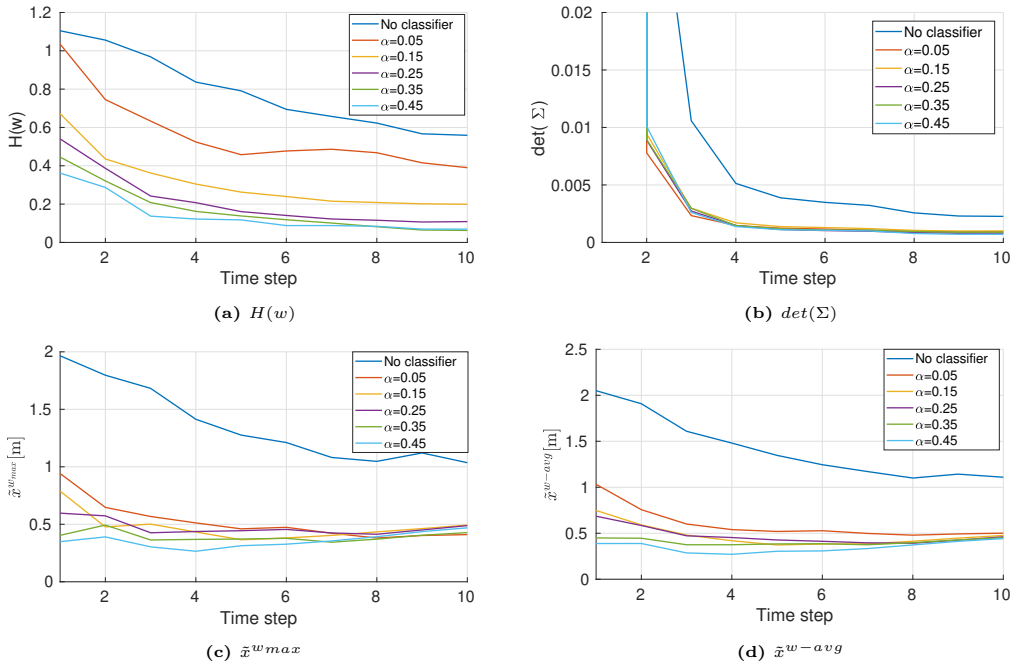
(a) $k = 1$, no classifier **(b)** $k = 5$, no classifier **(c)** $k = 15$, no classifier

(d) $k = 25$, no classifier **(e)** $k = 5$, with classifier **(f)** $k = 15$, with classifier

(g) $k = 15$, with classifier **(h)** $k = 25$, with classifier **(i)** $k = 1$

(j) $k = 5$ **(k)** $k = 15$ **(l)** $k = 25$

Figure 3.4: (a) - (h): Posterior over robot poses of all non-pruned realizations for times $k = 1, 5, 15, 25$, without (first row) and with a classifier model (second row). **Bolder** lines correspond to higher weights. Ground truth trajectory is shown in each of the plots (in terms of camera poses). **(i)-(l)**: Corresponding posterior over data association hypotheses, $\mathbb{P}(\beta_{1:k} \mid \mathcal{H}_k)$, at each time. **Blue** bars are without classifier model, **red** bars are with. **Green** circles represent ground truth data associations.



(a) $H(w)$ **(b)** $det(\Sigma)$ **(c)** $\tilde{x}^{w_{max}}$ **(d)** $\tilde{x}^{w-avg}$

Figure 3.5: Effects of different α values on DA disambiguation ability, estimation uncertainty and accuracy in terms of the metrics ($H(w)$, $det(\Sigma)$, $\tilde{x}^{w_{max}}$, and $\tilde{x}^{w-avg}$), averaged over 50 sampled tracks (see Fig. 3.3b).

Algorithm 3.1 Data Association-Aware Mapping and Localization. Inference at time k

Input: Prior belief $\mathbb{P}(\mathcal{X}_{k-1}, C, \beta_{1:k-1} \mid \mathcal{H}_{k-1})$, observations $\mathcal{Z}_k = (\mathcal{Z}^{geo}, \mathcal{Z}^{sem})$, action a_{k-1}

for every component $\beta_{1:k-1}, C$ **s.t.** $w^C_{\beta_{1:k-1}} > 0$ **do**
  ▷ Propagate component according to motion model
  $b[\mathcal{X}_k^-]^C_{\beta_{1:k-1}} \leftarrow \mathbb{P}(x_k | x_{k-1}, a_{k-1}) \cdot b[\mathcal{X}_{k-1}]^C_{\beta_{1:k-1}}$
  ▷ Propagate weights Eq. (3.16), Eq. (3.17)
  $w^C_{\beta_{1:k}} \leftarrow$ PropW. $\left(b[\mathcal{X}_k^-]^C_{\beta_{1:k-1}}, w^C_{\beta_{1:k-1}}, \mathcal{Z}_k\right)$
  ▷ Prune low-probability components
  $w^C_{\beta_{1:k}} \leftarrow$ PruneAndNormalize$(w^C_{\beta_{1:k}})$
  ▷ Propagate non-zero weight components
  for $\beta_{1:k}, C$ **s.t.** $w^C_{\beta_{1:k}} > 0$ **do**
    ▷ Add observation factors, Eqs. (3.9), and (3.10)
    $b[\mathcal{X}_k]^C_{\beta_{1:k}} \leftarrow b[\mathcal{X}_k^-]^C_{\beta_{1:k-1}} \cdot \mathbb{P}(\mathcal{Z}_k \mid \mathcal{X}_k, C, \beta_k)$
  end for
end for
return $\mathbb{P}(\mathcal{X}_k, C, \beta_{1:k} \mid \mathcal{H}_k) \equiv \{(b[\mathcal{X}_k]^C_{\beta_{1:k}}, w^C_{\beta_{1:k}})\}$

procedure PropWeights$(b[\mathcal{X}_k^-]^C_{\beta_{1:k-1}}, w^C_{\beta_{1:k-1}}, \mathcal{Z}_k)$
  for every possible assignment of β_k **do**
    ▷ Sample current poses by Eq. (3.18)
    Sample $\{x_k^{(i)}, \mathcal{X}^{o\ (i)}_{\beta_k}\}_{i=1}^{n_s} \sim b^-[x_k, \mathcal{X}^o_{\beta_k}]^C_{\beta_{1:k-1}}$
    ▷ Calculate update factor and propagate Eq. (3.17)
    $\psi \leftarrow (1/n_s) \cdot \sum_{i=1}^{n_s} \mathbb{P}(\mathcal{Z}_k, \beta_k \,|\, x_k^{(i)}, \mathcal{X}^{o(i)}_{\beta_k}, C, \beta_{1:k-1})$
    $\widetilde{w}^C_{\beta_{1:k}} \leftarrow w^C_{\beta_{1:k-1}} \cdot \psi$
  end for
  ▷ Normalize weights and return
  return $w^C_{\beta_{1:k}} \leftarrow \widetilde{w}^C_{\beta_{1:k}} / \sum_{\beta_k} \widetilde{w}^C_{\beta_{1:k}}$
end procedure

procedure PruneAndNormalize$(w^C_{\beta_{1:k}})$
  for $\beta_{1:k}, C$ **s.t.** $w^C_{\beta_{1:k}} <$ threshold **do**
    $\widetilde{w}^C_{\beta_{1:k}} \leftarrow 0$
  end for
  return $w^C_{\beta_{1:k}} \leftarrow \widetilde{w}^C_{\beta_{1:k}} / \sum_{\beta_k} \widetilde{w}^C_{\beta_{1:k}}$
end procedure

Chapter 4

Distributed Semantic SLAM with Viewpoint Dependent Classifier Model

In this chapter our main contributions are as follows. (i) we contribute a multi-robot approach that maintains a hybrid belief over robot and object poses, and object classes in a distributed setting, while addressing the coupling between semantic and geometric information via viewpoint-dependent classifier model; (ii) we address estimation consistency aspects considering both continuous and discrete random variables; (iii) we demonstrate the strength of this approach in simulation and real-world experiment, comparing to single robot and distributed multi-robot with double counting.

4.1 Notations and Problem Formulation

Consider a group of robots operating in an unknown environment represented by object landmarks. All of the robots aim to localize themselves, and map the environment geometrically and semantically within a distributed multi-robot framework. In this chapter we consider a closed-set setting, where each of the objects is of one of M possible classes. The number of objects in the environment prior to the scenario is unknown.

We denote states inferred by robot r with a superscript $\square^r$. Set R is the set of all robots communicating with robot r (including itself), either directly, or relayed through other robots. Note that R can increase its size with time. Let x_k denote robot pose at time k, x_n^o and c_n denote the n'th object pose and class respectively. Let $\mathcal{X}^o \doteq \{x_n^o\}_n$ and $C \doteq \{c_n\}_n$ denote poses and classes of objects, and $\mathcal{X}_k \doteq \{x_{0:k}, \mathcal{X}_k^o\}$ denotes all poses up to time k. Subscript new, k representing the objects newly observed at k.

Let $\mathcal{Z}_k^r$ be the set of measurements robot r receives at time k by its own sensors. $\mathcal{Z}_k^r$ is composed of geometric and semantic measurements $\mathcal{Z}_k^{geo,r}$, and $\mathcal{Z}_k^{sem,r}$ respectively. We assume independence between geometric and semantic measurements, as well as

Table 4.1: Main notations used in the chapter.

Parameters	
x	Robot pose
x_n^o, c_n	n'th object pose and class
$\mathcal{X}_k^o$	Poses of objects observed up to time k
$\mathcal{X}_{new,k}^o$	Poses of objects newly observed at time k
$\mathcal{X}_k$	Robot and object poses up to time k
C_k	Object seen up to time k class realization
$C_{new,k}$	Classes of objects newly observed at time k
$\mathcal{Z}_k$	Measurements at time k including geometric and semantic
$\mathcal{M}_k$	Motion model from x_{k-1} to x_k
$\mathcal{L}_k$	Measurement likelihood of $\mathcal{Z}_k$
$\mathcal{H}_k$	History of measurements and action up to time k
b_k	Conditional continuous belief at time k
w_k	Discrete weight at time k
ξ_k	Continuous object marginal belief at time k
ϕ_k	Discrete marginal belief at time k
$N_k(\cdot)$	Number of objects observed by a robot or a group up to time k
Superscripts	
r	States of robot r
R	States of robots communicating with r, directly and indirectly, including itself

between different time steps.

We assume Gaussian and known identical motion $\mathcal{M}_k \doteq \mathbb{P}(x_k|x_{k-1}, a_{k-1})$ and geometric $\mathbb{P}(z_k^{geo,r}|x_k^r, x^{o,r})$ models for all robots. At each time step, there is a subset of object poses involved in the geometric and classifier model that is determined by data association (DA). Unlike chapter 3, herein, DA is assumed to be externally determined.

Additionally, we use a viewpoint-dependent classifier model that "predicts" classification scores (a vector of class probabilities). This model couples classifier scores with viewpoint dependency between object and camera; this coupling can be used to improve pose inference performance (see Chapter 3). The viewpoint dependency is modeled as a Gaussian with parameters that depend on the relative viewpoint from the camera to the object $x^{o,r} \ominus x_k^r$ and object's class c:

$$\mathbb{P}(z_k^{sem,r}|x_k^r, x^o, c) = \mathcal{N}(h_c(x_k^r, x^{o,r}), \Sigma_c(x_k^r, x^{o,r})), \tag{4.1}$$

where $h_c(\cdot)$ and $\Sigma_c(\cdot)$ can be learned offline via a Gaussian Process (GP) [16] or a deep neural network [17]. Note that for M candidate classes, M viewpoint-dependent models have to be learned.

Let $\mathcal{L}_k^r \doteq \mathbb{P}(\mathcal{Z}_k^r|\mathcal{X}_k^r, C_k^r)$ be the local measurement likelihood of r that consists of geometric and classifier models:

$$\mathcal{L}_k^r \doteq \prod_{x^{o,r}, c^r} \mathbb{P}(\mathcal{Z}_k^{geo,r}|x_k^r, x^{o,r}) \mathbb{P}(\mathcal{Z}_k^{sem,r}|x_k^r, x^{o,r}, c^r), \tag{4.2}$$

where $x^{o,r} \in \mathcal{X}_{\beta_k}^{o,r}$ and $c^r \in C_{\beta_k}^r$; the term β_k represents the local DA of robot r at time k, i.e. the correspondences between observations and object IDs. Denote $\mathcal{X}_{\beta_k}^{o,r}$ the set of all poses of objects that observed by r at time k, and similarly denote $C_{\beta_k}^r$ for object classes. For the reader's convenience, Table 4.1 presents the important notations used in the paper, some will be defined in the next section.

Problem formulation For each robot r we aim to maintain the following hybrid belief:

$$\mathbb{P}(\mathcal{X}_k^R, C^R | \mathcal{H}_k^R), \tag{4.3}$$

where $\mathcal{H}_k^R \doteq \{\mathcal{Z}_{1:k}^{r'}, a_{0:k-1}^{r'}\}_{r' \in R}$ is the history of measurements of robot r itself and transmitted information to r, as well as actions from all robots in R. The belief in Eq. (4.3) is a hybrid belief over both continuous (camera and object poses), and discrete (object classes) random variables. We aim to update this hybrid belief per each robot in a recursive manner, using both local measurements and information sent from other robot in the neighborhood, as well as sending information by itself. We aim to keep estimation consistency by avoiding double counting, i.e. using every measurement only once.

4.2 Approach

We present a framework for distributed classification, localization, and mapping. As with many multi-robot distributed frameworks, over-confident estimations, due to double counting, is a key issue; We propose a framework that simplifies the book-keeping that allows relaying of information (e.g. robot 1 sends information to robot 2, then 2 sends to 3 information that also includes the received from robot 1). This framework requires the maintenance of a local belief $\mathbb{P}(\mathcal{X}_k^r, C^r | \mathcal{H}_k^r)$ per each robot that can be sent and relayed to other robots. From multiple local beliefs a distributed belief can be constructed. The local beliefs are stored by each robot, and updated accordingly when new information arrives, and the receiving robot filters out the old information, thus avoiding double counting.

In the next sections we derive a recursive formulation for maintenance of the local belief, the distributed hybrid belief, and the information stack each robot holds and transmits.

2.A Local Hybrid Belief Maintenance

Our formulation for maintaining local hybrid beliefs builds upon our previous work in Chapter 3, with the main differences being that here we assume the DA is solved, and the number of objects is unknown a-priori. In this section we present an overview of this approach.

We maintain the hybrid belief of robot r only from local information. This belief can be split into continuous and discrete parts as in:

$$\mathbb{P}(\mathcal{X}_k^r, C_k^r | \mathcal{H}_k^r) = \underbrace{\mathbb{P}(\mathcal{X}_k^r | C_k^r, \mathcal{H}_k^r)}_{b_k^r} \underbrace{\mathbb{P}(C_k^r | \mathcal{H}_k^r)}_{w_k^r}. \tag{4.4}$$

To maintain this hybrid belief, we must maintain a set of continuous beliefs conditioned

on the class realization of all objects observed in the scene by robot r thus far.

The continuous part can be updated as follows:

$$b_k^r \propto b_{k-1}^r \cdot \mathcal{L}_k^r \cdot \mathcal{M}_k^r \cdot \mathbb{P}(\mathcal{X}_{\text{new},k}^{o,r}), \tag{4.5}$$

where $\mathbb{P}(\mathcal{X}_{\text{new},k}^{o,r}) = \frac{\mathbb{P}(\mathcal{X}_k^{o,r})}{\mathbb{P}(\mathcal{X}_{k-1}^{o,r})}$ is the prior over object poses newly observed at time k. As opposed to Chapter 3, this formulation also supports an increasing number of objects known at each time step, with both $\mathcal{X}_k^{o,r}$ and C_k^r increasing in dimension. Note that in general b_k^r is different for each class realization, as models (4.1) are different for each class.

The discrete part is the weight associated to its corresponding continuous belief. As our measurement models depend on continuous variables, we use Bayes rule on $\mathbb{P}(C_k^r|\mathcal{H}_k^r)$ and marginalize the measurement likelihood as follows:

$$w_k^r \propto w_{k-1}^r \mathbb{P}(C_{\text{new},k}^r) \int_{\mathcal{X}_k^r} \mathcal{L}_k^r \cdot b_{k-1}^r \cdot \mathcal{M}_k^r d\mathcal{X}_k^r, \tag{4.6}$$

where $\mathbb{P}(C_{\text{new},k}^r) = \frac{\mathbb{P}(C_k^r)}{\mathbb{P}(C_{k-1}^r)}$ is the prior over classes of new objects locally observed by r at time k. We compute the integral in Eq. (4.6) by sampling the continuous variables that participate in $\mathbb{P}(\mathcal{Z}_k^r|\mathcal{X}_k^r, C_k^r)$, i.e. the last robot pose x_k^r and the poses of observed objects $\mathcal{X}_{\beta_k}^{o,r}$ at time k. These variables are sampled from the propagated belief $b_{k-1}^r \cdot \mathcal{M}_k^r$. Variables that do not participate in $\mathcal{L}_k^r$ can be marginalized analytically.

2.B Distributed Hybrid Belief Maintenance

In this section we extend the formulation presented in Sec. 2.A to include updates from other robots, considering a distributed multi-robot setting. As will be seen, our formulation uses each measurement only once, thus keeping estimation consistency and avoiding double counting. Similarly to (4.4), we factorize the distributed hybrid belief (4.3)

$$\mathbb{P}(\mathcal{X}_k^R, C_k^R|\mathcal{H}_k^R) = \underbrace{\mathbb{P}(\mathcal{X}_k^R|C_k^R, \mathcal{H}_k^R)}_{b_k^R} \underbrace{\mathbb{P}(C_k^R|\mathcal{H}_k^R)}_{w_k^R}. \tag{4.7}$$

As in the single robot case, maintaining this belief requires managing multiple hypotheses of class realizations. Compared to the single robot case, the number of objects observed will be equal or greater for distributed belief, therefore the number of possible realizations increases as well. Importantly, information transmitted by other robots impacts both b_k^R and w_k^R. Furthermore, the classifier viewpoint-dependent model induces coupling between localization uncertainty and classification of different robots.

We present a recursive formulation for maintaining each of the parts in (4.7). The distributed measurement history $\mathcal{H}_k^R$ can be split to a prior part, and a new part, defined as $\Delta\mathcal{H}_k^R$, that consists of measurements and actions from time k, s.t: $\mathcal{H}_k^R = \mathcal{H}_{k-1}^R \cup \Delta\mathcal{H}_k^R$. Similarly, let $\mathcal{H}_k^r \doteq \mathcal{H}_{k-1}^r \cup \{\mathcal{Z}_k^r, a_{k-1}^r\}$ for the single robot case. Note

information in $\Delta\mathcal{H}_k^R$ transmitted by other robots can potentially be from earlier time instances (as each robot during communication transmits to robot r its own stack of local beliefs of other robots, see Section 2.C). Crucially, each measurement must be used once to avoid double counting. We also denote history *without* local measurements and action at time k as

$$\mathcal{H}_k^{R-} \doteq \mathcal{H}_k^R \backslash \{\mathcal{Z}_k^r, a_{k-1}^r\}\,,\ \Delta\mathcal{H}_k^{R-} \doteq \Delta\mathcal{H}_k^R \backslash \{\mathcal{Z}_k^r, a_{k-1}^r\}. \tag{4.8}$$

Using the above notations, one can observe $\mathcal{H}_k^{R-} = \mathcal{H}_{k-1}^R \cup \Delta\mathcal{H}_k^{R-}$. Next, we detail our approach for maintaining both the conditional continuous part b_k^R and the discrete part w_k^R recursively for a realization of object classes C_k^R.

Maintaining b_k^R

First, we present a relation that is used in equation derivation; Let A be a random variable conditioned on the set $\{B_i\}$ of random variables B_i that are independent from each other. By using Bayes Law, we can split the conditional probability $\mathbb{P}(A|\{B_i\})$ to a product of conditional probabilities:

$$\mathbb{P}(A|\{B_i\}) = \frac{\mathbb{P}(\{B_i\}|A)\mathbb{P}(A)}{\mathbb{P}(\{B_i\})} = \frac{\prod_i \mathbb{P}(B_i|A)}{\prod_i \mathbb{P}(B_i)}\mathbb{P}(A). \tag{4.9}$$

Using Bayes Law again on each element in the product, we reach the following expression:

$$\mathbb{P}(A|\{B_i\}) = \prod_i \left(\frac{\mathbb{P}(A|B_i)}{\mathbb{P}(A)}\right)\mathbb{P}(A). \tag{4.10}$$

This allow to express a random variable as a multiplication of conditionals, which will be useful to separate local and external measurements.

Using Bayes rule, we rewrite b_k^R as:

$$b_k^R = \eta \cdot \mathcal{L}_k^r \cdot b_k^{R-} \tag{4.11}$$

where $\eta \doteq \mathbb{P}(\mathcal{Z}_k^r|C_k^r, \mathcal{H}_k^R \backslash \mathcal{Z}_k^r)^{-1}$ is a normalization constant the does not participate in inference of the continuous belief. The local measurement likelihood, $\mathcal{L}_k^r$, is defined in Eq. (4.2).

The term $b_k^{R-} \doteq \mathbb{P}(\mathcal{X}_k^R|C_k^R, \mathcal{H}_k^R \backslash \mathcal{Z}_k^r)$ is the distributed propagated belief that is conditioned on information transmitted by other robots at time k, and on the latest action of robot r but not on its local measurement. During update, b_k^{R-} is saved to be used in maintenance of w_k^R, as seen in the next subsection. Using chain rule, we can extract the motion model of the latest action as well:

$$b_k^{R-} = \mathcal{M}_k^r \cdot \mathbb{P}(\mathcal{X}_k^R \backslash x_k^r | C_k^R, \mathcal{H}_k^{R-}). \tag{4.12}$$

We can express $\mathbb{P}(\mathcal{X}_k^R \backslash x_k^r | C_k^R, \mathcal{H}_k^{R-})$ in terms of the distributed continuous prior $b_{k-1}^R \doteq$

$\mathbb{P}(\mathcal{X}_{k-1}^R|C_{k-1}^R, \mathcal{H}_{k-1}^R)$, and the new information received from other robots:

$$\mathbb{P}(\mathcal{X}_k^R \backslash x_k^r | C_k^R, \mathcal{H}_k^{R-}) = b_{k-1}^R \cdot \frac{\mathbb{P}(\mathcal{X}_k^{o,R} | C_k^{o,R}, \Delta\mathcal{H}_k^{R-})}{\mathbb{P}(\mathcal{X}_{k-1}^{o,R})} \tag{4.13}$$

Recall splitting $\mathcal{H}_k^{R-}$ into prior history $\mathcal{H}_{k-1}^R$ and non-local measurements & actions:

$$\mathcal{H}_k^{R-} = \mathcal{H}_{k-1}^R \cup \Delta\mathcal{H}_k^{R-}. \tag{4.14}$$

We then use the above definition and relation (4.10) to split $\mathbb{P}(\mathcal{X}_k^R \backslash x_k^r | C_k^R, \mathcal{H}_k^{R-})$ into a product of two beliefs, one that depends on prior history, and one that depends on external new measurements:

$$\mathbb{P}(\mathcal{X}_k^R \backslash x_k^r | C_k^R, \mathcal{H}_k^{R-}) = \mathbb{P}(\mathcal{X}_k^R \backslash x_k^r | C_k^R) l \frac{\mathbb{P}(\mathcal{X}_k^R \backslash x_k^r | C_k^R, \Delta\mathcal{H}_k^{R-})}{\mathbb{P}(\mathcal{X}_k^R \backslash x_k^r | C_k^R)} \frac{\mathbb{P}(\mathcal{X}_k^R \backslash x_k^r | C_k^R, \mathcal{H}_{k-1}^R)}{\mathbb{P}(\mathcal{X}_k^R \backslash x_k^r | C_k^R)}. \tag{4.15}$$

This formulation allows us to isolate the new information sent by other robots at time k, from information already used for inference at previous times. Next, we have to address that not all known objects are present in the sent local beliefs. Because the priors are assumed independent between poses and classes, $\mathbb{P}(\mathcal{X}_k^R \backslash x_k^r | C_k^R) = \mathbb{P}(\mathcal{X}_k^R \backslash x_k^r)$. From $\mathbb{P}(\mathcal{X}_k^R \backslash x_k^r | C_k^R, \mathcal{H}_{k-1}^R)$ we can split $\mathcal{X}_k^R \backslash x_k^r$ into poses of objects that are involved in $\mathcal{H}_{k-1}^R$ and ones that do not as:

$$\mathbb{P}(\mathcal{X}_k^R \backslash x_k^r | C_k^R, \mathcal{H}_{k-1}^R) = \mathbb{P}(\mathcal{X}_{\text{new},k}^{o,R} | C_{\text{new},k}^R, \mathcal{H}_{k-1}^R) \mathbb{P}(\mathcal{X}_{k-1}^R | C_k^R, \mathcal{H}_{k-1}^R). \tag{4.16}$$

Poses of objects that r wasn't aware of at time $k-1$ are independent of $\mathcal{H}_{k-1}^R$, and without measurements, $\mathcal{X}_{\text{new},k}^{o,R}$ are independent of $C_{\text{new},k}^R$ as well. In addition, $\mathcal{X}_{k-1}^R$ is independent of classes of objects that are observed only at time k, thus we can write:

$$\mathbb{P}(\mathcal{X}_k^R \backslash x_k^r | C_k^R, \mathcal{H}_{k-1}^R) = \mathbb{P}(\mathcal{X}_{\text{new},k}^{o,R}) \cdot \mathbb{P}(\mathcal{X}_{k-1}^R | C_{k-1}^R, \mathcal{H}_{k-1}^R), \tag{4.17}$$

which is the prior for poses of newly known objects at time step k, multiplied by the conditional continuous belief for objects already known. Similarly to Eq. (4.17), the prior for $\mathcal{X}_k^R \backslash x_k^r$ is separated to previously known and new objects:

$$\mathbb{P}(\mathcal{X}_k^R \backslash x_k^r) = \mathbb{P}(\mathcal{X}_{\text{new},k}^{o,R}) \mathbb{P}(\mathcal{X}_{k-1}^R), \tag{4.18}$$

therefore we can write:

$$\frac{\mathbb{P}(\mathcal{X}_k^R \backslash x_k^r | C_k^R, \mathcal{H}_{k-1}^R)}{\mathbb{P}(\mathcal{X}_k^R \backslash x_k^r | C_k^R)} = \frac{b_{k-1}^R}{\mathbb{P}(\mathcal{X}_{k-1}^R)}, \tag{4.19}$$

and substitute it into the rightmost fracture in Eq. (4.15). Then we cancel out $\mathbb{P}(\mathcal{X}_k^R \backslash x_k^r | C_k^R)$

and proceed to remove r's poses from the prior by:

$$\frac{\mathbb{P}(\mathcal{X}_k^R \backslash x_k^r | C_k^R, \Delta\mathcal{H}_k^{R-})}{\mathbb{P}(\mathcal{X}_{k-1}^R)} = \frac{\mathbb{P}(\mathcal{X}_k^{o,R} | C_k^R, \Delta\mathcal{H}_k^{R-})}{\mathbb{P}(\mathcal{X}_{k-1}^{o,R})}, \tag{4.20}$$

as robot r's poses up until time $k-1$ are independent from the new external measurements. Finally, after factoring out $\mathbb{P}(\mathcal{X}_k^R \backslash x_k^r | C_k^R)$, and Eq. (4.19) and (4.20) with Eq. (4.15) we reach the following expression that is used in the chapter:

$$\mathbb{P}(\mathcal{X}_k^R \backslash x_k^r | C_k^R, \mathcal{H}_k^{R-}) = b_{k-1}^R \cdot \frac{\mathbb{P}(\mathcal{X}_k^{o,R} | C_k^{o,R}, \Delta\mathcal{H}_k^{R-})}{\mathbb{P}(\mathcal{X}_{k-1}^{o,R})} \tag{4.21}$$

Finally, we substitute Eq. (4.20) to Eq. (4.12) and in turn to Eq. (4.11), and get the following recursive formulation:

$$b_k^R \propto b_{k-1}^R \cdot \mathcal{L}_k^r \cdot \mathcal{M}_k^r \cdot \mathbb{P}(\mathcal{X}_{\text{new},k}^{o,R}) \frac{\mathbb{P}(\mathcal{X}_k^{o,R} | C_k^{o,R}, \Delta\mathcal{H}_k^{R-})}{\mathbb{P}(\mathcal{X}_k^{o,R})}, \tag{4.22}$$

where the measurement likelihood $\mathcal{L}_k^r$ accounts for the new local measurement, $\mathcal{M}_k^r$ accounts for the latest action of robot r, and $\mathbb{P}(\mathcal{X}_k^{o,R} | C_k^{o,R}, \Delta\mathcal{H}_k^{R-})$ (shown in blue) accounts for new information sent to r by other robots in R at time k. This pdf is only over object poses ($\mathcal{X}_k^{o,R}$), while the other robots' poses are marginalized out. Thus, robots communicate the environment states, which are implicitly affected by the robots' pose estimation. Computation of the blue part is further discussed in Sec. 2.C. Compared to the local belief update (4.5), the blue part is the main difference. The expression $\mathbb{P}(\mathcal{X}_{\text{new},k}^{o,R})$ represents pose prior of objects newly known by r at time k.

The distributed belief has at worst $M^{N_k(R)}$ continuous beliefs with corresponding weights, where the number of objects $N_k(R)$ known by r can increase with time. Naturally, a multi-robot system will observe more objects than a single robot, therefore the computational burden for distributed belief will be larger than for the local belief. Therefore, the significance of pruning beliefs with small weight grows. We set a threshold for the ratio between a weight and the largest weight in the distributed hybrid belief.

Maintaining w_k^R

To maintain w_k^R, we use a similar derivation to the weight update via local information only, presented in Sec. 2.A. We use Bayes rule to extract the last local measurement likelihood:

$$w_k^R = \eta \cdot w_k^{R-} \cdot \mathbb{P}(\mathcal{Z}_k^r | C_k^R, \mathcal{H}_k^R \backslash \mathcal{Z}_k^r), \tag{4.23}$$

where $w_k^{R-} \doteq \mathbb{P}(C_k^R | \mathcal{H}_k^R \backslash \mathcal{Z}_k^r)$ is the posterior distributed weight without accounting for the latest local measurements, and $\eta \doteq \mathbb{P}(\mathcal{Z}_k^r | \mathcal{H}_k^R \backslash \mathcal{Z}_k^r)^{-1}$ is a normalization constant that is identical in all realizations of C_k^R, thus does not participate in weight inference.

As we use a viewpoint dependent classifier model that utilizes the coupling between relative viewpoint and object class, we need to marginalize $\mathbb{P}(\mathcal{Z}_k^r|C_k^R, \mathcal{H}_k^R \backslash \mathcal{Z}_k^r)$ over the involved poses in this likelihood: the last robot pose x_k^r, and poses of objects observed at time k. We denote the latter by $\mathcal{X}_{\beta_k}^{o,r}$, and to shorten notations denote $\mathcal{X}_{\text{inv},k}^r \doteq \{x_k^r, \mathcal{X}_{\beta_k}^{r,k}\}$, and by $\neg\mathcal{X}_{\text{inv},k}^r$. Thus, $\mathbb{P}(\mathcal{Z}_k^r|C_k^R, \mathcal{H}_k^R \backslash \mathcal{Z}_k^r)$ is marginalized as

$$\mathbb{P}(\mathcal{Z}_k^r|C_k^R, \mathcal{H}_k^R \backslash \mathcal{Z}_k^r) = \int_{\mathcal{X}_{\text{inv},k}^r} \mathcal{L}_k^r \cdot \mathbb{P}(\mathcal{X}_{\text{inv},k}^r|C_k^r, \mathcal{H}_k^R \backslash \mathcal{Z}_k^r) d\mathcal{X}_{\text{inv},k}^r, \tag{4.24}$$

where $\mathbb{P}(\mathcal{X}_{\text{inv},k}^r|C_k^r, \mathcal{H}_k^R \backslash \mathcal{Z}_k^r)$ is computed by marginalizing b_k^{R-} over the uninvolved variables $\neg\mathcal{X}_{\text{inv},k}^r$, with $\mathcal{X}_k^R = \mathcal{X}_{\text{inv},k}^r \cup \neg\mathcal{X}_{\text{inv},k}^r$, as

$$\mathbb{P}(\mathcal{X}_{\text{inv},k}^r|C_k^r, \mathcal{H}_k^R \backslash \mathcal{Z}_k^r) = \int_{\neg\mathcal{X}_{\text{inv},k}^r} b_k^{R-} d\neg\mathcal{X}_{\text{inv},k}^r. \tag{4.25}$$

The propagated distributed belief b_k^{R-} is given to us from the continuous belief with Eq. (4.12), and includes the external information, shown in blue.

In practice, we sample the involved variables $\mathcal{X}_{\text{inv},k}^r$ in the current measurement likelihood and compute its value. As b_k^R and $\mathcal{L}_k^r$ are Gaussian, η does not play a role in the sampling process. Despite the classifier outputs being modeled as Gaussian, we integrate over poses; In the general case, expectation and covariance of the classifier model are a function of the relative viewpoint, thus we need to sample $\mathcal{X}_{\text{inv},k}^r$ as presented in Sec. 2.A at Eq. (4.6).

The other term we will address from Eq. (4.23) is w_k^{R-}. We express w_k^{R-} in terms of w_{k-1}^R:

$$w_k^{R-} \propto w_{k-1}^R \cdot \mathbb{P}(C_{k-1}^R)^{-1} \cdot \mathbb{P}(C_k^R|\Delta\mathcal{H}_k^R \backslash \mathcal{Z}_k^r). \tag{4.26}$$

Finally, we substitute Eq. (4.24) and (4.26) to Eq. (4.23) to reach our final recursive form for the discrete belief update:

$$\begin{aligned} w_k^R \propto w_{k-1}^R \cdot \mathbb{P}(C_{\text{new},k}^R) \frac{\mathbb{P}(C_k^R|\Delta\mathcal{H}_k^R \backslash \mathcal{Z}_k^r)}{\mathbb{P}(C_k^R)} \int_{\mathcal{X}_{\text{inv},k}^r} \mathcal{L}_k^r \cdot \\ \cdot \mathbb{P}(\mathcal{X}_{\text{inv},k}^r|C_k^r, \mathcal{H}_k^R \backslash \mathcal{Z}_k^r) d\mathcal{X}_{\text{inv},k}^r, \end{aligned} \tag{4.27}$$

with $\mathbb{P}(\mathcal{X}_{\text{inv},k}^r|C_k^r, \mathcal{H}_k^R \backslash \mathcal{Z}_k^r)$ computed via Eq. (4.25). This is a recursive formulation that includes the discrete prior w_{k-1}^R, external updates for the class probability from other robots (shown in red), and the external updates for the continuous belief contained within the integral.

Dependency Between Object Classes

One might be tempted to infer the class of each object separately, but it is not accurate due to the coupling between relative viewpoint and object class, as each object class is possibly implicitly dependent on all poses: robot and objects. We present a simple example where c_1 and c_2 be the underlying classes of objects 1 and 2 respectively. Let

$\mathcal{H}^R$ be the total measurement history, including semantic measurements z_1^{sem} and z_2^{sem} for objects 1 and 2 respectively. Recall that measurements are assumed independent from each other. Using the Bayes Law:

$$\mathbb{P}(c_1, c_2|\mathcal{H}^R) \propto \mathbb{P}(c_1, c_2|\mathcal{H}^R \backslash z_1^{sem}, z_2^{sem})\mathbb{P}(z_1^{sem}|c_1)\mathbb{P}(z_2^{sem}|c_2). \tag{4.28}$$

We use a viewpoint dependent classifier model, so we must marginalize $\mathbb{P}(z_1^{sem}|c_1)\mathbb{P}(z_2^{sem}|c_2)$ by the corresponding relative viewpoints, denoted x_1^{rel} and x_2^{rel} respectively:

$$\mathbb{P}(z_1^{sem}|c_1)\mathbb{P}(z_2^{sem}|c_2) = \int_{x_1^{rel}, x_2^{rel}} \mathbb{P}(z_1^{sem}|c_1, x_1^{rel})\mathbb{P}(z_2^{sem}|c_2, x_2^{rel})\mathbb{P}(x_1^{rel}, x_2^{rel}|\mathcal{H}^R)dx_1^{rel}dx_2^{rel}. \tag{4.29}$$

From the above equation, the condition for c_1 and c_2 to be independent is that x_1^{rel} and x_2^{rel} must be independent, which is not true in the general case, thus c_1 and c_2 are dependent.

2.C Communication Between Robots

In Sec. 2.B we presented a framework to maintain a hybrid belief of r given information obtained from other robots in R. That information was represented by the continuous blue expression in Eq. (4.22) and implicitly in Eq. (4.27), and the discrete red expression in Eq. (4.27). In this section, we present our approach for computing these parts, thus describing the management of this information and what each robot sends when communicating. We aim to achieve two goals:

1. Simple double counting prevention when maintaining the distributed belief without complex bookkeeping.

2. Distributed belief inference also via data not directly transmitted (e.g. robot r_1 sends data to r_2, r_2 to r_3, and r_3 is using data from r_1).

As will be shown next, the blue and red terms in Eqs. (4.22) and (4.27) can be expressed via local information transmitted by different robots in R to robot r. To that end, each robot r maintains and broadcasts a *stack* of local hybrid beliefs of other robots it is aware of. In contrast to (4.4), these local beliefs are marginal beliefs over object poses and classes, i.e. robot poses are marginalized out.

Each slot for robot r' in the stack of robot r contains $N_k(r')$ continuous and discrete marginal beliefs (defined below as $\xi_k^{r,r'}$ and $\phi_k^{r,r'}$), one pair per class realization, following a factorization similar to (4.4). Additionally, each slot includes a time-stamp that indicates on what data the local hybrid belief is conditioned upon. All in all, every stack contains $\sum_{i=1}^{|R|} N_k(r_i)$ continuous and discrete beliefs. Eq. (4.30) presents the stack of robot r as a set of slots, where each slot contains a hybrid belief of a particular

robot $r_i \in R$ over object poses and classes, normalized by their priors.

$$\mathcal{S}_k^r \doteq \left\{ \left(\frac{\mathbb{P}(\mathcal{X}_{k_i}^{o,r_i} | C_{k_i}^{r_i}, \mathcal{H}_{k_i}^{r_i}) \mathbb{P}(C_{k_i}^{r_i} | \mathcal{H}_{k_i}^{r_i})}{\mathbb{P}(\mathcal{X}_{k_i}^{o,r_i}) \mathbb{P}(C_{k_i}^{r_i})}, k_i \right) \right\}_{r_i \in R}, \tag{4.30}$$

where k_i is the time-stamp when robot r received information about r_i. In general, time k_i is not synchronized with k. The marginal continuous and discrete beliefs that robot r has about robot $r_i \in R$ are denoted $\xi_k^{r,r_i} \doteq \mathbb{P}(\mathcal{X}_{k_i}^{o,r_i} | C_{k_i}^{r_i}, \mathcal{H}_{k_i}^{r_i}) / \mathbb{P}(\mathcal{X}_{k_i}^{o,r_i})$ for the continuous part, and $\phi_k^{r,r_i} \doteq \mathbb{P}(C_{k_i}^{r_i} | \mathcal{H}_{k_i}^{r_i}) / \mathbb{P}(C_{k_i}^{r_i})$ for the discrete part.

With these definitions of ξ_k^{r,r_i} and ϕ_k^{r,r_i}, it is possible to show that the blue part in Eq. (4.22) can be expressed as:

$$\frac{\mathbb{P}(\mathcal{X}_k^{o,R} | C_k^R, \Delta\mathcal{H}_k^{R-})}{\mathbb{P}(\mathcal{X}_k^{o,R})} = \prod_{r_i \in R} \frac{\xi_k^{r,r_i}}{\xi_{k-1}^{r,r_i}} \tag{4.31}$$

Similarly, the red term in Eq. (4.27) can be expressed as:

$$\frac{\mathbb{P}(C_k^R | \Delta\mathcal{H}_k^R \backslash \mathcal{Z}_z^r)}{\mathbb{P}(C_k^R)} = \prod_{r_i \in R} \frac{\phi_k^{r,r_i}}{\phi_{k-1}^{r,r_i}}. \tag{4.32}$$

Eqs. (4.31) and (4.32) present the external update as a product of local beliefs, with only the updates from $k-1$ for robot r are present. The full derivation of these expression is shown in the next two subsections. This formulation avoids double counting by removing old information, ξ_{k-1}^{r,r_i} and ϕ_{k-1}^{r,r_i}, in each communication and uses measurements only once. Specifically for ξ_{k-1}^{r,r_i}, we use the approach presented in [63]. Doing so by maintaining stacks of individual information does not require complex bookkeeping, only time-stamps for each slot; Thus we fulfill the first goal. Robots can also relay information transmitted to them, thus the distributed belief can be updated by information from robots that did not transmit to the inferring robot, thus fulfilling the second goal.

Robot r_i sends the entire stack during information broadcast. When robot r receives information, it integrates the broadcast in as follows: recall that r_i's stack is divided to slots, with a time stamp per each slot. Robot r compares time stamps with the received information per slot, and replaces the information within the slot if the received time stamps is newer. If r receives information from more than one other robot at the same time, it will select the newest information per slot. This procedure is dependent on the relations between time-stamps, thus it is not necessary to synchronize time between the robots.

Derivation of $\frac{\mathbb{P}(\mathcal{X}_k^{o,R}|C_k^R,\Delta\mathcal{H}_k^{R-})}{\mathbb{P}(\mathcal{X}_k^{o,R})}$ (Continuous Belief Update)

Using the relation (4.10) we can split the blue part by separating the new measurements and actions per robot, excluding r itself as it is not present in $\Delta\mathcal{H}_k^{R-}$:

$$\mathbb{P}(\mathcal{X}_k^{o,R}|C_k^R,\Delta\mathcal{H}_k^{R-}) = \prod_{k',r'\in R\setminus r}\left(\frac{\mathbb{P}(\mathcal{X}_{k'}^{o,R}|C_{k'}^{r'},\Delta\mathcal{H}_{k'}^{r'})}{\mathbb{P}(\mathcal{X}_{k'}^{o,R})}\right)\mathbb{P}(\mathcal{X}_k^{o,R}). \tag{4.33}$$

From that, we will address every element in the product. Poses of objects that r' doesn't observe locally can be canceled out as follows, leaving only the object poses that r' observed:

$$\frac{\mathbb{P}(\mathcal{X}_{k'}^{o,R}|C_{k'}^{r'},\Delta\mathcal{H}_{k'}^{r'})}{\mathbb{P}(\mathcal{X}_{k'}^{o,R})} = \frac{\mathbb{P}(\mathcal{X}_{k'}^{o,r'}|C_{k'}^{r'},\Delta\mathcal{H}_{k'}^{r'})}{\mathbb{P}(\mathcal{X}_{k'}^{o,r'})}. \tag{4.34}$$

Then, using relation (4.10) again, we can expand $\mathbb{P}(\mathcal{X}_{k'}^{o,r'}|C_{k'}^{r'},\mathcal{H}_{k'}^{r'})$ to separate between known prior and new measurements:

$$\mathbb{P}(\mathcal{X}_{k'}^{o,r'}|C_{k'}^{r'},\mathcal{H}_{k'}^{r'}) = \mathbb{P}(\mathcal{X}_{k'}^{o,r'})\frac{\mathbb{P}(\mathcal{X}_{k'}^{o,r'}|C_{k'}^{r'},\mathcal{H}_{k'-l'}^{r'})}{\mathbb{P}(\mathcal{X}_{k'}^{o,r'})}\frac{\mathbb{P}(\mathcal{X}_{k'}^{o,r'}|C_{k'}^{r'},\Delta\mathcal{H}_{k'}^{r'})}{\mathbb{P}(\mathcal{X}_{k'}^{o,r'})}, \tag{4.35}$$

with l' being the time difference between subsequent slots of r (that can be 0 if the slot isn't updated). Then we take out all the poses that aren't dependent on the prior information, and we reach the definition of $\xi_{k-1}^{r,r'}$, i.e. the marginal object poses at the previous time.

$$\frac{\mathbb{P}(\mathcal{X}_{k'}^{o,r'}|C_{k'}^{r'},\mathcal{H}_{k'-l'}^{r'})}{\mathbb{P}(\mathcal{X}_{k'}^{o,r'})} = \frac{\mathbb{P}(\mathcal{X}_{k'-l'}^{o,r'}|C_{k'-l'}^{r'},\mathcal{H}_{k'-l'}^{r'})}{\mathbb{P}(\mathcal{X}_{k'-l'}^{o,r'})} \doteq \xi_{k-1}^{r,r'}. \tag{4.36}$$

With the definition of $\xi_k^{r,r'}$, and by substituting Eq. (4.36) into Eq. (4.35) we reach the expression for a single element of the product in Eq. (4.33):

$$\frac{\mathbb{P}(\mathcal{X}_{k'}^{o,r'}|C_{k'}^{r'},\Delta\mathcal{H}_{k'}^{r'})}{\mathbb{P}(\mathcal{X}_{k'}^{o,r'})} = \frac{\xi_k^{r,r'}}{\xi_{k-1}^{r,r'}}. \tag{4.37}$$

Finally, substituting Eq. (4.37) we reach the expression for the external continuous update belief:

$$\frac{\mathbb{P}(\mathcal{X}_k^{o,R}|C_k^R,\Delta\mathcal{H}_k^{R-})}{\mathbb{P}(\mathcal{X}_k^{o,R})} = \prod_{r'\in R}\frac{\xi_k^{r,r'}}{\xi_{k-1}^{r,r'}}. \tag{4.38}$$

Derivation of $\frac{\mathbb{P}(C_k^R|\Delta\mathcal{H}_k^{R-})}{\mathbb{P}(C_k^R)}$ (Discrete Belief Update)

The discrete belief update is similar to the continuous in its derivation, with probability over class realization, rather than object poses. Again, using the relation (4.10) we can split the red part by separating the new measurements and actions per robot, excluding

r itself as it is not present in $\Delta\mathcal{H}_k^{R-}$:

$$\mathbb{P}(C_k^R|\Delta\mathcal{H}_k^{R-}) = \prod_{k',r'\in R\setminus r}\left(\frac{\mathbb{P}(C_{k'}^R|\Delta\mathcal{H}_{k'}^{r'})}{\mathbb{P}(C_{k'}^R)}\right)\mathbb{P}(C_{k'}^R) \tag{4.39}$$

From that, we will address every element in the product. Classes of objects that r' doesn't observe locally can be canceled out as follows, leaving only the classes of object that r' observed:

$$\frac{\mathbb{P}(C_{k'}^R|\Delta\mathcal{H}_{k'}^{r'})}{\mathbb{P}(C_{k'}^R)} = \frac{\mathbb{P}(C_{k'}^{r'}|\Delta\mathcal{H}_{k'}^{r'})}{\mathbb{P}(C_{k'}^{r'})}. \tag{4.40}$$

Then, using relation (4.10) again, we can expand $\mathbb{P}(C_{k'}^{r'}|\mathcal{H}_{k'}^{r'})$ to separate between known prior and new measurements:

$$\mathbb{P}(C_{k'}^{r'}|\mathcal{H}_{k'}^{r'}) = \mathbb{P}(C_{k'}^{r'})\frac{\mathbb{P}(C_{k'}^{r'}|\mathcal{H}_{k'-l'}^{r'})}{\mathbb{P}(C_{k'}^{r'})}\frac{\mathbb{P}(C_{k'}^{r'}|\Delta\mathcal{H}_{k'}^{r'})}{\mathbb{P}(C_{k'}^{r'})}, \tag{4.41}$$

with l' being the time difference between subsequent slots of r (that can be 0 if the slot isn't updated). Then we take out all the classes of objects that not appear in prior information, and we reach the definition of $\phi_{k-1}^{r,r'}$, i.e. the marginal object poses at the previous time.

$$\frac{\mathbb{P}(C_{k'}^{r'}|\mathcal{H}_{k'-l'}^{r'})}{\mathbb{P}(C_{k'}^{r'})} = \frac{\mathbb{P}(C_{k'-l'}^{r'}|\mathcal{H}_{k'-l'}^{r'})}{\mathbb{P}(C_{k'-l'}^{r'})} \doteq \phi_{k-1}^{r,r'} \tag{4.42}$$

With the definition of $\phi_k^{r,r'}$, and by substituting Eq. (4.42) into Eq. (4.41) we reach the expression for a single element of the product in Eq. (4.39):

$$\frac{\mathbb{P}(C_{k'}^{r'}|\Delta\mathcal{H}_{k'}^{r'})}{\mathbb{P}(C_{k'}^{r'})} = \frac{\phi_k^{r,r'}}{\phi_{k-1}^{r,r'}} \tag{4.43}$$

Finally, substituting Eq. (4.43) we reach the expression for the external continuous update belief:

$$\frac{\mathbb{P}(C_k^R|\Delta\mathcal{H}_k^{R-})}{\mathbb{P}(C_k^R)} = \prod_{r'\in R}\frac{\phi_k^{r,r'}}{\phi_{k-1}^{r,r'}}. \tag{4.44}$$

In the following section we discuss double counting aspects of discrete random variables, corresponding to Eq. (4.44).

2.D Double Counting of Discrete Random Variables

Double counting leads to over-confident estimations, and if an erroneous measurement is counted multiple times, it may lead to a large error in the state's estimation in turn. While the implications of double counting on continuous random variables (e.g. camera poses and objects) have been investigated, it is not so for discrete random variables. Both cases have a common thread: measurements counted multiple times will 'push' the posterior estimation to a certain direction while leading to lower uncertainty than

when double counting is appropriately avoided (i.e. each measurement is used at most once). In the continuous Gaussian case, it manifests in a covariance matrix with smaller eigenvalues. Comparatively, in the discrete case the highest probability category will have its probability increase while the probability of not being in this category decreases.

To illustrate the above, consider an example with a categorical random variable c; we receive two sets of data $Z_a = \{z_1, z_2\}$, and $Z_b = \{z_2, z_3\}$, with a common measurement z_2. Considering a measurement likelihood $\mathbb{P}(z|c)$, the posterior over c is (see e.g. Bailey et al. [108]):

$$\mathbb{P}(c|Z_a, Z_b) \propto \mathbb{P}(c)\mathbb{P}(Z_a, Z_b|c) = \mathbb{P}(c)\frac{\mathbb{P}(z_1|c)\mathbb{P}(z_2|c)^2\mathbb{P}(z_3|c)}{\mathbb{P}(z_2|c)}. \tag{4.45}$$

If the common data (measurement z_2) is not removed via the denominator in Eq. (4.45), it will be double counted. Compared to Eq. (4.44), the above nominator and denominator correspond, respectively, to the terms ϕ_k^{r,r_i} and ϕ_{k-1}^{r,r_i}.

Denote $\mathbb{P}(z_2|c = i) \doteq a_i$, and to shorten the notations $\mathbb{P}(c = i)\mathbb{P}(z_1|c = i)\mathbb{P}(z_3|c = i) \doteq \mathcal{L}_i$. The normalized posterior can be written as:

$$\mathbb{P}(c = i|Z_a, Z_b) = \frac{a_i\mathcal{L}_i}{\sum_{j=1}^m a_j\mathcal{L}_j} = \frac{a_i^2\mathcal{L}_i}{\sum_{j=1}^m a_j\mathcal{L}_j \cdot a_i} \tag{4.46}$$

where m is the number of candidate categories. Double counting, i.e. without the denominator in Eq. (4.45), gives after normalization $\frac{a_i^2\mathcal{L}_i}{\sum_{j=1}^m a_j^2\mathcal{L}_j}$.

The largest a_i is denoted a_{max}, with i_{max} being the category corresponding to a_{max}, and subsequently the product of all other terms for i_{max} is denoted $\mathcal{L}_{max}$. Double counting of $\mathbb{P}(z_2|c_i)$ will increase the probability of i_{max}:

$$\mathbb{P}(c = i_{max}|Z_a, Z_b) = \frac{a_{max}^2\mathcal{L}_{max}}{\sum_{j=1}^m a_j\mathcal{L}_j \cdot a_{max}} \leq \frac{a_{max}^2\mathcal{L}_{max}}{\sum_{j=1}^m a_j^2\mathcal{L}_j}. \tag{4.47}$$

Similarly, it can be shown that with higher power (i.e. counting the data more) can increase the posterior probability even further; In addition, the reverse can be shown for the lowest probability in a. This increase in influence can be disastrous if the category of the highest probability likelihood is not correct, possibly leading to pruning of the correct class hypothesis when maintaining the hybrid belief (4.3).

A visualization can be seen in Fig. 4.1, where there are 4 categories with uninformed prior and a measurement likelihood; in Figs. 4.1a, 4.1b and 4.1c the likelihood is counted once, twice and thrice respectively. Evidently, the strongest category's probability (cat. 3) is increased when counted more times while all other have their probability diminish.

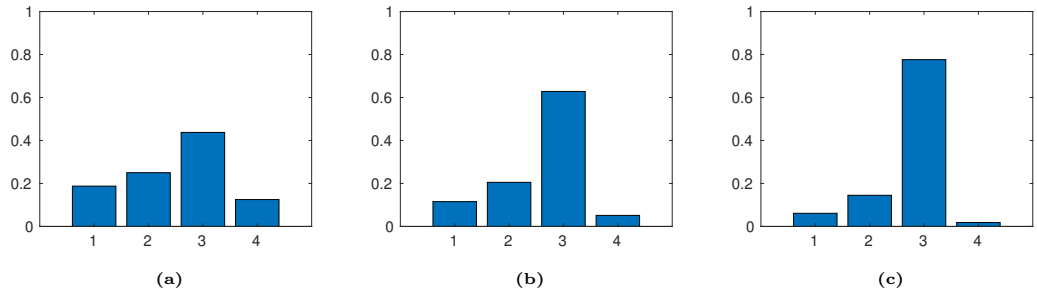


Figure 4.1: Conceptual demonstration of the effects of double counting on discrete random variables. Consider 4 possible categories with an uninformative prior over them. **(a)** is the measurement likelihood for the categories. Considering the uninformative prior, it is the posterior distribution as well. **(b)** and **(c)** counts the same likelihood twice and thrice respectively.

4.3 Experiments

We evaluated our approach in a multi-robot SLAM simulation and with real-world data where we consider an environment comprising several scattered objects observed by multiple mobile cameras from different viewpoints. Fig. 4.2a and Fig. 4.13a present the ground truth for simulation and experiment respectively. Our implementation uses the GTSAM library [107] with a python wrapper. The hardware used is an Intel i7-7700 processor running at 2.8GHz and 16GB RAM, with GeForce GTX 1050Ti with 4GB RAM.

3.A Simulation Setting, Compared Approaches and Metrics

Consider 3 robots, denoted r_1, r_2, and r_3, moving in a 2D environment represented by $N = 15$ scattered objects. We consider a closed-set setting and assume, for simplicity, $M = 2$ classes, where each object can be one of the two. In this scenario the maximum number of possible class realizations is $M^N = 32768$.

Our approach is evaluated for both classification, and pose inference accuracy, as we maintain a hybrid belief. We consider an ambiguous scenario where the classifier model cannot distinguish between the two classes from a certain viewpoint, thus requiring additional viewpoints to correctly disambiguate between the two classes. The robots communicate between themselves, increasing performance for discrete and continuous variables, i.e. classification and SLAM. Additionally, the distributed setting extends the sensing horizon, allowing robots to reason about objects that are not directly observed, while keeping estimation consistency.

Each robot only communicates with robots within a 10 meter communication range, relaying the local information stored in its stack. In particular, initially r_2 and r_3 share information with each other, then r_1 and r_2, relaying information from r_3 through r_2. For a complete table of communication in the considered scenario, see Appendix A.1. Further, we assume the robots share a common reference frame (this assumption can be relaxed as in [66]). We simulate relative pose odometry and geometric measurements, and we crafted a classifier model that simulates perceptual aliasing.

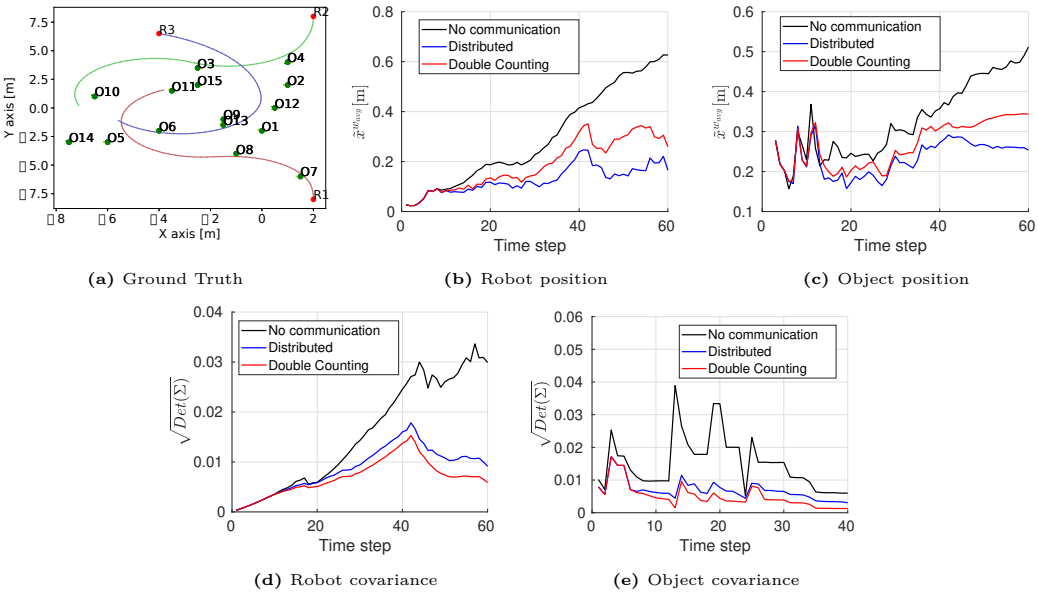
In the evaluation we compare between three approaches: local estimations, our approach, and our approach with double counting, i.e. $\xi_{k-1}^{r,r_i} = 1$ and $\phi_{k-1}^{r,r_i} = 1$ in Eq. (4.31) and (4.32) respectively. In all benchmarks we average the results for each robot.

We consider a motion model with noise covariance $\Sigma_w = diag(0.003, 0.003, 0.001)$, and geometric model with noise covariance $\Sigma_v^{geo} = diag(0.1, 0.1, 0.01)$, both corresponding to position coordinates in meters and orientation in radians.

Our semantic model parameters are defined as:

$$
\begin{aligned}
h_c(c = 1, \psi) &\doteq [0.25 \cdot \sin(\psi) + 0.75, 0.25(1 - \sin(\psi))]^T \\
h_c(c = 2, \psi) &\doteq [0.25(1 - \sin(\psi)), 0.25 \cdot \sin(\psi) + 0.75]^T ,
\end{aligned}
$$

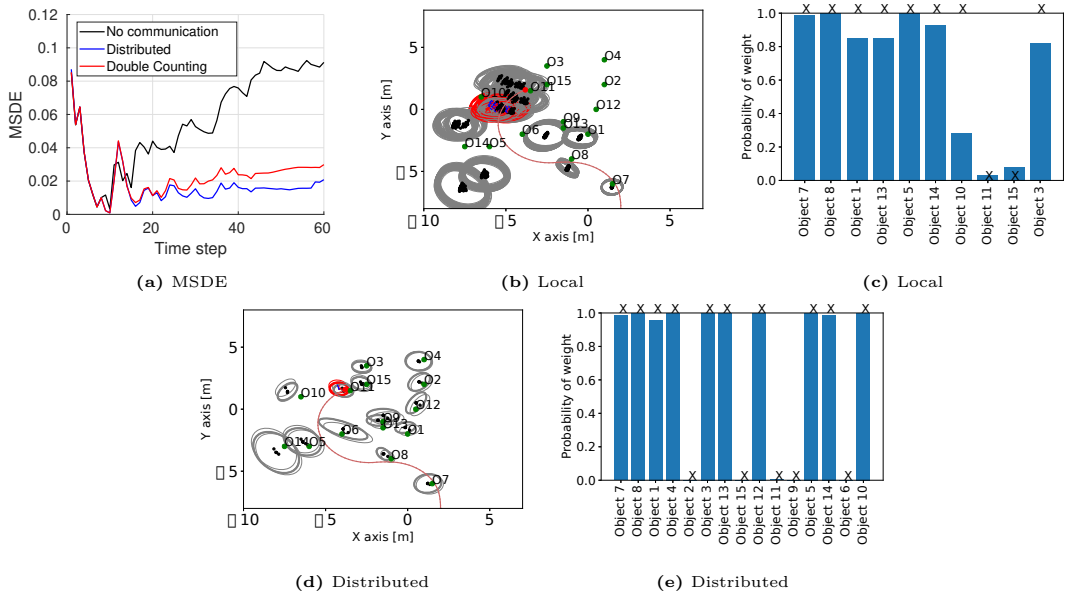
where $h_c(c = i, \psi) \in \mathbb{R}^M$ is the predicted probability vector given object class c is i. Recall that our semantic measurements $z_k^{sem,r}$ are probability vectors as well. ψ is the relative orientation between robot and object, computed from the relative pose $x_k^{rel} \doteq x^o \ominus x_k$. The measurement covariance is defined via the square root information matrix, such that $\Sigma_c \doteq (R^T R)^{-1}$, and $R = \begin{bmatrix} 1.5 & -0.75 \\ 0 & 1.5 \end{bmatrix}$. Both the geometric and semantic measurements are limited to 10 meters from the robot's pose. The highest probability for ambiguous class measurements is at $\psi = -90°$, where $h_c = [0.5, 0.5]^T$ for both classes.



(a) Ground Truth **(b)** Robot position **(c)** Object position

(d) Robot covariance **(e)** Object covariance

Figure 4.2: Simulation figures; **(a)** present the ground truth of the scenario. Red points represent the initial position of the robots, with different colored lines represent different robots. The green points represent the object poses. **(b)** and **(c)** represent the average $\tilde{x}^{w_{\text{avg}}}$ for robot and object position respectively as a function of time. **(d)** and **(e)** present the corresponding square-root of the position covariance for the robot and object average respectively.



(a) MSDE **(b)** Local **(c)** Local

(d) Distributed **(e)** Distributed

Figure 4.3: **(a)** presents average MSDE for the robots over 100 runs with different measurements. The rest are figures for time $k = 60$ of r_1. **(b)** and **(d)** represent multiple SLAM hypotheses for local and distributed setting respectively; Black dots with gray ellipse represent object pose estimation, red & blue signs with red ellipse represent robot pose estimation. Green and red points represent ground truth for object and robot positions respectively. **(c)** and **(e)** represent class probabilities for $c = 1$ for objects observed thus far for local and distributed respectively. The X notations represent ground truth (1 for class $c = 1$, 0 for class $c = 2$).

As explained in Sec. 2.D, when double counting occurs, the posterior class probability will converge to extreme results quicker, and may result on either completely right or wrong classifications. Therefore, reasoning about a single run is insufficient, and a statistical study is required. To quantify classification accuracy, we sample 100 times different geometric and semantic measurements, and perform a statistical study over the results. For that, we use mean square detection error (MSDE) averaged over all objects, robots, and runs (also used by Teacy et al. [15] and Feldman & Indelman [16]). We define MSDE per robot and object as follows:

$$MSDE \doteq \frac{1}{m}\sum_{i=1}^{m}(\mathbb{P}_{gt}(c=i) - \mathbb{P}(c=i|\mathcal{H}_k^R))^2, \tag{4.48}$$

where $\mathbb{P}_{gt}(c = i)$ represents the classification ground truth and can be either 1 for the correct class or 0 for all other classes. Therefore $MSDE = 1$ for completely incorrect classification, thus allowing us to perform statistical study of the effects of double counting of discrete random variables. To quantify localization accuracy, we use estimation error $\tilde{x}^{w_{avg}}$ which is the weighted average of Euclidean distance between the estimated and ground truth poses.

3.B Simulation Results

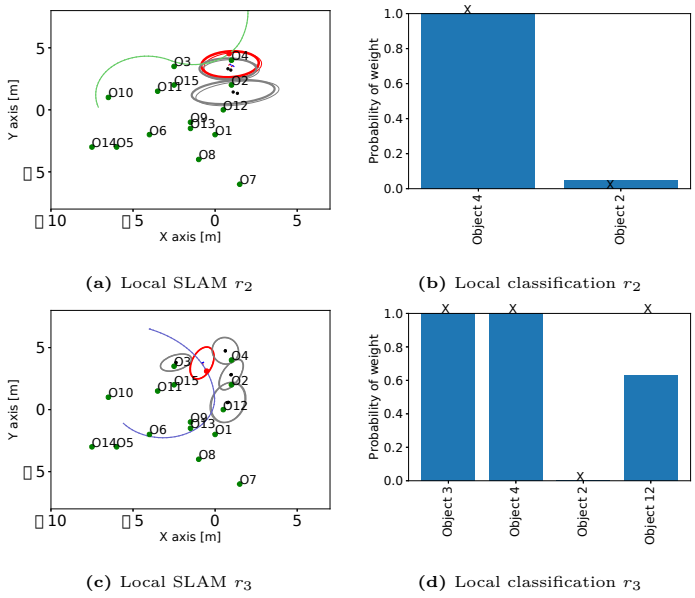
Fig. 4.2 presents results for continuous variables, i.e. robot and object poses. Figs. 4.2b and 4.2c show a clear advantage to our approach, where the localization error is the smallest for robots and objects respectively after the first 10 time steps. In Figs. 4.2d

and 4.2e the estimation covariance is presented, where the double counted approach has the smallest values as expected. Fig. 4.2e shows 'spikes' in the average objects' position covariance; these correspond to new object detections where the localization uncertainty is still high.

Fig. 4.3 visualizes classification and estimations at time $k = 60$ for local only and for distributed beliefs of robot r_2. At that time, robot r_2 communicated earlier with r_3, and for the first time communicates with r_1. When comparing Fig. 4.3b (local) to Fig. 4.3d (distributed), the number of possible class realizations is reduced. In addition, the estimate of r_2's pose, as well as the objects, is more certain and accurate. When comparing Figs. 4.3c and 4.3e, the latter presents a larger map, i.e. more objects observed, and the class estimations (classification) are closer to the ground truth.

Fig. 4.3a presents the average MSDE over 100 runs, where as a whole our approach shows lower MSDE values, i.e. statistically stronger classification results. In supplementary material [109, Sec. 8] we present additional classification and SLAM results.

In Fig. 4.4, 4.5, 4.6, and 4.7 we show the beliefs at various stages of the path.



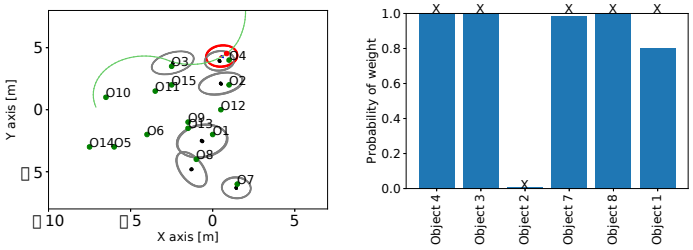
(a) Local SLAM r_2 **(b)** Local classification r_2

(c) Local SLAM r_3 **(d)** Local classification r_3

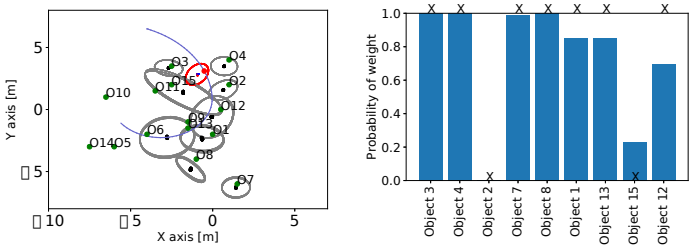
Figure 4.4: Figures for robot r_2 and r_3, local beliefs for time $k = 15$ and $k = 20$ respectively. **(a)** and **(b)** show results for r_2, **(c)** and **(d)** for r_3. **(a)** and **(c)** present SLAM results, **(b)** and **(d)** present classification results.

The results of all the graphs support the paper results, where both classification and SLAM in general are more accurate for the distributed belief. In addition, the robots inferring the distributed belief take into account objects that they didn't observe directly.

In Fig. 4.8 we show the time each inference time-step takes to compute for the distributed case, without and with double-counting. In general, computation time is influenced by the number of class realizations that aren't pruned, and is higher when robots communicate between each other. For each newly observed object the algorithm

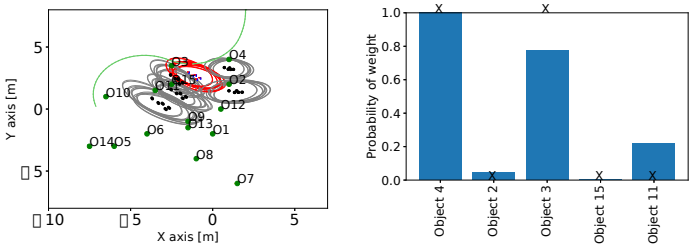


(a) Dis. SLAM r_2 **(b)** Dis. classification r_2

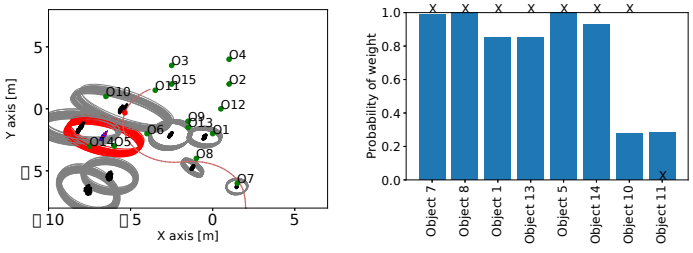


(c) Dis. SLAM r_3 **(d)** Dis. classification r_3

Figure 4.5: Figures for robot r_2 and r_3, distributed beliefs for time $k = 15$ and $k = 20$ respectively. **(a)** and **(b)** show results for r_2, **(c)** and **(d)** for r_3. **(a)** and **(c)** present SLAM results, **(b)** and **(d)** present classification results.

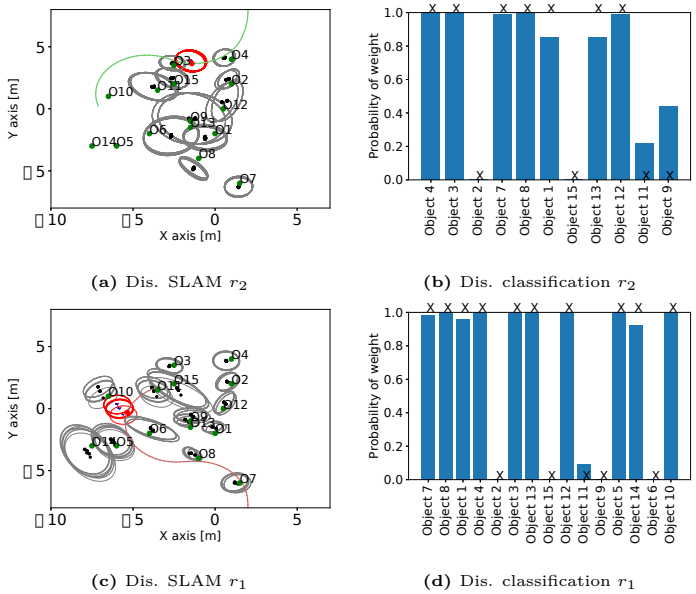


(a) Local SLAM r_2 **(b)** Local classification r_2



(c) Local SLAM r_1 **(d)** Local classification r_1

Figure 4.6: Figures for robot r_2 and r_1, local beliefs for time $k = 25$ and $k = 50$ respectively. **(a)** and **(b)** show results for r_2, **(c)** and **(d)** for r_1. **(a)** and **(c)** present SLAM results, **(b)** and **(d)** present classification results.



(a) Dis. SLAM r_2

(b) Dis. classification r_2

(c) Dis. SLAM r_1

(d) Dis. classification r_1

Figure 4.7: Figures for robot r_2 and r_1, distributed beliefs for time $k = 25$ and $k = 50$ respectively. **(a)** and **(b)** show results for r_2, **(c)** and **(d)** for r_1. **(a)** and **(c)** present SLAM results, **(b)** and **(d)** present classification results.

must consider all realizations for the said object, thus the computation time "spikes" at the first step the new object is observed.

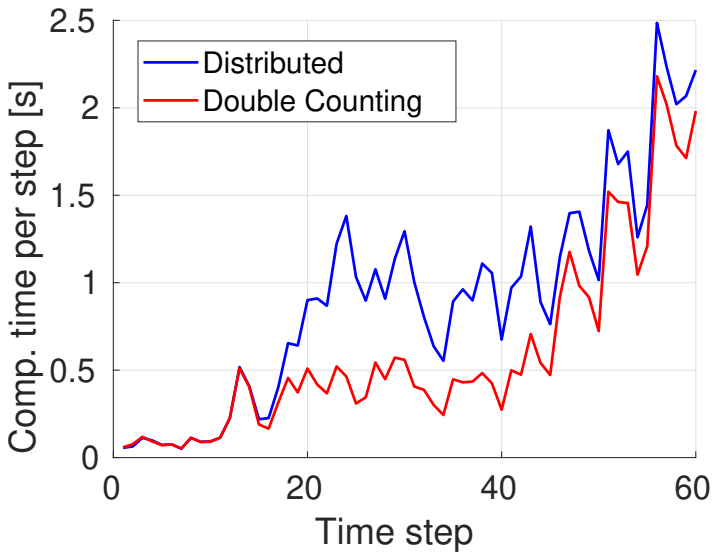


Figure 4.8: Calculation time as a function of the time step in seconds.

3.C Experiment Setting

In our scenario 3 robots are moving within an environment with multiple objects within it. We scattered 6 chairs within the environment and photographed them using a camera on a stand, keeping a constant height. The chairs were detected with YOLO3 DarkNet detector [110], which provided bounding boxes, and then each bounding box was classified using a ResNet50 convolutional neural network [4]. We considered 3 candidate classes out of 1000: 'barber chair', 'punching bag', and 'traffic light', as $c = 1, 2, 3$ respectively with $c = 1$ being the ground truth class. We trained three viewpoint-dependent classifier models using three sets of relative pose and class probability vector pairs, with the spatial parameters being the yaw and pitch angles from camera to object; For the ground truth class we photographed an objects from multiple viewpoints, and then classified it using ResNet 50. For the other two classifier models, we sampled class probability vectors with larger probability for the corresponding class of the model, and used the same relative poses as the first model.

In the experiment (deployment phase), we utilized both geometric and semantic measurements, using the corresponding (learned) measurement likelihood models. Relative pose geometric measurements for odometry and between camera and objects were generated by corrupting ground truth with Gaussian noise, while the semantic measurements are provided by YOLO3 and ResNet from real images. The same metrics as the simulation are used here.

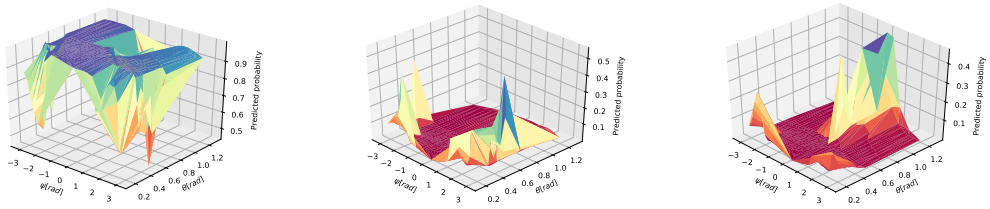
We consider a motion model with noise covariance $\Sigma_w = diag(0.0003, 0.0003, 0.0001)$, and geometric model with noise covariance $\Sigma_v^{geo} = diag(0.04, 0.04, 0.005)$, both corresponding to position coordinates in meters and orientation in radians. We simulated noisy odometry and geometric measurements, while using YOLO3 to create object proposals and a classifier to classify them. The communication radius in this scenario is 3 meters. The robot's and chair ground truth was measured via motion capture cameras with OptiTrack. The chairs' center of mass is used as a frame of reference for relative poses.

The classifier used in our experiment is the Pytorch implementation of ResNet 50, pre-trained on ImageNet dataset [111]. We trained three classifier models, one per each class. Class $c = 1$ is 'Barber Chair' and is considered our ground truth. Class $c = 2$ is 'Punching Bag' and class $c = 3$ is 'Traffic Light'. We trained the classifiers using pairs of relative pose and probability vectors; for $c = 1$, we used images of a chair used in the experiment, while for $c = 2$ and $c = 3$, we sampled measurements from Dirichlet Distribution with parameters $\alpha = [5, 15, 3]$ and $\alpha = [5, 3, 15]$ respectively. Each relative pose was parametrized by the relative yaw angle ψ, and the relative θ, with the camera being viewed from the object's frame of reference.

Fig. 4.9 presents 4 of the images used in the experiment, with bounding boxes for the chairs. Fig. 4.10, Fig 4.11, and 4.12 present the trained expected probability values for each relative ψ and θ values, i.e. $\mathbb{P}(z^{sem}|c = i, \psi, \theta)$ for each figure with different i.

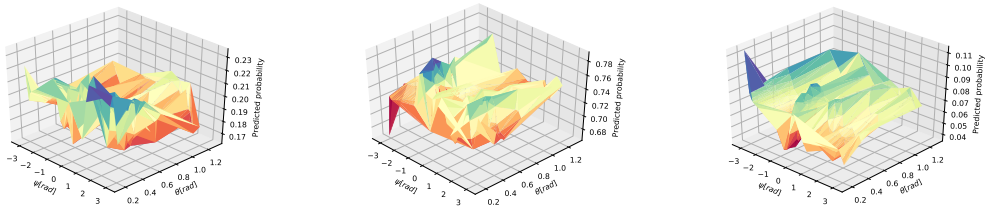
Each subfigure **(a)** to **(c)** representing measurement probability of class $c = 1$ to $c = 3$ respectively.

Figure 4.9: Four of the experiment images shown with corresponding bounding boxes.



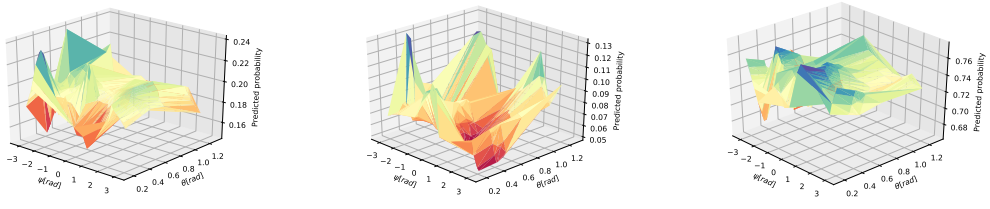
(a) $c = 1$ model, $c = 1$ prob. **(b)** $c = 1$ model, $c = 2$ prob. **(c)** $c = 1$ model, $c = 3$ prob.

Figure 4.10: Classifier model for $c = 1$, trained on real images: probabilities of classes 1 to 3 depending on relative yaw and pitch angles presented i **(a)** to **(c)** respectively. Higher surfaces go have bluer color.



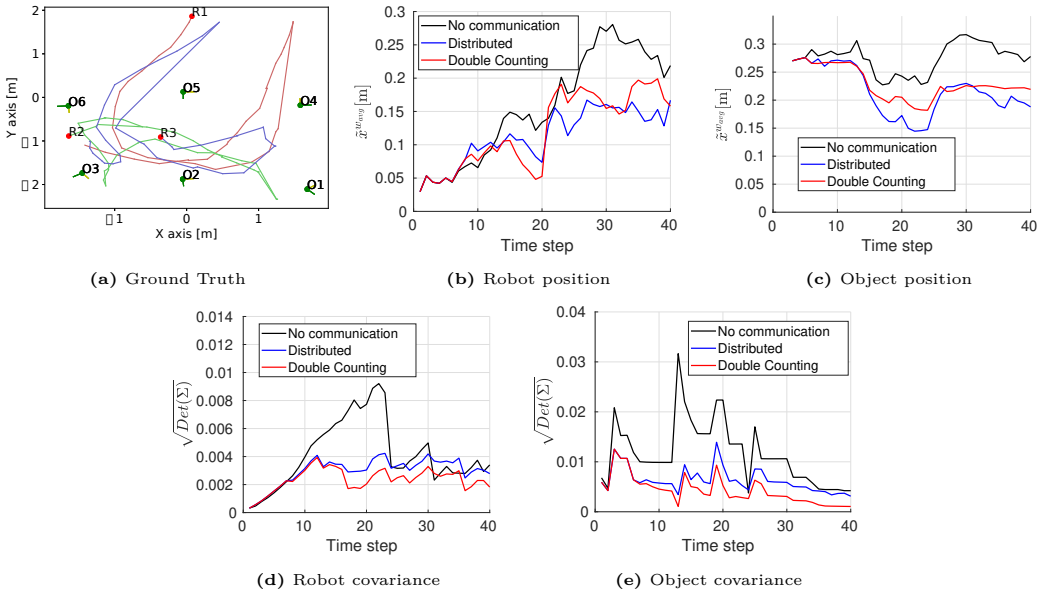
(a) $c = 2$ model, $c = 1$ prob. **(b)** $c = 2$ model, $c = 2$ prob. **(c)** $c = 2$ model, $c = 3$ prob.

Figure 4.11: Classifier model for $c = 2$, trained on real images: probabilities of classes 1 to 3 depending on relative yaw and pitch angles presented i **(a)** to **(c)** respectively. Higher surfaces go have bluer color.



(a) $c = 3$ model, $c = 1$ prob. **(b)** $c = 3$ model, $c = 2$ prob. **(c)** $c = 3$ model, $c = 3$ prob.

Figure 4.12: Classifier model for $c = 3$, trained on real images: probabilities of classes 1 to 3 depending on relative yaw and pitch angles presented i **(a)** to **(c)** respectively. Higher surfaces go have bluer color.



(a) Ground Truth **(b)** Robot position **(c)** Object position

(d) Robot covariance **(e)** Object covariance

Figure 4.13: Experiment figures; **(a)** present the ground truth of the scenario. Red points represent the initial position of the robots, with different colored lines represent different robots. The green points represent the object poses. **(b)** and **(c)** represent the average $\tilde{x}^{w_{\text{avg}}}$ for robot and object positions respectively as a function of time for the experiment. **(d)** and **(e)** present the corresponding square-root of the position covariance for the robot and object average respectively.

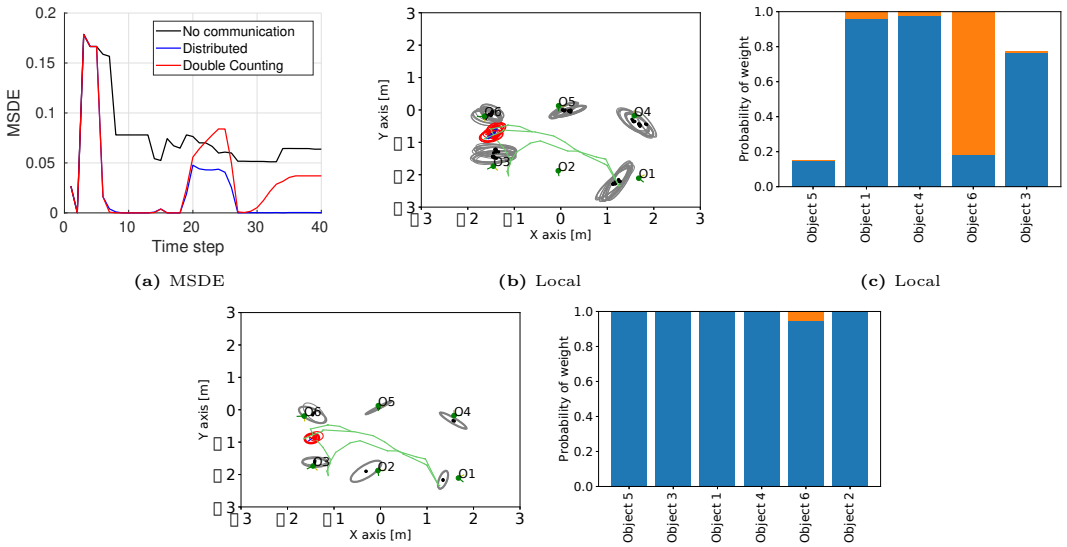
3.D Experimental Results

Fig. 4.13 presents SLAM results for the same benchmarks as in Fig. 4.2. Figs. 4.13b and 4.13c present an average $\tilde{x}^{w_{\text{avg}}}$ over all robots for robot and object positions, respectively. In general, the advantage of our approach is evident with lower errors. In addition, Figs. 4.13d and 4.13e present a similar pattern to Figs. 4.2d and 4.2e, respectively, where the covariance of our approach is smaller than the single robot case, but larger than the over-confident double counting case.

For classification results, Fig. 4.14a shows the average MSDE per robot as a function of time step, where eventually our approach out-performs both the single robot and the double counting cases, with higher probability for the correct class realization. In Fig. 4.14, SLAM and classification results for Robot 2 at time step $k = 35$ are presented, showing similar resulting trends to Fig. 4.3. Comparing Fig. 4.14b and Fig. 4.14d, the later shows more accurate SLAM compared to the former, with less class realizations. In addition, compared to Fig. 4.14e, Fig. 4.14c shows more accurate classification with an additional object classified.

The results of all the graphs support the paper results as well, where both classification and SLAM in general are more accurate for the distributed belief. In addition, the robots inferring the distributed belief take into account objects that they didn't observe directly.

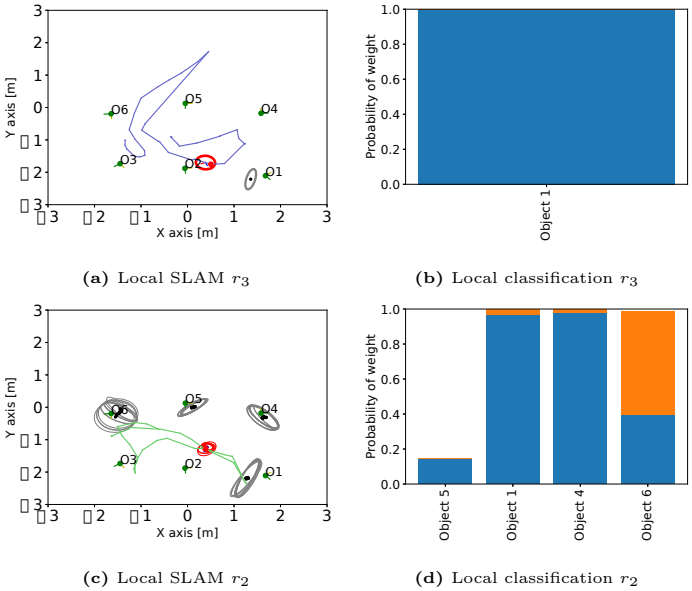
In Fig. 4.19 we show the time each inference time-step takes to compute for the distributed case, without and with double-counting. In general, computation time is influenced by the number of class realizations that aren't pruned, and is higher when



(a) MSDE **(b)** Local **(c)** Local

(d) Distributed **(e)** Distributed

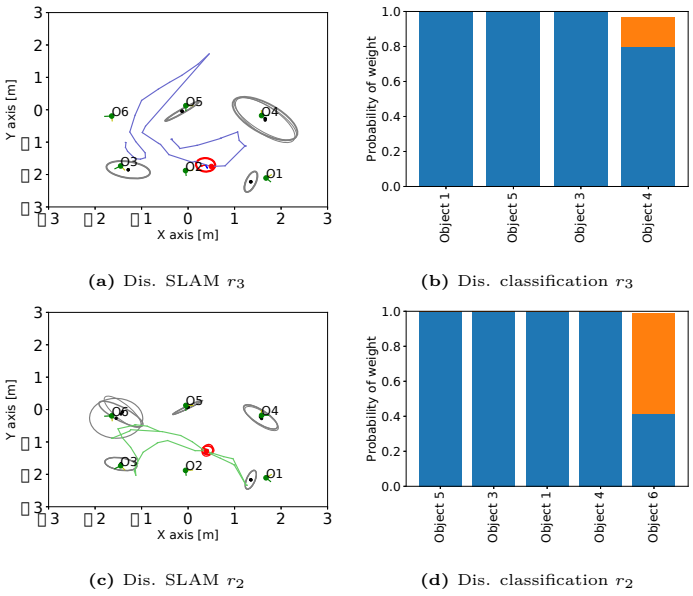
Figure 4.14: **(a)** presents average MSDE for the robots over 100 runs with different measurements. The rest are figures for time $k = 35$ of r_2. **(b)** and **(d)** represent multiple SLAM hypotheses for local and distributed setting respectively; Black dots with gray ellipse represent object pose estimation, red & blue signs with red ellipse represent robot pose estimation. Green and red points represent ground truth for object and robot poses respectively. **(c)** and **(e)** represent class probabilities for $c = 1$ and $c = 2$ for objects observed thus far for local and distributed respectively, with blue and orange for classes 1 and 2 respectively. In this case, the ground truth class of all objects is $c = 1$.



(a) Local SLAM r_3 **(b)** Local classification r_3

(c) Local SLAM r_2 **(d)** Local classification r_2

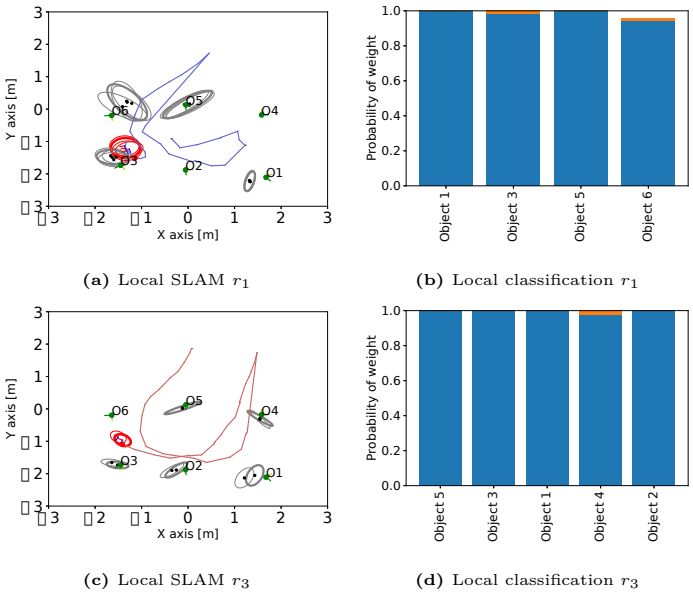
Figure 4.15: Figures for robot r_3 and r_2, local beliefs for time $k = 15$ and $k = 20$ respectively. **(a)** and **(b)** show results for r_3, **(c)** and **(d)** for r_2. **(a)** and **(c)** present SLAM results, **(b)** and **(d)** present classification results.



(a) Dis. SLAM r_3 **(b)** Dis. classification r_3

(c) Dis. SLAM r_2 **(d)** Dis. classification r_2

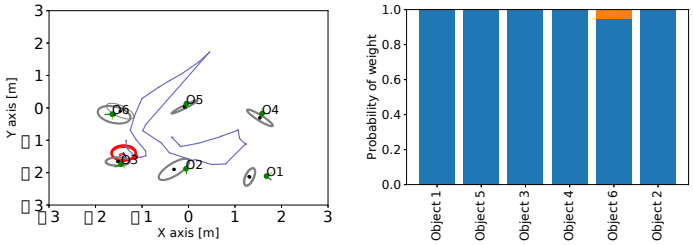
Figure 4.16: Figures for robot r_2 and r_1, distributed beliefs for time $k = 15$ and $k = 20$ respectively. **(a)** and **(b)** show results for r_3, **(c)** and **(d)** for r_2. **(a)** and **(c)** present SLAM results, **(b)** and **(d)** present classification results.



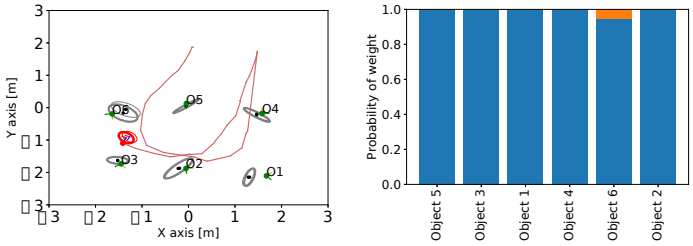
(a) Local SLAM r_1 **(b)** Local classification r_1

(c) Local SLAM r_3 **(d)** Local classification r_3

Figure 4.17: Figures for robot r_3 and r_1, local beliefs for time $k = 35$ and $k = 40$ respectively. **(a)** and **(b)** show results for r_3, **(c)** and **(d)** for r_1. **(a)** and **(c)** present SLAM results, **(b)** and **(d)** present classification results.



(a) Dis. SLAM r_3 **(b)** Dis. classification r_3



(c) Dis. SLAM r_1 **(d)** Dis. classification r_1

Figure 4.18: Figures for robot r_3 and r_1, distributed beliefs for time $k = 35$ and $k = 40$ respectively. **(a)** and **(b)** show results for r_3, **(c)** and **(d)** for r_1. **(a)** and **(c)** present SLAM results, **(b)** and **(d)** present classification results.

robots communicate between each other. For each newly observed object the algorithm must consider all realizations for the said object, thus the computation time ”spikes” at the first step the new object is observed. Because the classifier model in the experiment uses deep neural networks, the computation is slower than in the simulation where hand crafted models were used.

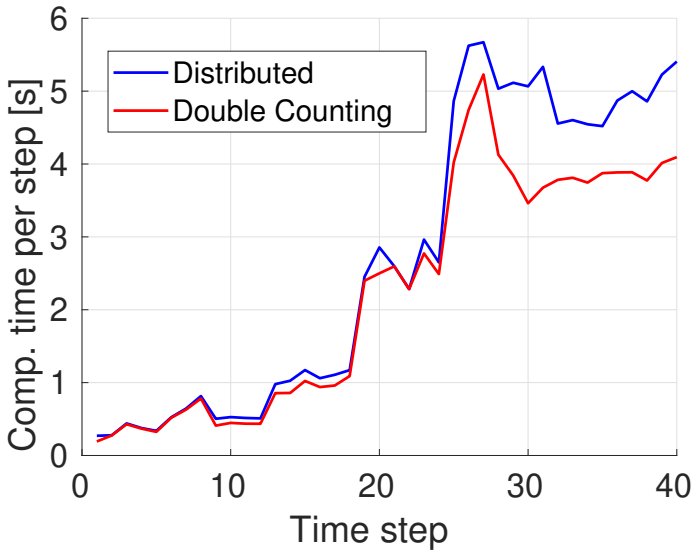


Figure 4.19: Calculation time as a function of the time step in seconds.

Chapter 5

Model Uncertainty Aware Sequential Inference of Posterior Class Probability

In this chapter we propose to maintain a *distribution over posterior class probabilities* while accounting for model uncertainty. This distribution enables reasoning about uncertainty in posterior classification, which is crucial for robust classification, and for safe autonomy in general. In particular, we derive equations to sequentially update the distribution over posterior class probabilities. We evaluate our approach in simulation, and using real images fed into a deep learning classifier.

5.1 Notations and Problem Formulation

Consider a robot observing a single object from multiple viewpoints, aiming to infer its class while quantifying uncertainty in the latter. Each class probability vector is $\gamma_k \doteq \begin{bmatrix} \gamma_k^1 & \cdots & \gamma_k^i & \cdots & \gamma_k^M \end{bmatrix}$, where M is the number of candidate classes. Each element γ_k^i is the probability of object class c being i given image z_k, i.e. $\gamma_k^i \equiv \mathbb{P}(c = i|z_k)$, while γ_k resides in the $(M-1)$ simplex such that

$$\gamma_k^i \geqslant 0 \qquad ||\gamma_k||_1 = 1. \tag{5.1}$$

Existing Bayesian sequential classification approaches do not consider model uncertainty, and thus maintain a posterior distribution λ_k for time k over c,

$$\lambda_k \doteq \mathbb{P}(c|\gamma_{1:k}), \tag{5.2}$$

given history $\gamma_{1:k}$ obtained from images $z_{1:k}$. In other words, λ_k is inferred from a single sequence of $\gamma_{1:k}$, where each γ_t for $t \in [1, k]$ corresponds to an input image z_t. However, the posterior class probability λ_k by itself does not provide any information

regarding how reliable the classification result is due to model uncertainty. For example, a classifier output γ_k may have a high score for a certain class, but if the input is far from the classifier training set the result is not reliable and may vary greatly with small changes in the scenario and classifier weights.

In this chapter we wish to reason about model uncertainty, i.e. quantify how "far" an image input z_t is from the training set D by modeling the distribution $\mathbb{P}(\gamma_t|z_t, D)$. Given a training set D and classifier weights w, the output γ_t is a deterministic function of input z_t for all $t \in [1, k]$:

$$\gamma_t = f_w(z_t), \tag{5.3}$$

where the function f_w is a classifier with weights w. However, w are stochastic given D, thus inducing a probability $\mathbb{P}(w|D)$ and making γ_t a random variable. Gal and Ghahramani [21] showed that an input far from the training set will produce vastly different classifier outputs for small changes in weights. Unfortunately, $\mathbb{P}(w|D)$ is not given explicitly. To combat this issue, Gal and Ghahramani [21] proposed to approximate $\mathbb{P}(w|D)$ via dropout, i.e. sampling w from another distribution closest to $\mathbb{P}(w|D)$ in a sense of KL divergence. Practically, we run an input image z_t through a classifier with dropout multiple times to get many different γ_t's for corresponding w realizations, creating a point cloud of class probability vectors. Note that every distribution in this chapter is dependent on the training set D, so we omit it from further expressions to avoid clutter.

In this chapter, a class-dependent likelihood $\mathbb{P}(\gamma_k|c = i)$, referred as a classifier model, is utilized. We use a Dirichlet distributed classifier model with a different hyperparameter vector $\theta_i \in \mathbb{R}^{M\times 1}$ per class $i \in [1, M]$, rewriting $\mathbb{P}(\gamma_k|c = i)$ as:

$$\mathbb{P}(\gamma_k|c = i) = Dir(\gamma_k; \theta_i). \tag{5.4}$$

This distribution is the conjugate prior of the categorical distribution, thus it supports class probability vectors, particularly γ_k. Sampling from Dirichlet distribution necessarily satisfies conditions (5.1), unlike other distributions such as Gaussian. The probability density function (PDF) of the above distribution is as follows:

$$Dir(\gamma_k; \theta_i) = C(\theta_i) \prod_{j=1}^{M} \left(\gamma_k^j\right)^{\theta_i^j - 1}, \tag{5.5}$$

where $C(\theta_i)$ is a normalizing constant dependent on θ_i, and θ_i^j is the j-th element of vector θ_i. To shorten notation, we will write this likelihood as:

$$\mathbb{P}(\gamma_k|c = i) \doteq \mathcal{L}_i(\gamma_k), \qquad \mathbb{P}(\cdot|c = i) \doteq \mathcal{L}_i. \tag{5.6}$$

We denote the likelihood vector as $\mathcal{L}(\gamma_k) \doteq \begin{bmatrix} \mathcal{L}_1(\gamma_k) \cdots \mathcal{L}_M(\gamma_k) \end{bmatrix}$. For simplicity, we consider these hyperparameter vectors to be known or inferred. Furthermore, in this

chapter we assume an uninformative prior $\mathbb{P}(c = i) = 1/M$.

We must distinguish between the classifier model $\mathcal{L}_i(\gamma_k)$, and the model uncertainty derived from $\mathbb{P}(\gamma_k|z_k)$ for class i and time step k. The classifier model $\mathcal{L}_i(\gamma_k)$ is the likelihood of a single γ_k given a class hypothesis i; it is computed *prior to the scenario* for each class from the training set, and it is assumed constant within the scenario. On the other hand, $\mathbb{P}(\gamma_k|z_k)$ is the probability of γ_k given an image z_k, and is computed *during the scenario*. Note that if the true object class is i and it is "close" to the training set, the probabilities $\mathbb{P}(\gamma_k|z_k)$ and $\mathcal{L}_i(\gamma_k)$ will be "close" to each other as well.

A *key observation* is that λ_k is a random variable, as it depends on $\gamma_{1:k}$ (see Eq. (5.2)) while each γ_t, with $t \in [1, k]$, is a random variable distributed according to $\mathbb{P}(\gamma_t|z_t, D)$. Thus, rather than maintaining the posterior Eq. (5.2), our goal is to maintain a *distribution over posterior class probabilities* for time k, i.e.

$$\mathbb{P}(\lambda_k|z_{1:k}). \tag{5.7}$$

This distribution allows to calculate the posterior class distribution, $\mathbb{P}(c|z_{1:k})$, via expectation

$$\begin{aligned} \mathbb{P}(c = i|z_{1:k}) &= \textstyle\int_{\lambda_k^i} \mathbb{P}(c = i|\lambda_k^i, z_{1:k})\mathbb{P}(\lambda_k^i|z_{1:k})d\lambda_k^i \\ &= \textstyle\int_{\lambda_k^i} \mathbb{P}(c = i|\lambda_k^i)\mathbb{P}(\lambda_k^i|z_{1:k})d\lambda_k^i = \mathbb{E}[\lambda_k^i], \end{aligned} \tag{5.8}$$

where we utilized the identity $\mathbb{P}(c = i|\lambda_k^i) = \lambda_k^i$.

Moreover, as will be seen, Eq. (5.7) allows to quantify the *posterior uncertainty*, thereby providing a measure of confidence in the classification result given all data thus far.

At this point, it is useful to summarize our assumptions:

1. A single object is observed multiple times.
2. $\mathbb{P}(\gamma_t|z_t, D)$ is approximated by a point cloud $\{\gamma_t\}$ for each image z_t.
3. An uninformative prior for $\mathbb{P}(c = i)$.
4. A Dirichlet distributed classifier model with with designated parameters for each class $c \in [1, \ldots, M]$. These parameters are constant and given (e.g. learned).

5.2 Approach

We aim to find a distribution over the posterior class probability vector λ_k for time k, i.e. $\mathbb{P}(\lambda_k|z_{1:k})$. First, λ_k is expressed given some specific sequence $\gamma_{1:k}$. Using Bayes' law:

$$\lambda_k^i = \mathbb{P}(c = i|\gamma_{1:k}) \propto \mathbb{P}(c = i|\gamma_{1:k-1})\mathbb{P}(\gamma_k|c = i, \gamma_{1:k-1}). \tag{5.9}$$

Algorithm 5.1 $\mathbb{P}(\lambda_k|z_{1:k})$ inference algorithm with sub-sampling.

Input: $z_{1:k}$: k images of an object, $\mathbb{P}(c=i)\ \forall\ i=1,...,M$: a prior for object class, $\mathcal{L}_i\ \forall\ i=1,...,M$: a classifier model, $N_{ss,n}$ maximum points per time step.

$\lambda_0^i = \mathbb{P}(c=i)$
for $t=1:k$ **do**
  Classify image z_t, and produce a point cloud $\{\gamma_t\}$.
  for All possible γ_t and λ_{t-1} pairs: **do**
    for $i=1:M$ **do**
      $\lambda_t^i \propto \mathcal{L}_i(\gamma_t)\lambda_{t-1}^i$.
    end for
  end for
  Select randomly $N_{ss,n}$ pairs to form $\{\lambda_t\}$
end for
return $\{\lambda_k\}$

We assume, for simplicity, classifier outputs are statistically independent and re-write Eq. (5.9) as

$$\lambda_k^i \propto \mathbb{P}(c=i|\gamma_{1:k-1})\mathbb{P}(\gamma_k|c=i). \tag{5.10}$$

Per the definition for λ_{k-1} (Eq. (5.2)) and $\mathbb{P}(\gamma_k|c=i)$ (Eq. (5.6)), λ_k^i assumes the following recursive form:

$$\lambda_k^i \propto \lambda_{k-1}^i \mathcal{L}_i(\gamma_k). \tag{5.11}$$

We now recall that γ_t (for each time step $t \in [1,k]$) is a random variable, making also λ_{k-1}^i and λ_k^i random variables. Thus, our problem is to infer $\mathbb{P}(\lambda_k|z_{1:k})$, where, according to Eq. (5.11), for each realization of the sequence $\gamma_{1:k}$, λ_k is a function of λ_{k-1} and γ_k.

We present our approach in Algorithm 5.1. At each time step t, a new image z_t is classified using multiple forward passes through a CNN with dropout, yielding a point cloud $\{\gamma_t\}$. Each forward pass gives a probability vector $\gamma_t \in \{\gamma_t\}$, which is used to compute the class likelihood $\mathcal{L}(\gamma_t)$, that is modeled as a Dirichlet distribution. In addition, we have a point cloud $\{\lambda_{t-1}\}$ from the previous step. We multiply all possible pairs of λ_{t-1}^i and $\mathcal{L}_i(\gamma_t)$, as in Eq. (5.11). Finally $N_{ss,n}$ pairs are chosen for the next step, in a sub-sampling algorithm that will be detailed in Section 2.B. We eventually get a point cloud $\{\lambda_t\}$ that approximates $\mathbb{P}(\lambda_t|z_{1:t})$.

We need to initialize the algorithm for the first image. Recalling Eq. (5.2), we define λ_1^i (first image) for class i and time $k=1$ as:

$$\lambda_1^i \doteq \mathbb{P}(c=i|\gamma_1). \tag{5.12}$$

Using Bayes law:

$$\mathbb{P}(c=i|\gamma_1) = \frac{\mathbb{P}(\gamma_1|c=i)\mathbb{P}(c=i)}{\mathbb{P}(\gamma_1)} \tag{5.13}$$

where $\mathbb{P}(c=i)$ is a prior probability of class i, $\mathbb{P}(\gamma_1)$ serves as a normalizing term, and

$\mathbb{P}(\gamma_1|c=i)$ is the classifier model for class i. Per definition Eq. (5.6), Eq. (5.13) can be written as:

$$\lambda_1^i \propto \mathbb{P}(c=i)\mathcal{L}_i(\gamma_1), \tag{5.14}$$

thus λ_1^i is a function of prior $\mathbb{P}(c=i)$ and γ_1, and in the subsequent steps we can use the update rule of Eq. (5.11) to infer $\mathbb{P}(\lambda_k|z_{1:k})$.

Remark: There is a numerical issue where λ_k^i for sufficiently large k can practically become 0 or 1, preventing any possible change for future time steps. In our implementation, we overcome this by calculating $\log \lambda_k^i$ instead of λ_k^i.

In the next section we discuss the properties of $\mathbb{P}(\lambda_k|z_{1:k})$), analyze the corresponding posterior uncertainty versus time, and consider two inference approaches that approximate this PDF.

2.A Inference over the Posterior $\mathbb{P}(\lambda_k|z_{1:k})$

In this section we consider how the distribution $\mathbb{P}(\lambda_k|z_{1:k})$ develops and seek to find an inference method to track this distribution over time. As discussed in Section 5.1, we consider all γ_t as random variables; hence, according to Eq. (5.11), $\mathbb{P}(\lambda_k|z_{1:k})$ accumulates all model uncertainty data from all $\mathbb{P}(\gamma_t|z_t)$ up until time step k, with $t \in [1,k]$.

Fig. 5.1 illustrates an example for inference of $\mathbb{P}(\lambda_k|z_{1:k})$ from $\mathbb{P}(\gamma_k|z_k)$ and $\mathbb{P}(\lambda_{k-1}|z_{1:k})$ using a known classifier model, considering three possible classes. Fig. 5.1a-5.1c present example distributions for the classifier model. Fig. 5.1d presents a point cloud that describes the distribution of λ_{k-1}. Fig. 5.1e presents $\mathbb{P}(\gamma_k|z_k)$ represented by a point cloud of γ_k instances. Each γ_k is projected via $\mathcal{L}(\gamma_k)$ to a different cloud in the simplex, presented in Fig. 5.1f. Finally, based on Eq. (5.11), the multiplication of points from Fig. 5.1d and 5.1f creates a $\{\lambda_k\}$ point cloud, shown in Fig. 5.1g. In the presented scenario, the spread of $\{\lambda_k\}$ (Fig. 5.1g) point cloud was smaller than $\{\lambda_{k-1}\}$ (Fig. 5.1d), because both point clouds $\{\lambda_{k-1}\}$ and $\{\mathcal{L}(\gamma_k)\}$ are near the same simplex edge. In general, classifier models with large parameters (see Eq. 5.5) create $\{\mathcal{L}(\gamma_t)\}$ point clouds that are closer to the simplex edge. In turn, the $\{\lambda_k\}$ point cloud (updated via Eq. (5.11)) will converge faster to a single simplex edge.

In this paragraph we discuss the behavior of $\mathbb{P}(\lambda_k)$, dependent on both λ_{k-1} and γ_k. The spread of $\{\lambda_k\}$ is indicative of accumulated model uncertainty, and is dependent on the expectation and spread of both $\{\lambda_{k-1}\}$ and $\{\gamma_k\}$. For specific realizations of λ_{k-1} and γ_k, as seen in Eq. (5.11), λ_k^i is a multiplication of λ_{k-1}^i and $\mathcal{L}_i(\gamma_k)$. Therefore, when $\mathcal{L}(\gamma_k)$ is within the simplex center, i,e. $\mathcal{L}_i(\gamma_k) = \mathcal{L}_j(\gamma_k)$ for all $i,j = 1,...,M$, the resulting λ_k will be equal to λ_{k-1}. On the other hand, when $\mathcal{L}(\gamma_k)$ is at one of the simplex' edges, its effect on λ_k will be the greatest. Expanding to the probability $\mathbb{P}(\lambda_k|z_{1:k})$, there are several cases to consider. If $\mathbb{P}(\lambda_{k-1}|z_{1:k-1})$ and $\{\mathcal{L}(\gamma_k)\}$ "agree" with each other, i.e. the highest probability class is the same, and both are far enough from the simplex center, the resulting $\mathbb{P}(\lambda_k|z_{1:k})$ will have a smaller spread compared to

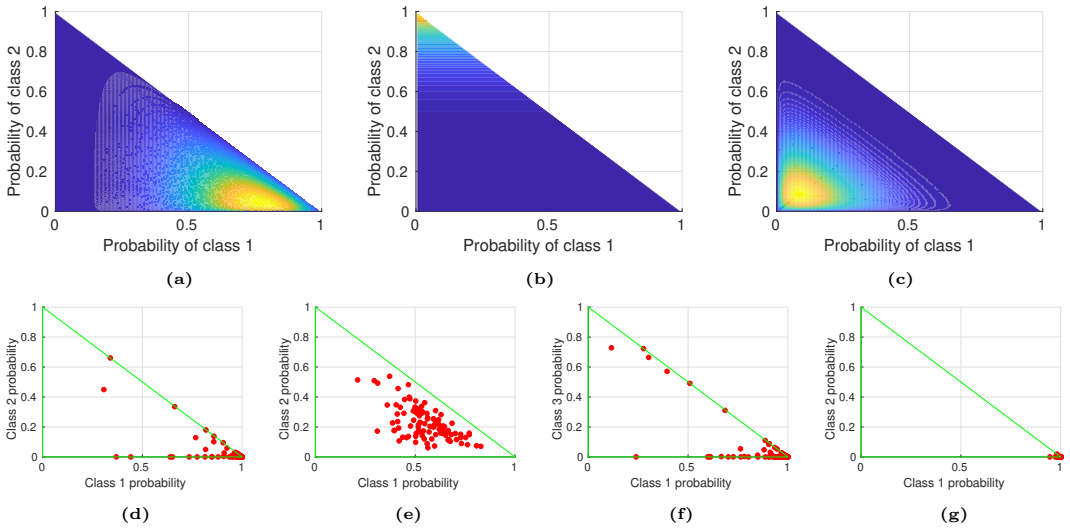


Figure 5.1: An example to illustrate the inference process of $\mathbb{P}(\lambda_k|z_{1:k})$. **(a)**, **(b)**, and **(c)** $\mathcal{L}_i$ classifier model for classes $1, 2$ and 3, respectively, with higher probability zones presented in yellow. **(d)** distribution of λ_{k-1} from the previous step. Note that for $k = 1$, λ_0 is given by the prior $\mathbb{P}(c)$. **(e)** a point cloud $\{\gamma_k\}$ approximating $\mathbb{P}(\gamma_k|z_k)$ via multiple forward passes of the (CNN) classifier with dropout, given a new measurement z_k (an image) at current time step k. **(f)** The corresponding likelihood $\mathcal{L}(\gamma_k)$ for each $\gamma_k \in \{\gamma_k\}$ from **(e)**. Finally, multiplying λ_{k-1} and $\mathcal{L}(\gamma_k)$ (Eq. (5.11)) results in the point cloud shown in **(g)** representing a distribution over λ_k. λ_k's spread is smaller in this case than λ_{k-1}'s, as both $\mathcal{L}(\gamma_k)$ and $\mathbb{P}(\lambda_{k-1}|z_{k-1})$ are close to the same simplex corner.

$\mathbb{P}(\lambda_{k-1}|z_{1:k-1})$ and its expectation will have the dominant class with a high probability. On the other hand, if $\mathbb{P}(\lambda_{k-1}|z_{1:k-1})$ and $\{\mathcal{L}(\gamma_k)\}$ "disagree" with each other, i.e. they are close to the same simplex corner, the spread of $\mathbb{P}(\lambda_k|z_{1:k})$ will become larger; an example for this case is illustrated in Fig. 5.2. In practice such a scenario can occur when an object of a certain class is observed from a viewpoint where it appears like a different class. If both $\mathbb{P}(\lambda_{k-1}|z_{1:k-1})$ and $\{\mathcal{L}(\gamma_k)\}$ are near the simplex center, the spread of $\mathbb{P}(\lambda_k|z_{1:k})$ will increase as well. Finally, if only one of $\mathbb{P}(\lambda_{k-1}|z_{1:k-1})$ and $\{\mathcal{L}(\gamma_k)\}$ is near the simplex center, $\mathbb{P}(\lambda_k|z_{1:k})$ will be similar to the one that is farther from the simplex center.

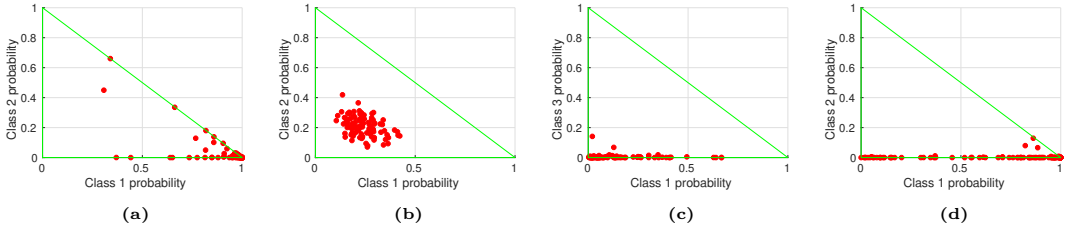


Figure 5.2: An example to illustrate a case where the posterior uncertainty *grows* with an additional image. The classifier model is the same as in Fig. 5.1, as well as the inference steps. **(a)** represents $\mathbb{P}(\lambda_{k-1}|z_{k-1})$. Here, in **(b)** the point cloud $\{\gamma_k\}$ is closer to class 3, compared to $\{\lambda_{k-1}\}$ cloud from **(a)** that is closer to class 1. The classifier model translates γ_k into $\mathcal{L}(\gamma_k)$ in **(c)**, projecting the point cloud around class 3, and thus after the multiplication in **(d)** the distribution is more spread out compared to **(a)**.

From $\mathbb{P}(\lambda_k|z_{1:k})$ we can infer the expectation $\mathbb{E}(\lambda_k)$ (computed as in Eq. (5.8)) and covariance matrix $Cov(\lambda_k)$ of λ_k. As $\mathbb{E}(\lambda_k)$ takes into account model uncertainty from each image, unlike existing approaches (e.g. [10]), we can achieve a posterior classification that is more resistant to possible aliasing. The covariance matrix $Cov(\lambda_k)$

represents the spread of λ_k, and in turn accumulates the model uncertainty from all images $z_{1:k}$. In general, lower $Cov(\lambda_k)$ values represent smaller λ_k spread, and thus higher confidence with the classification results. Practically, this can be used in a decision making context, where higher confidence answers are preferred. In this chapter we compare between values of $Var(\lambda_k^i)$ for all classes $i = 1, ..., M$, as it is simpler to describe the uncertainty per class.

There is a correlation between the expectation $\mathbb{E}(\lambda_k)$ and $Cov(\lambda_k)$. The largest covariance values will occur when $\mathbb{E}(\lambda_k)$ is at the simplex' center. In particular, it is not difficult to show that the highest possible value for $Var(\lambda_k^i)$ for any i is 0.25; it can occur when $\lambda_k^i = 0.5$. In general, if $\mathbb{E}(\lambda_k)$ is close to the simplex' boundaries, the uncertainty is lower. Therefore, to reduce uncertainty, $\mathbb{E}(\lambda_k)$ should be concentrated in a single high probability class.

To the author's knowledge, the expression $\mathbb{P}(\lambda_k|z_{1:k})$, where the expression for λ_k is described in Eq. (5.11), has no known analytical solution. The next most accurate method available is multiplying all possible permutations of point clouds $\{\gamma_t\}$, for all images at times $t \in [1, k]$. This method is computationally intractable as the number of λ_k points grows exponentially. In the next section we propose a simple sub-sampling method to approximate this distribution and keep computational tractability.

2.B Sub-Sampling Inference

As mentioned previously in section 5.1, each measurement we receive a cloud of N_k probability vectors $\{(\gamma_k)^n\}_{n=1}^{N_k}$. Each probability vector is projected via the classifier model to a different point with the simplex, which provides a new point cloud $\{\mathcal{L}(\gamma_k)^n\}_{n=1}^{N_k}$. We assume that $\mathbb{P}(\lambda_{k-1}|z_{1:k-1})$ is described by a cloud of N_{k-1} points. Given the data for γ_k and λ_{k-1}, the most accurate approximation to $\mathbb{P}(\lambda_k|z_{1:k})$ is given by multiplying all possible pairs of λ_{k-1} and $\mathcal{L}(\gamma_k)$. Thus, $\mathbb{P}(\lambda_k|z_{1:k})$ is described with a cloud of $N_{k-1} \times N_k$ points. For subsequent steps the cloud size grows exponentially, making it computationally intractable. We address this problem by randomly sampling from the point cloud for λ_k a subset of $N_{ss,n}$ points and use them for the next time step. In practice, we keep $N_{ss,n}$ constant across all time steps, see line 16 in Algorithm 5.1.

5.3 Experiments

In this section we study our method in simulation and using real images fed into an AlexNet [1] CNN classifier. We used a PyTorch implementation of AlexNet for classification, and Matlab for sequential data fusion. Our hardware is an Intel i7-7700HQ CPU running at 2.8GHz, and 16GB of RAM. We compare between four different approaches:

1. Method-$\mathbb{P}(c|z_{1:k})$-w/o-model: Naive Bayes that infers the posterior of $\mathbb{P}(c|z_{1:k})$ where the classifier model is not taken into account (SSBF in [10]).

2. Method-$\mathbb{P}(c|z_{1:k})$-w-model: A Bayesian approach that infers the posterior of $\mathbb{P}(c|z_{1:k})$ and uses a classifier model; essentially using Eq. (5.11) with a known classifier model.

3. Method-$\mathbb{P}(\lambda_k|z_{1:k})$-AP: Inference of $\mathbb{P}(\lambda_k|z_{1:k})$ multiplying all possible combinations of λ_{k-1} and $\mathcal{L}(\gamma_k)$. Note that the number of combinations grows exponentially with k, thus the results are presented up until $k = 5$.

4. Method-$\mathbb{P}(\lambda_k|z_{1:k})$-SS: Inference of $\mathbb{P}(\lambda_k|z_{1:k})$ using the sub-sampling method.

Our proposed approaches are 3 and 4.

3.A Simulated Experiment

This experiment is a simulation to demonstrate the algorithm's performance. This simulation is designed to emulate a scenario of a robot traveling in a predetermined trajectory and observing an object from multiple viewpoints. This object's class is one of three possible candidates. We infer the posterior over λ and display the results as expectation $\mathbb{E}(\lambda_k^i)$ and standard deviation per class i:

$$\sigma_i \doteq \sqrt{Var(\lambda_k^i)}. \tag{5.15}$$

This simulation is a study on the effect of using classifier model in the inference for highly ambiguous measurements. In addition, we analyze the uncertainty behavior for this scenario. We use a categorical uninformative prior of $\mathbb{P}(c = i) = 1/M$ for all $i = 1, ..., M$.

Each of the three classes has its own (known) classifier model Eq. (5.16), as shown in Figures 5.3a-5.3c. This classifier model is assumed Dirichlet distributed with the following hyperparameters θ_i for all $i \in [1, 3]$:

$$\begin{aligned} \theta_1 &= [6\ 1\ 1] \\ \theta_2 &= [2\ 7\ 2] \\ \theta_3 &= [1\ 1.5\ 2]\,. \end{aligned} \tag{5.16}$$

In this experiment the true class is 3. These hyperparameters were selected to simulate a case where the γ measurements are spread out (corresponds to ambiguous appearance of the class), thus leading to incorrect classification without a classifier model. The classifier model for this class $\mathcal{L}_3$ predicts highly variable γ's using the training data (Fig. 5.3c). The $\{\gamma_t\}$ point clouds for each $t \in [1, k]$ are different from each other (Fig. 5.3e), representing an object photographed by a robot from multiple viewpoints.

We simulate a series of 5 images. Each image at time step t has its own different $\mathbb{P}(\gamma_t|z_t)$. For the approaches that infer $\mathbb{P}(c|z_{1:k})$, we sample a single γ_t per image z_t for all $t \in [1, k]$ (Fig. 5.3f, also we present the γ_t order). This sample simulates the usual single classifier forward pass that is used. For our approaches we sample 10 γ_t's from

each $\mathbb{P}(\gamma_t|z_t)$, except for the first step $t = 1$ where we sample 100 γ_1's. For Method-$\mathbb{P}(\lambda_k|z_{1:k})$-SS each $\{\lambda_t\}$ point cloud is capped at 100 points. The expectation of these generated measurements are presented in Fig. 5.3d, along with the cloud order. In Fig. 5.3e $\{\gamma_t\}$ point clouds for three different t's are presented in distinct colors. The input for methods 1 and 2 is shown in Fig. 5.3f, and some of the input for methods 3 and 4 is shown in Fig. 5.3e

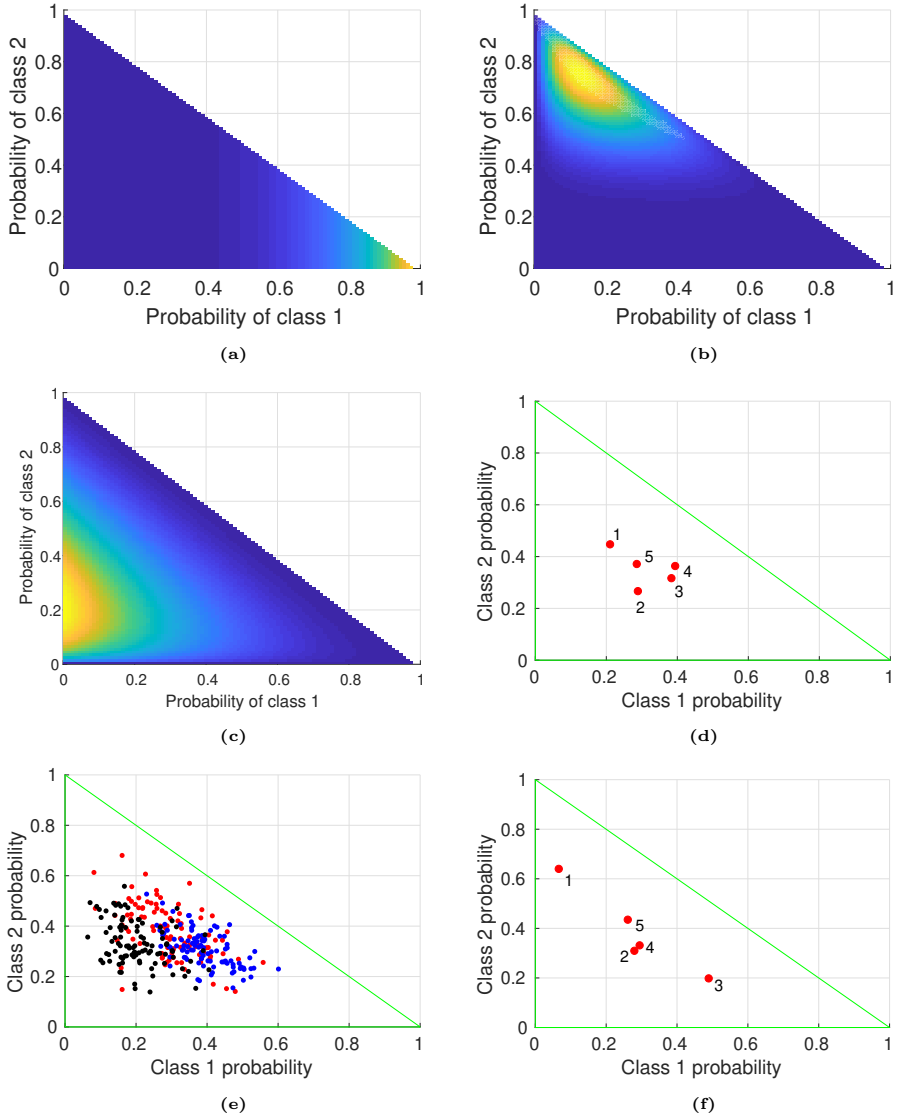


Figure 5.3: **(a)-(c)** Classifier likelihood model (Eq. (5.16)) for classes 1 to 3 respectively. Blue and orange colors correspond, respectively, to low and high probability values. **(d)** $\mathbb{E}(\gamma_t)$ for $t \in [1, 5]$ (i.e. 5 images). **(e)** Point cloud $\{\gamma_t\}$ for 3 images. **(f)** CNN classifier without dropout. In **(d)** and **(f)**, image indices are shown.

Fig. 5.4 presents results obtained with our algorithm, in terms of expectation $\mathbb{E}(\lambda_k^i)$ and $\sqrt{Var(\lambda_k^i)}$ for each class i, as a function of classifier measurements. In Fig. 5.4a and 5.4b we use a single sampled γ_t for z_t (see Fig. 5.3f), while in Fig. 5.4c and 5.4d we create a $\{\gamma_t\}$ point cloud for z_t (see Fig. 5.3e). In Fig. 5.4a and 5.4b results for Method-$\mathbb{P}(c|z_{1:k})$-w/o-model and Method-$\mathbb{P}(c|z_{1:k})$-w-model respectively. Without

classifier model the results generally favor class 2 incorrectly, as the measurements tend to give that class the higher chances. With classifier models the results favor class 3, the correct class. Because the classifier model for class 3 is more spread out than for the other classes, γ's in the simplex middle (as in Fig. 5.3e) have higher $\mathcal{L}_3(\gamma)$ values than $\mathcal{L}_1(\gamma)$ and $\mathcal{L}_2(\gamma)$. While method Method-$\mathbb{P}(c|z_{1:k})$-w-model gives eventually correct classification results, it does not account for model uncertainty, i.e. uses a single classifier output γ obtained with a forward run through the classifier without dropout. In this simulation we sample a single γ from each point cloud to simulate this forward run.

Figs. 5.4c and 5.4d present the results for Method-$\mathbb{P}(\lambda_k|z_{1:k})$-SS and Method-$\mathbb{P}(\lambda_k|z_{1:k})$-AP, expectation and standard deviation respectively. Throughout the scenario class 3 has the highest probability correctly, and the deviation drops as more measurements are introduced. Compared to Fig. 5.4b where class 3 has high probability only at time step $t = 3$, in Fig. 5.4c class 3 is the most probable from time step $t = 1$. Both Method-$\mathbb{P}(\lambda_k|z_{1:k})$-SS and Method-$\mathbb{P}(\lambda_k|z_{1:k})$-AP behave similarly. Note that class 1 has much smaller deviation than the other two because its probability is close to 0 through the entire scenario.

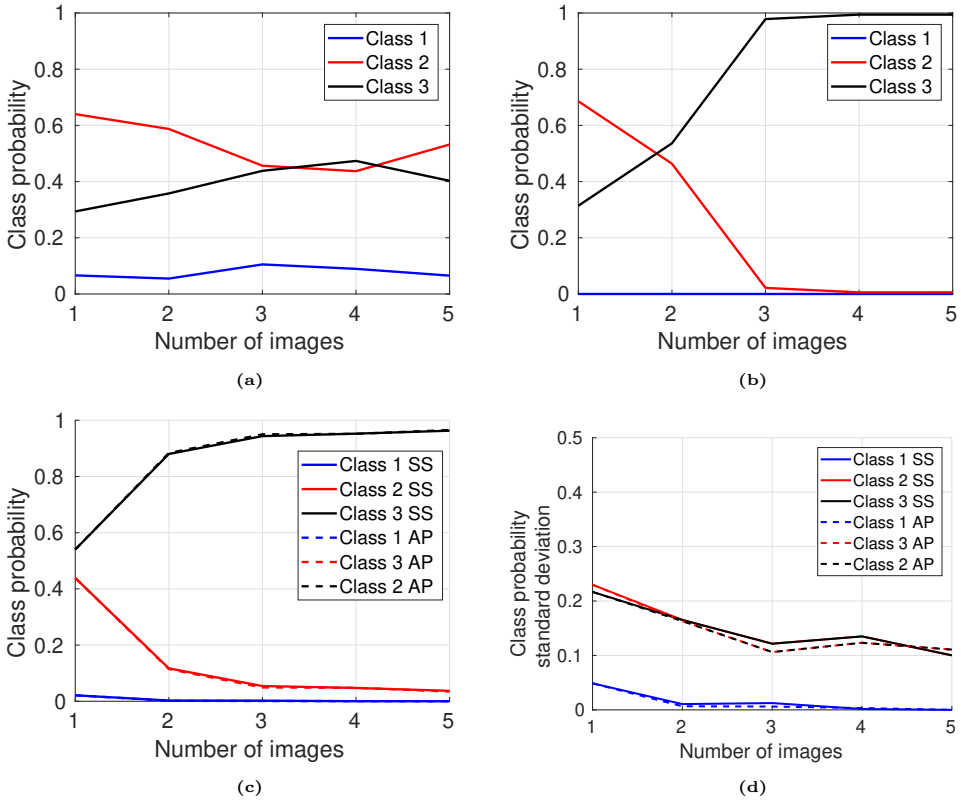
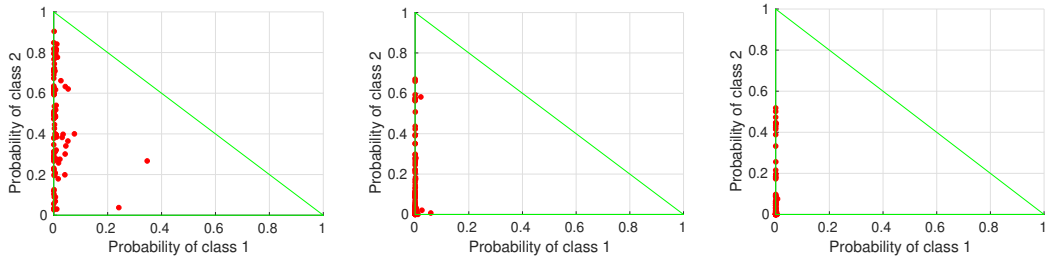


Figure 5.4: **(a)-(c)** Posterior class probabilities: **(a)** Method-$\mathbb{P}(c|z_{1:k})$-w/o-model; **(b)** Method-$\mathbb{P}(c|z_{1:k})$-w-model; **(c)** $\mathbb{P}(c|z_{1:k})$ calculated via expectation (5.8) for Method-$\mathbb{P}(\lambda_k|z_{1:k})$-SS and Method-$\mathbb{P}(\lambda_k|z_{1:k})$-AP; **(d)** presents the posterior standard deviation Eq. (5.15)

for both of our methods.

Fig. 5.5 presents the development of $\{\lambda_k\}$ point clouds for Method-$\mathbb{P}(\lambda_k|z_{1:k})$-SS

at different time steps. Those figures show the gradual decrease in $\{\lambda_k\}$'s spread, coinciding with the corresponding standard deviation at Fig. 5.4d.



(a) k=1 **(b)** k=2 **(c)** k=3

Figure 5.5: The figure depicts the evolution of the $\{\lambda_k\}$ point cloud, as calculated by Method-$\mathbb{P}(\lambda_k|z_{1:k})$-SS, for different time instances k.

3.B Experiment with Real Images

Our algorithm is tested using a series of images of an object (space heater) with conflicting classifier outputs when observed from different viewpoints. This corresponds to a scenario where a robot in a predetermined path observes an object that is obscured by occlusions and different lighting conditions. The experiment presents our algorithm's robustness to these difficulties in classification, and addressing them is important for real-life robotic applications.

The database photographed is a series of 10 images of a space heater with artificially induced blur and occlusions. Each of the images is run through an AlexNet convolutional neural network [1] with 1000 possible classes. Similar to Section 3.A, we use an uninformative classifier prior on $\mathbb{P}(c)$ with $\mathbb{P}(c = i) = 1/M$ for all $i = 1, ..., M$ classes. Our algorithm is used to fuse the classification data into a posterior distribution of the class probability and infer deviation for each class. As in the previous section, we present results with and without classifier model. Fig. 5.6 presents four of the dataset images, exhibiting occlusions, blur and different colored filters in a monotone environment.

(a) **(b)** **(c)** **(d)**

Figure 5.6: Four of the 10 images used in the dataset with occlusions and different viewpoints. Blurring and colored filters were introduced to some images artificially.

We compare between the same methods that are used in the previous sub-sections. For Method-$\mathbb{P}(c|z_{1:k})$-w/o-model and Method-$\mathbb{P}(c|z_{1:k})$-w-model, we forward the images through the classifier without dropout and use a single output γ for each image. For

Method-$\mathbb{P}(\lambda_k|z_{1:k})$-SS, we run each image 10 times through the classifier with dropout, producing a point cloud $\{\gamma\}$ per image. The cap for number of λ_k points with Method-$\mathbb{P}(\lambda_k|z_{1:k})$-SS is 100. For Method-$\mathbb{P}(\lambda_k|z_{1:k})$-AP method, we present results only for the first five images as the calculations become infeasible due to the exponential complexity.

As AlexNet has 1000 possible classes (one of them is "Space Heater"), it is difficult to clearly present results for all of them. Because we wish to compare between the most likely classes, we select 3 likely classes by averaging all γ classifier outputs and selecting the three with highest probability. The probabilities for those classes are then normalized, and utilized in the scenario. All other classes outside those three are ignored. We require a classifier model for each class; assuming the classifier model is Dirichlet distributed, we classified multiple images unrelated to the scenario for each class with the same AlexNet classifier but without dropout. The classifier produced multiple γ's, one per image, and via a Maximum Likelihood Estimator [112] we inferred the Dirichlet hyperparameters for each class $i \in [1, 3]$. The classifier model $\mathbb{P}(\gamma_k|c = i) = Dir(\gamma_k; \theta_i)$ was used with the following hyperparameters θ_i:

$$\begin{aligned} \theta_1 &= [5.103\ 1.699\ 1.239] \\ \theta_2 &= [0.143\ 208.7\ 5.31] \\ \theta_3 &= [0.993\ 14.31\ 25.21] \end{aligned} \tag{5.17}$$

In this experiment, class 1 is the correct class (i.e. "Space Heater"). Fig. 5.7 presents the simplex representation of the classifier model per class, and a normalized simplex of classifier outputs for three high probability classes, similarly to Fig. 5.3. The classifier model for class 1 is much more spread than the other two (Fig. 5.7a), therefore the likelihood of measurements within a larger area will be higher for this class. Interestingly, the classifier model for class 3 predicts $\mathbb{P}(\gamma_k|c = 3)$ will be between classes 2 and 3 (Fig. 5.7c). Fig. 5.7e presents 4 of the 10 $\{\gamma_t\}$ point clouds used in the scenario. Fig. 5.7d presents the expectation of each $\{\gamma_t\}$ point cloud for $t \in [1, 10]$. Fig. 5.7f presents classifier outputs without dropout, i.e. a single γ_t per image. Both Fig. 5.7d and 5.7f have indices that represent the images order.

Fig. 5.8 presents the classification results for all the methods presented. Fig. 5.8a and 5.8b show results for Method-$\mathbb{P}(c|z_{1:k})$-w/o-model and Method-$\mathbb{P}(c|z_{1:k})$-w-model respectively. Without a classifier model, i.e. the former method, incorrectly indicates class 2 as the most likely, because the classifier outputs often show class 2 as the most likely (see Fig. 5.7f). With a classifier model, the results jump between classes 1 and 3 as most probable. This can be explained by the likelihood vector $\mathcal{L}$ from Eq. (5.17) that projects the γ's from different images approximately to different simplex edges (e.g. γ_2 and γ_4 for class 1, and γ_3 and γ_5 for class 3).

Figs. 5.8c and 5.8d present results for Method-$\mathbb{P}(\lambda_k|z_{1:k})$-SS and Method-$\mathbb{P}(\lambda_k|z_{1:k})$-AP, expectation and standard deviation respectively. Fig. 5.8c presents class 1 as most likely correctly in both methods from $k = 2$ onwards, and the results are smoother than

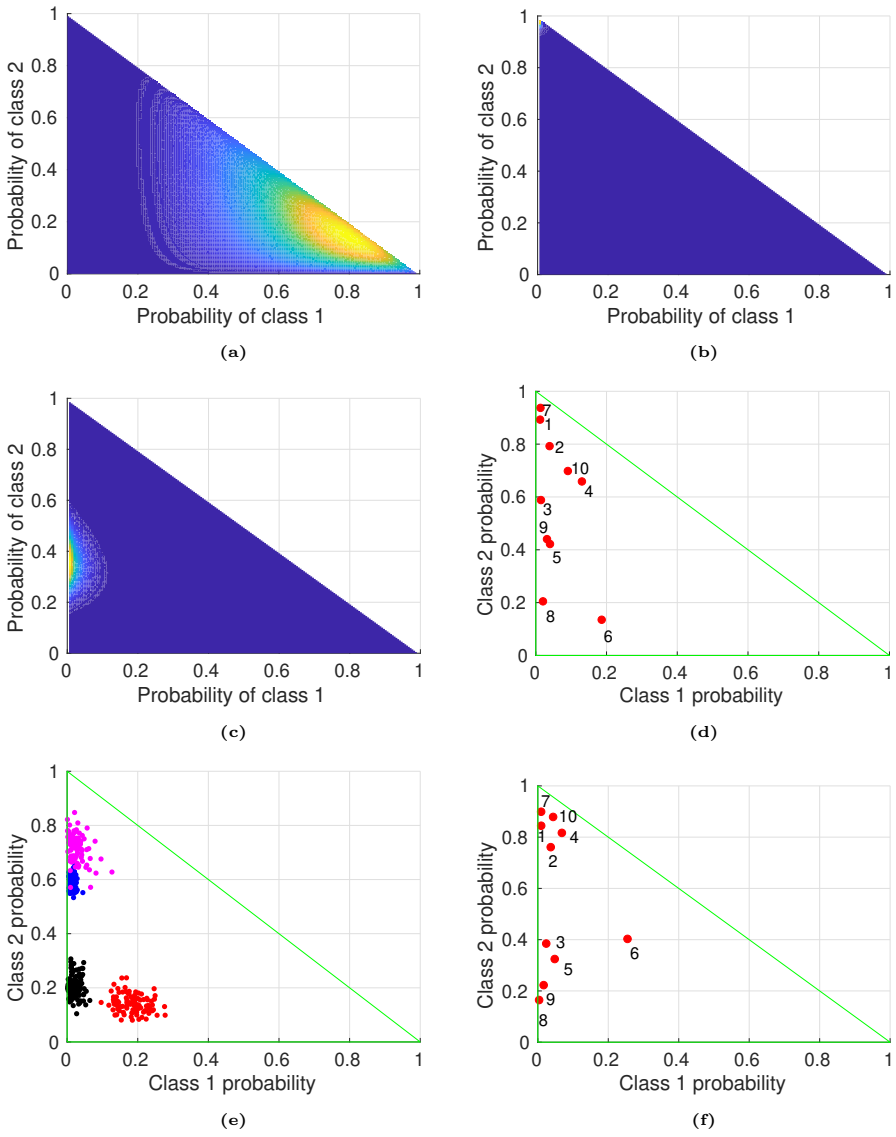


Figure 5.7: A simplex representation of the classifier model for **(a)** class 1, **(b)** class 2, and **(c)** class 3. In **(b)**, note the distribution is very tight centered at the top left corner of the simplex. **(d)** $\mathbb{E}(\gamma_t)$ for $t \in [1, 10]$ (i.e. 10 images). **(e)** Pointcloud $\{\gamma_t\}$ for 4 images. **(f)** CNN classifier output without dropout. In **(d)** and **(f)**, image indices are shown.

in Fig. 5.8b because our algorithm takes into account multiple realizations of γ_1 to γ_{10} - we recall that for each image we use a point cloud of γ's. In addition, we can reason about the standard deviation of λ_k, representing the posterior uncertainty, as seen in Fig. 5.8d. Note that starting from the 4th image, the uncertainty *increases*, as later measurement likelihoods do not agree with λ_{k-1} about the most likely class at those time steps, similar to the example presented in Fig. 5.2.

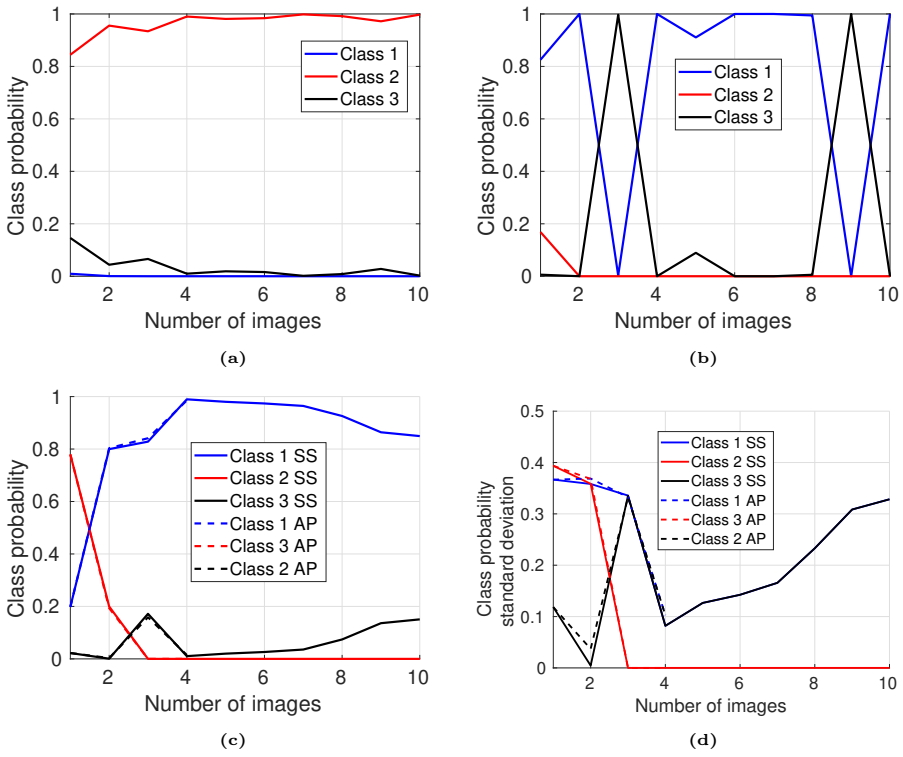
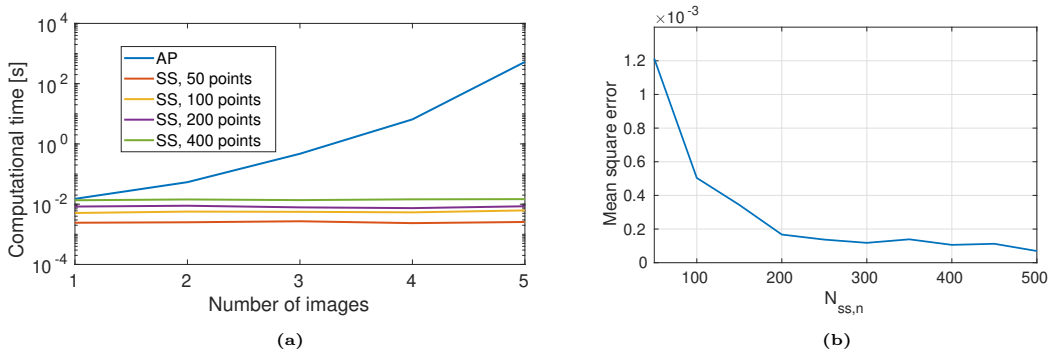


Figure 5.8: **(a)-(c)** Posterior class probabilities: **(a)** Method-$\mathbb{P}(c|z_{1:k})$-w/o-model; **(b)** Method-$\mathbb{P}(c|z_{1:k})$-w-model; **(c)** $\mathbb{P}(c|z_{1:k})$ calculated via expectation (5.8) for Method-$\mathbb{P}(\lambda_k|z_{1:k})$-SS and Method-$\mathbb{P}(\lambda_k|z_{1:k})$-AP; **(d)** presents the posterior standard deviation Eq. (5.15) for both of our methods.

Fig. 5.9a presents the computational time comparison between those two methods for the scenario presented in this section, including different number of samples $N_{ss,n}$ per time step. Importantly, the results for Method-$\mathbb{P}(\lambda_k|z_{1:k})$-SS are similar to Method-$\mathbb{P}(\lambda_k|z_{1:k})$-AP while offering significantly shorter computational times. Note that the computational time per step is constant as well for Method-$\mathbb{P}(\lambda_k|z_{1:k})$-SS. Fig. 5.9b presents mean square error (MSE) of Method-$\mathbb{P}(\lambda_k|z_{1:k})$-SS compared to Method-$\mathbb{P}(\lambda_k|z_{1:k})$-AP, as a function of $N_{ss,n}$. As expected, larger $N_{ss,n}$ values produce lower MSE.



(a) (b)

Figure 5.9: (a) Computational time comparison between Method-$\mathbb{P}(\lambda_k|z_{1:k})$-AP and Method-$\mathbb{P}(\lambda_k|z_{1:k})$-SS per time step. The figure presents computational times for $N_{ss,n} \in \{50, 100, 200, 400\}$ points per time step for Method-$\mathbb{P}(\lambda_k|z_{1:k})$-SS. (b) The statistical mean square error of Method-$\mathbb{P}(\lambda_k|z_{1:k})$-SS as a function of $N_{ss,n} \in [50, 500]$ relative to Method-$\mathbb{P}(\lambda_k|z_{1:k})$-AP.

Chapter 6

Epistemic Uncertainty Aware Semantic Localization and Mapping for Inference and Belief Space Planning

In this chapter we contribute a unified framework for epistemic uncertainty aware inference and belief space planning in the context of semantic perception and SLAM. Our framework considers prominent sources of uncertainty — classification aliasing, classifier epistemic uncertainty, and localization uncertainty — within inference and BSP.

Specifically, the main contributions of this chapter are as follows.

1. We develop two methods for maintaining a joint distribution over robot and object poses, and over the posterior class probability vector that considers epistemic uncertainty in a Bayesian fashion. The first approach is Multi-Hybrid (MH), where multiple hybrid beliefs over poses and classes are maintained to approximate the joint belief over poses and posterior class probability. The second approach is Joint Lambda Pose (JLP), where the joint belief maintained directly using a novel JLP factor.

2. We extend both methods to a BSP framework, planning over posterior epistemic uncertainty indirectly, or directly via a novel information-theoretic reward over the distribution of posterior class probability.

3. Our inference and BSP methods utilize a novel viewpoint dependent classifier model that predicts epistemic classifier uncertainty given a candidate class and relative viewpoint, allowing us to reason about the coupling between poses and classification scores, and predict future epistemic classifier uncertainty, while avoiding predicting and generating entire images.

4. We extensively study our inference and BSP methods in simulation and using real data from the Active Vision Dataset [113].

This chapter is structured as follows: In Sec. 6.3 we address epistemic-uncertainty-aware inference; MH and JLP are introduced in first for the single object case and afterwards to multiple objects case. In Sec. 6.4 we expand both approaches to BSP, and discuss the information-theoretic cost over the distribution of posterior class probability. Finally, we validate our approaches in Sec. 6.5 first in simulation in Sec. 5.B, and then using Active Vision Dataset and BigBIRD in Sec. 5.C.

6.1 Preliminaries

In this section we introduce notations, provide preliminary material, and formulate the problem addressed in this work. First, we introduce our setting and simultaneous localization and mapping (SLAM) notations. Afterwards, we introduce notations specifically for classification in the context of epistemic uncertainty. Finally, we briefly introduce belief space planning (BSP), and present the problem formulation for epistemic uncertainty aware semantic inference and planning.

1.A Simultaneous Localization and Mapping (SLAM)

Consider a robot operating in an unknown environment represented by object landmarks. For inference and planning over a distribution of posterior class probabilities, we need to solve an underlying object based simultaneous localization and mapping problem (SLAM). The robot's and objects pose, and objects' classes are all unknown. Let x_k denote the robot pose at time k; Let x^o and c denote object pose and class respectively. To shorten notations, denote $\mathcal{X}_x \triangleq \{x^o, x_{1:k}\}$ as all poses of robot and the observed (expanded later to multiple objects) up until time k.

The robot receives from observed objects both geometric and semantic measurements. Let z_k denote a measurement received at time k from the object. This measurement is split into geometric z_k^g and semantic z_k^s measurements; All those measurements are aggregated to a set $z_k \triangleq \{z_k^g, z_k^s\}$. The robot action at time k is denoted a_k, and finally we denote the measurement history as $\mathcal{H}_k \triangleq \{z_{1:k}, a_{0,k-1}\}$. We assume independence between semantic and geometric measurements, as well between different time steps.

We utilize a known Gaussian motion model with constant parameters, denoted $\mathcal{M}_k$, and defined as:

$$\mathcal{M}_k \triangleq \mathbb{P}(x_k|x_{k-1}, a_{k-1}) \tag{6.1}$$

and a known geometric model $\mathbb{P}(z_k^g|x^o, x_k)$. In addition, we use an externally trained viewpoint dependent classifier and uncertainty model $\mathbb{P}(z_{k,n}^s|c_n, x^o, x_k)$ that will be discussed in Section 3.A. Let us denote the corresponding measurement likelihood term,

$$\mathcal{L}_k \triangleq \mathbb{P}(z_k^g | x^o, x_k) \cdot \mathbb{P}(z_k^s | c, x^o, x_k), \tag{6.2}$$

where, both geometric and classifier models are considered Gaussian as well.

1.B Distribution Over Class Probability Vector

During inference the robot receives a raw image in which observed objects are segmented. In standard (deep-learning) approaches a classification model, i.e. a classifier, is learned beforehand and used to classify the objects within each segment (e.g. bounding box) by producing an output of a class probability vector. Given fixed classifier weights w, we denote a probability vector from a classifier at time k as

$$\gamma_k \triangleq \mathbb{P}(c | I_k, w), \tag{6.3}$$

where I_k is the raw image of the object. Also, denote $\gamma_{k,w}$ as the probability vector given a specific w. In practice, the image fed into the classifier is a cropped image of an object via a bounding box. Note that $\gamma_k \triangleq [\gamma_k^1, ..., \gamma_k^m] \in \mathbb{R}^m$ is a probability vector, thus it must satisfy the following conditions:

- All its elements must sum to 1, i.e. $\sum_{i=1}^m \gamma_k^i = 1$.
- Each element is bounded between 0 and 1, i.e. $0 \leq \gamma_k^i \leq 1, \ \ \forall i = 1, ..., m$.

In contrast to this standard approach, in this work we reason about classifier epistemic uncertainty. Denote D as the classifier's training set. In literature, these approaches rely on describing the trained weights w as random variables by themselves distributed $w \sim \mathbb{P}(w|D)$, thus making γ_k a random variable. In this chapter we create a set W of sampled w to produce a point cloud of γ_k vectors per object and time step, such that we can describe the distribution over γ_k with the delta Dirac function $\delta(\cdot)$:

$$\gamma_k \sim \mathbb{P}(\gamma_k | I_k, D) = \int_w \delta(\gamma_k = \mathbb{P}(c | I_k, w)) \mathbb{P}(w|D) dw, \tag{6.4}$$

which we approximate via sampling as:

$$\mathbb{P}(\gamma_k | I_k, D) \approx \frac{1}{|W|} \sum_w \delta(\gamma = \mathbb{P}(c | I_k, w)). \tag{6.5}$$

Thus for each time step we get a point cloud $\{\gamma_k\}$ per object where its spread describes the epistemic model uncertainty of the classifier. See a simplified illustration in Fig. 6.1, where an object is observed from multiple viewpoints, and the classifier outputs a cloud of γ's for each viewpoint. For example, the cloud $\{\gamma_k\}$ obtained by observing the object from the bottom right corner is spread widely, therefore the epistemic uncertainty from that viewpoint is high. Contrast it with the upper-right viewpoint where the spread is tight, representing low epistemic uncertainty. In this chapter the semantic

measurements are those point clouds within the $m-1$ simplex, such that $z_k^s = \{\gamma_k\}$. The set of sampled w can be created by, for example, MC-dropout [21] or Bootstrapping [18].

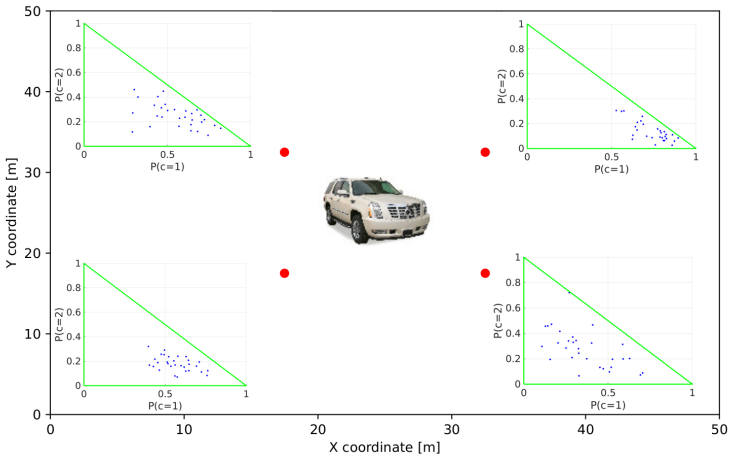


Figure 6.1: Illustration of viewpoint dependency for both classification scores and epistemic uncertainty. The figure presents simplex graphs for different viewpoints, where $m = 3$. The individual class probability scores are shown as blue points in the simplex, where it's borders are in green. The red points represent possible viewpoints observing the SUV in the middle.

1.C Distribution Over Posterior Class Probability Vector

Eventually the posterior over a *sequence* of γ vectors can be inferred. This posterior takes into account both the epistemic uncertainty from multiple observations of an object, as well as localization uncertainty induced by coupling between relative poses and class probabilities. The posterior is defined as follows:

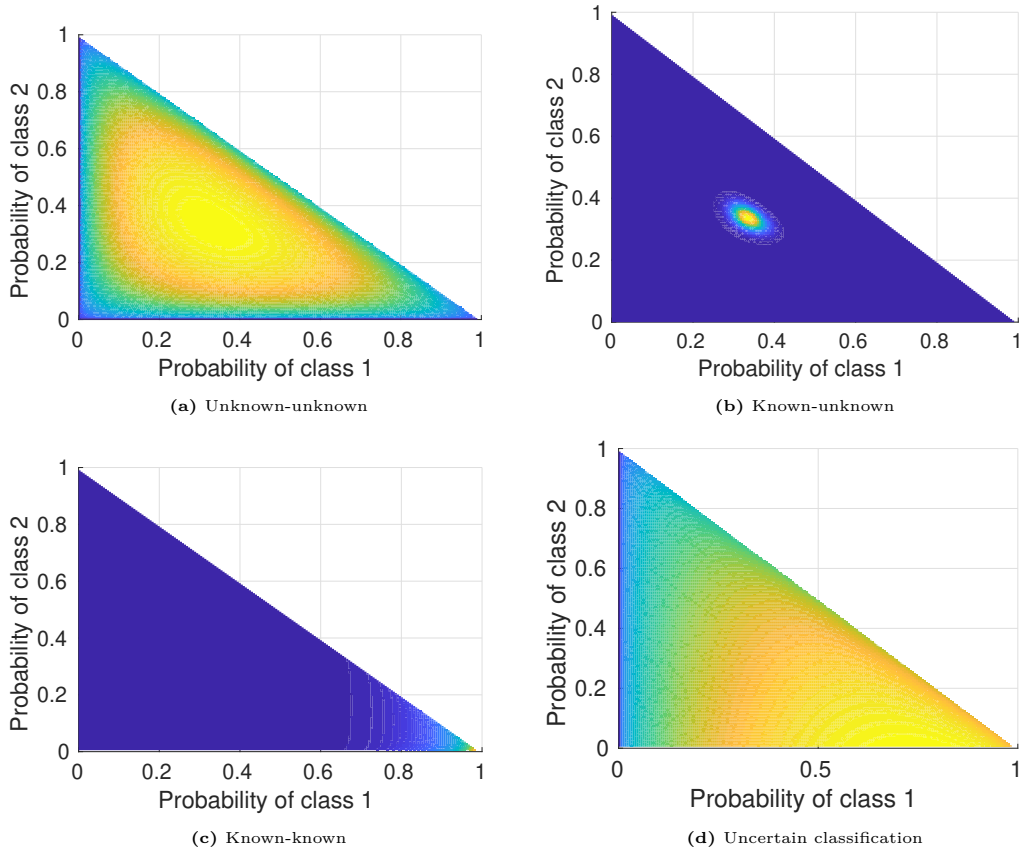
$$\lambda_k \triangleq \mathbb{P}(c|\gamma_{1:k}, z^g_{1:k}), \tag{6.6}$$

where λ_k is deterministically determined by both a sequence $\gamma_{1:k}$ and the geometric measurement history. For a specific $\gamma_{1:k,w}$ sequence which is created by a specific w, we use the notation $\lambda_{k,w}$. Because we consider $\gamma_{1:k}$ to be a random variable (as w is a random variable), so is λ_k. As such, we can define a belief over λ_k the following way:

$$b[\lambda_k] \triangleq \mathbb{P}(\lambda_k|I_{1:k}, \mathcal{H}^g_k, D). \tag{6.7}$$

The belief $b[\lambda_k]$ encompasses both the posterior classification probability vector via $\mathbb{E}(\lambda_k)$, and the epistemic and localization uncertainty via $Cov(\lambda_k)$. The belief $b[\lambda_k]$ representation is more expressive than a single class probability vector representation, and it can reflect four possible archetypes, as seen in Fig. 6.2 (see [7]). Fig. 6.2a presents an out-of-distribution case where the inputs to the classifier are totally alien, therefore the output is completely unpredictable. Fig. 6.2c represent a case where the classifier can safely identify the object with high degree of certainty, i.e. the input is close to the training set. Intuitively, this is the case that we aim for, and generally

has the highest reward. Fig. 6.2b represents the case of high data uncertainty where the classifier certainly cannot disambiguate between different classes, i.e. the classifier "knows" that it does not know. This can be resulted from ambiguity in the training set between different classes, when objects from different classes look identical from certain viewpoints. Finally, Fig. 6.2d represent a case where the classifier can vaguely infer the object class, but it's still far from the training set (e.g. a car of an unusual shape that there are no similar images in the training set), therefore with a large degree of uncertainty.



(a) Unknown-unknown

(b) Known-unknown

(c) Known-known

(d) Uncertain classification

Figure 6.2: The 4 archetypes of $b[\lambda_k]$, shown in a 3 dimensional Dirichlet simplex example, where blue to yellow correspond to low to high probability respectively. **(a)** is an out-of-distribution setting, where the classifier does not identify the object and the epistemic uncertainty is high. **(b)** is high data uncertainty setting, which close to D, but the class is identifiable in the training set itself. In **(c)** the classifier recognizes that the object of a certain class with certainty, a scenario we aim for. In **(d)** the classifier gives preference to one of the classes, but with a high degree of uncertainty.

As we shall see, this belief can be used within belief space planning, e.g. going to relative poses where the epistemic uncertainty is the smallest to safely classify objects, or vice-versa going to relative poses with high epistemic uncertainty to potentially learn a model online.

More generally λ_k is coupled with object and camera poses $\mathcal{X}_k$. A joint belief over λ_k and $\mathcal{X}_k$ can be maintained, denoted as:

$$b[\lambda_k, \mathcal{X}_k] \triangleq \mathbb{P}(\lambda_k, \mathcal{X}_k | I_{1:k}, \mathcal{H}_k^g, D). \tag{6.8}$$

We may require to compute $\mathbb{E}(\lambda_k)$ to, for example, compute a reward function that depends on $\mathbb{E}(\lambda_k)$. Consider that $\mathbb{P}(c = i|\lambda_k, \mathcal{X}_k) = \lambda_k^i$, then for every object class $c = i$:

$$\mathbb{P}(c = i|I_{1:k}, Z_k^g, D) = \int_{\lambda_k, \mathcal{X}_k} \lambda_k^i \cdot b[\lambda_k, \mathcal{X}_k] d\lambda_k d\mathcal{X}_k = \mathbb{E}(\lambda_k^i). \tag{6.9}$$

In chapter 5 we presented an approach to maintain $b[\lambda_k]$ in a setting with a single object, didn't consider the coupling between $\mathcal{X}_k$ and λ_k, and the approach was limited to inference. On the other hand, here we account for the coupling between λ_k and $\mathcal{X}_k$, present an active approach, and expand to a multi-object setting. First we consider the formulation for a single object. In the approach sections 6.3 and 6.4 we extend the formulation to the multiple object case, with each method having its specific notations.

1.D Belief Space Planning (BSP)

Given a general current belief b_k, one can reason about the best future action from a set of action to maximize (or minimize) an object function. With b_k and a set of future actions $a_{k:k+L}$, it is common to define the objective function as the expected cumulative reward,

$$J(b_k, a_{k:k+L}) = \mathbb{E}_{\mathcal{Z}_{k+1:k+L}}(\sum_{i=0}^{L} r(b_{k+i}(\mathcal{Z}_{k+1:k+i})), a_{k+i}), \tag{6.10}$$

where $r(\cdot)$ is a belief-dependent reward function, and L is the planning horizon. This formulation can be extended to policies as well.

The above equation can also be also written in a recursive form as in,

$$\begin{aligned} J(b_k, a_{k:k+L}) = \int_{\mathcal{Z}_{k+1}} & \mathbb{P}(\mathcal{Z}_{k+1}|\mathcal{H}_k, a_k) \cdot \\ & \cdot J(b_{k+1}, a_{k+1:k+L}) d\mathcal{Z}_{k+1}, \end{aligned} \tag{6.11}$$

where $b_{k+1} = b_{k+1}(\mathcal{Z}_{k+1})$. The term $\mathbb{P}(\mathcal{Z}_{k+1}|\mathcal{H}_k, a_k)$ is the measurement likelihood of future measurement history thus far and and a_k, and is essential for BSP. In practice, most of the time the integral in Eq. (6.11) cannot be analytically computed, thus it is approximated in sampled form:

$$J(b_k, a_{k:k+L}) \approx \frac{1}{N_z} \sum_{\mathcal{Z}_{k+1}} J(b_{k+1}(\mathcal{Z}_{k+1}), a_{k+1:k+L}), \tag{6.12}$$

where N_z is the number of Z_{k+1} samples, and $Z_{k+1} \sim \mathbb{P}(Z_{k+1}|\mathcal{H}_k, a_k)$.

The optimal action sequence $a_{k:k+L}^*$ is chosen such that it maximizes the objective function:

$$a_{k:k+L}^* = \arg\max_{a_{k:k+L}} \left(J(b_k, a_{k:k+L}) \right). \tag{6.13}$$

To evaluate the optimal action sequence, one must consider all possible sequences (possibly via search algorithms) and select the one that produces the highest objective

function.

Specifically, in this chapter we consider the belief $b_k = b[\lambda_k, \mathcal{X}_k]$ for BSP, and discuss planning using various reward functions, while focusing on classifier epistemic uncertainty reward function, namely the entropy of $b[\lambda_{k+i}]$ for a future time $k+i$. Yet, first, we must address the corresponding inference problem.

1.E Problem Formulation

Given geometric measurement history $\mathcal{H}_k^g$, an image sequence $I_{1:k}$, actions $a_{0;k-1}$, an epistemic-uncertainty-aware classifier trained on training dataset D with a set W of weight realizations $w \in W$, the problems of inference and planning are defined as follows:

1. *Inference:* Infer the posterior joint belief $b[\lambda_k, \mathcal{X}_k]$, as defined in Eq. (6.8).

2. *Planning:* Given $b[\lambda_k, \mathcal{X}_k]$, find the future action sequence $a^*_{k:k+L}$ that maximizes $J(b[\lambda_k, \mathcal{X}_k], a_{k:k+L})$ with the reward function $r(b[\lambda_k, \mathcal{X}_k])$.

In this chapter we address the inference and planning problems in Sections 6.3 and 6.4, respectively.

6.2 Approach Overview

Two approaches are presented for solving each of the problems presented in Sec. 1.E. The first approach is Multi-Hybrid (MH), a particle-based approach where multiple hybrid beliefs are maintained simultaneously. The second approach is Joint Lambda Pose (JLP), where a single continuous belief is maintained and the posterior class probabilities are states within this belief.

The approach sections are divided to inference and planning; Starting with inference, we introduce the viewpoint dependent classifier uncertainty model, which predicts the distribution of the classifier output and is used by both methods for inference and planning. In particular, the classifier uncertainty model is used to generate predicted measurements during planning. Then, we introduce MH and JLP for inference; First, for simplicity we consider the single object case, afterwards the formulation is expanded to multiple objects. The section concludes with a computation complexity analysis and comparison.

Section 6.4 addresses the planning problem; First we discuss measurement generation in general for a single object, then delve into the specifics of both MH and JLP of generating measurements for multiple objects. Afterwards we discuss reward functions, and specifically expand upon information-theoretic reward for $b[\lambda]$. Finally, we discuss Dirichlet distribution and LG as possible distributions of λ when using MH (JLP is limited to LG).

For the reader's convenience, main notations used in this chapter are summarized in Table. 6.1.

6.3 Approach- Inference

3.A Viewpoint Dependent Classifier Uncertainty Model

We use a classifier uncertainty model that accounts both for the coupling between localization and classification, and epistemic model uncertainty. As an example, Fig. 6.1 illustrates that $\{\gamma_k\}$ measurements varies across different viewpoints, with some containing high epistemic uncertainty and some low. The model we propose learns to predict these measurements, and subsequently which viewpoints will contain high epistemic uncertainty. In contrast, previous works that used a viewpoint dependent classifier model (e.g. chapters 3, 4 and [12, 15]) did not consider epistemic uncertainty while learning the model.

The conditions for γ_k being a probability vector must be considered when one requires to sample from the classifier model, thus unlike previous chapters 3 and 4 we cannot use a Gaussian distributed classifier model. One possible solution is to consider the classifier model as Dirichlet distributed (see chapter 5), but that model cannot be incorporated into a Gaussian optimization framework (e.g. iSAM2 [36]) with unknown poses which are coupled with classification results. Instead, we consider the following solution: we use a logit transformation for γ to a vector $l\gamma \in \mathbb{R}^{m-1}$ space, such that the support of each element $(-\infty, \infty)$:

$$l\gamma \triangleq \left[\log\left(\frac{\gamma^1}{\gamma^m}\right), \log\left(\frac{\gamma^2}{\gamma^m}\right), ..., \log\left(\frac{\gamma^{m-1}}{\gamma^m}\right)\right]^T. \tag{6.14}$$

Then, $l\gamma$ can be assumed Gaussian such that:

$$\mathbb{P}(l\gamma_k|c, x^o, x_k) = \mathcal{N}(h_c(x^o, x_k), \Sigma_c(x^o, x_k)), \tag{6.15}$$

and as a consequence γ_k is distributed Logistical Gaussian with parameters $\{h_c, \Sigma_c\}$. The probability density function (PDF) of γ_k is as follows:

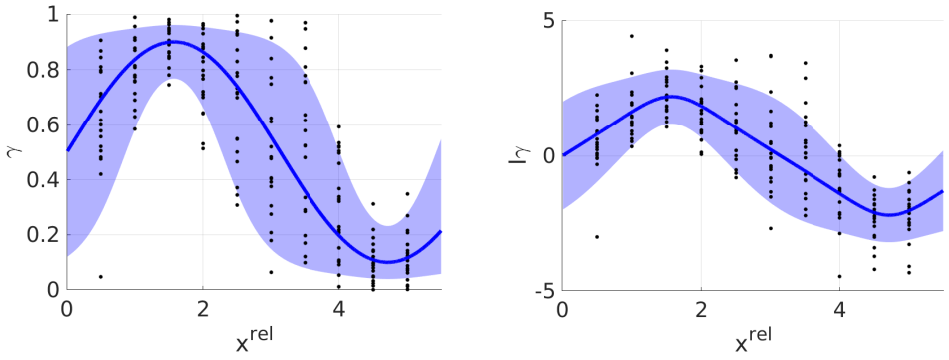
$$\mathbb{P}(\gamma_k|c, x^o, x_k) = \frac{1}{\sqrt{|2\pi\Sigma_c|}} \cdot \frac{1}{\prod_{i=1}^m \gamma_k^i} \cdot e^{\left(-\frac{1}{2}||l\gamma_k - h_c||^2_{\Sigma_c}\right)}. \tag{6.16}$$

In practice, a classifier provides us with a cloud $\{\gamma_k\}$, and each $\gamma_k \in \{\gamma_k\}$ is transformed to $l\gamma_k$. There are m such models, one for each class. The training set consists of tuples of relative pose and $l\gamma$ point clouds such that $D_{cm} \triangleq \{x^{rel}, \{l\gamma\}\}$ for each class, where $x^{rel} \triangleq x^o \ominus x$ is the relative pose between object and robot; the expectation (classification scores) and covariance (epistemic uncertainty) is extracted from $\{l\gamma\}$ and fitted as known points either in the model using e.g. Gaussian Processes or deep-learning

Table 6.1: Main notations used in the paper.

Parameters	
x	Robot pose
x^o	Object o's pose
$\mathcal{X}_k$	All robot and object poses up to k
x^{rel}	Relative pose between x and x^o
O_k	Set of all objects observed at time k
x_k^{inv}	Set that contains the last robot pose and all object poses from O_k
c^o	Object o's class
C	Class realization of all objects
z^g	Geometric measurement
z^s	Semantic measurement
n	The amount of all objects in the environment
n_k	Number of objects observed at time k
N_k	Number of objects observed up to time k
$\mathcal{M}_k$	Motion model from x_{k-1} to x_k
a	Robot action
$\mathcal{H}_k$	History of measurements and action up to time k
$\mathcal{H}_k^g$	History of geometric measurements and action up to time k
Z_k^g	All geometric measurements for all objects at time k
$\mathcal{L}^s$	Semantic measurement likelihood
h_c	Expectation of class c's classifier uncertainty model
Σ_c	Covariance of class c's classifier uncertainty model
$\mathcal{L}_k$	Geometric and semantic measurement likelihood at time k
D	Classifier training dataset
$\{\cdot\}$	Set or point cloud
I	Raw image
$l\square$	Logit transformation of probability vector
γ	Probability vector classifier output
γ^c	Element of γ of class c
Γ_k	Set of all γ observed at time k, one per object
$l\Gamma_k$	Set of all logit transformations for all $\gamma_k \in \Gamma_k$
λ	Posterior class probability vector
λ^c	Element of λ of class c
Λ_k	Posterior probability vector for class realizations
$l\bar{\lambda}_k$	Set of $l\lambda$ of all objects observed up to k
W	Set of all possible classifier weight realizations w
$b[\cdot]$	Belief, probability conditioned on history $\mathbb{P}(\cdot\|I_{1:k}, \mathcal{H}_k^g, D)$.
b_w^c	Continuous belief conditioned on history, c, and w
hb_w	Hybrid belief conditioned on w
$l\mathcal{L}^s$	Logit transformation of semantic measurement likelihood
Subscripts	
w	Classifier weight realization
k	Time step
L	Planning horizon
Superscript	
o	Object o
c	Class hypothesis of an object
C	Class hypothesis of all objects

based approaches. Real-life application may require creating D_{cm} from multiple different instances of the same objects, e.g. for class "car" multiple types of cars may be used. Fig. 6.3 illustrates the training data shown in black dots versus the trained model shown in blue. The model attempts to "predict" the epistemic uncertainty based on a given training set.



(a) γ space **(b)** $l\gamma$ space

Figure 6.3: Simplified illustration of the classifier uncertainty model we use in the chapter. **(a)** and **(b)** represent γ and $l\gamma$ space respectively. The black dots represent the corresponding $\gamma(x^{rel}, w) \in \{\gamma\}(x^{rel})$ and $l\gamma(x^{rel}, w) \in \{l\gamma\}(x^{rel})$. The expectation and covariance are learned in **(b)** and interpolated for new queries of x^{rel}, potentially returning to **(a)** via the inverse logit transformation. The expectation is represented in dark blue, while the one sigma covariance is represented in light blue.

3.B Multi-Hybrid Inference

In this section we present the Multi-Hybrid (MH) inference approach to maintain the belief from $b[\lambda_k, \mathcal{X}_k]$ (6.8). With this method, we maintain $b[\lambda_k, \mathcal{X}_k]$ indirectly via a set of hybrid beliefs, each for a realization of classifier weights. The posterior class probabilities from each hybrid belief together represent the posterior classifier epistemic uncertainty. From there, we can compute marginal distributions for both λ_k and $\mathcal{X}_k$ if needed, e.g. when computing reward functions for planning (see Section 6.4). We present this approach first when observing a single object, then we extend it to multiple objects, and finally address computational complexity aspects.

Single Object

The belief $b[\lambda_k, \mathcal{X}_k]$, as defined by Eq. (6.8), is conditioned both on geometric measurements $\mathcal{H}_k^g$ and raw images $I_{1:k}$, and the classifier training set D. As discussed in Sec. 1.B, an epistemic uncertainty aware classifier provides us a cloud $\{l\gamma_{1:k}\}$, which we consider as semantic measurements. In the MH approach, we maintain $b[\lambda_k, \mathcal{X}_k]$ by splitting it to components by marginalizing over object classes and classifier weight realization $w \in W$, where W is a predetermined discrete set of w that are used throughout

the entire scenario. First, we marginalize $b[\lambda_k, \mathcal{X}_k]$ over w:

$$\begin{aligned} b[\lambda_k, \mathcal{X}_k] &= \int_w \mathbb{P}(\mathcal{X}_k, \lambda_k | I_{1:k}, \mathcal{H}_k^g, w) \cdot \mathbb{P}(w|D) dw \\ &\approx \frac{1}{|W|} \sum_w \mathbb{P}(\mathcal{X}_k, \lambda_k | I_{1:k}, \mathcal{H}_k^g, w). \end{aligned} \tag{6.17}$$

Then, using chain rule yields

$$b[\lambda_k, \mathcal{X}_k] \approx \frac{1}{|W|} \sum_w \mathbb{P}(\mathcal{X}_k | \lambda_k, w, I_{1:k}, \mathcal{H}_k^g) \cdot \mathbb{P}(\lambda_k | w, I_{1:k}, \mathcal{H}_k^g). \tag{6.18}$$

Each term in the right-hand side of the above is addressed separately; $\mathbb{P}(\mathcal{X}_k | \lambda_k, w, I_{1:k}, \mathcal{H}_k^g)$ is marginalized over c and using chain-rule can be split into the following distributions:

$$\begin{aligned} \mathbb{P}(\mathcal{X}_k | \lambda_k, w, I_{1:k}, \mathcal{H}_k^g) = &\sum_c \mathbb{P}(\mathcal{X}_k | c, \lambda_k, w, I_{1:k}, \mathcal{H}_k^g) \\ &\cdot \mathbb{P}(c | \lambda_k, w, I_{1:k}, \mathcal{H}_k^g) \end{aligned} \tag{6.19}$$

$\mathcal{X}_k$ is conditioned on c, thus λ_k can be omitted. For c, given the posterior probability vector λ_k, the rest can be omitted, and $\mathbb{P}(c|\lambda_k) = \lambda_k^c$ where λ_k^c is the element c of λ_k.

λ_k is a function of w, $I_{1:k}$, and $\mathcal{H}_k^g$; therefore, $\mathbb{P}(\lambda_k | w, I_{1:k}, \mathcal{H}_k^g)$ is a Dirac function $\delta(\cdot)$, such that

$$\mathbb{P}(\lambda_k | w, I_{1:k}, \mathcal{H}_k^g) = \delta(\lambda_k - \lambda_{k,w}). \tag{6.20}$$

As such, Eq. (6.18) is rewritten as:

$$b[\lambda_k, \mathcal{X}_k] \approx \frac{1}{|W|} \sum_c \sum_w \underbrace{\mathbb{P}(\mathcal{X}_k | c, l\gamma_{1:k,w}, \mathcal{H}_k^g)}_{b_w^c[\mathcal{X}_k]} \cdot \lambda_k^c \cdot \delta(\lambda_k - \lambda_{k,w}), \tag{6.21}$$

where $b_w^c[\mathcal{X}_k]$ is the continuous belief conditioned on w and c. Each $w \in W$ is constant throughout the scenario with the reasoning of keeping the number of particles constant, thus avoiding managing an exponentially increasing number of components (such as in chapter 5). This can be achieved by, e.g., training multiple models on the same dataset via bootstrapping, or re-using classifier weight sets created by MC-dropout. That way, Eq. (6.21) shows that maintaining $b[\lambda_k, \mathcal{X}_k]$ is equivalent to maintaining $b_w^c[\mathcal{X}_k]$ and λ_k^c for all $w \in W$ and c.

Each class probability $\lambda_{k,w}^c$ within the particle $\lambda_{k,w}$ is updated using Bayes rule as follows:

$$\lambda_{k,w}^c = \eta \cdot \lambda_{k-1,w}^c \cdot \mathbb{P}(z_k^g, l\gamma_{k,w} | c), \tag{6.22}$$

where η is a normalizing constant such that $\sum_c \lambda_k^c = 1$, and does not affect inference. As our classifier model is viewpoint dependent (Eq. (6.15)), we must marginalize

$\mathbb{P}(z_k^g, l\gamma_{k,w}|c)$ over x^o and x_k to fully utilize our models as follows:

$$\lambda_{k,w}^c \propto \lambda_{k-1,w}^c \int_{x^o,x_k} \mathcal{L}_k \cdot b_w^{c-}[x^o, x_k] dx^o dx_k, \tag{6.23}$$

where $b_w^{c-}[x^o, x_k]$ is the propagated conditional continuous belief, constructed as follows, as we marginalize out all other variables from $\mathcal{X}_k$ beside x^o and x_k:

$$b_w^{c-}[x^o, x_k] \triangleq \int_{\mathcal{X}_k/x^o,x_k} \mathcal{M}_k \cdot b_w^c[\mathcal{X}_{k-1}] d(\mathcal{X}_k/x^o, x_k). \tag{6.24}$$

$b_w^c[\mathcal{X}_k]$ from (6.21) is incrementally updated using standard SLAM state of the art approaches (e.g. iSAM2 [36]):

$$b_w^c[\mathcal{X}_k] \propto b_w^c[\mathcal{X}_{k-1}] \cdot \mathcal{M}_k \cdot \mathcal{L}_k. \tag{6.25}$$

Essentially, for every w, we maintain a hybrid belief over robot and object poses, and classes, which we define as:

$$hb_w[\mathcal{X}_x, c] \triangleq b_w^c[\mathcal{X}_k] \cdot \lambda_{k,w}^c, \tag{6.26}$$

and using the above definition, and considering that the Dirac function only "blocks" all λ_k except for $\lambda_{k,w}$, we can rewrite Eq. (6.21) in terms of $hb_w[\mathcal{X}_x, c]$:

$$b[\lambda_k, \mathcal{X}_k] \approx \frac{1}{|W|} \sum_c \sum_w hb_w[\mathcal{X}_k, c] \cdot \delta(\lambda_k - \lambda_{k,w}). \tag{6.27}$$

Practically, for every $w \in W$, we maintain $b_w^c[\mathcal{X}_k]$ with the accompanying $\lambda_{k,w}^c$ for every object class realization, overall maintaining $|W|$ hybrid beliefs $hb_w[\mathcal{X}_k, c]$ in parallel.

Further, one may require to infer the marginals $b[\lambda_k]$ or $\mathbb{P}(\mathcal{X}_k|I_{1:k}, \mathcal{H}_k^g, D)$, e.g. to compute an appropriate reward function, as we shall see in Section 6.4. We can describe $b[\lambda_k]$ in term of $\lambda_{k,w}$ particles by marginalizing $b[\lambda_k]$ over w:

$$\begin{aligned} b[\lambda_k] &\approx \frac{1}{|W|} \sum_w \mathbb{P}(\lambda_k|w, I_{1:k}, \mathcal{H}_k^g) \\ &= \frac{1}{|W|} \sum_w \mathbb{P}(\lambda_k|l\gamma_{1:k,w}, \mathcal{H}_k^g) \\ &= \frac{1}{|W|} \sum_w \delta(\lambda_k - \lambda_{k,w}). \end{aligned} \tag{6.28}$$

On the other hand, to compute $\mathbb{P}(\mathcal{X}_k|I_{1:k}, \mathcal{H}_k^g, D)$, we marginalize over w and c,

$$\mathbb{P}(\mathcal{X}_k|I_{1:k}, \mathcal{H}_k^g, D) \approx \frac{1}{|W|} \sum_w \sum_c hb_w[\mathcal{X}_k, c], \tag{6.29}$$

utilizing the already-calculated individual hybrid beliefs $hb_w[\mathcal{X}_k, c]$.

While theoretically this kind of maintenance is computationally expensive, in prac-

tice many class realizations can be with probability close to zero, allowing us to prune $b^c_w[\mathcal{X}_k]$ with its conditional $\lambda^c_{k,w}$ if needed (see e.g. [114]). In this chapter we set a fixed lower limit on $\lambda^c_{k,w}$ and remove the corresponding component if the value of $\lambda^c_{k,w}$ is lower than said limit. In total, $|W|$ hybrid beliefs are maintained to infer $\lambda_{k,w}$ for each w.

Multiple Objects

We now extend our formulation to consider the environment includes multiple objects observed by the robot. Let us introduce some notations to support this extension. First, we denote variables corresponding to object o with a superscript $\square^o$. At time k a robot may observe a subset of n_k objects within the environment, and up until time k, N_k objects. The subset of n_k objects is denoted as O_k. Each object is segmented from the image and the classifier outputs $\{\gamma^o_{k,w}\}_{w\in W}$ corresponding to said object, and the set of all those clouds for all n_k objects in O_k is denoted as $\{\Gamma_k\}$ with Γ defined as a realization of γ measurements, one per each observation. For a specific $w \in W$, we define the realization of γ measurements as $\Gamma_{k,w} \triangleq \{\gamma^o_{k,w}\}_{o\in O_k}$, thus $\{\Gamma_k\} \triangleq \{\Gamma_{k,w}\}_{w\in W}$. We define $l\Gamma_k$ as the logit transformation of all $\gamma_k \in \Gamma_k$ as in Eq. (6.14). The set of all geometric measurements at time k is denoted Z^g_k, the history $\mathcal{H}^g_k \triangleq \{a_{0:k-1}, Z^g_k\}$ includes all geometric measurements and actions up until time k, and subsequently $\mathcal{H}_k \triangleq \{I_{1:k}, \mathcal{H}^g_k\}$ includes all measurement and action history up to time k.

We define the joint posterior class probability vector as:

$$\Lambda_k \triangleq \mathbb{P}(C|l\Gamma_{1:k}, \mathcal{H}^g_k), \tag{6.30}$$

where $C \triangleq \{c^o\}_{o\in O_{1:k}}$ is the class realization of all objects observed up to time k, with c^o being the o-th object class. In addition, we include in $\mathcal{X}_k$ the poses of all the objects, such that $\mathcal{X}_k \triangleq x_{0:k} \cup \{x^o\}_{o\in O_{1:k}}$. Subsequently, the belief over Λ_k and $\mathcal{X}_k$ is:

$$b[\Lambda_k, \mathcal{X}_k] \triangleq \mathbb{P}(\Lambda_k, \mathcal{X}_k|I_{1:k}, \mathcal{H}^g_k, D). \tag{6.31}$$

Observe that Λ_k is still a probability vector, but with m^{N_k} possible categories. To illustrate this, consider an example with two objects and three candidate classes, i.e. $O_{1:k} = \{o, o'\}$ and $m = 3$. Then each category contains a class hypothesis for all object classes, e.g. $c^o = 1$, $c^{o'} = 3$. As such, there are 9 possible class realizations and therefore Λ_k has 9 categories whose probabilities should sum to one. That way, the number of categories in Λ_k grows exponentially with the number of objects, potentially to intractable levels. Fortunately, this can be mitigated by pruning components with low probability, as was done in chapters 3 and 4.

For every $w \in W$, updating Λ_k is largely similar to updating λ_k in Sec. 3.B, except for a few differences. The likelihood terms include all objects O_k observed at time k, and the conditional probability over the poses is conditioned on class realization C

instead of the class of a single object,

$$\Lambda^C_{k,w} \propto \Lambda^C_{k-1,w} \int_{x^{inv}_k} \mathcal{L}_k \cdot b^{C-}_w[\mathcal{X}^{inv}_k] dx^{inv}_k, \tag{6.32}$$

where $\Lambda^C_{k,w}$ denotes the posterior probability of class realization C at time k for weight realization w, and $\mathcal{X}^{inv}_k$ represents the last robot pose and all poses of objects observed at time k, i.e. $\mathcal{X}^{inv}_k \triangleq x_k \cup \{x^o\}_{o \in O_k}$. Likelihood $\mathcal{L}_k$ now encompasses all the measurement likelihoods of all objects as follows:

$$\mathcal{L}_k \triangleq \prod_{o \in O_{k,o}} \mathbb{P}(l\gamma^o_k | C, x^o, x_k) \cdot \mathbb{P}(z^g_k | x^o, x_k). \tag{6.33}$$

The belief $b^{C-}_w[\mathcal{X}^{inv}_k]$ is the propagated belief conditioned on C, and marginalized over the uninvolved variables such that $\mathcal{X}_k = \mathcal{X}^{inv}_k \cup \mathcal{X}^{\neg inv}_k$:

$$b^{C-}_w[\mathcal{X}^{inv}_k] = \int_{\mathcal{X}^{\neg inv}_k} \mathcal{M}_k \cdot b^C_w[\mathcal{X}_{k-1}] d\mathcal{X}^{\neg inv}_k. \tag{6.34}$$

Similarly to Sec. 3.B we can rewrite $b[\Lambda_k, \mathcal{X}_k]$ as:

$$b[\Lambda_k, \mathcal{X}_k] \approx \frac{1}{|W|} \sum_w \sum_C hb_w[\mathcal{X}_k, C] \cdot \delta(\Lambda_k - \Lambda_{k,w}), \tag{6.35}$$

where $hb_w[\mathcal{X}_k, C]$ is the hybrid belief conditioned on w:

$$hb_w[\mathcal{X}_k, C] \triangleq b^C_w[\mathcal{X}_k] \cdot \Lambda^C_{k,w}, \tag{6.36}$$

and $b^C_w[\mathcal{X}_k] \triangleq \mathbb{P}(\mathcal{X}_k | c, l\gamma_{1:k,w}, \mathcal{H}^g_k)$. Similarly to Eq. (6.27), maintaining $b[\Lambda_k, \mathcal{X}_k]$ is equivalent to maintaining $hb_w[\mathcal{X}_k, C]$ for all $w \in W$ and C. In case $b[\Lambda_k]$ is required, for the multi-object case Eq. (6.28) becomes:

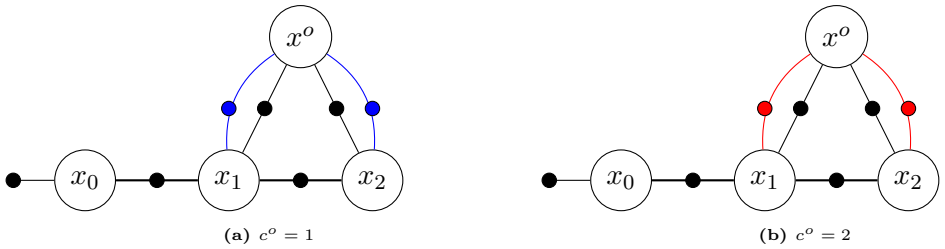
$$b[\Lambda_k] \approx \frac{1}{|W|} \sum_w \delta(\Lambda_k = \Lambda_{k,w}); \tag{6.37}$$

Similarly; In case $\mathbb{P}(\mathcal{X}_k | I_{1:k}, \mathcal{H}^g_k, D)$ is required, for the multi-object case Eq. (6.29) becomes:

$$\mathbb{P}(\mathcal{X}_k | I_{1:k}, \mathcal{H}^g_k, D) \approx \frac{1}{|W|} \cdot \sum_w \sum_C hb_w[\mathcal{X}_k, C]. \tag{6.38}$$

In general, all $\mathcal{X}_k$ and C are coupled, and subsequently so do $\mathcal{X}_k$ and Λ_k. There are two possible sources of coupling: class priors that depend on other objects' classes (e.g. a computer mouse may be expected to appear next to a monitor), and the coupling between poses and classes induced by the viewpoint-dependent classifier uncertainty model (6.15).

Specifically, if the classifier model is not viewpoint dependent, i.e. $\mathbb{P}(l\gamma_k | c, \mathcal{X}_k) = \mathbb{P}(l\gamma_k | c)$, then $\mathbb{P}(\mathcal{X}_k, c | \mathcal{H}_k) = \mathbb{P}(\mathcal{X}_k | \mathcal{H}_k) \cdot \mathbb{P}(c | \mathcal{H}_k)$, and each one can be maintained



(a) $c^o = 1$ **(b)** $c^o = 2$

Figure 6.4: Factor graphs for a toy scenario where the camera observes an object for a specific w,there are $|W|$ such factor graph pairs. The object has two candidate classes. Each dot and line represents a separate factor. The black factors between the camera and object represent the geometric model, while the colored factors represent the classifier models, $c = 1$ and $c = 2$ by blue and red respectively.

separately. This simplified case can be represented as a single factor graph for the continuous variables, and the discrete variables are maintained via $\mathbb{P}(c|\mathcal{H}_k) \propto \mathbb{P}(c|\mathcal{H}_{k-1}) \cdot \mathbb{P}(l\gamma_k|c)$.

However, in our case, the viewpoint-dependent model $\mathbb{P}(l\gamma_k|c, \mathcal{X}_k)$ couples between relevant continuous and discrete variables; specifically, it is represented as a factor between robot and object poses at time steps when the object is observed. Thus, for $l\gamma_k$ that corresponds to a semantic observation of some object o at time instant k, the factor is $\mathbb{P}(l\gamma_k|c, x_k, x^o)$, and, according to (6.15), it differs for each class realization c. This is represented by multiple factor graphs as illustrated in a simple example in Fig. 6.4. Between the graphs, the topology is identical, but the factor $\mathbb{P}(l\gamma_k|c, \mathcal{X}_k)$ changes according to class c hypothesis. Further, each classifier weight $w \in W$ corresponds to its own instance of those factor graphs. Of course, if a given factor graph is connected, all variables in it are coupled.

Remark: While here we described a straightforward extension of the MH approach to the multi-object, its worst-case computational complexity (discussed in Section 3.B) scales poorly with the number of objects. One could also consider maintaining a marginal distribution for each object (e.g. $b[\lambda_k^o, x^o]$) instead of the joint distribution (6.35); yet, without introducing approximations, this would still involve inferring poses via (6.38), as all object poses and classes are dependent as discussed in Sec. 3.B. The marginal poses, i.e. Eq. (6.38), requires the maintenance of all $hb_w[\mathcal{X}_k, C]$ as maintaining $b[\Lambda_k, \mathcal{X}_k]$, thus the computational time and memory complexity does not change. Computing a marginal distribution for each object is outside the scope of this chapter and might be addressed in future work.

Computational Complexity and Discussion

With m candidate classes, and n objects, the number of possible class realizations per $\Lambda_{k,w}$ is m^n, as C considers all possible class realizations of all objects observed thus far, making C combinatorial in nature. Inference over continuous states (k camera poses and n object poses), i.e. the conditional belief $\mathbb{P}(\mathcal{X}_k|C, l\gamma_{1:k}, \mathcal{H}_k^g)$, can be efficiently done using, e.g. the state of the art iSAM2 approach [36], with a computational complexity of $O((k+n)^{1.5})$, and at worst $O((k+n)^3)$ for loop closures. To compute an individual λ_k

particle, we must account for the attached $\mathbb{P}(\mathcal{X}_k|C, l\gamma_{1:k}, \mathcal{H}_k^g)$, and thus the worst-case computational complexity is $O(m^n(k+n)^3)$. Eventually we maintain $|W|$ particles, and therefore the overall time computational complexity for $b[\lambda_k]$ inference is $O(|W|m^n(k+n)^3)$ at worst without pruning.

The computational complexity can be further reduced by pruning e.g. low probability classes for individual particles, but as with any pruning this can induce a problem where a certain realization gets "locked" in either probability 0 or 1, rendering the possibility of probability changing for the said realization impossible.

For memory complexity, we require the latest robot pose, and the n poses and m sized class probability vector for every object per λ particle. All in all, we must maintain $O(nm^n)$ random variables in memory per particle, and $O(|W|nm^n)$ variables for $b[\lambda]$. This also can be reduced by e.g. pruning low probability class realizations or incremental inference methods.

To conclude, while accurate„ due to the combinatorial nature of C which considers all possible class realizations, the need to simultaneously maintain $|W|$ hybrid beliefs, the worst-case complexity of MH scales poorly with number of objects and candidate classes. In practice, pruning class realizations with low probability can reduce computational complexity to manageable levels. Incremental inference approaches for hybrid beliefs, inline with [56], could further reduce computational complexity. These, however, are outside the scope of this chapter.

As an alternative, in the next section we propose the JLP algorithm, which is by far computationally more efficient than MH.

3.C Joint Lambda Pose Inference

In this subsection we present an alternative approach for inference, which maintains a joint belief over $\mathcal{X}_k$ and λ_k. This approach is significantly less computationally expensive than the Multi-Hybrid approach. Its accuracy depends on conditions that we discuss below. This approach is denoted as Joint Lambda Pose (JLP). Similarly to MH, we first consider the single object case, and then extend JLP to the multiple object case.

To the best of our knowledge, there are no approaches that combine Gaussian distributed variables with random variables within a simplex besides sampling based methods. Thus, we cannot maintain $b[\lambda_k, \mathcal{X}_k]$ as a single continuous belief, e.g. MH requires maintaining multiple hybrid beliefs, as discussed in Sec. 3.B.

Instead, we define $l\lambda_k$ as the logit transformation of λ_k, and maintain the belief (considering a single object, for now):

$$b[l\lambda_k, \mathcal{X}_k] \triangleq \mathbb{P}(l\lambda_k, \mathcal{X}_k|I_{1:k}, \mathcal{H}_k^g, D). \tag{6.39}$$

For a general λ_k, each λ_k^c can be updated using Bayes rule:

$$\lambda_k^c = \eta \cdot \lambda_{k-1}^c \cdot \mathbb{P}(l\gamma_k|c, x_k^{rel}). \tag{6.40}$$

When λ_k is cast into logit space, the above equation transforms into the following sum, written in a vector form:

$$l\lambda_k = l\lambda_{k-1} + l\mathcal{L}_k^s, \tag{6.41}$$

where for each element in λ_k, the normalizer η gets canceled as it is identical for all elements of λ_k, and $l\mathcal{L}_k^s$ is defined as:

$$\begin{aligned} l\mathcal{L}_k^s \triangleq \Bigg[&\log\left(\frac{\mathbb{P}(l\gamma_k|c=1,x_k^{rel})}{\mathbb{P}(l\gamma_k|c=m,x_k^{rel})}\right), ..., \\ &\log\left(\frac{\mathbb{P}(l\gamma_k|c=m-1,x_k^{rel})}{\mathbb{P}(l\gamma_k|c=m,x_k^{rel})}\right)\Bigg]^T. \end{aligned} \tag{6.42}$$

To recursively update a Gaussian $l\lambda_k$ in closed form from a Gaussian $l\lambda_{k-1}$, $l\mathcal{L}_k^s$ needs to be Gaussian as well. We now discuss conditions for which $l\mathcal{L}_k^s$ is indeed Gaussian.

Accuracy Conditions

In this section we analyze the condition under which $l\mathcal{L}_k^s$ is accurately Gaussian distributed. This is formulated in the following Lemma:

Lemma 6.3.1. *Given m Gaussian distributed viewpoint-dependent classifier uncertainty models $\mathbb{P}(l\gamma_k|c,x_k^{rel})$ as in Eq. (6.15), if $\Sigma_{c=i}(x_k^{rel}) \equiv \Sigma_{c=j}(x_k^{rel})\ \forall i,j \in [1,m]$, then $l\mathcal{L}_k^s$ is Gaussian distributed.*

Proof In this proof, we will omit time index k and sometimes omit x_k^{rel} from h_c and Σ_c to reduce clutter. We prove by construction, with writing the PDF of the classifier uncertainty model for the i-th element of $l\mathcal{L}^s$. The model for class i has an expectation $h_{c=i}(x^{rel})$ and a covariance matrix $\Sigma_{c=i}(x^{rel})$. Thus:

$$\begin{aligned} l\mathcal{L}_i^s &= \log\left(\frac{(2\pi)^{\frac{m-1}{2}}\sqrt{|\Sigma_{c=m}|}e^{-\frac{1}{2}||l\gamma-h_{c=i}||^2_{\Sigma_{c=i}}}}{(2\pi)^{\frac{m-1}{2}}\sqrt{|\Sigma_{c=i}|}e^{-\frac{1}{2}||l\gamma-h_{c=m}||^2_{\Sigma_{c=m}}}}\right) \\ &= -\frac{1}{2}\log(|\Sigma_{c=i}|) + \frac{1}{2}\log(|\Sigma_{c=m}|) \\ &\quad - \frac{1}{2}||l\gamma-h_{c=i}||^2_{\Sigma_{c=i}} + \frac{1}{2}||l\gamma-h_{c=m}||^2_{\Sigma_{c=m}}. \end{aligned} \tag{6.43}$$

Now, applying the condition $\Sigma_{c=i}(x^{rel}) \equiv \Sigma_{c=j}(x^{rel})$, and denoting both as Σ_c we get the following expression:

$$\begin{aligned} l\mathcal{L}_i^s =& l\gamma^T\Sigma_c^{-1}h_{c=i} - \frac{1}{2}h_{c=i}^T\Sigma_c^{-1}h_{c=i} \\ &- l\gamma^T\Sigma_c^{-1}h_{c=m} + \frac{1}{2}h_{c=m}^T\Sigma_c^{-1}h_{c=m}. \end{aligned} \tag{6.44}$$

From the above equation, if $l\gamma$ is a multi-variate Gaussian random variable, then $l\mathcal{L}_i^s$ is a linear combination of Gaussian random variables, therefore Gaussian by itself. This is valid for every $i \in [1, m-1]$. ■

In general, the classifier model covariance functions may not be equivalent; Therefore, Eq. (6.43) includes a quadratic expression of $l\gamma$, making $l\mathcal{L}_i^s$ a mixture of Gaussian and Generalized Chi distributions. To counter this, the models' covariances must be "close" to each other to approximately describe $l\mathcal{L}_i^s$ as a Gaussian.

If $l\mathcal{L}^s$ is assumed Gaussian via moment matching or other methods, it will only approximate the true distribution of $l\mathcal{L}^s$ with the accuracy dependent on the "distance" between $\Sigma_{c=i}(x^{rel})$ and $\Sigma_{c=j}(x^{rel})$ for all $i, j \in [1, m]$. This distance can be represented by, for example, Forbenius Norm.

Remark: The Forbenius norm can be inserted into the loss function while training the viewpoint-dependent classifier uncertainty models (6.15), thereby enforcing sufficiently close covariance functions between different models such that the approach presented in this section can be used. Our implementation utilizes this concept, as we further explain in Sec. 5.C.

Having discussed conditions for $l\mathcal{L}^s$ to be Gaussian (accurately or approximately), in the following section we introduce a new factor, termed joint Lambda pose (JLP) factor, which constructs $b[l\lambda_k, \mathcal{X}_k]$.

Joint Lambda Pose (JLP) Factor

Assume the conditions in Lemma 6.3.1 are satisfied. Considering Eq. (6.44) for all $i \in [1, m-1]$, we can describe $l\mathcal{L}_k^s$ as follows:

$$l\mathcal{L}_k^s = \Phi l\gamma_k - \frac{1}{2}\phi, \tag{6.45}$$

where the matrix $\Phi \in \mathbb{R}^{(m-1)\times(m-1)}$ and the vector $\phi \in \mathbb{R}^{m-1}$ depend on the individual classifier models (6.15) and x_k^{rel}. Using Eq. (6.44), the matrix Φ is defined as

$$\Phi \triangleq \begin{bmatrix} h_{c=1}^T \Sigma_{c=1}^{-1} - h_{c=m}^T \Sigma_{c=m}^{-1} \\ \vdots \\ h_{c=m-1}^T \Sigma_{c=m-1}^{-1} - h_{c=m}^T \Sigma_{c=m}^{-1} \end{bmatrix}, \tag{6.46}$$

and ϕ is defined as

$$\phi \triangleq \begin{bmatrix} h_{c=1}^T \Sigma_{c=1}^{-1} h_{c=1} - h_{c=m}^T \Sigma_{c=m}^{-1} h_{c=m} \\ \vdots \\ h_{c=m-1}^T \Sigma_{c=m-1}^{-1} h_{c=m-1} - h_{c=m}^T \Sigma_{c=m}^{-1} h_{c=m} \end{bmatrix}. \tag{6.47}$$

If the conditions of Lemma 6.3.1 are satisfied, we can substitute $l\mathcal{L}_k^s$ in Eq. (6.41) with the expression in Eq. (6.45):

$$l\lambda_k = l\lambda_{k-1} + \Phi l\gamma_k - \frac{1}{2}\phi. \tag{6.48}$$

Now, as $l\gamma_k$ is assumed Gaussian, its distribution is defined by expectation $\mathbb{E}(l\gamma_k)$ and covariance $\Sigma(l\gamma_k)$. Assuming a non-singular matrix Φ, we define the JLP factor as:

$$\begin{aligned}\mathbb{P}(l\lambda_k|l\lambda_{k-1}, I_k, D, x_k^{rel}) \triangleq \\ \mathcal{N}\left(l\lambda_{k-1} + \Phi\mathbb{E}(l\gamma_k) - \tfrac{1}{2}\phi, \Phi\Sigma(l\gamma_k)\Phi^T\right),\end{aligned} \tag{6.49}$$

As mentioned before, we utilize a classifier that outputs a set $\{\gamma_k\}$ instead of a single γ_k. Each $\gamma_k \in \{\gamma_k\}$ is then transformed via the logit transformation (6.14) to $l\lambda_k$, thus the entire set $\{\gamma_k\}$ is transformed to $\{l\gamma_k\}$. From there $\mathbb{E}(l\gamma_k)$ and $\Sigma(l\gamma_k)$ are inferred, and the JLP factor can be written as $\mathbb{P}(l\lambda_k|l\lambda_{k-1}, \{l\gamma_k\}, x_k^{rel})$. As in Sec. 3.B, $\{l\gamma_k\}$ represents the classifier's epistemic uncertainty.

The factor (6.49) is a four variable factor of $l\lambda_k$, $l\lambda_{k-1}$, x^o, and x, with the latter two used to compute x^{rel} via $x^{rel} \triangleq x^o \ominus x$. The factor can be inserted into a graph structure that can be optimized using standard SLAM methods, where $l\lambda_k$ for different k are separate variable nodes. This factor enables us to maintain $b[l\lambda_k, \mathcal{X}_k]$ using a single continuous belief as we discuss in the next section, and in turn be faster computationally than MH.

The term $\Phi\Sigma(l\gamma_k)\Phi^T$ is positive definite when Φ is not singular, but in practice we cannot guarantee this condition. If there is some x_k^{rel} for classes $c = i$ and $c = j$ where $h_{c=i}(x^{rel}) = h_{c=j}(x^{rel})$ and $\Sigma_{c=i}(x^{rel}) = \Sigma_{c=j}(x^{rel})$, then at that point Φ is singular. This means: at that certain x_k^{rel}, we cannot differentiate between the two classes with the given classifier models. To keep $\Phi\Sigma(l\gamma_k)\Phi^T$ non-singular, we add to it an identity matrix multiplied by a small positive constant $\epsilon \cdot I^{(m-1)\times(m-1)}$.

Recursive Update Formulation

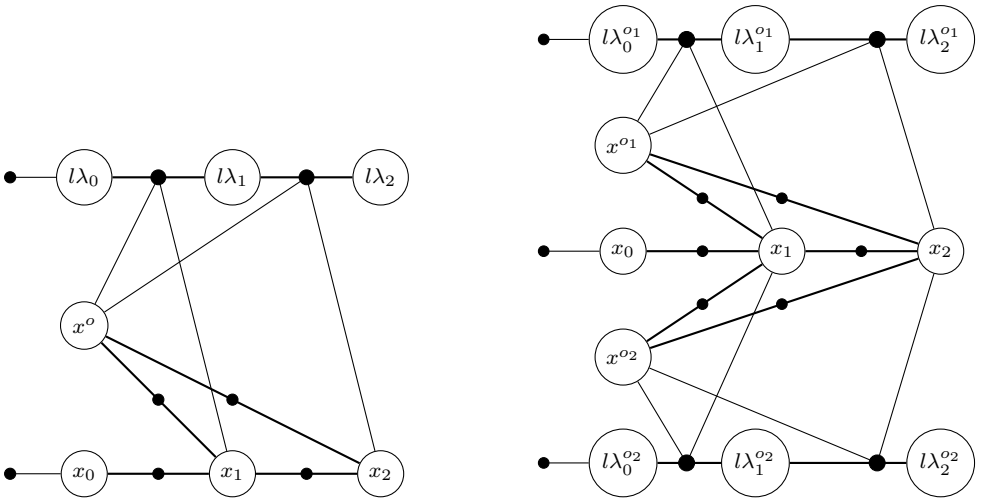
In Section 3.C we introduced a novel four variable factor[1] $\mathbb{P}(l\lambda_k|l\lambda_{k-1}, I_k, D, x_k^{rel})$, denoted as the JLP factor. This factor allows to update $b[l\lambda_{1:k}, \mathcal{X}_k]$ as a single continuous belief, instead of multiple conditioned ones as with MH.

We may consider a smoothing formulation where we maintain the joint belief $b[l\lambda_{1:k}, \mathcal{X}_k]$. Using Bayes and chain rules, $b[l\lambda_{1:k}, \mathcal{X}_k]$ can then be updated as:

$$\begin{aligned}b[l\lambda_{1:k}, \mathcal{X}_k] =& \eta \cdot \mathbb{P}(l\lambda_k|l\lambda_{k-1}, I_k, D, x_k^{rel}) \cdot \mathcal{M}_k \cdot \\ & \cdot \mathbb{P}(z_k^g|x_k^{rel}) \cdot b[l\lambda_{1:k-1}, \mathcal{X}_{k-1}],\end{aligned} \tag{6.50}$$

where η is a normalization constant. In practice, previous $l\lambda_{1:k-1}$ are typically not required for classification inference and planning, so we can consider the belief $b[\lambda_k, \mathcal{X}_k]$, without maintaining a large number of states per object. To update this belief recursively, we must express it as a function of the prior $b[l\lambda_{k-1}, \mathcal{X}_{k-1}]$. To do so, we

[1] In case the object is not observed at $k-1$, instead of $l\lambda_{k-1}$ we connect the factor to the latest previous $l\lambda$.



(a) A single object. **(b)** Two objects.

Figure 6.5: Example of a factor graph for a scenario until $k = 2$, where at each time step an object **(a)** or objects **(b)** are observed. There are priors for x_0, and $l\lambda_0$ for every object. Between the camera poses are motion factors, connecting camera and object poses are geometric measurement factors, and between $l\lambda$'s at different time steps the 4 variable factors are connecting.

marginalize over $l\lambda_{k-1}$ and use the Bayes rule:

$$\begin{aligned} b[l\lambda_k, \mathcal{X}_k] \propto \int_{l\lambda_{k-1}} & \mathbb{P}(l\lambda_k | l\lambda_{k-1}, I_k, D, x_k^{rel}) \cdot \mathcal{M}_k \cdot \\ & \cdot \mathbb{P}(z_k^g | x_k^{rel}) \cdot b[l\lambda_{k-1}, \mathcal{X}_{k-1}] dl\lambda_{k-1}. \end{aligned} \tag{6.51}$$

Fig. 6.5a presents a simple example to illustrate the factor graph structure using JLP. In this figure, we present a scenario with two time steps in which the robot observes a single object.

Multiple Objects

The extension to multiple objects within the JLP framework is straight-forward. Each object o has its own set of $l\lambda_o$ nodes, as seen in the example in Fig. 6.5b. The set of all $l\lambda_k^o$ for objects observed thus far is denoted as $l\bar{\lambda}_k \triangleq \{l\lambda_k^o\}_{o \in O_{1:k}}$. In contrast with Λ_k, which is a single probability vector over class realization with m^{N_k} categories, $l\bar{\lambda}_k$ is a set of vectors with $m-1$ elements each, being the logit transformation of a probability vector of an object, to a total of $(m-1) \cdot N_k$ elements for $l\bar{\lambda}_k$. As such, the joint belief is updated in a similar manner to the single object case:

$$\begin{aligned} b[l\bar{\lambda}_k, \mathcal{X}_k] \; = \; \eta \int_{l\bar{\lambda}_{k-1}} & \prod_{o \in O_k} \mathbb{P}(l\lambda_k^o | l\lambda_{k-1}^o, \{l\gamma_k^o\}, x_k^{rel}) \cdot \mathcal{M}_k \cdot \\ & \cdot \mathbb{P}(Z_k^g | \mathcal{X}_k) \cdot b[l\bar{\lambda}_{k-1}, \mathcal{X}_{k-1}] dl\bar{\lambda}_{k-1}, \end{aligned} \tag{6.52}$$

with η being a normalization constant that does not participate in inference.

With this formulation, the difference between maintaining $b[\Lambda_k, \mathcal{X}_k]$ in MH and $b[l\bar{\lambda}_k, \mathcal{X}_k]$ must be discussed.

Computation Complexity and Discussion

MH maintains Λ_k, which is a posterior joint probability vector for all class realizations, with m^{N_k} categories as seen in Sec. 3.B. Thus Λ_k grows exponentially with the number of objects observed, and considering that the inference is done for $|W|$ times, each w with its own Λ_k, MH is intractable unless pruning methods are applied. On the other hand, JLP maintains $\bar{l\lambda}_k$ which essentially maintains a separate $l\lambda_k$ per object, resulting in size of $(m-1)\cdot N_k$, which grows linearly with the number of objects, and results in better scaling. Moreover, the inference is done only a single time, as the set W only plays a role in classifier output γ.

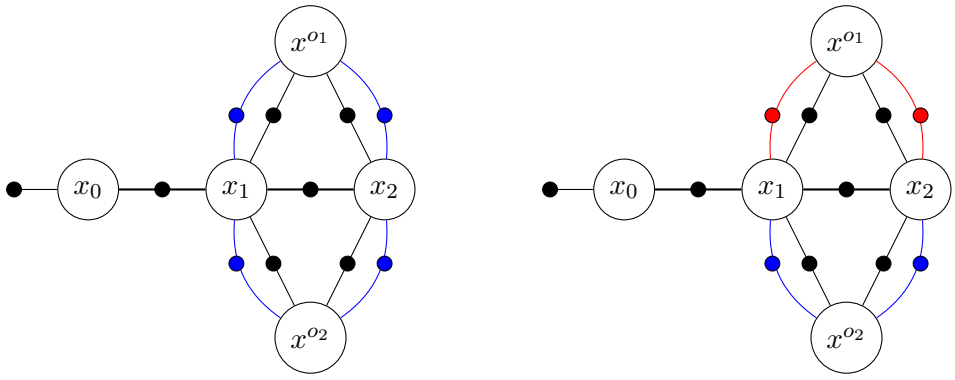
If we consider again the example scenario with two objects and three candidate classes, Λ_k is a probability vector with 9 categories. On the other hand, $\bar{l\lambda}_k = \{l\lambda_k^o, l\lambda_k^{o'}\}$ where $l\lambda_k^o$ and $l\lambda_k^{o'}$, each, is a vector with two elements as they are the logit transformation of λ_k^o and $\lambda_k^{o'}$, respectively, totaling in 4 elements for $\bar{l\lambda}_k$.

JLP has a single continuous belief with at most k pose states and n object pose states. In addition each object has $l\lambda$ with $m-1$ variables. Consider pose states with d variables each, at worst the total number of variables in JLP is $dk+dn+n\cdot(m-1)$. In total, the computational time complexity is $O\left((dk+dn+nm)^3\right)$ at worst. Compared to MH, as discussed in Sec. 3.B where the time computational complexity is at worst $O\left(|W|m^n(k+n)^3\right)$, JLP scales significantly better with the number of objects.

We can compare Fig. 6.6 and Fig. 6.5b for illustration of the difference between MH and JLP inference. In those figures, a scenario with two objects and two candidate is presented. Fig. 6.6 presents the factor graphs for a specific w, therefore in this case we have to maintain $4|W|$ factor graphs. On the other hand, with JLP we have to maintain a single one that also contains additional $l\lambda$ nodes.

As we can see, the advantage of the JLP approach compared to the MH approach is that it does not require maintaining multiple hybrid beliefs. It maintains a single continuous belief that encompasses all poses and object classes while reasoning about classifier epistemic uncertainty, allowing a rich, viewpoint dependent representation of object class probabilities.

The main disadvantage of this approach is the requirement of Lemma 6.3.1 to hold for accuracy. While the requirement can be offset by enforcing additional constrains on the training of classifier uncertainty models, using the resulting models will render JLP as approximation compared to MH. Another potential drawback is that JLP forces λ to be LG distributed, which, as we will see in Sec. 4.E, results in slower entropy computation. Despite of that, the advantage in computational efficiency of JLP is significant enough to offset slower entropy computation relative to MH, thus practically significantly more feasible.



(a) $c^{o1} = 1$, $c^{o2} = 1$ **(b)** $c^{o1} = 2$, $c^{o2} = 1$

(c) $c^{o1} = 1$, $c^{o2} = 2$ **(d)** $c^{o1} = 2$, $c^{o2} = 2$

Figure 6.6: Factor graphs used by MH with the same scenario as in Fig. 6.5b for a single w. Each dot and line represents a separate factor. The black factors between the camera and object represent the geometric model, while the colored factors represent the classifier models, $c = 1$ and $c = 2$ by blue and red respectively.

6.4 Approach- Planning

In this section we present a framework for epistemic uncertainty aware semantic BSP (EUS-BSP). Our framework incorporates reasoning about future posterior epistemic uncertainty within BSP; moreover, we appropriately generate future semantic and geometric observations while utilizing the coupling between λ and $\mathcal{X}$. Importantly, maintaining the corresponding future posterior belief $b[\lambda, \mathcal{X}]$ within BSP allows to utilize a variety of reward functions, and in particular, information-theoretic rewards over epistemic uncertainty. As such, EUS-BSP provides key capabilities for reliable autonomous semantic perception in uncertain environments.

Each of the inference approaches developed in Section 6.3 has its own BSP counterpart. As we discuss in detail below, they are not compatible with each other, i.e. MH planning must be used with inference, and the same for JLP.

This section is structured as follows; First, in Sec. 4.A, we discuss future measurement generation given candidate actions: For semantic measurements, we consider generating raw images, and then propose to generate semantic measurements directly from the viewpoint-dependent classifier uncertainty model from Eq. (6.15). Then, we detail the specifics of generating measurements from the model for MH in Sec. 4.B and JLP in Sec. 4.C. Afterwards, we discuss possible reward functions, first mentioning rewards in the form of $r(b[\mathcal{X}])$ and $r(\mathbb{E}(\lambda))$ in Sec. 4.D, both indirectly involving reasoning about epistemic uncertainty. Further, we discuss an epistemic uncertainty information-theoretic reward $r(b[\lambda])$ in Sec. 4.E, specifically the negative of differential entropy $-H(\lambda)$. We discuss computing $-H(\lambda)$ for both LG and Dirichlet distributed λ. For MH approach, λ can be distributed as either, but for the JLP approach, λ is limited to LG. Fig. 6.7 presents a diagram of all aspects considered in this section.

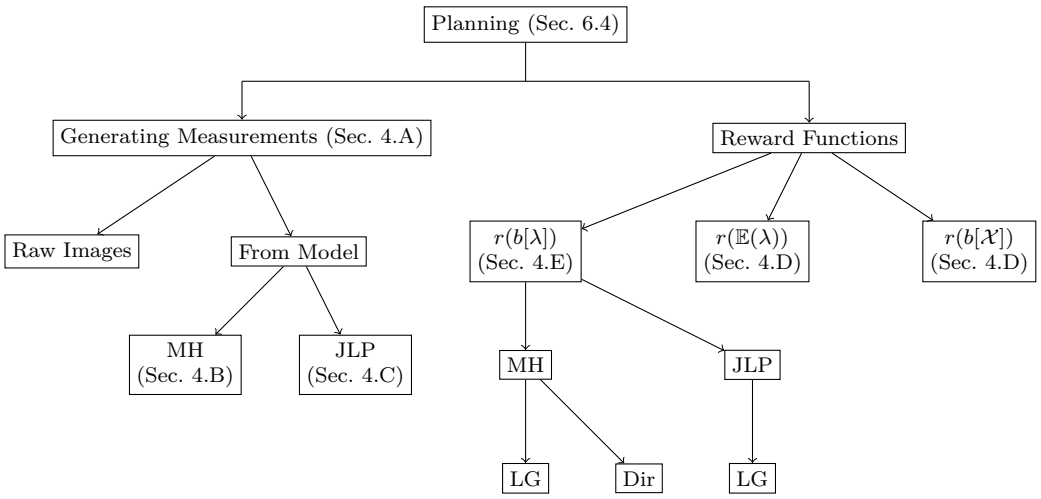


Figure 6.7: A diagram of aspects considered in Sec. 6.4. *Dir* stands for Dirichlet distribution.

4.A Measurement Generation

As part of the objective function (6.10) evaluation, we need to reason about future observations, both geometric and semantic. While geometric measurements can be sampled given through the geometric measurement model given sampled poses, the semantic measurement generation, especially when accounting for epistemic uncertainty is not immediate. For clarity, in this section we consider the single object case, while in the next sections we expand to the multiple object case in each method.

One alternative is to consider, for the ith look ahead step, generating $\{l\gamma_{k+i}\}$ by first predicting raw measurements, i.e. future images I_{k+i}. Given each such image, we can produce $\{l\gamma_{k+1}\}$ by forwarding I_{k+1} through a classifier for each $w \in W$, similarly to passive inference. In such a case, the objective function (6.10) becomes:

$$J(b[\lambda_k, \mathcal{X}_k], a_{k:k+L}) = \mathbb{E}_{I_{k+1:k+L}, z^g_{k+1:k+L}}(\sum_{i=1}^{l} r(b[\lambda_{k+i}, \mathcal{X}_{k+i}], a_{k+i})), \tag{6.53}$$

where

$$b[\lambda_{k+i}, \mathcal{X}_{k+i}] = \mathbb{P}(\lambda_{k+i}, \mathcal{X}_{k+i} | I_{k+1:k+i}, z^g_{k+1:k+i}, I_{1:k}, \mathcal{H}^g_k, D). \tag{6.54}$$

As presented in Sec. 1.D, we have to use a generative model for generating measurements with the general form of $\mathbb{P}(\mathcal{Z}_{k+1:k+L} | \mathcal{H}_k, a_k)$. In this case, it takes the form of $\mathbb{P}(I_{k+1:k+L}, z^g_{k+1:k+L} | \mathcal{H}_k, a_k)$, which is a generative model for generating raw images and geometric measurements.

This model generates images from a candidate viewpoint of a scene yet to be observed, given a set of environments it was trained on. While such works do exist (e.g. [115]), the problem is high dimensional and feasible only in specifically trained environments.

In contrast, we propose an alternative approach that generates semantic measurements directly via a learned viewpoint dependent classifier uncertainty model (6.15), thereby avoiding generating raw, high-dimensional images.

Specifically, we use the LG model presented in Eq. (6.16) for generating semantic measurements $z^s_{k+1:k+L}$ with the specifics discussed in Sec. 4.B and Sec. 4.C for MH and JLP respectively. Thus, as alternative to Eq. (6.53), the objective function (6.10) becomes:

$$J(b[\lambda_k, \mathcal{X}_k], a_{k:k+L}) = \mathbb{E}_{z^s_{k+1:k+L}, z^g_{k+1:k+L}}(\sum_{i=1}^{L} r(b[\lambda_{k+i}, \mathcal{X}_{k+i}], a_{k+i})), \tag{6.55}$$

where, as opposed to Eq. (6.54), $b[\lambda_{k+i}, \mathcal{X}_{k+i}]$ is conditioned on $z^s_{k+1:k+i}$, i.e.

$$b[\lambda_{k+i}, \mathcal{X}_{k+i}] = \mathbb{P}(\lambda_{k+i}, \mathcal{X}_{k+i} | z^s_{k+1:k+i}, z^g_{k+1:k+i}, I_{1:k}, \mathcal{H}^g_k, D). \tag{6.56}$$

As both the geometric $\mathbb{P}(z_k^g|x_k^{rel})$ and classifier (6.15) models require x_{k+i}^{rel}, in addition to the class hypothesis c, measurement generation involves sampling both. We now discuss the specifics for each method, addressing MH in Sec. 4.B and JLP in Sec. 4.C while expanding both to multiple objects.

4.B Multi-Hybrid Planning (MH-BSP)

In this section we discuss the specifics of generating measurements for planning using MH. Now considering multiple objects, we must generate future Z^g and $\{l\Gamma\}$, s.t. the objective function is as follows:

$$J(b[\Lambda_k, \mathcal{X}_k], a_{k:k+L}) = \mathbb{E}_{\{l\Gamma_{k+1:k+L}\}, Z^g_{k+1:k+L}}(\sum_{i=1}^{l} r(b[\Lambda_{k+i}, \mathcal{X}_{k+i}], a_{k+i})), \tag{6.57}$$

where $b[\Lambda_k, \mathcal{X}_k]$ is obtained by MH from Sec. 3.B, and:

$$b[\Lambda_{k+i}, \mathcal{X}_{k+i}] = \mathbb{P}(\Lambda_{k+i}, \mathcal{X}_{k+i}|\{l\Gamma_{k+1:k+i}\}, Z^g_{k+1:k+i}, I_{1:k}, \mathcal{H}^g_k, D). \tag{6.58}$$

As each $l\Gamma_{k+i,w}$ consists of separate $l\gamma^o_{k+i,w}$, and similarly Z^g_{k+i} consists of $z^{g,o}_{k+i}$, we must first predict which objects will be observed at time $k+i$. This can be done using an object observation model (see e.g. chapter 3) and sampled robot and object poses (either by sampling all objects or using a heuristic, see e.g. [116]); These objects are included in the predicted O_{k+i} set, and form $\mathcal{X}^{inv}_{k+1} \triangleq x_{k+1} \cup \{x^o\}_{o \in O_{k+1}}$.

To present that generative model, we first consider the generation of $\{l\Gamma_{k+1}\}$ and Z^g_{k+1} from $b[\Lambda_k, \mathcal{X}_k]$ conditioned on action a_k. We present a sampling hierarchy that is described by the following marginalization scheme:

$$\begin{gathered}\mathbb{P}(\{l\Gamma_{k+1}\}, Z^g_{k+1}|\mathcal{H}_k, a_k) = \\ \textstyle\sum_C \int_{\Lambda_k, \mathcal{X}_{k+1}} \mathcal{L}_{k+1} \cdot \mathbb{P}(C|\Lambda_k) \cdot \mathcal{M}_{k+1} \cdot b[\Lambda_k, \mathcal{X}_k] d\Lambda_k d\mathcal{X}_{k+1},\end{gathered} \tag{6.59}$$

which induces the following sampling hierarchy, for every object $o \in O_{k+1}$:

$$\{l\gamma^o_{k+1}\} \sim \mathbb{P}(l\gamma^o_{k+1}|c^o, x^{rel,o}_{k+1}) \tag{6.60}$$

$$z^{g,o}_{k+1} \sim \mathbb{P}(z^{g,o}_{k+1}|x^{rel,o}_{k+1}) \tag{6.61}$$

$$C \sim Cat(\Lambda_{k,w}) \tag{6.62}$$

$$\mathcal{X}_{k+1} \sim \mathcal{M}_{k+1}\mathbb{P}(\mathcal{X}_k|\Lambda_{k,w}, l\gamma_{1:k,w}, \mathcal{H}^g_k) \tag{6.63}$$

$$w \sim \mathbb{P}(w|D), \tag{6.64}$$

where $x^{rel,o}_{k+1} \triangleq x^o \ominus x_{k+1}$, and is determined by $\mathcal{X}^{inv}_{k+1}$. Recall that O_{k+1} must be determined by sampling $\mathcal{X}_{k+1}$. First, w is sampled uniformly from $|W|$. From there, as $b[\Lambda_k]$ is already represented by a set of samples $\{\Lambda_{k,w}\}$, sampling w chooses $\Lambda_{k,w}$ as well. Next, following from Eq. (6.19) for the multiple object case, definition (6.36) for

$hb_w[\mathcal{X}_k, C]$, and that $\Lambda_{k,w}$ is chosen:

$$\mathbb{P}(\mathcal{X}_k|\Lambda_{k,w}, l\gamma_{1:k,w}, \mathcal{H}_k^g) = \sum_c hb_w[\mathcal{X}_k, C]. \tag{6.65}$$

Then $\mathbb{P}(\mathcal{X}_k|\Lambda_{k,w}, l\gamma_{1:k,w}, \mathcal{H}_k^g)$ is propagated via:

$$\mathbb{P}(\mathcal{X}_{k+1}|\Lambda_{k,w}, l\gamma_{1:k,w}, \mathcal{H}_k^g, a_k) = \sum_c \mathcal{M}_{k+1} hb_w[\mathcal{X}_k, C], \tag{6.66}$$

and $\mathcal{X}_{k+1}$ is sampled, from there we determine O_{k+1}.

Now for each object $o \in O_{k+1}$ we determine the appropriate $x_{k+1}^{rel,o}$, and generate its own geometric measurement $z_{k+1}^{g,o}$. Next we sample class realization C; As the action a_k alone doesn't change Λ from time k to $k+1$ without measurements, $\Lambda_{k,w}$ is used to sample C. As such, C is a categorical random variable with the probability vector $\Lambda_{k,w}$ as its parameters. Finally, with $c \in C$ and $x_{k+1}^{rel,o}$ we sample a set of $|W|$ vectors $l\gamma_{k+1}^o$.

Often planning algorithms use Maximum Likelihood (ML) estimation to reduce computational effort compared to sampling; Note that in our case, taking the ML estimation of C can be problematic because it only considers the most likely class realization, ignoring all possible others.

For the following time steps, we use the generated $\{l\Gamma_{k+1}\}$ and Z_{k+1}^g to infer $b[\Lambda_{k+1}, \mathcal{X}_{k+1}]$ using MH inference from Sec. 3.B. Now using action a_{k+1}, we can generate $\{l\Gamma_{k+2}\}$ and Z_{k+2}^g, then $b[\Lambda_{k+2}, \mathcal{X}_{k+2}]$, and continue generating measurements and inferring corresponding belief until the end of planning horizon.

Alg. 6.1 presents the MH-BSP measurement generation algorithm, where the function *PredictObs* predicts which objects are observed given sampled camera and object poses.

We summarize our approach with the MH-BSP objective function computation Alg. 6.2, where the function *UpdateHB* is the hybrid belief update approach presented in Sec. 3.B, and *InferDist* infers $b[\Lambda_{k+1}, \mathcal{X}_{k+1}]$ from measurement generated in Alg. 6.1. Alg. 6.2 recursively calls itself until the action set only includes one action, allowing non-myopic planning.

As in inference, while accurate, MH-BSP can be computationally expensive. Subsequently, in the next section we propose the expansion of JLP for planning. As in inference, JLP is significantly computationally faster.

4.C Joint Lambda Pose Planning (JLP-BSP)

In this section we present JLP-BSP, an epistemic uncertainty aware semantic BSP framework that leverages JLP from Section 3.C as the inference engine. If the assumption in Lemma. 6.3.1 is exactly or approximately satisfied, we can utilize JLP for planning.

Algorithm 6.1 MH-BSP Measurement Generation

Input: Belief $b[\Lambda_k, \mathcal{X}_k]$, action a_k

$b[\Lambda_k, \mathcal{X}_{k+1}] \leftarrow \mathcal{M}_k \cdot b[\Lambda_k, \mathcal{X}_k]$
$\Lambda_k, \mathcal{X}_{k+1} \leftarrow \text{Sample}(b[\Lambda_k, \mathcal{X}_{k+1}])$
$O_{k+1} \leftarrow \text{PredictObs}(\mathcal{X}_{k+1})$
$\mathcal{X}^{inv}_{k+1} \leftarrow O_{k+1}, \mathcal{X}_{k+1}$
$Z^g_{k+1} \leftarrow \emptyset$
$\{l\Gamma_{k+1}\} \leftarrow \emptyset$
for $o \in O_{k+1}$ **do**
    $x^{rel,o}_{k+1} \leftarrow x^o, x_{k+1} \in \mathcal{X}^{inv}_{k+1}$
    $c^o \leftarrow C$
    $z^{g,o}_{k+1} \leftarrow \text{Sample}(\mathbb{P}(z^{g,o}_{k+1}|x^{rel,o}_{k+1}))$
    $Z^g_{k+1} \leftarrow Z^g_{k+1} \cup z^g_{k+1}$
    $\{l\gamma^o_{k+1}\} \leftarrow \emptyset$
    for $w \in W$ **do**
        $\{l\gamma^o_{k+1}\} \leftarrow \{l\gamma^o_{k+1}\} \cup \text{Sample}(\mathbb{P}(l\gamma^o_{k+1}|c^o, x^{rel,o}_{k+1}))$
    end for
    $\{l\Gamma_{k+1}\} \leftarrow \{l\Gamma_{k+1}\} \cup \{l\gamma^o_{k+1}\}$
end for
return $Z^g_k, \{l\Gamma_{k+1}\}$

Similarly to MH-BSP, we should reason about the generation of new measurement. As described in Sec. 4.B, MH-BSP uses the classifier uncertainty model (6.15) parameters $h_i(x^{rel}_{k+1})$ and $\Sigma_i(x^{rel}_{k+1})$ to generate $\{l\gamma_{k+1}\}$ given class $c = i$ and x^{rel}_{k+1}; On the other hand, JLP-BSP doesn't require generating $\{l\gamma_{k+1}\}$ and elegantly uses $h_i(x^{rel}_{k+1})$ and $\Sigma_i(x^{rel}_{k+1})$ as generated measurements.

With this, the objective function takes the following form:

$$J(b[\bar{l\lambda}_k, \mathcal{X}_k], a_{k:k+L}) = \mathbb{E}_{\mathbb{E}(l\Gamma_{k+1:k+L}),\Sigma(l\Gamma_{k+1:k+L}),Z^g_{k+1:k+L}}(\sum_{i=1}^{L} r(b[\bar{l\lambda}_k, \mathcal{X}_{k+i}], a_{k+i})) \tag{6.67}$$

where,

$$b[\bar{l\lambda}_{k+i}, \mathcal{X}_{k+i}] = \mathbb{P}(\bar{l\lambda}_{k+i}, \mathcal{X}_{k+i}|\mathbb{E}(l\Gamma_{k+1:k+L}), \Sigma(l\Gamma_{k+1:k+L}), Z^g_{k+1:k+i}, I_{1:k}, \mathcal{H}^g_k, D), \tag{6.68}$$

where $\mathbb{E}(l\Gamma_k) \triangleq \{\mathbb{E}(l\gamma^o_k)\}_{o\in O_k}$ and similarly $\Sigma(l\Gamma_k) \triangleq \{\Sigma(l\gamma^o_k)\}_{o\in O_k}$

As in Sec. 4.B, we consider measurement generation for time $k+1$ from time k. This time, we present a sampling hierarchy that is described by the following marginalization scheme:

$$\begin{gathered}\mathbb{P}(\mathbb{E}(l\Gamma_{k+1}), \Sigma(l\Gamma_{k+1}), Z^g_{k+1}|\mathcal{H}_k, a_k) = \\ \int_{\mathcal{X}_{k+1},\bar{l\lambda}_k} \prod_{o\in O_{k+1}} \mathbb{P}(\mathbb{E}(l\gamma^o_{k+1}), \Sigma(l\gamma^o_{k+1})|l\lambda^o_k, \mathcal{H}_k, a_k)\cdot \\ \cdot\mathbb{P}(z^{g,o}_{k+1}|\mathcal{X}_{k+1}) \cdot b[\bar{l\lambda}_k, \mathcal{X}_{k+1}]d\mathcal{X}_{k+1}.\end{gathered} \tag{6.69}$$

By using the above equation, we can write the generative model that is used to generate

Algorithm 6.2 MH-BSP Objective Function

Input: MH Belief $b[\Lambda_k, \mathcal{X}_k]$, a set of actions $a_{k:k+L}$

$J \leftarrow 0$
for number of samples N_s **do**
  $Z^g_{k+1}, \{l\Gamma_{k+1}\} \leftarrow$
    MH Measurement Generation($b[\Lambda_k, \mathcal{X}_k], a_k$)(Alg. 6.1)
  $b[\Lambda_{k+1}, \mathcal{X}_{k+1}] \leftarrow$ UpdateMH($b[\Lambda_k, \mathcal{X}_k], \{l\Gamma_{k+1}\}, Z^g_{k+1}, a_k$)
  $r(b[\Lambda_{k+1}, \mathcal{X}_{k+1}]) \leftarrow$ Reward($b[\Lambda_{k+1}, \mathcal{X}_{k+1}]$
  $J \leftarrow J + r(b[\Lambda_{k+1}, \mathcal{X}_{k+1}])/N_s$
  if $L \neq 0$ **then**
    $J \leftarrow J$
    +MH Objective Function($b[\Lambda_{k+1}, \mathcal{X}_{k+1}], a_{k+1:k+L}$)$/N_s$
  end if
end for
return J

measurements for every $o \in O_{k+1}$. First, we need to determine the set O_{k+1}, and sample the hypothesized object class c^o from $l\lambda^o_k \in \bar{l\lambda}_k$. We do so by sampling $\mathcal{X}_{k+1}$ and $\bar{l\lambda}_k$ using $b[\bar{l\lambda}_k, \mathcal{X}_k]$ as follows:

$$\bar{l\lambda}_k, \mathcal{X}_{k+1} \sim \mathcal{M}_{k+1} \cdot b[\bar{l\lambda}_k, \mathcal{X}_k]. \tag{6.70}$$

Similar to MH-BSP, $\bar{l\lambda}_k$ stays the same conditioned on a_k, thus not propagated. Then we determine O_{k+1}, $\mathcal{X}^{inv}_{k+1}$ and $x^{rel,o}_{k+1}$ per object as we did in Sec. 4.B. From there, for $o \in O_{k+1}$ we sample c^o and afterwards generate the measurements:

$$\mathbb{E}(l\gamma^o_{k+1}) = h_c(x^{rel,o}_{k+1}) \tag{6.71}$$
$$\Sigma(l\gamma^o_{k+1}) = \Sigma_c(x^{rel,o}_{k+1}) \tag{6.72}$$
$$z^g_{k+1} \sim \mathbb{P}(z^g_k | x^{rel,o}_{k+1}) \tag{6.73}$$
$$c^o \sim Cat(\lambda^o_k), \tag{6.74}$$

where $h_c(x^{rel,o}_{k+1})$ and $\Sigma_c(x^{rel,o}_{k+1})$ are the Gaussian parameters of $\mathbb{P}(l\gamma^o_{k+1}|c, x^{rel,o}_{k+1})$, as in Eq. (6.15).

Alg. 6.3 presents the JLP measurement generation algorithm, where $\mathbb{E}(l\Gamma_k) \triangleq \{\mathbb{E}(l\gamma^o_k)\}_{o\in O_k}$, and similarly $\Sigma(l\Gamma_k) \triangleq \{\Sigma(l\gamma^o_k)\}_{o\in O_k}$.

The objective function computation is presented in Alg. 6.4. *UpdateJLP* refers to updating $b[\bar{l\lambda}_k, \mathcal{X}_k]$ as in Sec. 3.C given generated measurements. Alg. 6.4 calls itself recursively until there is only one action left in the set. The algorithm is similar to Alg. 6.2, except for the measurement generation and update functions which are specific for JLP.

Algorithm 6.3 JLP-BSP Measurement Generation

Input: Belief $b[l\bar{\lambda}_k, \mathcal{X}_k]$, action a_k

$b[l\bar{\lambda}_k, \mathcal{X}_{k+1}] \leftarrow \mathcal{M}_k \cdot b[l\bar{\lambda}_k, \mathcal{X}_k]$
$l\bar{\lambda}_k, \mathcal{X}_{k+1} \leftarrow \text{Sample}(b[\Lambda_k, \mathcal{X}_{k+1}])$
$O_{k+1} \leftarrow \text{PredictObs}(\mathcal{X}_{k+1})$
$\mathcal{X}^{inv}_{k+1} \leftarrow O_{k+1}, \mathcal{X}_{k+1}$
$Z^g_{k+1} \leftarrow \emptyset$
$\mathbb{E}(l\Gamma_{k+1}) \leftarrow \emptyset$
$\Sigma(l\Gamma_{k+1}) \leftarrow \emptyset$
for $o \in O_{k+1}$ **do**
    $x^{rel}_{k+1} \leftarrow x^o, x_{k+1} \in \mathcal{X}^{inv}_{k+1}$
    $z^{g,o}_{k+1} \leftarrow \text{Sample}(\mathbb{P}(z^{g,o}_{k+1}|x^{rel}_{k+1}))$
    $Z^g_{k+1} \leftarrow Z^g_{k+1} \cup z^g_{k+1}$
    $c^o \leftarrow \text{Sample}(Cat(\lambda^o_k))$
    $\mathbb{E}(l\gamma^o_{k+1}) \leftarrow h_c(x^{rel}_{k+1})$
    $\mathbb{E}(l\Gamma_{k+1}) \leftarrow \mathbb{E}(l\Gamma_{k+1}) \cup \mathbb{E}(l\gamma^o_{k+1})$
    $\Sigma(l\gamma^o_{k+1}) \leftarrow \Sigma_c(x^{rel}_{k+1})$
    $\Sigma(l\Gamma_{k+1}) \leftarrow \Sigma(l\Gamma_{k+1}) \cup \Sigma(l\gamma^o_{k+1})$
end for
return $Z^g_k, \mathbb{E}(l\Gamma_{k+1}), \Sigma(l\Gamma_{k+1})$

4.D Reward Functions Over $b[\lambda, \mathcal{X}]$

Predicting future $b[\lambda_{k+i}, \mathcal{X}_{k+i}]$ at a future time $k+i$ allows us to consider multiple reward functions, all captured by the general formulation $r(b[\lambda, \mathcal{X}])$. To the best of our knowledge, we are the first to consider reasoning about *future posterior epistemic uncertainty* within a BSP setting. For rewards based on the poses $r(\mathcal{X})$ e.g. distance-to-goal, or rewards based on the belief over the poses $r(b[\mathcal{X}])$ e.g. information-theoretic costs, we can compute the marginal $b[\mathcal{X}_{k+i}]$ as in Eq. (6.29) for MH, or by marginalizing out $l\lambda_{k+i}$ from $b[\lambda_{k+i}, \mathcal{X}_{k+i}]$ for JLP.

In addition, we may also consider a reward over the posterior class probability $r(\mathbb{P}(c|\mathcal{H}))$ which can be extracted by computing $\mathbb{E}(\lambda_{k+i})$ from the marginal $b[\lambda_{k+i}]$:

$$\mathbb{P}(c \mid \{l\gamma_{k+1:k+i}\}, z^g_{k+1,k+i}, I_{1:k}, \mathcal{H}^g_k, D) = \int_{\lambda_{k+i}} \mathbb{P}(c|\lambda_{k+i}) \cdot b[\lambda_{k+i}] d\lambda_{k+i} = \mathbb{E}(\lambda_{k+i}), \tag{6.75}$$

therefore we can write $r(\mathbb{P}(c|\mathcal{H}))$ as $r(\mathbb{E}(\lambda))$. An example for such reward is the minus of Shannon Entropy, such that $r(\mathbb{E}(\lambda)) = \sum_c \lambda^c \log(\lambda^c)$. This reward favors class probability vectors when one of the candidates has probability close to one, and others close to zero.

Crucially, as $b[\Lambda_{k+i}, \mathcal{X}_{k+i}]$ for MH-BSP and $b[\bar{l\lambda}_{k+i}, \mathcal{X}_{k+i}]$ for JLP-BSP both reason about epistemic uncertainty, it affects implicitly every reward. Thus, we account for future posterior epistemic uncertainty indirectly in all the cases discussed in this section.

Algorithm 6.4 JLP-BSP Objective Function

Input: JLP Belief $b[\bar{l\lambda}_k, \mathcal{X}_k]$, a set of actions $a_{k:k+l}$

$J \leftarrow 0$
for number of samples N_s **do**
  $Z^g_{k+1}, \mathbb{E}(l\Gamma_{k+1}), \Sigma(l\Gamma_{k+1}) \leftarrow$
    JLP Measurement Generation$(b[\bar{\lambda_k}, \mathcal{X}_k], a_{k:k+l})$(Alg. 6.3)
  $b[\bar{l\lambda}_{k+1}, \mathcal{X}_{k+1}] \leftarrow$ UpdateJLP$(b[\bar{l\lambda}_k, \mathcal{X}_k])$,
    $Z^g_{k+1}, \mathbb{E}(l\Gamma_{k+1}), \Sigma(l\Gamma_{k+1})$
  $r(b[\bar{l\lambda}_{k+1}, \mathcal{X}_{k+1}]) \leftarrow$ Reward$(b[\bar{l\lambda}_{k+1}, \mathcal{X}_{k+1}]$
  $J \leftarrow J + r(b[\bar{l\lambda}_{k+1}, \mathcal{X}_{k+1}])/N_s$
  if $l \neq 0$ **then**
    $J \leftarrow J$
      +JLP Objective Function$(b[\bar{l\lambda}_{k+1}, \mathcal{X}_{k+1}], a_{k+1:k+l})/N_s$
  end if
end for
return J

4.E Information-Theoretic Reward Over $b[\lambda]$

In Sec. 4.D we discussed reward functions in the form of $r(b[\mathcal{X}])$ and $r(\mathbb{P}(c|\mathcal{H}))$. But crucially, maintaining $b[\lambda_{k+i}, \mathcal{X}_{k+i}]$ opens the possibility of planning directly over $b[\lambda]$. We consider info-theoretical rewards over λ in the form of $r(b[\lambda])$. Specifically, we consider the differential entropy of λ_{k+i}, denoted $H(\lambda_{k+i})$, and is defined as:

$$H(\lambda_{k+1}) \triangleq -\int_{\lambda_{k+1}} b[\lambda_{k+1}] \cdot \log b[\lambda_{k+1}] d\lambda_{k+1}. \tag{6.76}$$

The reward considered is the minus of the entropy, i.e. $r(b[\lambda]) = -H(\lambda)$, which, as we will see in Sec. 4.E and 4.E, is dependent both on $\mathbb{E}(\lambda)$ and the epistemic model uncertainty.

A possible alternative is a reward of the following general form for λ (see e.g. [88]):

$$r(b[\lambda]) = \omega_1 \cdot f_1(\mathbb{E}(\lambda)) + \omega \cdot f_2(\Sigma(\lambda)), \tag{6.77}$$

where ω_1 and ω_2 are hyperparameters, and f_1 and f_2 are general functions. Here λ can be interchangeable with its logit transformation $l\lambda$. This reward requires the tuning of ω_1 and ω_2 manually, as opposed to using $r(b[\lambda]) = -H(\lambda)$ which does not require parameter tuning at all. In particular, as we will see in Sec. 4.E and Sec. 4.E, $H(\lambda)$ addresses both $\mathbb{E}(\lambda)$ and $\Sigma(\lambda)$ simultaneously; $H(\lambda)$ diminishes (i.e. $r(b[\lambda])$ grows) when $\mathbb{E}(\lambda)$ is closer to the simplex corners, i.e. when one category has its probability close to 1 and the rest close to 0. Also, $H(\lambda)$ diminishes the smaller $\Sigma(\lambda)$ becomes, which corresponds to smaller epistemic uncertainty.

However, computing $H(\lambda_{k+i})$ requires the PDF value of $b[\lambda]$, according to Eq. (6.76), thus requiring us to model the distribution of λ_{k+i}. This distribution can be either parametric e.g. Dirichlet or LG, which we will discuss here, or non-parametric such as

Kernel Density Estimation (KDE). MH provides us with $\{\lambda\}$, therefore any distribution that supports probability vectors can be chosen. On the other hand, JLP limits λ to be LG distributed per definition. Sec. 4.E and Sec. 4.E detail Dirichlet and Logistical Gaussian distributions for $b[\lambda_k]$ respectively in the context of computing entropy. Sec. 4.E discusses the differences between utilizing both distributions. To simplify notations, all of the variables in these sections are considered at the same time step, so we drop the time step index. In addition, we use the single-object notation, i.e. λ and c.

Logistic Gaussian For $b[\lambda]$

One option is to model $b[\lambda]$ as Logistic Gaussian (LG) distributed. This option is supported by both MH and JLP, as illustrated in Fig. 6.7. This distribution (with PDF as in Eq. (6.16)) supports probability vectors with conditions presented in Sec. 1.B for γ, thus samples from LG are probability vectors. This distribution does not have an analytical expression for expectation and covariance, and must be computed numerically or approximated, e.g. via bounds, as we will discuss later.

To compute the parameters from a point cloud of probability vectors, e.g. $\{\lambda\}$, we apply the logit transformation for each $\lambda \in \{\lambda\}$, and get $\{l\lambda\}$. Then, as $l\gamma$ is modeled Gaussian the LG parameters $\mathbb{E}(l\lambda)$ and $\Sigma(l\lambda)$ are inferred.

In addition to expectation and covariance, the LG distribution does not have a closed form solution for its differential entropy. However, LG variable is a transformation of a Gaussian variable with a known expression for entropy. As such, we can express the entropy using the following lemma.

Lemma 6.4.1. *Let $\lambda = [\lambda^1, ..., \lambda^m]^T$ be Logistical-Gaussian distributed, and $l\lambda$ its logit transformation as in Eq. (6.14), thus $l\lambda$ is Gaussian with parameters $\mathbb{E}(l\lambda)$ and $\Sigma(l\lambda)$. As such, the differential entropy $H(\lambda)$ is described by:*

$$\begin{aligned} H(\lambda) =& H(l\lambda) + \sum_{i=1}^{m-1} \mathbb{E}[l\lambda^i] \\ & - \int_{l\lambda} \log\left(1 + \sum_{i=1}^{m-1} e^{l\lambda^i}\right) \mathbb{P}(l\lambda) dl\lambda. \end{aligned} \tag{6.78}$$

Proof The reverse logit transformation from $l\lambda$ to λ is given by:

$$\lambda = \left[\frac{e^{l\lambda^1}}{1 + \sum_{i=1}^{m-1} e^{l\lambda^i}}, ..., \frac{e^{l\lambda^{m-1}}}{1 + \sum_{i=1}^{m-1} e^{l\lambda^i}}, \frac{e^1}{1 + \sum_{i=1}^{m-1} e^{l\lambda^i}}\right]^T. \tag{6.79}$$

Thus, λ is LG distributed, and the probability density function is given as:

$$\mathbb{P}(\lambda) = \frac{1}{\sqrt{2\pi|\Sigma|}} \cdot \frac{1}{\prod_{i=1}^{m} \lambda^i} \cdot e^{-\frac{1}{2}||l\lambda - \mu||^2_\Sigma}, \tag{6.80}$$

with $\mu \in \mathbb{R}^{m-1}$ and $\Sigma \in \mathbb{R}^{(m-1)\times(m-1)}$ being the LG parameters. The term $\frac{1}{\prod_{i=1}^{m}\lambda^i}$ is the determinant of the transformation Jacobian, and is denoted as $|J(\lambda)|$. Thus we write $\mathbb{P}(\lambda)$ as:

$$\mathbb{P}(\lambda) = \mathbb{P}(l\lambda)|J(\lambda)|, \tag{6.81}$$

with $\mathbb{P}(l\lambda) \triangleq \mathbb{P}(l\lambda) = \mathbb{P}(l\lambda)(\mu, \Sigma)$, and write $H(\lambda)$ as:

$$H(\lambda) = -\int_{\lambda} \mathbb{P}(\lambda) \cdot \log \mathbb{P}(\lambda) d\lambda \tag{6.82}$$

Then, we transform the integral variable back to $l\lambda$, as we have a closed form expression for $H(l\lambda)$. As $|J(\lambda)|$ is the transformation Jacobian, $|J(\lambda)|d\lambda = dl\lambda$. From there we can write the integral in Eq. (6.82) as a function of $l\lambda$:

$$\begin{aligned} H(\lambda) = &-\int_{\lambda} \mathbb{P}(l\lambda) \cdot |J(\lambda)| \cdot \log(\mathbb{P}(l\lambda) \cdot |J(\lambda)|) d\lambda = \\ &-\int_{l\lambda} \mathbb{P}(l\lambda) \cdot \log(\mathbb{P}(l\lambda) \cdot |J(\lambda)|) dy = \\ &-\int_{l\lambda} \mathbb{P}(l\lambda) \cdot \log(\mathbb{P}(l\lambda)) dl\lambda - \int_{l\lambda} \mathbb{P}(l\lambda) \cdot \log(|J(\lambda)|) dl\lambda = \\ &H(l\lambda) - \int_{l\lambda} \mathbb{P}(l\lambda) \cdot \log(|J(\lambda)|) dl\lambda. \end{aligned} \tag{6.83}$$

The term $\int_{l\lambda} \mathbb{P}(l\lambda) \cdot \log(|J(\lambda)|) dl\lambda$ is positive, as $\mathbb{P}(l\lambda)$ is always positive and $J(\lambda) > 1$, therefore $H(\lambda) < H(l\lambda)$. Next, we describe $\log(|J(\lambda)|)$ in as a function of $l\lambda$:

$$\begin{aligned} \log|J(\lambda)| = &\log\left(\frac{1}{\prod_{i=1}^{m}\lambda^i}\right) = -\sum_{i=1}^{m-1} \log\lambda^i - \log\lambda^m = \\ &-\sum_{i=1}^{m-1}\left[\log(e^{l\lambda^i}) + \log\left(1 + \sum_{j=1}^{m-1} e^{l\lambda^j}\right)\right] \\ &-\log(1) + \log\left(1 + \sum_{j=1}^{m-1} e^{l\lambda^j}\right) = \\ &-\sum_{i=1}^{m-1} l\lambda^i + m \cdot \log\left(1 + \sum_{j=1}^{m-1} e^{l\lambda^j}\right). \end{aligned} \tag{6.84}$$

Now we plug the above expression for $\log|J(\lambda)|$ into Eq. (6.83) and express $H(\lambda)$ as a function of $l\lambda$:

$$\begin{aligned} H(\lambda) = &H(l\lambda) \\ &+\int_{l\lambda} \mathbb{P}(l\lambda) \cdot \left[\sum_{i=1}^{m-1} l\lambda^i - m \cdot \log\left(1 + \sum_{j=1}^{m-1} e^{l\lambda^j}\right)\right] dl\lambda. \end{aligned} \tag{6.85}$$

As $\int_{l\lambda} \mathbb{P}(l\lambda) l\lambda^i dl\lambda = \mathbb{E}[l\lambda^i]$, we can simplify the above equation into the form shown in

Lemma 6.4.1:

$$H(\lambda) = H(l\lambda) + \sum_{i=1}^{m-1} \mathbb{E}(l\lambda^i) - m \int_{l\lambda} \log\left(1 + \sum_{j=1}^{m-1} e^{l\lambda^j}\right) \cdot \mathbb{P}(l\lambda) dl\lambda. \tag{6.86}$$

As $l\lambda$ is Gaussian, $H(l\lambda) = 0.5 \cdot \log(2\pi e |Cov(l\lambda)|)$. The integral in Eq. (6.78) to the best of our knowledge does not have an analytical solution. One approach is to compute the entropy numerically from $\{\lambda\}$ that we already have, but it is computationally expensive to do so for a large number of candidate classes. Another option is to compute bounds for the entropy, which are presented in the following lemma.

Lemma 6.4.2. *Let $\lambda = [\lambda^1, ..., \lambda^m]^T$ be Logistical-Gaussian distributed, and $l\lambda$ its logit transformation as in Eq.* (6.14)*, thus $l\lambda$ is Gaussian with parameters $\mathbb{E}(l\lambda)$ and $\Sigma(l\lambda)$. As such, an upper bound for $H(\lambda)$ is given by:*

$$H(\lambda) \leq H(l\lambda) + \sum_{i=1}^{m-1} \mathbb{E}(l\gamma^i) - m \cdot \max_i\{0, \mathbb{E}(l\gamma^i)\}, \tag{6.87}$$

and similarly a lower bound is given by:

$$\begin{aligned} H(\lambda) \geq & H(l\lambda) + \sum_{i=1}^{m-1} \mathbb{E}(l\gamma^i) \\ & - m \cdot \max_i\{0, \mathbb{E}(l\gamma^i)\} - m \log m - \sqrt{\frac{\sigma_{ii}^{max}}{2\pi}}, \end{aligned} \tag{6.88}$$

where $\sigma_{ii}^{max} \triangleq \max_i \Sigma_{ii}(l\lambda)$ is the largest value element in the covariance of $l\lambda$.

Proof **Upper Bound**

Let us look at the integral in Eq. (6.86). The term $\log\left(1 + \sum_{j=1}^{m-1} e^{l\lambda^j}\right)$ can be bounded from below by:

$$\log\left(1 + \sum_{j=1}^{m-1} e^{l\lambda^j}\right) \geq \log(\max_i\{1, e^{l\lambda^i}\}) = \max_i\{0, l\lambda^i\}. \tag{6.89}$$

Substituting the above equation to Eq. (6.86) yields the following inequality:

$$H(\lambda) \leq H(l\lambda) + \sum_{i=1}^{m-1} \mathbb{E}(l\lambda^i) - m \int_{l\lambda} \max_i\{0, l\lambda^i\} \cdot \mathbb{P}(l\lambda) dl\lambda. \tag{6.90}$$

The integral term is similar to the expectation definition for $l\lambda_i$, except that it considers only positive $l\lambda_i$, making the resulting value from the integral larger than $\mathbb{E}(l\lambda_i)$. For the next step, we consider the case where there is at least a single $\mathbb{E}[l\lambda^i] \geq 0$, and the

case where for all i, $\mathbb{E}[l\lambda^i] < 0$. Considering both cases we can write:

$$\begin{cases} \int_{l\lambda} \max_i\{0, l\lambda^i\} \cdot \mathbb{P}(l\lambda) dl\lambda \geq 0 & \mathbb{E}(l\lambda^i) < 0 : \forall i \\ \int_{l\lambda} \max_i\{0, l\lambda^i\} \cdot \mathbb{P}(l\lambda) dl\lambda \geq \max \mathbb{E}(l\lambda^i) & \exists \mathbb{E}(l\lambda^i) \geq 0. \end{cases}$$

Considering both cases:

$$\int_{l\lambda} \max_i\{0, l\lambda^i\} \cdot \mathbb{P}(l\lambda) dl\lambda \geq \max\{0, \mathbb{E}l\lambda^i\}. \tag{6.91}$$

Finally, we can substitute the above expression into Eq. (6.90) and get the expression in Lemma 6.4.2:

$$H(\lambda) \leq H(l\lambda) + \sum_{i=1}^{m-1} \mathbb{E}(l\lambda^i) - m \max\{0, \mathbb{E}(l\lambda^i)\}. \tag{6.92}$$

Lower Bound

Let us look again at the integral in Eq. (6.86). This time, the term $\log\left(1 + \sum_{j=1}^{m-1} e^{l\lambda^j}\right)$ can be bounded from above by:

$$\begin{aligned} \log\left(1 + \sum_{j=1}^{m-1} e^{l\lambda^j}\right) \leq & \log(\max_i\{m, me^{l\lambda^i}\}) = \\ & \max_i\{0, l\lambda^i\} + \log(m). \end{aligned} \tag{6.93}$$

Now, we substitute the above inequality into Eq. (6.86), and we get the following expression:

$$H(\lambda) \leq H(l\lambda) + \sum_{i=1}^{m-1} \mathbb{E}(l\lambda^i) - m\log(m) - m \int_{l\lambda} \max_i\{0, l\lambda^i\} \cdot \mathbb{P}(l\lambda) dl\lambda. \tag{6.94}$$

This time we look for an upper bound for $\int_{l\lambda} \max_i\{0, l\lambda^i\} \cdot \mathbb{P}(l\lambda) dl\lambda$. Let us consider that:

$$\int_0^\infty \frac{l\lambda^i}{\sqrt{2\pi}} e^{-\frac{(l\lambda^i)^2}{2\Sigma_{ii}}} dl\lambda^i = \sqrt{\frac{\Sigma_{ii}}{2\pi}} \leq \sqrt{\frac{\Sigma_{ii}^{max}}{2\pi}} \tag{6.95}$$

where Σ_{ii} is the element (i, i) in the diagonal of matrix Σ, and Σ_{ii}^{max} is the largest element of Σ. Then we can bound $\int_{l\lambda} \max_i\{0, l\lambda^i\} \cdot \mathbb{P}(l\lambda) dl\lambda$ by:

$$\begin{cases} \int_{l\lambda} \max_i\{0, l\lambda^i\} \cdot \mathbb{P}(l\lambda) dl\lambda \leq \sqrt{\frac{\Sigma_{ii}^{max}}{2\pi}} & \mathbb{E}(l\lambda^i) < 0 : \forall i \\ \int_{l\lambda} \max_i\{0, l\lambda^i\} \cdot \mathbb{P}(l\lambda) dl\lambda \leq \max \mathbb{E}(l\lambda^i) + \sqrt{\frac{\Sigma_{ii}^{max}}{2\pi}} & \exists \mathbb{E}(l\lambda^i) \geq 0, \end{cases}$$

From the above equation, we reach:

$$\int_{l\lambda} \max_i\{0, l\lambda^i\} \cdot \mathbb{P}(l\lambda) dl\lambda \leq \max_i\{0, \mathbb{E}(l\lambda^i)\} + \sqrt{\frac{\Sigma_{ii}^{max}}{2\pi}}, \tag{6.96}$$

and by substituting into Eq. (6.86), we reach the lower bound presented in Lemma 6.4.2:

$$\begin{aligned} H(\lambda) \geq & H(l\lambda) + \sum_{i=1}^{m-1} \mathbb{E}(l\gamma^i) \\ & - m \cdot \max_i\{0, \mathbb{E}(l\gamma^i)\} - m \log m - \sqrt{\frac{\sigma_{ii}^{max}}{2\pi}}. \end{aligned} \tag{6.97}$$

One can observe from the upper bound that $H(l\lambda)$ is necessarily larger than $H(\lambda)$ as $l\gamma$ is not subjected to the probability vector constraints, thus $\mathbb{E}(l\gamma^i)$ can be negative for every i and $\sum_{i=1}^{m-1} \mathbb{E}(l\gamma^i) - m \cdot \max_i\{0, \mathbb{E}(l\gamma^i)\}$ is necessarily non-positive.

Fig. 6.8 presents the entropy values of $b[\lambda]$ as a function of its LG parameters $\mathbb{E}(l\lambda)$ and $Var(l\lambda)$ in the case of two candidate classes. As it has a single degree of freedom, two parameters can fully describe the distribution. The figure shows that the farther $\mathbb{E}[l\lambda]$ is from zero, i.e. the closer $\mathbb{E}[\lambda^1]$ to either one or zero, the smaller the entropy gets in general. The effect is more pronounced in the case where $Var(l\lambda)$ is small. If we aim to minimize entropy during planning, the robot will aim to reach regions where $\mathbb{E}[\lambda]$ is close to the edges of the simplex, and have smaller posterior epistemic uncertainty.

The scenarios presented in Fig. 6.2 correspond to the following cases in Fig. 6.8:

- The unknown-unknown case (Fig. 6.2a) corresponds to $\mathbb{E}(l\lambda)$ close to 0, and large $Var(l\lambda)$, i.e. the upper central part of Fig. 6.8.
- The known-unknown case (Fig. 6.2b) corresponds to $\mathbb{E}(l\lambda)$ close to 0, and small $Var(l\lambda)$, i.e. the lower central part of Fig. 6.8.
- The known-known case (Fig. 6.2c) corresponds to $\mathbb{E}(l\lambda)$ with large absolute value, and small $Var(l\lambda)$, i.e. the lower areas at the sides.
- The uncertain classification case (Fig. 6.2d) corresponds to $\mathbb{E}(l\lambda)$ with large absolute value, and large $Var(l\lambda)$, i.e. the upper areas at the sides.

Dirichlet Distribution For $b[\lambda]$

The other option assumes $b[\lambda]$ is Dirichlet distributed, which is supported only by MH, as illustrated in Fig. 6.7. This distribution is a natural representation of distribution over probability vectors in which samples necessarily satisfy all the conditions of probability vectors presented in Sec. 1.B.

The Dirichlet distribution is parametrized by a parameter set $\alpha \triangleq \{\alpha_1, ..., \alpha_m\}$, and the PDF is:

$$Dir(\lambda; \alpha) = \frac{1}{B(\alpha)} \prod_{i=1}^{m} (\lambda^i)^{\alpha_i - 1}, \tag{6.98}$$

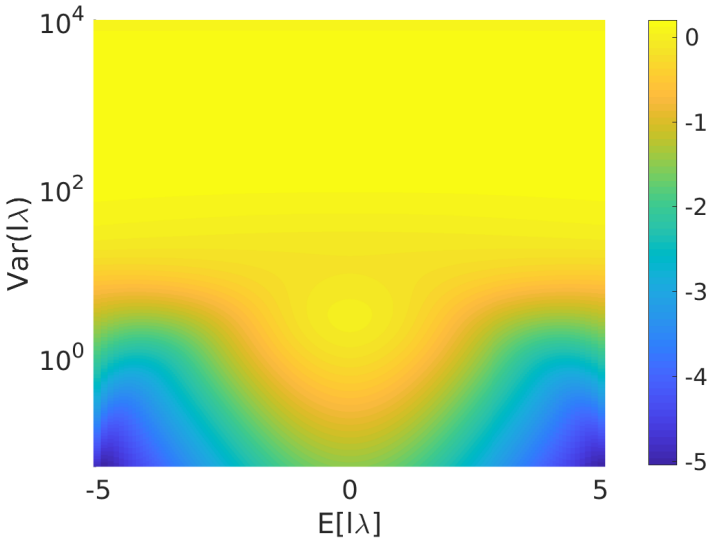


Figure 6.8: Entropy of a one dimensional Logistical Gaussian that corresponds to two dimensional probability vector γ. The x and y axis represent $\mathbb{E}(l\gamma)$ and $Var(l\gamma)$ respectively. Blue to yellow colors correspond to low to high entropy.

with λ^i being the i-th class probability. $B(\alpha)$ is a normalization constant defined as

$$B(\alpha) \triangleq \frac{\prod_{i=1}^{m} \Gamma(\alpha_i)}{\Gamma\left(\alpha_0\right)}, \tag{6.99}$$

where $\Gamma(\cdot)$ is the Gamma function and $\alpha_0 \triangleq \sum_{i=1}^{m} \alpha_i$ for shorthand.

Recall that in MH $b[\lambda]$ is maintained via maintaining each $\lambda_w \in \{\lambda_w\}_{w \in W}$ as in Eq. 6.22 and Eq. 6.32. Dirichlet's distribution parameters, given $\{\lambda\}$, can be estimated in an iterative manner as follows [117]:

$$\psi(\alpha_i^{\text{new}}) = \psi(\sum_{j=1}^{m} \alpha_j^{old}) + \log \hat{\lambda}^i, \tag{6.100}$$

where $\log \hat{\lambda}^i \triangleq \frac{1}{|W|} \log \lambda_w^i$, and $\psi(\cdot)$ is the digamma function. The following expression shows the entropy of the Dirichlet distribution given α parameters:

$$H(\lambda) = \log B(a) + (\alpha_0 - m)\psi(\alpha_0) - \sum_{i=1}^{m} (\alpha_i - 1)\psi(\alpha_i). \tag{6.101}$$

The term $B(\alpha)$ needs to be numerically computed. While it is not an analytical solution, the computation is significantly faster than computing differential entropy using samples.

This entropy takes the maximal value when $\alpha_i = 1$, $\forall i$, and at the "edges" of the distribution, where a single parameter is much larger than the others, the entropy is the lowest. If one of the parameters is zero, then $H(\lambda) = -\infty$, as $\psi(0) = -\infty$. This behavior of entropy can be observed in Fig. 6.9 that shows an example for a two dimensional distribution.

The scenarios presented in Fig. 6.2 correspond to the following cases in Fig. 6.9:

- The unknown-unknown case (Fig. 6.2a) corresponds to α_1 and α_2 that are close to 1, i.e. the central part of Fig. 6.9.
- The known-unknown case (Fig. 6.2b) corresponds to α's with large and similar values, i.e. the upper right part of Fig. 6.9.
- The known-known case (Fig. 6.2c) corresponds to the case where one α is significantly larger than the other, and larger than 1, i.e. left or bottom areas of Fig. 6.9.
- The uncertain classification case (Fig. 6.2d) corresponds to the case where one α is not significantly larger than the other, i.e. the areas between the high and low entropy in Fig. 6.9.

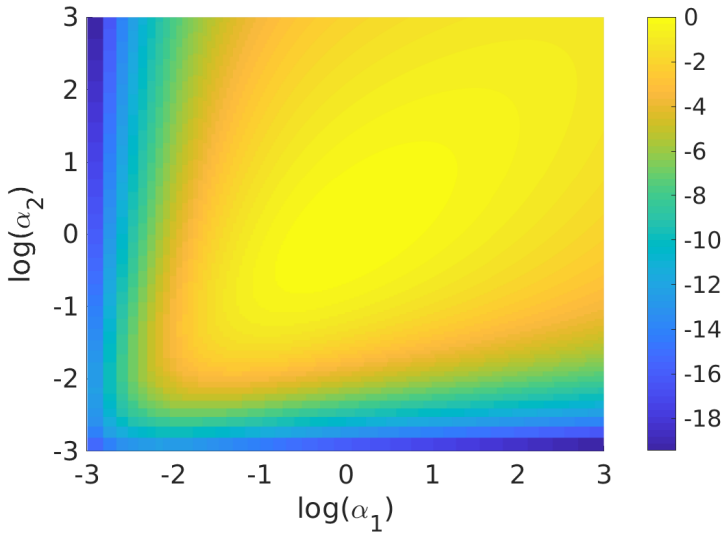
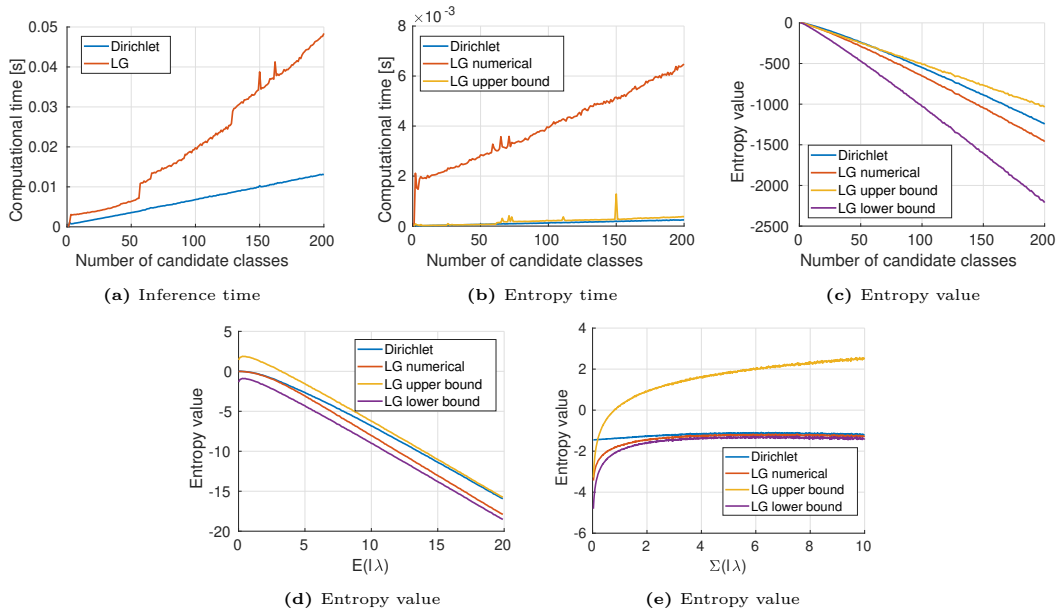


Figure 6.9: Entropy of a two dimensional Dirichlet distribution as a function of log of parameters. Blue to yellow colors correspond to low to high entropy values.

Comparison Between Dirichlet and Logistic Gaussian

When considering the reward $H(\lambda)$ we have to consider two steps:

1. Computation of $b[\lambda]$ parameters; With MH we maintain separately $\lambda_w \in \{\lambda_w\}_{w \in W}$ and subsequently describe $b[\lambda]$ using $\{\{\lambda_w\}_{w \in W}\}$. As such, to compute $H(\lambda)$ we must assume a distribution for $b[\lambda]$ and infer its parameters. JLP on the other hand limits $b[\lambda]$ to be LG distributed.
2. Calculation of $H(\lambda)$ to use as a reward function for planning, either via numerical computation, or by using bounds.



(a) Inference time **(b)** Entropy time **(c)** Entropy value

(d) Entropy value **(e)** Entropy value

Figure 6.10: This figure presents a comparison between Dirichlet and Logistical Gaussian (denoted LG) in terms of computational time, entropy, and log-likelihood values. **(a)** presents a computational time comparison between Dirichlet and LG for inference as a function of probability vector dimension. Similarly **(b)** presents a computational time for entropy computation, both for numerical and upper bound. **(c)** presents a value comparison between the different entropy computations. Note that in **(b)** and **(c)** the plots are given the parameters calculated for **(a)**. **(d)** presents an entropy value comparison between distributions with different expectations with a fixed covariance; Similarly, **(e)** presents an entropy value comparison between distributions with different covariance value with a fixed expectation.

We remind that this discussion is relevant for $H(\lambda)$ computation for MH, as in JLP λ is LG distributed by definition.

Parameter inference for Dirichlet is faster than for LG although the process is numeric. On the other hand, LG is more expressive; For m classes, Dirichlet distribution has m parameters, while LG has $m-1$ parameters for $\mathbb{E}(l\lambda)$, and $\frac{m\cdot(m-1)}{2}$ parameters for $\Sigma(l\lambda)$, totaling in $(1+\frac{m}{2})\cdot(m-1)$ parameters.

LG does not have an analytical solution for computing entropy, and its numeric computation is slower than Dirichlet's. On the other hand, computing bounds of entropy for LG is comparable in terms of computational effort to computing Dirichlet entropy. One must note that while Dirichlet is less computationally expensive than LG, when MH with Dirichlet and JLP are compared, JLP is still computationally much more efficient.

Fig. 6.10 presents a comparison between the two distributions in terms of computational effort and value of the entropy. In all figures, the x-axis is number of candidate classes, and for each, a dataset of 1000 class probability vectors was sampled. Fig. 6.10a presents the measured time of parameter computation, clearly showing an advantage for Dirichlet distribution for high dimensional probability vectors despite the parameter computation process for Dirichlet distribution containing functions that must be numerically computed.

Fig. 6.10b presents the computational time of the entropy, for numerical computation for both LG and Dirichlet, and the bounds for LG(computation time is identical

both for upper and lower bounds; thus only upper bound is shown). Here entropy computation for Dirichlet holds a significant advantage over numerical computation of entropy for LG, and the bound computation time is comparable to Dirichlet.

Fig. 6.10c presents entropy values for Dirichlet, numerical LG, lower and upper bounds for LG. In general, the upper bound tends to be close to the numerical solution for fewer candidate classes. In addition, entropy for Dirichlet distribution tends to be higher.

In Fig. 6.10d the number of candidate classes is fixed to two, i.e. the dimension of $l\lambda$ is $\mathbb{R}^1$; The covariance $\Sigma(l\lambda)$ is fixed at 3, and $\mathbb{E}(l\lambda)$ goes from 0 to 20. In this figure the entropy value monotonically decreases when increasing $\mathbb{E}(l\lambda)$. The lower bound is tighter between the two bounds as $\mathbb{E}(l\lambda)$ increases.

In Fig. 6.10e $\mathbb{E}(l\lambda)$ is fixed instead at $\mathbb{E}(l\lambda) = 3$ and $\Sigma(l\lambda)$ varies between 0 and 10. We can see that the entropy value increases with the increase in $\Sigma(l\lambda)$, but the bigger effect is for LG compared to Dirichlet distribution. The upper bound is tighter at lower $\mathbb{E}(l\lambda)$ values, while the lower bound is tighter for higher values.

One may ask: which distribution should be used? For JLP, as mentioned previously, we are limited to LG. For MH, the tradeoff is between distribution expressiveness and computational effort; While Logistical Gaussian is more expressive because of a larger number of parameters, the computation effort is significantly higher than for Dirichlet distribution. Also, Dirichlet distribution, unlike the Logistical Gaussian, can manage a very small number of probability vector samples.

6.5 Simulation and Experiments

We evaluate our approaches for semantic SLAM inference and planning in simulation (Sec. 5.B) and an experiment (Sec. 5.C) over the Active Vision Dataset scenario Home-3-01 [113], with viewpoint dependent classifier uncertainty models trained using the BigBIRD dataset [118]. We considered environments with multiple spatially scattered objects, and the robot's task is to accurately classify them while localizing. Our implementation uses the GTSAM library [107] with a Python wrapper. The hardware used is an Intel i7-7700 processor running at 2.8GHz and 16GB RAM, with GeForce GTX 1050Ti with 4GB RAM.

5.A Compared Approaches and Metrics

We consider three approaches for inference and planning: our MH and JLP methods with the corresponding MH-BSP and JLP-BSP, and an approach that does not consider model uncertainty, denoted as Without Epistemic Uncertainty (WEU). In this approach we maintain a single hybrid belief and use it for inference and planning, similar to approaches presented in [95] and chapter 3.

We require a metric to evaluate classification where a completely incorrect clas-

sification would not result in infinite error, unlike cross entropy loss. Therefore, our approach is evaluated for classification accuracy using the Mean Square Detection Error metric (MSDE, also used by Teacy et al. [15] and Feldman & Indelman [16]). Given $b[\lambda_k]$, MSDE is defined as follows:

$$MSDE \triangleq \frac{1}{m}\sum_{i=1}^{m}\left(\lambda_{gt}^{i} - \mathbb{E}(\lambda_k^i)\right)^2, \tag{6.102}$$

where λ_{gt}^{i} is the ground truth probability of the object being of class $c = i$, and is equal to 1 if the object is class i and 0 otherwise. For a completely incorrect classification $MSDE \leq 1$, ideal classification $MSDE = 0$, and for classification results where all class probabilities are equal, $MSDE = \frac{m-1}{m^2}$.

5.B Simulation

Simulation Setting

We consider a closed set setting and assume, for simplicity, that the number of classes $m = 2$, i.e., each object can be one of the two classes. The camera senses objects up to 10 meters distance, with an opening angle of 120°. We choose two sets of models for the simulation; The first is a model that satisfies Lemma 6.3.1, and the second does not to show the effect of using JLP with such models. The baseline MSDE score for $m = 2$ where all class probabilities are equal is $MSDE = 0.25$.

Inference: Single Run

The setting for this comparison is an environment with 5 objects; These object are placed within the environment, which is presented in Fig. 6.11 along with the ground truth trajectory. The robot passes through an area in which the objects have classification scores with high degree of epistemic uncertainty. Normally, with methods that do not consider epistemic uncertainty, classification results will have a high chance of being incorrect, but our approach provide more accurate results as it considers epistemic uncertainty. Denote ψ as the relative orientation between the object's orientation (chosen during the classifier uncertainty model training) and the camera's pose. We simulate a classifier model that considers the following cases:

1. The classifier differentiates well between classes with low epistemic uncertainty, $\psi = 0°$.
2. The classifier does not differentiate well between the two classes, $\psi = 90°, 270°$.
3. The classifier differentiates between classes well, but with high epistemic uncertainty, $\psi = 180°$.

As such, $\psi = 0°$ is the relative orientation where the best classification with the lowest uncertainty is expected (corresponding to the blue cone in Fig. 6.11 that represent

this relative orientation), and $\psi = 180°$ is the relative orientation that most prone to classification errors when not considering epistemic uncertainty. Considering the specific ground truth trajectory for the presented scenario, objects 1 and 2 represent case 3; as such, we expect our approaches to infer the correct class within a large number of steps because of the uncertainty. Object 3 represents case 1, and as such when it is observed the classification will be accurate on the first view. Objects 4 and 5 represent case 2, where classification is difficult as the model doesn't differentiate well between the classes of those objects. The object ground truth classes are $c = 1$ for objects 1, 2 and 5, and $c = 2$ for objects 3 and 4.

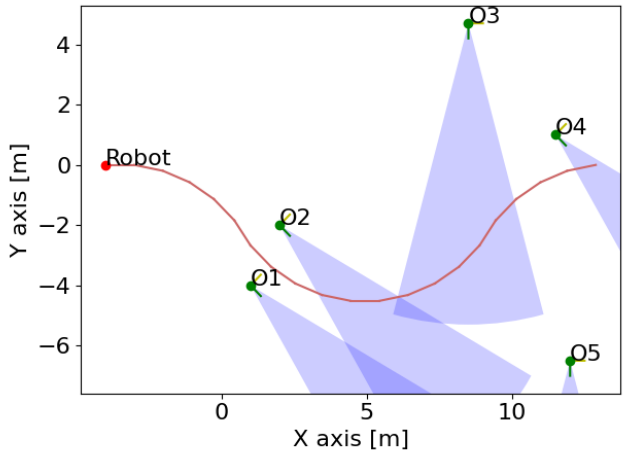
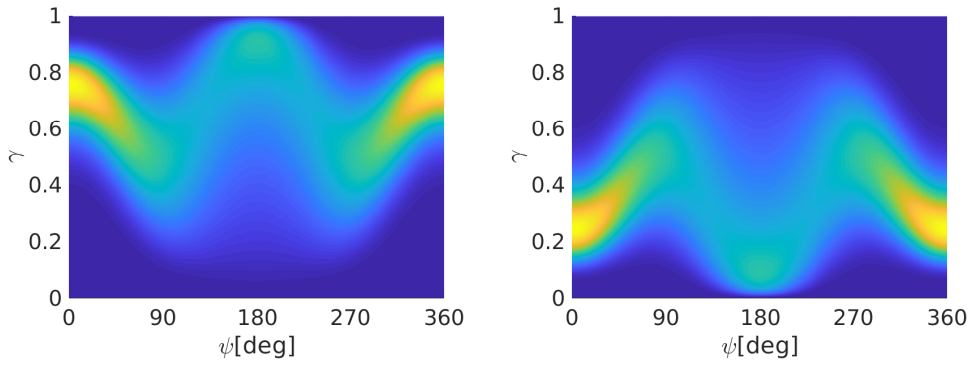


Figure 6.11: The ground truth of the scenario in Sec. 5.B. The red dot represents the robot's starting point, with the red curve being the path. The green dots represent the objects' location with the corresponding object labels. The green line represents the object orientation, with the yellow line present 90° of that orientation. The blue cones represent the observation viewpoints in which the classifier identifies the object class well with low uncertainty, i.e. case 1.

A visualization of the models presented can be seen in Fig. 6.12.



(a) $\mathbb{P}(\gamma^{c=1}|c=1,\psi)$ **(b)** $\mathbb{P}(\gamma^{c=1}|c=2,\psi)$

Figure 6.12: A visualization of the classifier uncertainty model used in Sec. 5.B. We present the value of $\mathbb{P}(\gamma^{c=1}|c, x^{rel}) \equiv \mathbb{P}(\gamma^{c=1}|c, \psi)$ as a function of relative orientation ψ and $\gamma^{c=1}$ value, for classes $c = 1$ and $c = 2$ in **(a)** and **(b)** respectively. Blue and yellow colors correspond to low and high PDF values respectively.

We consider noisy geometric measurements of relative pose, and cloud point semantic measurements, i.e. the classifier gives $\{\gamma\}$ per each object, sampled from the

classifier uncertainty model. We use a classifier uncertainty model with the following function for expectation (see Eq. (6.15)):

$$\begin{aligned} h_{c=1}(x^{rel}) &= \frac{1}{2}\cos(2\cdot\psi)+\frac{1}{2} \\ h_{c=2}(x^{rel}) &= -\frac{1}{2}\cos(2\cdot\psi)-\frac{1}{2}, \end{aligned} \tag{6.103}$$

and the following parameter for root-information:

$$R_{c=1}(x^{rel}) = R_{c=2}(x^{rel}) = 1.4 + 0.6\cdot\cos(\psi), \tag{6.104}$$

Subsequently, the covariance parameter from Eq. (6.15) in the two class case is computed as follows:

$$\Sigma_c(x^{rel}) = \sqrt{\frac{1}{R_c(x^{rel})}}. \tag{6.105}$$

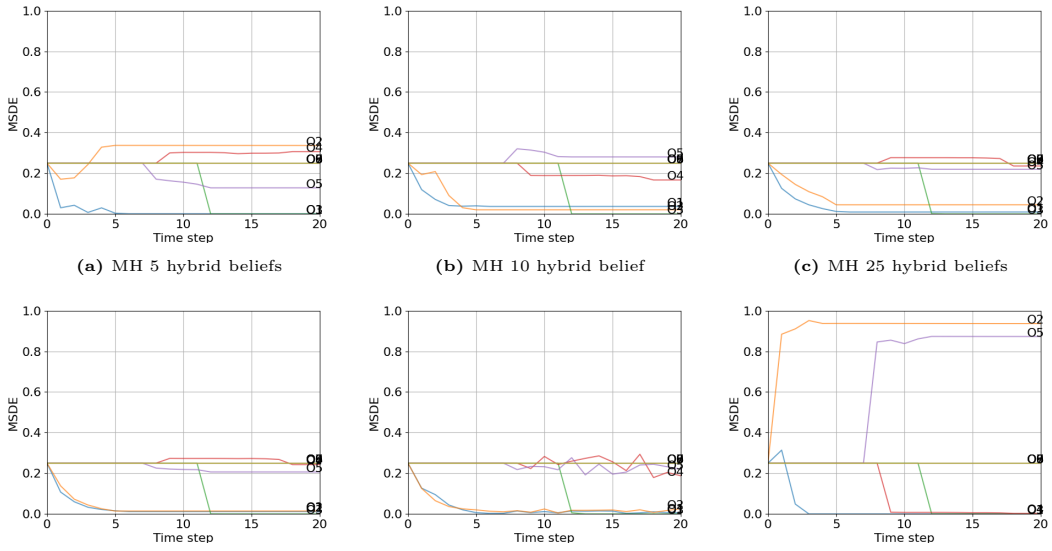
With the covariance parameter being equal, the presented model satisfies the assumption of Lemma 6.3.1, allowing us to use the JLP approach.

Fig. 6.13 presents MSDE results for each object separately. We perform inference with MH with a different number of hybrid beliefs, and compare it to JLP and WEU. With MH and JLP, the class of objects 1 and 2 is inferred using multiple observations, eventually inferring the correct class. The class of object 3, once seen, is quickly and accurately inferred. The class of objects 4 and 5 remain ambiguous (MSDE of approximately 0.25) because they are observed from viewpoints that correspond to case 2. In general, MH in Fig. 6.13a-6.13d tends to present smoother results the more hybrid beliefs are used, and also compared to JLP in Fig. 6.13e where for each time step the entropy must be computed numerically from new λ_k samples. WEU in Fig. 6.13f shows that objects can be classified incorrectly if not considering epistemic uncertainty, such as object 4, as shown in the figure.

As a summary, Fig. 6.14a presents average MSDE results for all the objects combined, showing that epistemic uncertainty aware approaches outperform WEU, while MH with 10 beliefs and JLP perform similarly. Fig 6.14b presents a computation time comparison between WEU, JLP and MH for different number of hybrid beliefs. From this figure, we can see that JLP is comparable to WEU, with MH being significantly more computationally intensive as the number of the simultaneous beliefs increase.

Inference: Statistical Study

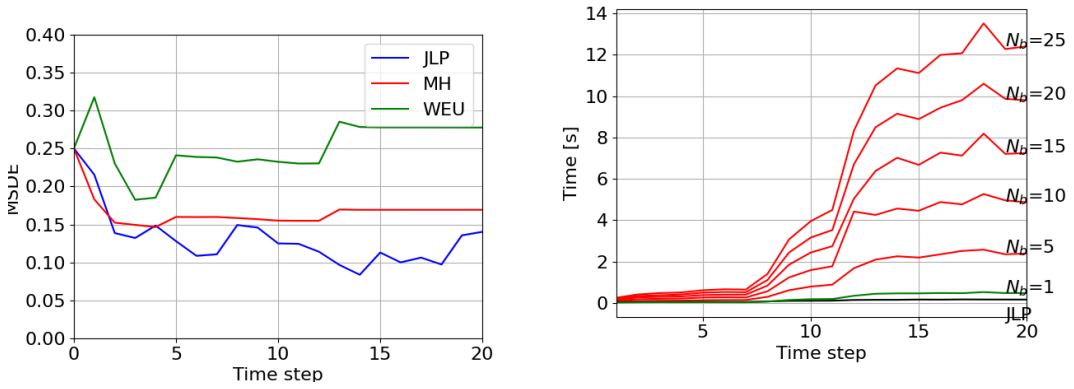
In this section we perform a Monte-Carlo study to compare between MH, JLP, and WEU. We run the simulation 10 times and present results for MSDE and computational time. The setting for this comparison is an environment with 5 objects with randomized poses, with examples presented in Fig. 6.15. Otherwise, we use the same setting and classifier uncertainty model as in Sec. 5.B.



(a) MH 5 hybrid beliefs **(b)** MH 10 hybrid belief **(c)** MH 25 hybrid beliefs

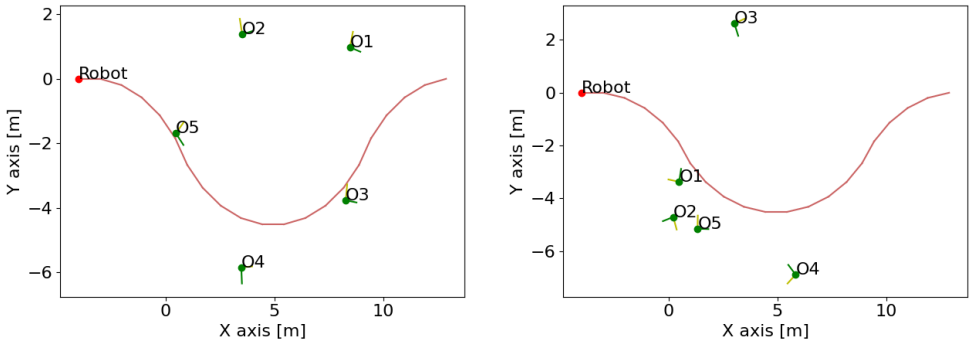
(d) MH 100 hybrid beliefs **(e)** JLP **(f)** WEU

Figure 6.13: **(a)**, **(b)**, **(c)**, and **(d)** show MSDE results per time step for MH per object, each in a different color, for 5, 10, 25, and 100 respectively. **(e)** shows MSDE results for JLP. **(f)** shows MSDE results for WEU



(a) **(b)**

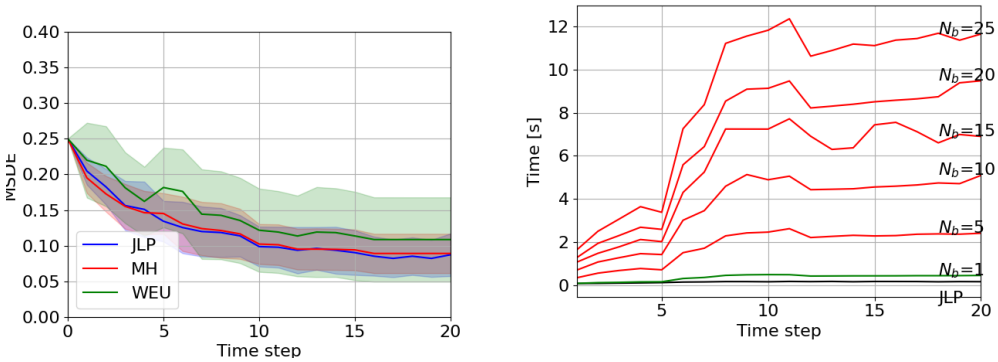
Figure 6.14: **(a)** compares MSDE to time step between MH in red, JLP in blue, and without uncertainty in green. **(b)** compares run-time per inference step between realizations of MH with different number of hybrid beliefs in red, JLP in black, and WEU in green.



(a) **(b)**

Figure 6.15: Examples of a sampled environment in which the inference is performed. The red trajectory is the robot path. The green dots denote the objects, numbered O1 to O5. The green and yellow lines represent their orientation, 0° and 90° respectively.

We present MSDE statistical results in Fig. 6.16a, with one σ uncertainty. While MH and JLP perform similarly, both outperform the approach that does not consider epistemic uncertainty, especially in cases where the camera goes through areas that correspond to $\psi = 180°$. Fig. 6.16b presents run-time results for the algorithms. Expectedly, as the number of simultaneous beliefs increase for MH, the algorithm runs slower. JLP is comparable to maintaining a single hybrid belief in this case, demonstrating that it is more practical when the conditions of Lemma 6.3.1 are satisfied.



(a) (b)

Figure 6.16: **(a)** compares MSDE to time step between MH in red, JLP in blue, and without uncertainty in green. The transparent colors correspond to the respective plot in one σ value. **(b)** compares run-time per inference step between realizations of MH with different number of hybrid beliefs in red, JLP in black, and WEU in green.

Inference: Joint Lambda Pose Assumption

One may consider the ramifications of using JLP with models that don't satisfy Lemma 6.3.1; The most straightforward result is that MH and JLP results don't coincide with each other, and that may result in either erroneous or overconfident classification (i.e. large values of $\mathbb{E}(l\lambda)$ and/or too small values of $\Sigma(l\lambda)$).

This time, the classifier model uses the following parameters; For expectation:

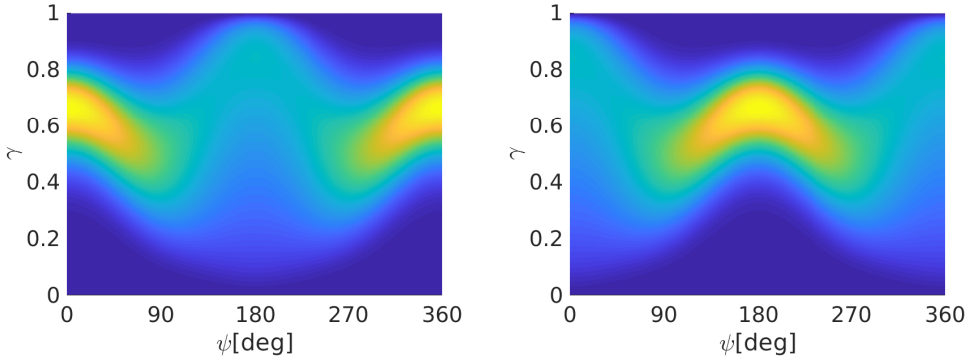
$$\begin{aligned} h_{c=1}(x^{rel}) &= 0.3 \cdot \cos(2 \cdot \psi) + 0.3 \\ h_{c=2}(x^{rel}) &= -0.3 \cdot \cos(2 \cdot \psi) - 0.3, \end{aligned} \tag{6.106}$$

and the following parameters for root-information:

$$\begin{aligned} R_{c=1}(x^{rel}) &= 1.4 + 0.6 \cdot \cos(\psi) \\ R_{c=2}(x^{rel}) &= 1.4 - 0.6 \cdot \cos(\psi). \end{aligned} \tag{6.107}$$

The goal is creating opposing $\Sigma_c(\psi)$ for the two classes, such that Lemma 6.3.1 does not hold and may result in inaccurate classification while using JLP. Fig. 6.17 presents a visualization of the model. Specifically, the problematic areas are around $\psi = 0°$ where the models actually predict that when $\gamma^{c=1} > 0.8$ the likelihood is actually higher for $c = 2$, and vice-versa when $\psi = 180°$ when $\gamma^{c=2} > 0.8$ predicts a higher likelihood for

$c = 1$.



(a) $\mathbb{P}(\gamma^{c=1}|c=1,\psi)$ **(b)** $\mathbb{P}(\gamma^{c=1}|c=2,\psi)$

Figure 6.17: A visualization of the classifier uncertainty model used in Sec. 5.B. We present the value of $\mathbb{P}(\gamma^{c=1}|c,\psi)$ as a function of relative orientation ψ and $\gamma^{c=1}$ value, for classes $c=1$ and $c=2$ in **(a)** and **(b)** respectively. Blue and yellow colors correspond to low and high PDF values respectively.

Fig. 6.18 presents the PDF values of $l\gamma_k$, and $\mathcal{L}_k^s$ with and without JLP assumption at $\psi = 0°$. Here $l\gamma_k \sim \mathcal{N}(0.6, 0.25)$, coinciding with the classifier uncertainty model parameters of class $c = 1$. In this figure the difference between the approximation and real $\mathcal{L}_k^s$ are evident, with the approximated $\mathcal{L}_k^s$ risking a larger chance of incorrect classification as the area below 0 is larger than that for the real $\mathcal{L}_k^s$.

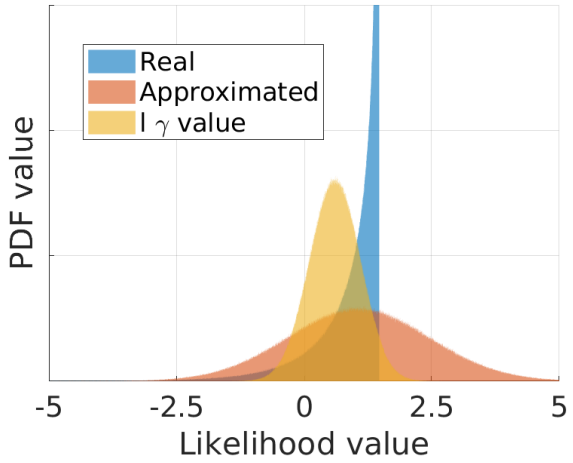
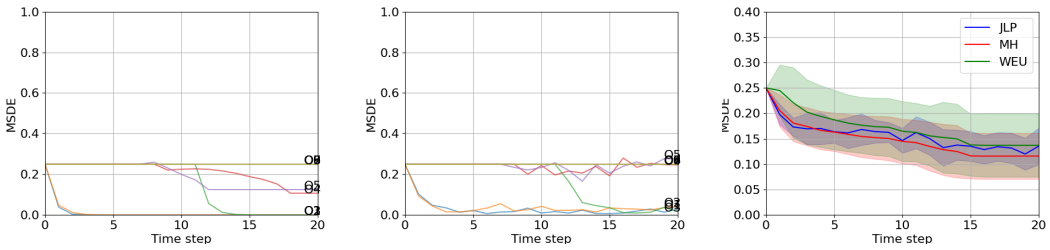


Figure 6.18: An approximate PDF value graph for JLP assumption where $\psi = 0°$ The yellow area represents the distribution of $l\gamma_k$, the red area represents $\mathcal{L}_k^s$ PDF when JLP assumption is used, and the blue area represents the real PDF of $\mathcal{L}_k^s$.

In this scenario we use the same setting we used at Sec. 5.B, 5.B, and 5.B. and compare the MSDE scores in Fig. 6.19. For Fig. 6.19a and 6.19b we use the scenario from Sec. 5.B. In Fig. 6.19a MSDE results for MH with 100 hybrid beliefs are shown as the most accurate, where objects 1 and 2 have more accurate classification than the rest. Fig. 6.19b presents MSDE results for JLP, where we see significantly less accurate results. Finally, we perform a statistical study with 5 random object locations of 10 runs as in Sec. 5.B, comparing between MH with 10 hybrid belief, JLP, and WEO,

and see that statistically the difference between MH and JLP is not large even without Lemma 6.3.1 holding, as opposed to the specific run from Figs. 6.19a and 6.19b.



(a) MH 100 hybrid beliefs **(b)** JLP **(c)** Average MSDE

Figure 6.19: **(a)** shows MSDE results to time step per object for MH with 100 hybrid beliefs which we consider most accurate, while **(b)** shows JLP results for MSDE. **(c)** is a statistical study that compares between average MSDE over all objects for WEU in green, JLP in blue, and MH in red. The line corresponds to MSDE expectation and the colored area to one σ range.

Empirical Analysis of Entropy Reward

First, we present how the reward function of entropy over $b[\lambda]$ as in Eq. (6.76) behaves from different relative viewpoints using the classifier model presented in Sec. 5.B Fig. 6.12. Around a single object, as seen in the ground truth Fig. 6.20, we have a robot placed in different initial points. From each point, the prior $\mathbb{P}(\lambda_0)$ is LG distributed with parameters $\mu = 1.218$ and $\Sigma = 1.219$. We compute sampled rewards for performing a candidate action for each initial position that takes the robot close to the target; Each starting position is denoted with a number index. We show results for both MH and JLP in term of predicted $-H(\lambda_1)$ values, compared to the value of $-H(\lambda_0)$, which for the selected parameters for $\mathbb{P}(\lambda_0)$ is 0.434.

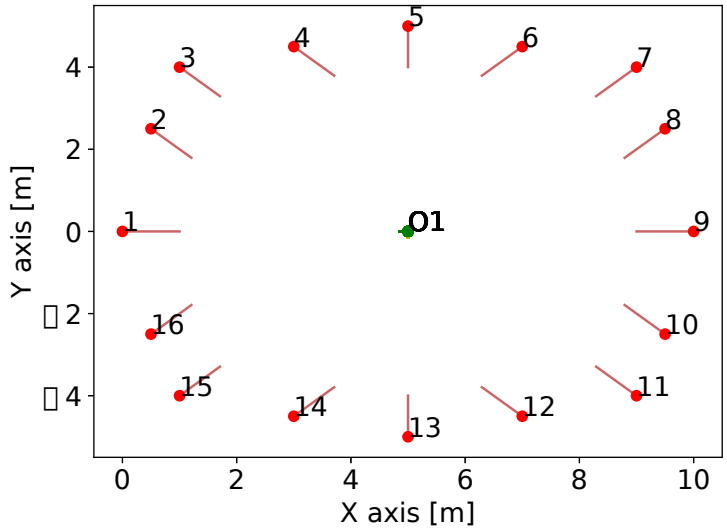
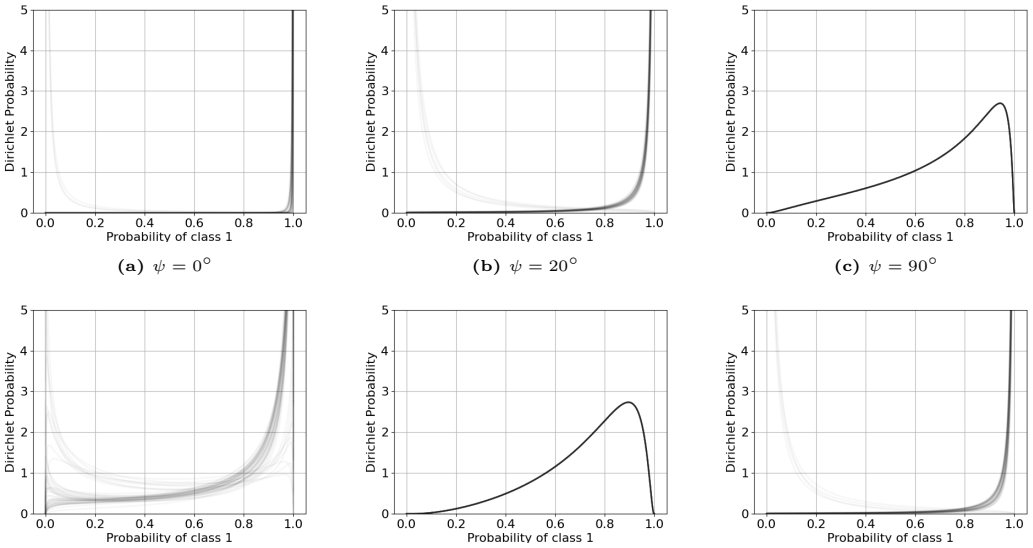


Figure 6.20: Ground truth of robot initial positions relative to the object at the center. Initial positions and corresponding paths are shown in red, the object position in green, and the orientation green and yellow.

Fig. 6.21 and 6.22 present two dimensional simplexes for predicted future $b[\lambda_1]$. In each sub-figure from those figures there are 50 predicted realizations of $b[\lambda_1]$ created from sampled measurements. Each sub-figure represent a different viewpoint. Recall the classifier models at Fig. 6.12, at $\psi = 0°$ the epistemic uncertainty is the lowest, with high degree of separation between the models. This is reflected in Fig. 6.21a and 6.22a, where the predicted $\mathbb{P}(\lambda_0)$ is the closest to the edges of the simplex. At $\psi = 90°$ and $\psi = 270°$, corresponding to Fig. 6.21c, 6.21e, 6.22c, and 6.22e, the classifier model cannot differentiate between the classes, and that is reflected by the simplexes that stay close to the prior. Fig. 6.21b, 6.21f, 6.22b, and 6.22f present the case in which $\psi = \pm 20°$, a middle-of-the-road case between $\psi = 0°$ and $\psi = 90°$. At $\psi = 180°$ the separation is large, but the epistemic uncertainty is high. As such, Fig. 6.21d and 6.22d present a case where the plots are close to the edges, but with large epistemic uncertainty.

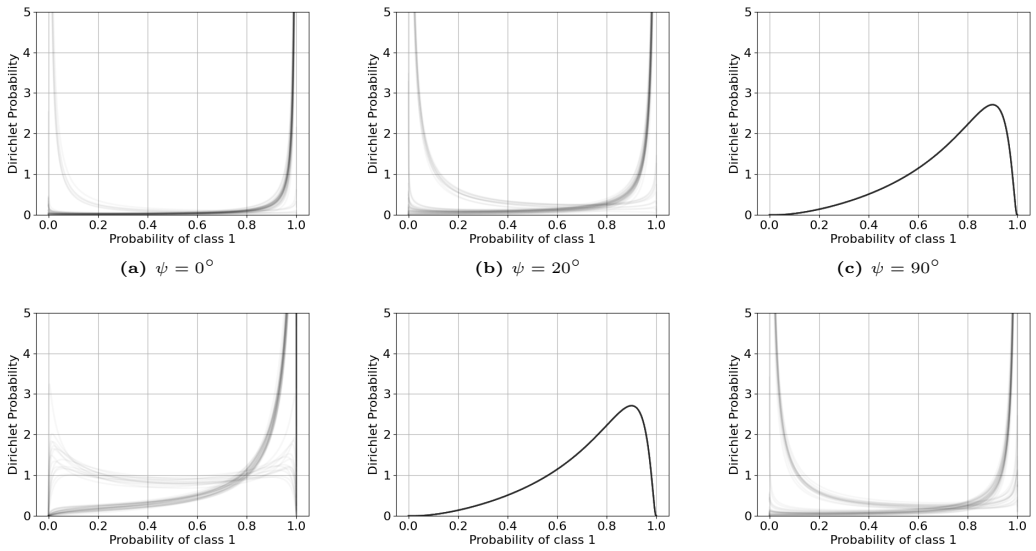


(a) $\psi = 0°$ **(b)** $\psi = 20°$ **(c)** $\psi = 90°$

(d) $\psi = 180°$ **(e)** $\psi = 270°$ **(f)** $\psi = 340°$

Figure 6.21: 2D simplexes of $b[\lambda_1]$ realization created from measurement samples for MH. The x axis is the value of $\lambda_1^{c=1}$, while the y axis is $b[\lambda_1^{c=1}]$. Each figure contains multiple plots with low opacity.

Fig. 6.23 and 6.24 present the rewards for each viewpoints within the scenario, with the red dot corresponding to the objective function, and the blue dot corresponding to $-H(\lambda_0)$, for MH in 6.23 and JLP in 6.24. Both figures present a similar behavior of the reward; The highest rewards are around $\psi = 0°$, i.e. indices 1,2 and 16 where the epistemic uncertainty is the lowest. On the other hand, he lowest rewards are for $\psi = 180°$ where the epistemic uncertainty is the highest. For $\psi = 90, 270°$, the reward stays close to $-H(\lambda_0)$. Therefore, to get the highest reward in planning, the robot must travel to zones where the relative viewpoint is close to $\psi = 0°$. Note that different class hypotheses for the object produce different predicted $b[\lambda_1]$ and therefore $-H(\lambda_1)$, as evident by the fact that at some viewpoints the reward was split to two "clusters".



(a) $\psi = 0°$ **(b)** $\psi = 20°$ **(c)** $\psi = 90°$

(d) $\psi = 180°$ **(e)** $\psi = 270°$ **(f)** $\psi = 340°$

Figure 6.22: 2D simplexes of $b[\lambda_1]$ realization created from measurement samples for JLP. The x axis is the value of $\lambda_1^{c=1}$, while the y axis is $b[\lambda_1^{c=1}]$. Each figure contains multiple plots with low opacity.

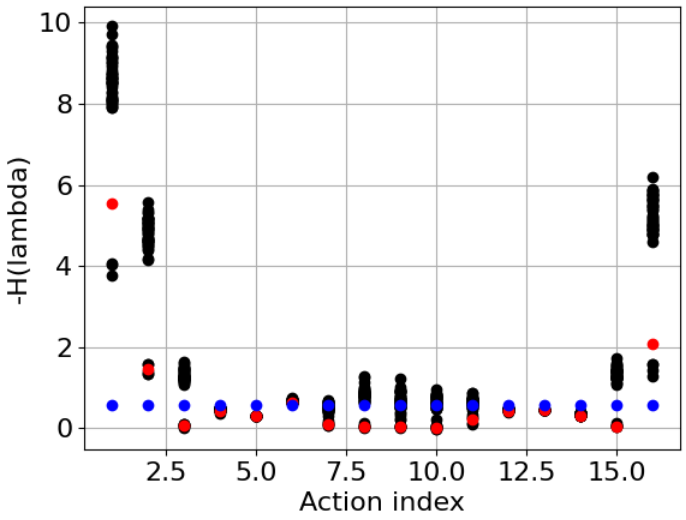


Figure 6.23: Rewards as a function of scenario and action index, logarithmic scale, MH. The black dots represent the rewards, the blue dots represent $-H(\lambda_0)$.

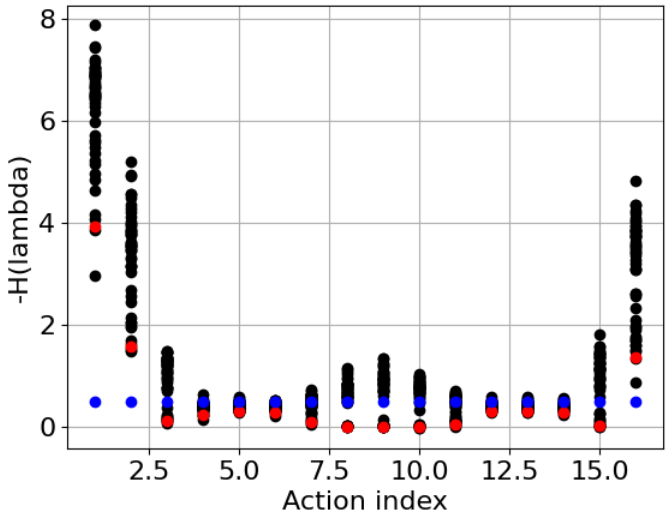


Figure 6.24: Rewards as a function of scenario and action index, logarithmic scale, JLP. The black dots represent the rewards, the blue dots represent $-H(\lambda_0)$.

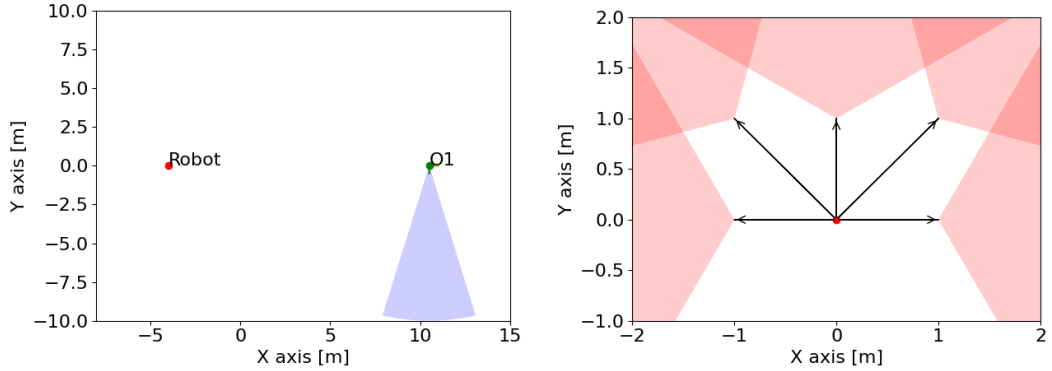
Planning: Single Object

Next, we simulate a planning scenario of a single object using EUS-BSP. Relative to the object, there is an area with low epistemic uncertainty and high separation between classes, represented as a blue cone in Fig. 6.25a. We compare between two reward functions for planning. R_1 is the negative of the entropy of λ as defined in Eq. (6.76), while R_2 is the entropy of $\mathcal{E}(\lambda)$ as defined in Sec. 4.D. For a future belief $b[\lambda_{k+1}]$:

$$\begin{aligned} R_1 &= -H(\lambda_{k+1}) \\ R_2 &= -H(\mathbb{E}(\lambda^o_{k+1})) \end{aligned} \tag{6.108}$$

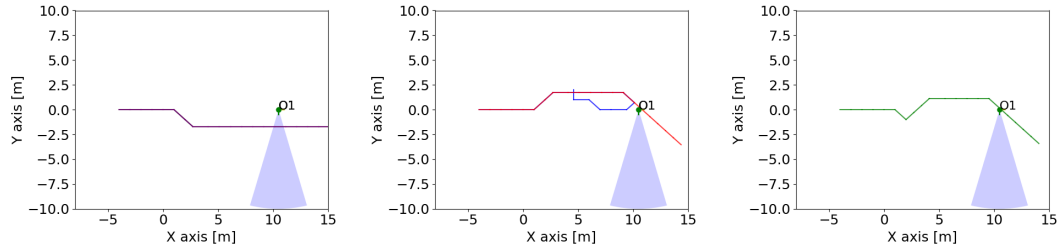
For both reward functions we use MH-BSP and JLP-BSP. We use only R_2 for WEU as R_1 is not applicable because it does not consider epistemic uncertainty, while R_2 can use the posterior class probability as $\mathbb{E}(\lambda^o_{k+1})$. Optimally, the robot would plan to go through the high separation low uncertainty zone. We have five possible motion primitives, as represented in Fig. 6.25b with a vision cone of 120° emanating from the camera. We explore the planning decision tree using Monte Carlo Tree Search with a horizon length $L = 10$ at each step, then perform the action with the highest reward. The setting for the classifier model, viewing radius and angle, motion, and geometric noise are the same as in the inference simulation. We use 10 hybrid beliefs for MH. The trajectory length is 20 time steps.

Fig 6.26 presents the ground truth trajectories calculated by performing planning over R_1 and R_2 both for JLP-BSP and MH-BSP, and planning for R_2 for WEU. It is evident that the epistemic-uncertainty-aware methods seek to pass near the blue-cone area for more accurate classification with lower epistemic uncertainty. Methods that plan over R_1 tend to pass through the cone.



(a) (b)

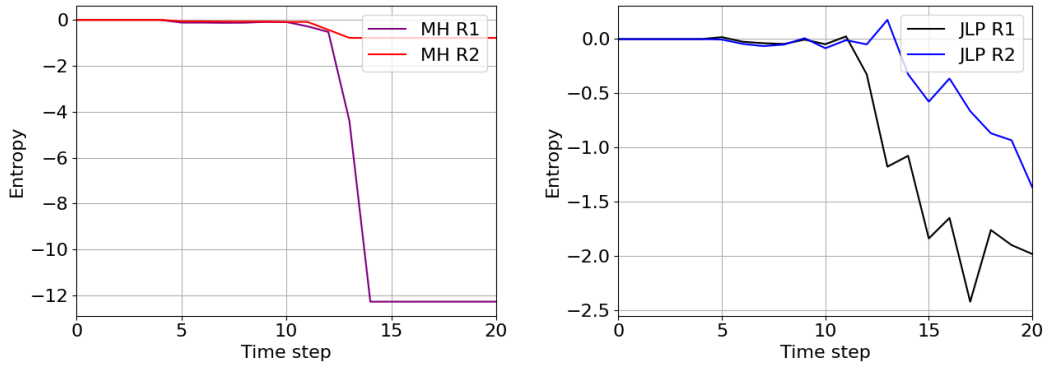
Figure 6.25: **(a)** is the ground truth of the scenario in Sec. 5.B. The red dot represents the robot's starting point. The green dots represent the objects' location with the corresponding object labels. The green line represents the object orientation, with the yellow line present 90° of that orientation. The blue cones represent the observation angles in which the object are classified most accurately with the lowest epistemic uncertainty. **(b)** presents the five motion primitives in the scenario. The red dot represents the origin point, the black arrows the possible actions, and the blue cone is the field of view after the action.



(a) R_1 (b) R_2 (c) WEU

Figure 6.26: This figure presents the ground truth of a planned trajectories. **(a)** for planning over R_1 for MH-BSP (purple) and JLP-BSP (black). **(b)** for planning over R_2 for MH-BSP (red) and JLP-BSP (blue). **(b)** for WEU (green). All are for the multiple object scenario. The object is shown in a green dot, with the green line representing the object orientation, with the yellow line present 90° of that orientation. The blue cones represent the areas where observations have the lowest epistemic uncertainty.

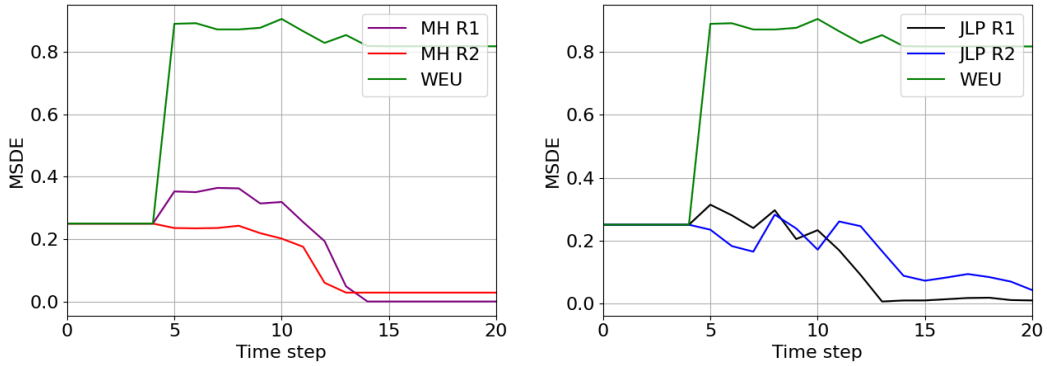
The behavior presented in Fig 6.26 is reflected in Fig. 6.27 where the values of $H(\lambda_k)$ are shown as a function of time during inference after the corresponding action has been performed. The values of $H(\lambda_k)$ correlate to the epistemic uncertainty. Evidently, planning over R_1 yields lower epistemic uncertainty for both MH-BSP (Fig. 6.27a) and JLP-BSP (Fig. 6.27b).



(a) (b)

Figure 6.27: $H(\lambda_{20})$ values for MH **(a)** and JLP **(b)** as a function of the time step. In **(a)**, the purple and red plots represent R_1 and R_2 respectively, and similarly in **(a)** , the black and blue plots represent R_1 and R_2 respectively.

Fig. 6.28 presents MSDE results for all the methods, split into results for MH in Fig. 6.28a and for JLP in Fig. 6.28b, both showing comparison to WEU in the green plot. WEU performs significantly worse in this setting than all the other methods. When comparing planning over R_1 and R_2, the first presents better results than the latter for both JLP and MH.



(a) (b)

Figure 6.28: MSDE values as a function of time step for MH **(a)** and JLP **(b)**, compared to WEU. In **(a)**, the purple and red plots represent R_1 and R_2 respectively, and similarly in **(a)** , the black and blue plots represent R_1 and R_2 respectively. WEU is represented in both figure with a green plot.

Fig. 6.29 presents the results at time $k = 20$ for all methods as a bar graph with error margins for the ground truth class. We can compare the entropy from Fig. 6.26 and MSDE from Fig. 6.28 with the bar graphs, with lower entropy values resulting in smaller posterior epistemic uncertainty. Similarly, lower MSDE values result in a more

"certain" result in the bar graph, as we can see for methods that plan over R_1.

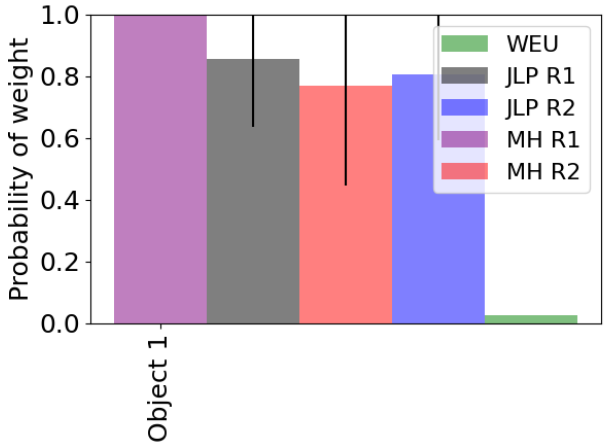


Figure 6.29: Probability of the object being class $c = 2$ (ground truth) for our methods at time $k = 20$. We compare planning over R_1 and R_2, JLP-BSP, MH-BSP, and WEU. Purple and red for R_1 and R_2 respectively using MH-BSP, black and blue for using for R_1 and R_2 respectively using JLP-BSP, and green for WEU. The one σ deviation is represented via the black line at each relevant bar, and represents the posterior model uncertainty.

In Fig. 6.30 we perform computation time comparisons between WEU, MH-BSP and JLP-BSP. The significant advantage in computational time for JLP-BSP is evident against MH-BSP, and while WEU is lower still, JLP-BSP also opens the possibility of reasoning about epistemic uncertainty.

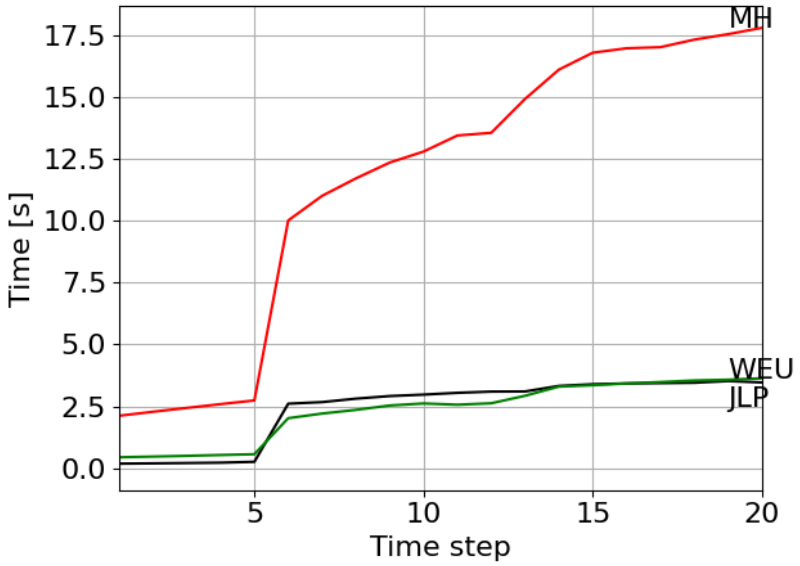


Figure 6.30: This figure compares run-time per inference step between realizations of MH with 5 hybrid beliefs in red, JLP in black, and WEU in green.

Planning: Single Run, Multiple Objects

We simulate a planning scenario of 9 objects, where they formed in a way that there are 3 zones of low uncertainty high expected classification scores, as shown in Fig. 6.31a. Reward function R_1 is now modified to include a cap of $R_{max} = 5$ per object to to encourage exploration and classification of all objects in the scene. We modify R_1 and R_2 to include all objects by summing the entropy of each marginal λ_{k+1} per object. All in all, the explicit expression for the cost functions for a future $b[\lambda_k]$ is:

$$\begin{aligned} R_1 &= \sum_o \min(-H(\lambda^o_{k+1}), R_{max}) \\ R_2 &= -\sum_o H(\mathbb{E}(\lambda^o_{k+1})) \end{aligned} \tag{6.109}$$

Optimally, the robot would plan to go through all three zones to achieve accurate classification of all objects. As in Sec. 5.B, We have five possible motion primitive, as presented in Fig. 6.31b with a cone of vision of 120° emanating from the camera. We use MCTS for a horizon $L = 10$. We use 10 hybrid beliefs for MH-BSP. The trajectory length is 20 time steps. As in the previous section, we plan for R_1 with MH-BSP and JLP-BSP, and for R_2 with MH-BSP, JLP-BSP, and WEU.

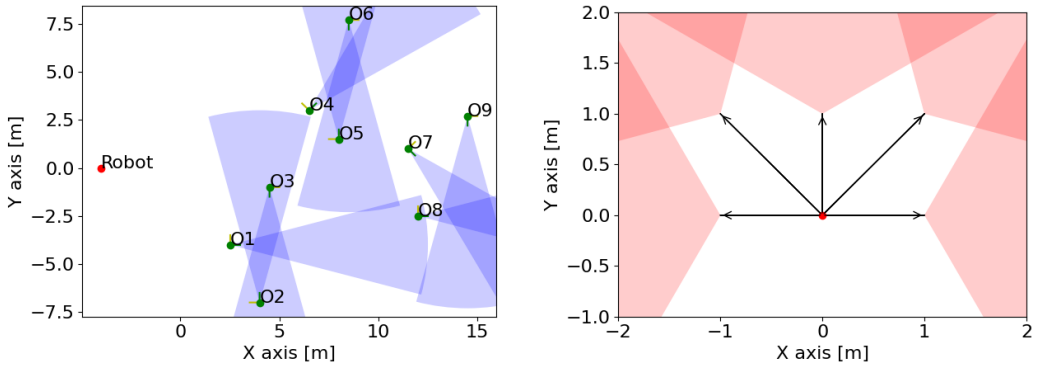
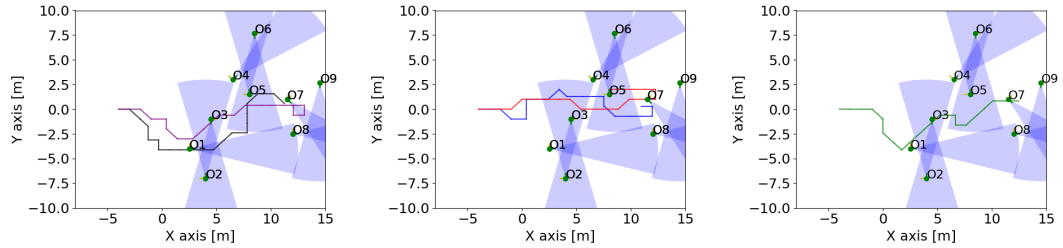


Figure 6.31: **(a)** is the ground truth of the scenario in Sec. 5.B. The red dot represents the robot's starting point. The green dots represent the objects' location with the corresponding object labels. The green line represents the object orientation, with the yellow line present 90° of that orientation. The blue cones represent the observation angles in which the objects are classified most accurately with the lowest epistemic uncertainty, with 3 overlapping areas as low epistemic uncertainty areas. **(b)** presents the five motion primitives in the scenario. The red dot represents the origin point, the black arrows the possible actions, and the blue cone is the field of view after the action.

Fig. 6.32 presents the trajectories created for all the methods. The ones that plan over R_1 create trajectories pass closer to the overlapping low uncertainty areas from Fig. 6.31a, resulting eventually in more accurate classification compared to planning over R_2 for all methods, especially WEU.

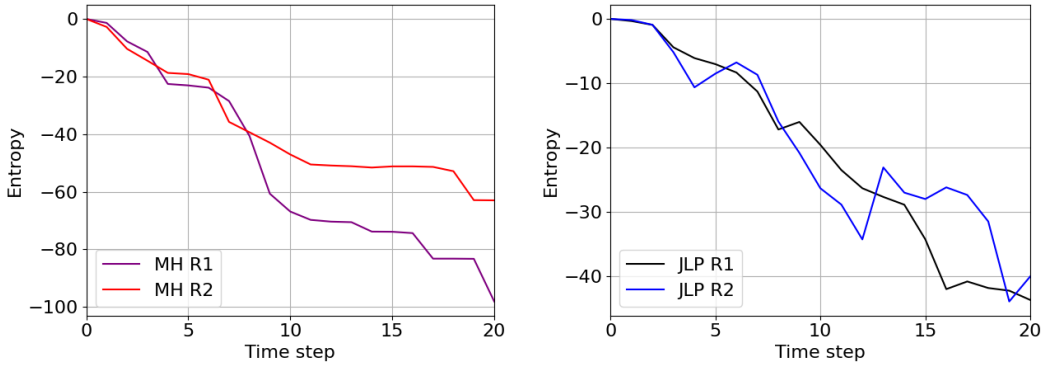
Fig. 6.33 presents a comparison for $H(\lambda_k)$ at the inference phase, when comparing planning over R_1, and R_2 for MH-BSP in Fig. 6.33a and JLP-BSP in Fig. 6.33b. In both figures planning over R_1 yields lower entropy, correlating to lower epistemic uncertainty.



(a) R_1 **(b)** R_2 **(c)** WEU

Figure 6.32: This figure presents the ground truth of a planned trajectories. **(a)** for planning over R_1 for MH-BSP (purple) and JLP-BSP (black). **(b)** for planning over R_2 for MH-BSP (red) and JLP-BSP (blue). **(b)** for WEU (green). All are for the multiple object scenario. The object is shown in a green dot, with the green line representing the object orientation, with the yellow line present 90° of that orientation. The blue cones represent the areas where observations have the lowest epistemic uncertainty.

The effect is more noticeable for MH-BSP than JLP-BSP.



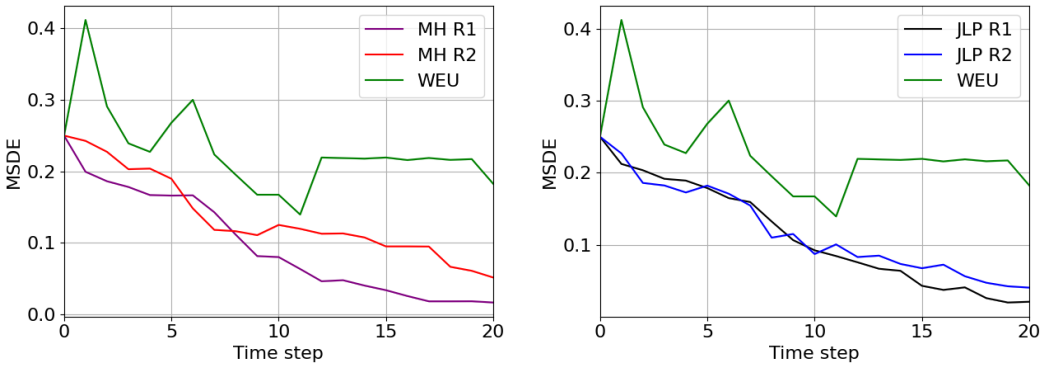
(a) **(b)**

Figure 6.33: $H(\lambda_{20})$ values for MH-BSP **(a)** and JLP-BSP **(b)** as a function of the time step. In **(a)**, the purple and red plots represent R_1 and R_2 respectively, and similarly in **(a)** , the black and blue plots represent R_1 and R_2 respectively.

Fig. 6.34 presents MSDE results for all the methods, split into results for MH-BSP in Fig. 6.34a and for JLP-BSP in Fig. 6.34b, both showing comparison to WEU in the green plot. As in Sec 5.B, planning over R_1 slightly outperforms planning over R_2, with WEU lagging far behind.

Fig. 6.35 presents a bar-graph with error representation of the classification results at time $k = 20$ for all objects. In general, planning over R_1 tend to have more accurate classification compared to planning over R_2 with lower uncertainty. On the other hand, WEU tends to go towards extremes of class probabilities 0 or 1, whether it is the correct class or not.

In Fig. 6.36 we present the computational time per step for all our approaches using R_2 reward function. For MH-BSP, we used 10 hybrid beliefs. This figure shows that JLP-BSP is slightly faster than WEU while also reasoning about posterior epistemic uncertainty, because the number of states in JLP-BSP scales linearly with the number of objects and candidate classes, as opposed to exponentially with WEU and MH-BSP. As in Sec. 5.B, JLP-BSP is significantly more computationally efficient than MH-BSP.



(a) (b)

Figure 6.34: Single-run study for multiple object scenario study for MSDE comparing planning over R_1 and R_2 for MH-BSP (**(a)**) and JLP-BSP (**(b)**), and WEU.

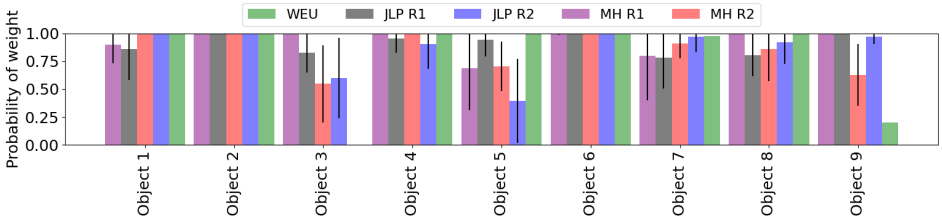


Figure 6.35: Probability of the objects being ground truth class for our methods at time $k = 20$ for all objects. We compare planning over R_1 and R_2, JLP-BSP, MH-BSP, and WEU. Purple and red for R_1 and R_2 respectively using MH, black and blue for using for R_1 and R_2 respectively using JLP-BSP, and green for WEU. The one σ deviation is represented via the black line at each relevant bar, and represents the posterior model uncertainty.

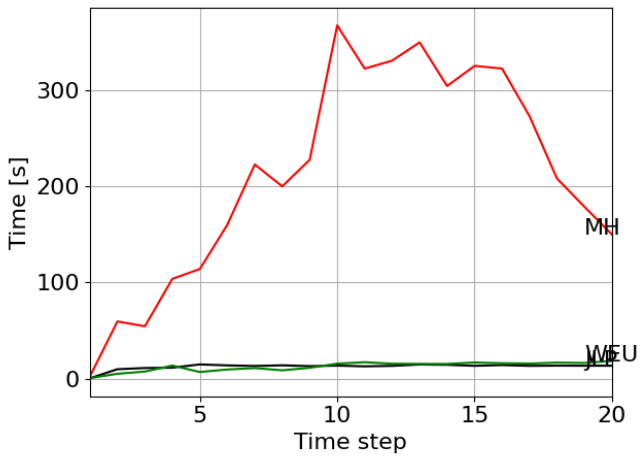
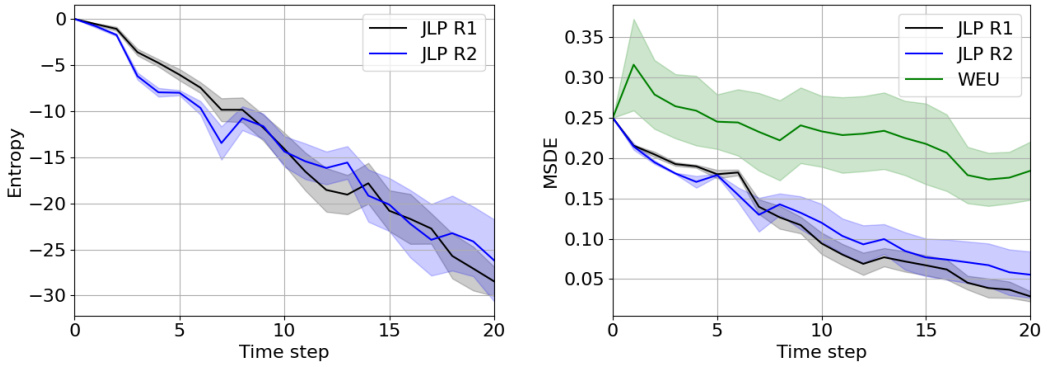


Figure 6.36: This figure compares run-time per inference step between realizations of MH-BSP with 5 hybrid beliefs in red, JLP-BSP in black, and WEU in green.

Planning: Statistical Study

For the statistical study, we randomly corrupt geometric and semantic measurements with noise. We use the scenario from Sec. 5.B, using R_1, and R_2 with JLP-BSP, and compare it to WEU. We perform 10 iteration, each with a planning horizon $L = 10$, and present results for entropy and MSDE. Each run was performed to 20 time-steps.

Fig. 6.37 presents the statistical results for the sum of the entropy in Fig. 6.37a, and the MSDE results in Fig. 6.37b, with the colored areas representing one σ deviation. All in all, planning over R_1 performs better over planning over R_2 for JLP-BSP, with lower entropy and MSDE. In addition, MSDE results compared to WEU are vastly superior for epistemic-uncertainty-aware methods.



(a) (b)

Figure 6.37: Statistical study for the scenario in Sec. 5.B **(a)** Presents an comparison for the sum of entropy over all objects between trajectories for R_1 and R_2 as a function of time step for MH-BSP. **(b)** presents an MSDE comparison between MH-BSP, JLP-BSP, and WEU as a function of time step. In both, the line represent the statistical expectation, while the colored area represents a one σ deviation.

5.C Experiment

Setup

For the experiment, we consider a myopic planning scenario in a semantic SLAM setting, using Active Vision Dataset (AVD) [113] Home 005 with example images presented in Fig. 6.38. In this scenario, the objects are grouped to two groups, one on a table near the window back-lit by sunlight as seen in Fig. 6.38a, and another on the kitchen counter seen in Fig. 6.38b. We perform planning for a 20 time step trajectory, at each step performing myopic planning. We aim to compare between JLP-BSP and WEU for classification accuracy using MSDE (6.102), differential entropy representing epistemic uncertainty, and computational time. The reward functions R_1 and R_2 are identical to those presented in Eq. (6.109).

We consider five candidate classes: "Packet", "Book Jacket", "Pop Bottle", "Digital Clock", and "Soap Dispenser". For each class, we trained classifier uncertainty models using images from BigBIRD dataset [118], with example images presented in Fig. 6.39.

(a) (b) (c)

Figure 6.38: [Experiment: example images with bounding boxes.]Example images of the Active Vision Dataset, home 005. The red boxes represent the bounding boxes for the objects, and the notation Ox represent the x'th object.

For classification, we used VGG convolutional neural network [2] with dropout activated during test time. The R_1 upper limit R_{max} per object is 500, as the increase number of objects increases the scale of R_1 values; Recall Lemma 6.4.1, the entropy depends on the covariance of $l\gamma$ via $H(l\lambda)$.

(a) (b) (c) (d)

Figure 6.39: Example images of the BigBIRD dataset for training the classifier models. **(a)** is an example for "pop bottle" class, while the rest are examples for "packet".

The classifier models were trained via PyTorch on fully connected networks. Recall Eq. (6.15), we train $h_c(x^{rel})$ and $\Sigma_c(x^{rel})$ from a dataset $D_c = \{x^{rel}, \{l\gamma\}\}$ per object, where $x^{rel} = [\psi, \theta]$ is parametrized by relative yaw angle ψ and relative pitch angle θ. h_c and Σ_c are represented by separate neural networks, up to a total of $2m$ networks. As seen in Sec. 3.C, all $\Sigma_{c=i} \equiv \Sigma_{c=j}$ for $i, j = 1, ..., m-1$ for the JLP factor to be Gaussian. This constraint limits the expressibility of Σ_c, thus not accurately representing the epistemic uncertainty from certain viewpoints of objects. As such, instead of enforcing a hard constraint on all Σ_c, we train the classifier uncertainty model with a loss function that imposes a penalty if Σ_c for different c are not similar, enforcing a soft constraint.

The loss function L_h for the h_c network is mean square error (MSE):

$$L_h(h_c, \{l\gamma\}) = MSE(h_c, \{l\gamma\}) = \sum_{i=1}^{m} \left(h_c^i - \mathbb{E}(l\gamma^i)\right)^2, \tag{6.110}$$

where h_c^i is the i'th element of h_c. The loss function L_Σ for the Σ_c uses MSE over the covariance matrix elements, and adds a Forbenius norm term that acts as the soft constraint that makes the values of Σ_c closer:

$$L_\Sigma(h_c, \Sigma_c, \{l\gamma\}) = MSE(\Sigma_c, \Sigma(l\gamma)) + \kappa \cdot F_N(h_c, \Sigma_c) \tag{6.111}$$

where the MSE for the above loss function is defined:

$$MSE(\Sigma_c, \Sigma(l\gamma)) = \frac{1}{(m-1)^2} \sum_{i=1}^{m} \sum_{j=1}^{m} ([\Sigma_c]_{ij} - [\Sigma(l\gamma)]_{ij})^2, \tag{6.112}$$

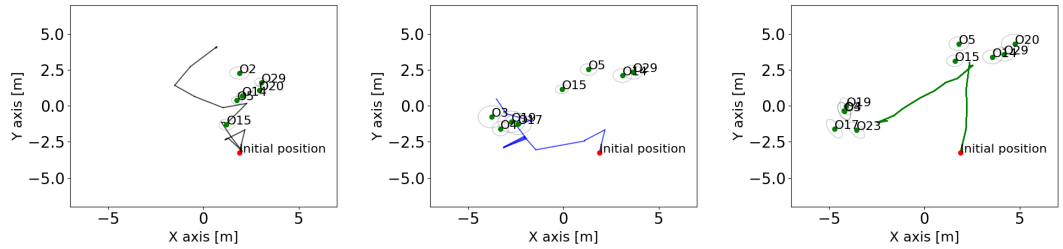
$F_N(\cdot)$ is the Forbenius Norm, defined:

$$F_N(\Sigma_c) = Tr\left((\Sigma_{c=i}^{-1} - \Sigma_{c=m}^{-1}) \cdot (\Sigma_{c=i}^{-1} - \Sigma_{c=m}^{-1})^T\right), \tag{6.113}$$

and κ is a positive constant. In our case, $\kappa = 0.005$.

Results

Fig. 6.40 presents the paths created by the planning session. The path for planning over R_1 focuses on the object group on the kitchen counter, while the others focus more on the object on the table by the window. This can be explained by poorer visibility of the objects near the window, induced by the sunlight, therefore inducing higher epistemic uncertainty than the objects on the counter.



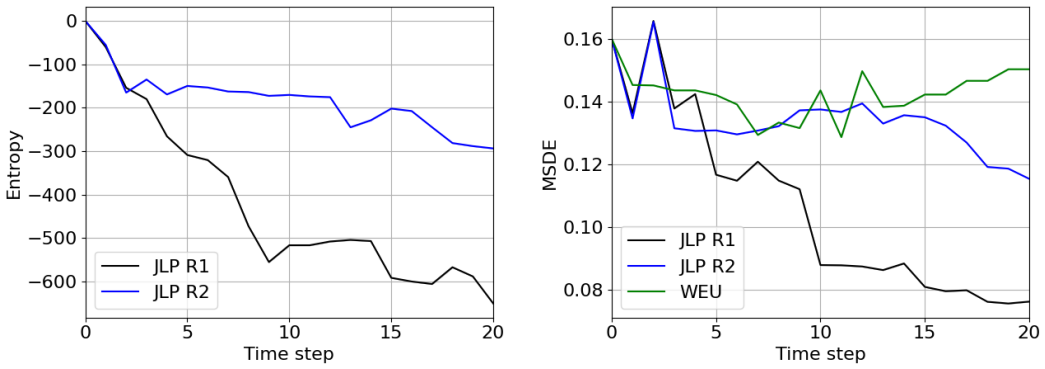
(a) R_1 **(b)** R_2 **(c)** WEU

Figure 6.40: This figure presents the ground truth of a planned trajectories with object pose estimations. **(a)** for planning over R_1 for JLP-BSP in black. **(b)** for planning over R_2 for JLP-BSP in blue. **(c)** for WEU (green). All for the AVD scenario. The object estimation is shown in a green dot with corresponding estimation covariance of 3σ in gray. The red dots represent the starting position of each trajectory.

The results of those trajectories chosen can be seen in Fig. 6.41, where the entropy and MSDE results are presented. In Fig. 6.41a the lower epistemic uncertainty for planning with R_2 can be evident. In addition, the MSDE comparison in Fig. 6.41b significantly favors planning over R_1 over R_2 and especially compared to WEU, with epistemic-uncertainty-aware planning outperforms both.

Fig. 6.42 shows the class probability of the ground truth class for all the objects for time-step $k = 20$. While both JLP-BSP with R_2 and WEU observe an object more as the group near the window contains more objects, the objects that JLP-BSP with R_1 observes are classified more accurately.

Fig. 6.43 presents a computational time comparison between JLP-BSP and WEU. The figure shows a significant advantage for JLP-BSP over WEU, as this time the number of candidate classes is 5, instead of 2 in the simulation. WEU computational time per step drops with time steps as some class realization are pruned. As evident



(a) (b)

Figure 6.41: Experimental results for the scenario in Sec. 5.C **(a)** Presents an comparison for the sum of entropy over all objects between trajectories for R_1 and R_2 as a function of time step for JLP-BSP. **(b)** presents an MSDE comparison between JLP-BSP, and WEU as a function of time step.

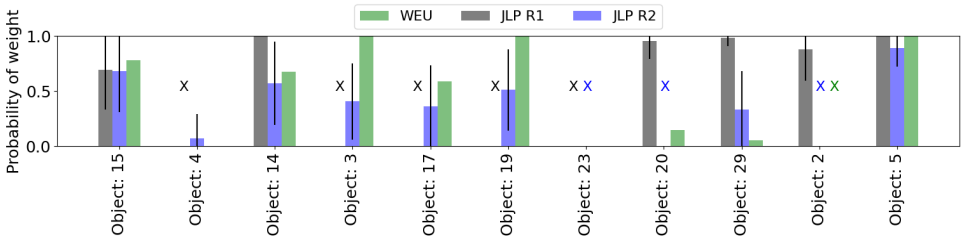


Figure 6.42: Probability of the objects being ground truth class for our methods at time $k = 20$ for all objects. We compare planning over R_1 and R_2, JLP-BSP,and WEU. Black and blue for using for R_1 and R_2 respectively using JLP-BSP, and green for WEU. The one σ deviation is represented via the black line at each relevant bar, and represents the posterior model uncertainty. The colored X marks represent that object wasn't observed by the corresponding method.

from the figure, JLP-BSP offers computational efficiency greater than WEU, while also opening access to model uncertainty, both for inference and planning.

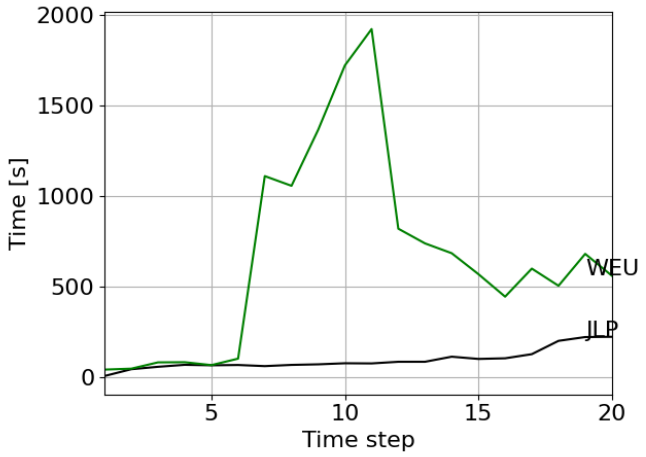


Figure 6.43: This figure compares run-time per inference step between realizations of JLP-BSP in black, and WEU in green for the AVD scenario.

Chapter 7

Conclusion and possible future research

In this thesis we addressed uncertainties in object classification, namely classification aliasing induced by observing objects from certain viewpoints, and classifier epistemic uncertainty. For addressing classification aliasing we proposed a semantic SLAM approach that maintains a hybrid belief over object poses and classes. For semantic measurements, we utilized an output of a class probability vector per object. We leveraged the coupling between poses and classes using a viewpoint dependent classifier model to assist in data association disambiguation. Eventually we presented in simulation that utilizing a viewpoint dependent classifier model assist in data association disambiguation, as well as enhancing SLAM performance.

Afterwards, we extended the approach to a distributed multi-robot setting, assuming solved data-association. For maintaining a consistent distributed hybrid belief estimation, we proposed an approach that maintains an individual belief and a joint belief simultaneously. In this approach the robots communicate the individual beliefs and store them in a stack, updating the older belief while removing old information. This way, the robots avoid double counting to keep the estimation consistent. We presented the advantage of our approach in term of classification and SLAM performance over the single robot approach and the approach that doesn't address double counting, both in simulation and experiment.

Then, we presented a sequential classification approach that infers the belief over posterior class probability, representing posterior uncertainty. We utilized an epistemic-uncertainty-aware classifier that provides a point cloud of probability vectors that represent the epistemic uncertainty from a single image. We computed a posterior class probability point cloud given measurements and a prior class probability point cloud. We showed in simulation and experiment the advantage of our approach against approaches that do not reason about epistemic uncertainty.

Finally, we presented two epistemic-uncertainty-aware approaches that address both classification aliasing and epistemic uncertainty in a unified inference and BSP semantic

SLAM setting. The first is MH, which maintains several hybrid belief simultaneously, each with semantic measurements of class probability vectors that correspond to different classifier weights. This approach proves computationally expensive, so we proposed a second method: JLP, which uses the novel JLP factor to maintain a single continuous belief that also reasons about posterior epistemic uncertainty. For the active BSP setting, we discussed the sampling of predicted measurements for both MH and JLP, namely the advantage of sampling from a viewpoint dependent classifier uncertainty model compared to generating raw images. The classifier uncertainty model is a modification of the viewpoint dependent model used previously that is able to generate probability vector, in addition to participate inference. We proposed a novel type of information-theoretic reward function over the belief of posterior class probability, and demonstrated in simulation and using Active Vision Dataset the advantages of planning over this reward function for classification. In addition, we showed the advantages in object classification of using epistemic-uncertainty-aware semantic SLAM.

7.1 Possible Future Research

Future research may reason about improving the computational efficiency of hybrid beliefs over poses and classes, either via parallel computation, incremental inference, utilizing simplifying assumptions that allows reducing computational effort, or smart hybrid belief component pruning schemes. JLP provides a promising direction for including epistemic-uncertainty-aware classification in SLAM, but still requires improvement for computational effort, e.g. sparsification. In addition, scenarios with many candidate classes per object may present challenges both for hybrid-belief-based methods and JLP, so addressing that concern may be a future research direction as well.

Appendix A

Communication Tables for Distributed Semantic SLAM

A.1 Communication Table for Distributed Semantic SLAM Simulation

In this section we present a table of stack time stamps that indicates direct and indirect communication between robots in our scenario. Recall that the maximal communication radius is 10 meters, thus robots r_2 and r_3 communicate from time $k = 6$, robots r_1 starts communicating to others from time $k = 13$.

Time step	Stack of r_1	Stack of r_2	Stack of r_3	Time step	Stack of r_1	Stack of r_2	Stack of r_3
	t(r_1): 1	t(r_1): 0	t(r_1): 0		t(r_1): 16	t(r_1): 15	t(r_1): 15
$k = 1$	t(r_2): 0	t(r_2): 0	t(r_2): 0	$k = 16$	t(r_2): 15	t(r_2): 16	t(r_2): 15
	t(r_3): 0	t(r_3): 0	t(r_3): 1		t(r_3): 15	t(r_3): 15	t(r_3): 16
	t(r_1): 2	t(r_1): 0	t(r_1): 0		t(r_1): 17	t(r_1): 16	t(r_1): 16
$k = 2$	t(r_2): 0	t(r_2): 0	t(r_2): 0	$k = 17$	t(r_2): 16	t(r_2): 17	t(r_2): 16
	t(r_3): 0	t(r_3): 0	t(r_3): 2		t(r_3): 16	t(r_3): 16	t(r_3): 17
	t(r_1): 3	t(r_1): 0	t(r_1): 0		t(r_1): 18	t(r_1): 17	t(r_1): 17
$k = 3$	t(r_2): 0	t(r_2): 3	t(r_2): 0	$k = 18$	t(r_2): 17	t(r_2): 18	t(r_2): 17
	t(r_3): 0	t(r_3): 0	t(r_3): 3		t(r_3): 17	t(r_3): 17	t(r_3): 18
	t(r_1): 4	t(r_1): 0	t(r_1): 0		t(r_1): 19	t(r_1): 18	t(r_1): 18
$k = 4$	t(r_2): 0	t(r_2): 4	t(r_2): 0	$k = 19$	t(r_2): 18	t(r_2): 19	t(r_2): 18
	t(r_3): 0	t(r_3): 0	t(r_3): 4		t(r_3): 18	t(r_3): 18	t(r_3): 19
	t(r_1): 5	t(r_1): 0	t(r_1): 0		t(r_1): 20	t(r_1): 19	t(r_1): 19
$k = 5$	t(r_2): 0	t(r_2): 5	t(r_2): 0	$k = 20$	t(r_2): 19	t(r_2): 20	t(r_2): 19
	t(r_3): 0	t(r_3): 0	t(r_3): 5		t(r_3): 19	t(r_3): 19	t(r_3): 20
	t(r_1): 6	t(r_1): 0	t(r_1): 0		t(r_1): 21	t(r_1): 20	t(r_1): 20
$k = 6$	t(r_2): 0	t(r_2): 6	t(r_2): 5	$k = 21$	t(r_2): 20	t(r_2): 21	t(r_2): 20
	t(r_3): 0	t(r_3): 5	t(r_3): 6		t(r_3): 20	t(r_3): 20	t(r_3): 21
	t(r_1): 7	t(r_1): 0	t(r_1): 0		t(r_1): 22	t(r_1): 21	t(r_1): 21
$k = 7$	t(r_2): 0	t(r_2): 7	t(r_2): 6	$k = 22$	t(r_2): 21	t(r_2): 22	t(r_2): 21
	t(r_3): 0	t(r_3): 6	t(r_3): 7		t(r_3): 21	t(r_3): 21	t(r_3): 22
	t(r_1): 8	t(r_1): 0	t(r_1): 0		t(r_1): 23	t(r_1): 22	t(r_1): 22
$k = 8$	t(r_2): 0	t(r_2): 8	t(r_2): 7	$k = 23$	t(r_2): 22	t(r_2): 23	t(r_2): 22
	t(r_3): 0	t(r_3): 7	t(r_3): 8		t(r_3): 22	t(r_3): 22	t(r_3): 23
	t(r_1): 9	t(r_1): 0	t(r_1): 0		t(r_1): 24	t(r_1): 23	t(r_1): 23
$k = 9$	t(r_2): 0	t(r_2): 9	t(r_2): 8	$k = 24$	t(r_2): 23	t(r_2): 24	t(r_2): 23
	t(r_3): 0	t(r_3): 8	t(r_3): 9		t(r_3): 23	t(r_3): 23	t(r_3): 24
	t(r_1): 10	t(r_1): 0	t(r_1): 0		t(r_1): 25	t(r_1): 24	t(r_1): 24
$k = 10$	t(r_2): 0	t(r_2): 10	t(r_2): 9	$k = 25$	t(r_2): 24	t(r_2): 25	t(r_2): 24
	t(r_3): 0	t(r_3): 9	t(r_3): 10		t(r_3): 24	t(r_3): 24	t(r_3): 25
	t(r_1): 11	t(r_1): 0	t(r_1): 0		t(r_1): 26	t(r_1): 25	t(r_1): 25
$k = 11$	t(r_2): 0	t(r_2): 11	t(r_2): 10	$k = 26$	t(r_2): 25	t(r_2): 26	t(r_2): 25
	t(r_3): 0	t(r_3): 10	t(r_3): 11		t(r_3): 25	t(r_3): 25	t(r_3): 26
	t(r_1): 12	t(r_1): 0	t(r_1): 0		t(r_1): 27	t(r_1): 26	t(r_1): 26
$k = 12$	t(r_2): 0	t(r_2): 12	t(r_2): 11	$k = 27$	t(r_2): 26	t(r_2): 27	t(r_2): 26
	t(r_3): 0	t(r_3): 11	t(r_3): 12		t(r_3): 26	t(r_3): 26	t(r_3): 27
	t(r_1): 13	t(r_1): 12	t(r_1): 12		t(r_1): 28	t(r_1): 27	t(r_1): 27
$k = 13$	t(r_2): 12	t(r_2): 13	t(r_2): 12	$k = 28$	t(r_2): 27	t(r_2): 28	t(r_2): 27
	t(r_3): 12	t(r_3): 12	t(r_3): 13		t(r_3): 27	t(r_3): 27	t(r_3): 28
	t(r_1): 14	t(r_1): 13	t(r_1): 13		t(r_1): 29	t(r_1): 28	t(r_1): 28
$k = 14$	t(r_2): 13	t(r_2): 14	t(r_2): 13	$k = 29$	t(r_2): 28	t(r_2): 29	t(r_2): 28
	t(r_3): 13	t(r_3): 13	t(r_3): 14		t(r_3): 28	t(r_3): 28	t(r_3): 29
	t(r_1): 15	t(r_1): 14	t(r_1): 14		t(r_1): 30	t(r_1): 29	t(r_1): 29
$k = 15$	t(r_2): 14	t(r_2): 15	t(r_2): 14	$k = 30$	t(r_2): 29	t(r_2): 30	t(r_2): 29
	t(r_3): 14	t(r_3): 14	t(r_3): 15		t(r_3): 29	t(r_3): 29	t(r_3): 30

Time step	Stack of r_1	Stack of r_2	Stack of r_3	Time step	Stack of r_1	Stack of r_2	Stack of r_3
$k = 31$	t(r_1): 31 t(r_2): 30 t(r_3): 30	t(r_1): 30 t(r_2): 31 t(r_3): 30	t(r_1): 30 t(r_2): 30 t(r_3): 31	$k = 46$	t(r_1): 46 t(r_2): 45 t(r_3): 45	t(r_1): 45 t(r_2): 46 t(r_3): 45	t(r_1): 45 t(r_2): 45 t(r_3): 46
$k = 32$	t(r_1): 32 t(r_2): 31 t(r_3): 31	t(r_1): 31 t(r_2): 32 t(r_3): 31	t(r_1): 31 t(r_2): 31 t(r_3): 32	$k = 47$	t(r_1): 47 t(r_2): 46 t(r_3): 46	t(r_1): 46 t(r_2): 47 t(r_3): 46	t(r_1): 46 t(r_2): 46 t(r_3): 47
$k = 33$	t(r_1): 33 t(r_2): 32 t(r_3): 32	t(r_1): 32 t(r_2): 33 t(r_3): 32	t(r_1): 32 t(r_2): 32 t(r_3): 33	$k = 48$	t(r_1): 48 t(r_2): 47 t(r_3): 47	t(r_1): 47 t(r_2): 48 t(r_3): 47	t(r_1): 47 t(r_2): 47 t(r_3): 48
$k = 34$	t(r_1): 34 t(r_2): 33 t(r_3): 33	t(r_1): 33 t(r_2): 34 t(r_3): 33	t(r_1): 33 t(r_2): 33 t(r_3): 34	$k = 49$	t(r_1): 49 t(r_2): 48 t(r_3): 48	t(r_1): 48 t(r_2): 49 t(r_3): 48	t(r_1): 48 t(r_2): 48 t(r_3): 49
$k = 35$	t(r_1): 35 t(r_2): 34 t(r_3): 34	t(r_1): 34 t(r_2): 35 t(r_3): 34	t(r_1): 34 t(r_2): 34 t(r_3): 35	$k = 50$	t(r_1): 50 t(r_2): 49 t(r_3): 49	t(r_1): 49 t(r_2): 50 t(r_3): 49	t(r_1): 49 t(r_2): 49 t(r_3): 50
$k = 36$	t(r_1): 36 t(r_2): 35 t(r_3): 35	t(r_1): 35 t(r_2): 36 t(r_3): 35	t(r_1): 35 t(r_2): 35 t(r_3): 36	$k = 51$	t(r_1): 51 t(r_2): 50 t(r_3): 50	t(r_1): 50 t(r_2): 51 t(r_3): 50	t(r_1): 50 t(r_2): 50 t(r_3): 51
$k = 37$	t(r_1): 37 t(r_2): 36 t(r_3): 36	t(r_1): 36 t(r_2): 37 t(r_3): 36	t(r_1): 36 t(r_2): 36 t(r_3): 37	$k = 52$	t(r_1): 52 t(r_2): 51 t(r_3): 51	t(r_1): 51 t(r_2): 52 t(r_3): 51	t(r_1): 51 t(r_2): 51 t(r_3): 52
$k = 38$	t(r_1): 38 t(r_2): 37 t(r_3): 37	t(r_1): 37 t(r_2): 38 t(r_3): 37	t(r_1): 37 t(r_2): 37 t(r_3): 38	$k = 53$	t(r_1): 53 t(r_2): 52 t(r_3): 52	t(r_1): 52 t(r_2): 53 t(r_3): 52	t(r_1): 52 t(r_2): 52 t(r_3): 53
$k = 39$	t(r_1): 39 t(r_2): 38 t(r_3): 38	t(r_1): 38 t(r_2): 39 t(r_3): 38	t(r_1): 38 t(r_2): 38 t(r_3): 39	$k = 54$	t(r_1): 54 t(r_2): 53 t(r_3): 53	t(r_1): 53 t(r_2): 54 t(r_3): 53	t(r_1): 53 t(r_2): 53 t(r_3): 54
$k = 40$	t(r_1): 40 t(r_2): 39 t(r_3): 39	t(r_1): 39 t(r_2): 40 t(r_3): 39	t(r_1): 39 t(r_2): 39 t(r_3): 40	$k = 55$	t(r_1): 55 t(r_2): 54 t(r_3): 54	t(r_1): 54 t(r_2): 55 t(r_3): 54	t(r_1): 54 t(r_2): 54 t(r_3): 55
$k = 41$	t(r_1): 41 t(r_2): 40 t(r_3): 40	t(r_1): 40 t(r_2): 41 t(r_3): 40	t(r_1): 40 t(r_2): 40 t(r_3): 41	$k = 56$	t(r_1): 56 t(r_2): 55 t(r_3): 55	t(r_1): 55 t(r_2): 56 t(r_3): 55	t(r_1): 55 t(r_2): 55 t(r_3): 56
$k = 42$	t(r_1): 42 t(r_2): 41 t(r_3): 41	t(r_1): 41 t(r_2): 42 t(r_3): 41	t(r_1): 41 t(r_2): 41 t(r_3): 42	$k = 57$	t(r_1): 57 t(r_2): 56 t(r_3): 56	t(r_1): 56 t(r_2): 57 t(r_3): 56	t(r_1): 56 t(r_2): 56 t(r_3): 57
$k = 43$	t(r_1): 43 t(r_2): 42 t(r_3): 42	t(r_1): 42 t(r_2): 43 t(r_3): 42	t(r_1): 42 t(r_2): 42 t(r_3): 43	$k = 58$	t(r_1): 58 t(r_2): 57 t(r_3): 57	t(r_1): 57 t(r_2): 58 t(r_3): 57	t(r_1): 57 t(r_2): 57 t(r_3): 58
$k = 44$	t(r_1): 44 t(r_2): 43 t(r_3): 43	t(r_1): 43 t(r_2): 44 t(r_3): 43	t(r_1): 43 t(r_2): 43 t(r_3): 44	$k = 59$	t(r_1): 59 t(r_2): 58 t(r_3): 58	t(r_1): 58 t(r_2): 59 t(r_3): 58	t(r_1): 58 t(r_2): 58 t(r_3): 59
$k = 45$	t(r_1): 45 t(r_2): 44 t(r_3): 44	t(r_1): 44 t(r_2): 45 t(r_3): 44	t(r_1): 44 t(r_2): 44 t(r_3): 45	$k = 60$	t(r_1): 60 t(r_2): 59 t(r_3): 59	t(r_1): 59 t(r_2): 60 t(r_3): 59	t(r_1): 59 t(r_2): 59 t(r_3): 60

A.2 Communication Table for Distributed Semantic SLAM Experiment

In this section we present a table of stack time stamps that indicates direct and indirect communication between robots in our scenario. Recall that the maximal communication radius is 3 meters, thus all robots start to communicate between them at step $k = 6$.

Time step	Stack of r_1	Stack of r_2	Stack of r_3	Time step	Stack of r_1	Stack of r_2	Stack of r_3
$k = 1$	t(r_1): 1 t(r_2): 0 t(r_3): 0	t(r_1): 0 t(r_2): 0 t(r_3): 0	t(r_1): 0 t(r_2): 0 t(r_3): 1	$k = 16$	t(r_1): 16 t(r_2): 15 t(r_3): 15	t(r_1): 15 t(r_2): 16 t(r_3): 15	t(r_1): 15 t(r_2): 15 t(r_3): 16
$k = 2$	t(r_1): 2 t(r_2): 0 t(r_3): 0	t(r_1): 0 t(r_2): 0 t(r_3): 0	t(r_1): 0 t(r_2): 0 t(r_3): 2	$k = 17$	t(r_1): 17 t(r_2): 16 t(r_3): 16	t(r_1): 16 t(r_2): 17 t(r_3): 16	t(r_1): 16 t(r_2): 16 t(r_3): 17
$k = 3$	t(r_1): 3 t(r_2): 0 t(r_3): 0	t(r_1): 0 t(r_2): 3 t(r_3): 0	t(r_1): 0 t(r_2): 0 t(r_3): 3	$k = 18$	t(r_1): 18 t(r_2): 17 t(r_3): 17	t(r_1): 17 t(r_2): 18 t(r_3): 17	t(r_1): 17 t(r_2): 17 t(r_3): 18
$k = 4$	t(r_1): 4 t(r_2): 0 t(r_3): 0	t(r_1): 0 t(r_2): 4 t(r_3): 0	t(r_1): 0 t(r_2): 0 t(r_3): 4	$k = 19$	t(r_1): 19 t(r_2): 18 t(r_3): 18	t(r_1): 18 t(r_2): 19 t(r_3): 18	t(r_1): 18 t(r_2): 18 t(r_3): 19
$k = 5$	t(r_1): 5 t(r_2): 0 t(r_3): 0	t(r_1): 0 t(r_2): 5 t(r_3): 0	t(r_1): 0 t(r_2): 0 t(r_3): 5	$k = 20$	t(r_1): 20 t(r_2): 19 t(r_3): 19	t(r_1): 19 t(r_2): 20 t(r_3): 19	t(r_1): 19 t(r_2): 19 t(r_3): 20
$k = 6$	t(r_1): 6 t(r_2): 5 t(r_3): 5	t(r_1): 5 t(r_2): 6 t(r_3): 5	t(r_1): 5 t(r_2): 5 t(r_3): 6	$k = 21$	t(r_1): 21 t(r_2): 19 t(r_3): 19	t(r_1): 19 t(r_2): 21 t(r_3): 20	t(r_1): 19 t(r_2): 20 t(r_3): 21
$k = 7$	t(r_1): 7 t(r_2): 6 t(r_3): 6	t(r_1): 6 t(r_2): 7 t(r_3): 6	t(r_1): 6 t(r_2): 6 t(r_3): 7	$k = 22$	t(r_1): 22 t(r_2): 19 t(r_3): 19	t(r_1): 19 t(r_2): 22 t(r_3): 21	t(r_1): 19 t(r_2): 21 t(r_3): 22
$k = 8$	t(r_1): 8 t(r_2): 7 t(r_3): 7	t(r_1): 7 t(r_2): 8 t(r_3): 7	t(r_1): 7 t(r_2): 7 t(r_3): 8	$k = 23$	t(r_1): 23 t(r_2): 19 t(r_3): 22	t(r_1): 21 t(r_2): 23 t(r_3): 21	t(r_1): 22 t(r_2): 21 t(r_3): 23
$k = 9$	t(r_1): 9 t(r_2): 8 t(r_3): 8	t(r_1): 8 t(r_2): 9 t(r_3): 8	t(r_1): 8 t(r_2): 8 t(r_3): 9	$k = 24$	t(r_1): 24 t(r_2): 19 t(r_3): 23	t(r_1): 21 t(r_2): 24 t(r_3): 21	t(r_1): 23 t(r_2): 21 t(r_3): 24
$k = 10$	t(r_1): 10 t(r_2): 9 t(r_3): 9	t(r_1): 9 t(r_2): 10 t(r_3): 9	t(r_1): 9 t(r_2): 9 t(r_3): 10	$k = 25$	t(r_1): 25 t(r_2): 23 t(r_3): 24	t(r_1): 21 t(r_2): 25 t(r_3): 24	t(r_1): 24 t(r_2): 24 t(r_3): 25
$k = 11$	t(r_1): 11 t(r_2): 10 t(r_3): 10	t(r_1): 10 t(r_2): 11 t(r_3): 10	t(r_1): 10 t(r_2): 10 t(r_3): 11	$k = 26$	t(r_1): 26 t(r_2): 24 t(r_3): 25	t(r_1): 24 t(r_2): 26 t(r_3): 25	t(r_1): 25 t(r_2): 25 t(r_3): 26
$k = 12$	t(r_1): 12 t(r_2): 11 t(r_3): 11	t(r_1): 11 t(r_2): 12 t(r_3): 11	t(r_1): 11 t(r_2): 11 t(r_3): 12	$k = 27$	t(r_1): 27 t(r_2): 25 t(r_3): 26	t(r_1): 25 t(r_2): 27 t(r_3): 26	t(r_1): 26 t(r_2): 26 t(r_3): 27
$k = 13$	t(r_1): 13 t(r_2): 12 t(r_3): 12	t(r_1): 12 t(r_2): 13 t(r_3): 12	t(r_1): 12 t(r_2): 12 t(r_3): 13	$k = 28$	t(r_1): 28 t(r_2): 26 t(r_3): 27	t(r_1): 26 t(r_2): 28 t(r_3): 27	t(r_1): 27 t(r_2): 27 t(r_3): 28
$k = 14$	t(r_1): 14 t(r_2): 13 t(r_3): 13	t(r_1): 13 t(r_2): 14 t(r_3): 13	t(r_1): 13 t(r_2): 13 t(r_3): 14	$k = 29$	t(r_1): 29 t(r_2): 27 t(r_3): 28	t(r_1): 27 t(r_2): 29 t(r_3): 28	t(r_1): 28 t(r_2): 28 t(r_3): 29
$k = 15$	t(r_1): 15 t(r_2): 14 t(r_3): 14	t(r_1): 14 t(r_2): 15 t(r_3): 14	t(r_1): 14 t(r_2): 14 t(r_3): 15	$k = 30$	t(r_1): 30 t(r_2): 29 t(r_3): 29	t(r_1): 29 t(r_2): 30 t(r_3): 29	t(r_1): 29 t(r_2): 29 t(r_3): 30

Time step	Stack of r_1	Stack of r_2	Stack of r_3
$k = 31$	t(r_1): 31 t(r_2): 30 t(r_3): 30	t(r_1): 30 t(r_2): 31 t(r_3): 30	t(r_1): 30 t(r_2): 30 t(r_3): 31
$k = 32$	t(r_1): 32 t(r_2): 31 t(r_3): 31	t(r_1): 31 t(r_2): 32 t(r_3): 31	t(r_1): 31 t(r_2): 31 t(r_3): 32
$k = 33$	t(r_1): 33 t(r_2): 32 t(r_3): 32	t(r_1): 32 t(r_2): 33 t(r_3): 32	t(r_1): 32 t(r_2): 32 t(r_3): 33
$k = 34$	t(r_1): 34 t(r_2): 33 t(r_3): 33	t(r_1): 33 t(r_2): 34 t(r_3): 33	t(r_1): 33 t(r_2): 33 t(r_3): 34
$k = 35$	t(r_1): 35 t(r_2): 34 t(r_3): 34	t(r_1): 34 t(r_2): 35 t(r_3): 34	t(r_1): 34 t(r_2): 34 t(r_3): 35
$k = 36$	t(r_1): 36 t(r_2): 35 t(r_3): 35	t(r_1): 35 t(r_2): 36 t(r_3): 35	t(r_1): 35 t(r_2): 35 t(r_3): 36
$k = 37$	t(r_1): 37 t(r_2): 36 t(r_3): 36	t(r_1): 36 t(r_2): 37 t(r_3): 36	t(r_1): 36 t(r_2): 36 t(r_3): 37
$k = 38$	t(r_1): 38 t(r_2): 37 t(r_3): 37	t(r_1): 37 t(r_2): 38 t(r_3): 37	t(r_1): 37 t(r_2): 37 t(r_3): 38
$k = 39$	t(r_1): 39 t(r_2): 38 t(r_3): 38	t(r_1): 38 t(r_2): 39 t(r_3): 38	t(r_1): 38 t(r_2): 38 t(r_3): 39
$k = 40$	t(r_1): 40 t(r_2): 39 t(r_3): 39	t(r_1): 39 t(r_2): 40 t(r_3): 39	t(r_1): 39 t(r_2): 39 t(r_3): 40

Bibliography

[1] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[2] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[3] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, Andrew Rabinovich, et al. Going deeper with convolutions. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[4] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[5] Sara Sabour, Nicholas Frosst, and Geoffrey E Hinton. Dynamic routing between capsules. In *Advances in Neural Information Processing Systems (NIPS)*, pages 3859–3869, 2017.

[6] A. Amini, A. Soleimany, S. Karaman, and D. Rus. Spatial uncertainty sampling for end-to-end control. *arXiv preprint arXiv:1805.04829*, 2018.

[7] Andrey Malinin and Mark Gales. Predictive uncertainty estimation via prior networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 7047–7058, 2018.

[8] A. Sharma, N. Azizan, and M. Pavone. Sketching curvature for efficient out-of-distribution detection for deep neural networks. *arXiv preprint arXiv:2102.12567*, 2021.

[9] Adam Coates and Andrew Y Ng. Multi-camera object detection for robotics. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 412–419. IEEE, 2010.

[10] Shayegan Omidshafiei, Brett T Lopez, Jonathan P How, and John Vian. Hierarchical bayesian noise inference for robust real-time probabilistic object classification. *arXiv preprint arXiv:1605.01042*, 2016.

[11] T. Patten, M. Zillich, R. Fitch, M. Vincze, and S. Sukkarieh. Viewpoint evaluation for online 3-d active object classification. *IEEE Robotics and Automation Letters (RA-L)*, 1(1):73–81, January 2016.

[12] N. Atanasov, B. Sankaran, J.L. Ny, G. J. Pappas, and K. Daniilidis. Nonmyopic view planning for active object classification and pose estimation. *IEEE Trans. Robotics*, 30:1078–1090, 2014.

[13] Sudeep Pillai and John Leonard. Monocular slam supported object recognition. In *Robotics: Science and Systems (RSS)*, 2015.

[14] Beipeng Mu, Shih-Yuan Liu, Liam Paull, John Leonard, and Jonathan How. Slam with objects using a nonparametric pose graph. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2016.

[15] WT Teacy, Simon J Julier, Renzo De Nardi, Alex Rogers, and Nicholas R Jennings. Observation modelling for vision-based target search by unmanned aerial vehicles. In *Intl. Conf. on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 1607–1614, 2015.

[16] Y. Feldman and V. Indelman. Bayesian viewpoint-dependent robust classification under model and localization uncertainty. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2018.

[17] D. Kopitkov and V. Indelman. Robot localization through information recovered from cnn classificators. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*. IEEE, October 2018.

[18] G. Paass. Assessing and improving neural network predictions by the bootstrap algorithm. In *Advances in Neural Information Processing Systems (NIPS)*, pages 196–203, 1993.

[19] Hugo Grimmett, Rudolph Triebel, Rohan Paul, and Ingmar Posner. Introspective classification for robot perception. *Intl. J. of Robotics Research*, 35(7):743–762, 2016.

[20] Yarin Gal and Zoubin Ghahramani. Bayesian convolutional neural networks with bernoulli approximate variational inference. *arXiv preprint arXiv:1506.02158*, 2016.

[21] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *Intl. Conf. on Machine Learning (ICML)*, 2016.

[22] Alex Kendall and Roberto Cipolla. Modelling uncertainty in deep learning for camera relocalization. *arXiv preprint arXiv:1509.05909*, 2015.

[23] Pavel Myshkov and Simon Julier. Posterior distribution analysis for bayesian inference in neural networks. In *Workshop on Bayesian Deep Learning, NIPS*, 2016.

[24] Alex Kendall and Yarin Gal. What uncertainties do we need in bayesian deep learning for computer vision? *arXiv preprint arXiv:1703.04977*, 2017.

[25] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2001.

[26] P. Felzenszwalb, R. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *IEEE Trans. Pattern Anal. Machine Intell.*, 32(9):1627–1645, 2010.

[27] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 580–587, 2014.

[28] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[29] Ross Girshick. Fast r-cnn. *arXiv preprint arXiv:1504.08083*, 2015.

[30] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: towards real-time object detection with region proposal networks. *IEEE Trans. Pattern Anal. Machine Intell.*, 39(6):1137–1149, 2017.

[31] V. Ila, J. M. Porta, and J. Andrade-Cetto. Information-based compact Pose SLAM. *IEEE Trans. Robotics*, 26(1):78–93, 2010. In press.

[32] F. Dellaert and M. Kaess. Square Root SAM: Simultaneous localization and mapping via square root information smoothing. *Intl. J. of Robotics Research*, 25(12):1181–1203, Dec 2006.

[33] D. Benson M. Svedman J. Ostrowski N. Karlsson P. Pirjanian L. Goncalves, E.D. Bernardo. A visual front-end for simultaneous localization and mapping. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pages 44–49, Apr 2005.

[34] N. Sunderhauf and P. Protzel. Towards a robust back-end for pose graph slam. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pages 1254–1261. IEEE, 2012.

[35] M. Kaess, A. Ranganathan, and F. Dellaert. iSAM: Incremental smoothing and mapping. *IEEE Trans. Robotics*, 24(6):1365–1378, Dec 2008.

[36] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. Leonard, and F. Dellaert. iSAM2: Incremental smoothing and mapping using the Bayes tree. *Intl. J. of Robotics Research*, 31(2):217–236, Feb 2012.

[37] Cesar Cadena, Luca Carlone, Henry Carrillo, Yasir Latif, Davide Scaramuzza, Jose Neira, Ian D Reid, and John J Leonard. Simultaneous localization and mapping: Present, future, and the robust-perception age. *IEEE Trans. Robotics*, 32(6):1309 – 1332, 2016.

[38] S. Vasudevan, S. Gachter, M. Berger, and R. Siegwart. Cognitive maps for mobile robots — an object based approach. *Robotics and Autonomous Systems*, 55(5):359–371, May 2007.

[39] Ioannis Kostavelis and Antonios Gasteratos. Semantic mapping for mobile robotics tasks: A survey. *Robotics and Autonomous Systems*, 2014.

[40] A. Gawel, C. Del Don, R. Siegwart, J. Nieto, and C. Cadena. X-view: Graph-based semantic multiview localization. *IEEE Robotics and Automation Letters*, 3(3):1687–1694, 2018.

[41] Y. Nakajima, K. Tateno, F. Tombari, and H. Saito. Fast and accurate semantic mapping through geometric-based incremental segmentation. *arXiv preprint arXiv:1803.02784*, 2018.

[42] J. Josifovski, M. Kerzel, C. Pregizer, L. Posniak, and S. Wermter. Object detection and pose estimation based on convolutional neural networks trained with synthetic data. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 6269–6276. IEEE, 2018.

[43] J. Wu, B. Zhou, R. Russell, V. Kee, S. Wagner, M. Hebert, A. Torralba, and D. Johnson. Real-time object pose estimation with pose interpreter networks. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 6798–6805. IEEE, 2018.

[44] T.E. Fortmann, Y. Bar-Shalom, and M. Scheffe. Multi-target tracking using joint probabilistic data association. In *Proc. 19th IEEE Conf. on Decision & Control*, 1980.

[45] N. Sünderhauf and P. Protzel. Switchable constraints for robust pose graph SLAM. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2012.

[46] L. Wong, L. P. Kaelbling, and T. Lozano-Pérez. Data association for semantic world modeling from partial views. In *International Symposium for Robotics Research*. Intl. Foundation of Robotics Research, 2013.

[47] E. Olson and P. Agarwal. Inference on networks of mixtures for robust robot mapping. *Intl. J. of Robotics Research*, 32(7):826–840, 2013.

[48] L. Carlone, A. Censi, and F. Dellaert. Selecting good measurements via l1 relaxation: A convex approach for robust estimation over graphs. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 2667–2674. IEEE, 2014.

[49] S. Pathak, A. Thomas, and V. Indelman. Nonmyopic data association aware belief space planning for robust active perception. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2017.

[50] S. Pathak, A. Thomas, and V. Indelman. A unified framework for data association aware robust belief space planning and perception. *Intl. J. of Robotics Research*, 32(2-3):287–315, 2018.

[51] A. Milan, S.H. Rezatofighi, A.R. Dick, I.D. Reid, and K. Schindler. Online multi-target tracking using recurrent neural networks. In *Nat. Conf. on Artificial Intelligence (AAAI)*, pages 4225–4232, 2017.

[52] Hafez Farazi and Sven Behnke. Online visual robot tracking and identification using deep lstm networks. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 6118–6125. IEEE, 2017.

[53] K. Doherty, D. Fourie, and J. Leonard. Multimodal semantic slam with probabilistic data association. In *2019 international conference on robotics and automation (ICRA)*, pages 2419–2425. IEEE, 2019.

[54] L. Bernreiter, A. Gawel, H. Sommer, J. Nieto, R. Siegwart, and C.C. Lerma. Multiple hypothesis semantic mapping for robust data association. *IEEE Robotics and Automation Letters (RA-L)*, 4(4):3255–3262, 2019.

[55] J. Wang and B. Englot. Robust exploration with multiple hypothesis data association. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 3537–3544. IEEE, 2018.

[56] M. Hsiao and M. Kaess. Mh-isam2: Multi-hypothesis isam using bayes tree and hypo-tree. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, May 2019.

[57] K. Ok, K. Liu, K. Frey, J.P. How, and N. Roy. Robust object-based slam for high-speed autonomous navigation. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pages 669–675, 2019.

[58] Aleksandr V Segal and Ian D Reid. Hybrid inference optimization for robust pose graph estimation. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 2675–2682. IEEE, 2014.

[59] Pierre-Yves Lajoie, Siyi Hu, Giovanni Beltrame, and Luca Carlone. Modeling perceptual aliasing in slam via discrete-continuous graphical models. *IEEE Robotics and Automation Letters (RA-L)*, 2019.

[60] S. Bowman, N. Atanasov, K. Daniilidis, and G. Pappas. Probabilistic data association for semantic slam. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pages 1722–1729. IEEE, 2017.

[61] Siddharth Choudhary, Luca Carlone, Carlos Nieto, John Rogers, Henrik I Christensen, and Frank Dellaert. Multi robot object-based slam. In *Intl. Sym. on Experimental Robotics (ISER)*, 2016.

[62] A. Cunningham, M. Paluri, and F. Dellaert. DDF-SAM: Fully distributed slam using constrained factor graphs. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2010.

[63] A. Cunningham, V. Indelman, and F. Dellaert. DDF-SAM 2.0: Consistent distributed smoothing and mapping. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, Karlsruhe, Germany, May 2013.

[64] V. Indelman, P. Gurfil, E. Rivlin, and H. Rotstein. Graph-based distributed cooperative navigation for a general multi-robot measurement model. *Intl. J. of Robotics Research*, 31(9), August 2012.

[65] V. Indelman, P. Gurfil, E. Rivlin, and H. Rotstein. Distributed vision-aided cooperative localization and navigation based on three-view geometry. *Robotics and Autonomous Systems*, 60(6):822–840, June 2012.

[66] V. Indelman, E. Nelson, J. Dong, N. Michael, and F. Dellaert. Incremental distributed inference from arbitrary poses and unknown data association: Using collaborating robots to establish a common reference. *IEEE Control Systems Magazine (CSM), Special Issue on Distributed Control and Estimation for Robotic Vehicle Networks*, 36(2):41–74, 2016.

[67] Jeffrey M Walls, Alexander G Cunningham, and Ryan M Eustice. Cooperative localization by factor composition over a faulty low-bandwidth communication channel. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2015.

[68] S.I. Roumeliotis and G.A. Bekey. Distributed multi-robot localization. *IEEE Trans. Robot. Automat.*, August 2002.

[69] A. Howard. Multi-robot simultaneous localization and mapping using particle filters. *Intl. J. of Robotics Research*, 25(12):1243–1256, 2006.

[70] Siddharth Choudhary, Luca Carlone, Carlos Nieto, John Rogers, Henrik I Christensen, and Frank Dellaert. Distributed mapping with privacy and communication constraints: Lightweight algorithms and object-based models. *Intl. J. of Robotics Research*, 36(12):1286–1311, 2017.

[71] A. Bahr, M.R. Walter, and J.J. Leonard. Consistent cooperative localization. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pages 3415–3422, May 2009.

[72] R.R. Brooks, P. Ramanathan, and A.M. Sayeed. Distributed target classification and tracking in sensor networks. *Proceedings of the IEEE*, 91(8):1163–1171, 2003.

[73] Y. Aloimonos, I. Weiss, and A. Bandyopadhyay. Active vision. *Intl. J. of Computer Vision*, 1(4), January 1988.

[74] Cl Connolly. The determination of next best views. In *Robotics and automation. Proceedings. 1985 IEEE international conference on*, volume 2, pages 432–435. IEEE, 1985.

[75] David Wilkes and John K Tsotsos. Active object recognition. In *Computer Vision and Pattern Recognition, 1992. Proceedings CVPR'92., 1992 IEEE Computer Society Conference on*, pages 136–141. IEEE, 1992.

[76] Mingyu Gao, Zhiwei He, and Yuanyuan Liu. Improved unscented kalman filter for bounded state estimation. In *Electronics, Communications and Control (ICECC), 2011 International Conference on*, pages 2101–2104. IEEE, 2011.

[77] G.A. Hollinger, U. Mitra, and G.S. Sukhatme. Active classification: Theory and application to underwater inspection. In *Robotics Research*, pages 95–110. Springer, 2017.

[78] A. Singh, A. Krause, C. Guestrin, and W.J. Kaiser. Efficient informative sensing using multiple robots. *J. of Artificial Intelligence Research*, 34:707–755, 2009.

[79] Javier Velez, Garrett Hemann, Albert S Huang, Ingmar Posner, and Nicholas Roy. Modelling observation correlations for active exploration and robust object detection. *J. of Artificial Intelligence Research*, 2012.

[80] Corina Gurău, Dushyant Rao, Chi Hay Tong, and Ingmar Posner. Learn from experience: probabilistic prediction of perception performance to avoid failure. *The International Journal of Robotics Research*, page 0278364917730603, 2017.

[81] T. Arbel and F.P. Ferrie. Entropy-based gaze planning. *Image and vision computing*, 19(11):779–786, 2001.

[82] J. Denzler and C.M. Brown. Information theoretic sensor data selection for active object recognition and state estimation. *IEEE Transactions on pattern analysis and machine intelligence*, 24(2):145–157, 2002.

[83] Michael A. Sipe and David Casasent. Feature space trajectory methods for active computer vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(12):1634–1643, 2002.

[84] K. Liu, M. Stadler, and N. Roy. Learned sampling distributions for efficient planning in hybrid geometric and object-level representations. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pages 9555–9562. IEEE, 2020.

[85] A. Wandzel, Y. Oh, M. Fishman, N. Kumar, L. Wong, and S. Tellex. Multi-object search using object-oriented pomdps. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pages 7194–7200. IEEE, 2019.

[86] R Faddoul, W Raphael, A-H Soubra, and Alaa Chateauneuf. Partially observable markov decision processes incorporating epistemic uncertainties. *European Journal of Operational Research*, 241(2):391–401, 2015.

[87] Akinobu Hayashi, Dirk Ruiken, Christian Goerick, and Tadaaki Hasegawa. Online adaptation of uncertain models using neural network priors and partially observable planning. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pages 2440–2446. IEEE, 2019.

[88] Björn Lütjens, Michael Everett, and Jonathan P How. Safe reinforcement learning with model uncertainty estimates. *arXiv preprint arXiv:1810.08700*, 2018.

[89] C. Papadimitriou and J. Tsitsiklis. The complexity of markov decision processes. *Mathematics of operations research*, 12(3):441–450, 1987.

[90] J. Van Den Berg, S. Patil, and R. Alterovitz. Motion planning under uncertainty using iterative local optimization in belief space. *Intl. J. of Robotics Research*, 31(11):1263–1278, 2012.

[91] R. Platt, L. Kaelbling, T. Lozano-Perez, and R. Tedrake. Efficient planning in non-gaussian belief spaces and its application to robot grasping. In *Proc. of the Intl. Symp. of Robotics Research (ISRR)*, 2011.

[92] R. Platt, R. Tedrake, L.P. Kaelbling, and T. Lozano-Pérez. Belief space planning assuming maximum likelihood observations. In *Robotics: Science and Systems (RSS)*, pages 587–593, Zaragoza, Spain, 2010.

[93] C. Stachniss, D. Haehnel, and W. Burgard. Exploration with active loop-closing for FastSLAM. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2004.

[94] V. Indelman, L. Carlone, and F. Dellaert. Planning in the continuous domain: a generalized belief space approach for autonomous navigation in unknown environments. *Intl. J. of Robotics Research*, 34(7):849–882, 2015.

[95] Timothy Patten, Wolfram Martens, and Robert Fitch. Monte carlo planning for active object classification. *Autonomous Robots*, 42(2):391–421, 2018.

[96] S. M. Chaves, J. M. Walls, E. Galceran, and R. M. Eustice. Risk aversion in belief-space planning under measurement acquisition uncertainty. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 2079–2086. IEEE, 2015.

[97] L. Burks, I. Loefgren, and N.R. Ahmed. Optimal continuous state pomdp planning with semantic observations: A variational approach. *IEEE Trans. Robotics*, 35(6):1488–1507, 2019.

[98] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[99] Oleksii Zhelo, Jingwei Zhang, Lei Tai, Ming Liu, and Wolfram Burgard. Curiosity-driven exploration for mapless navigation with deep reinforcement learning. *arXiv preprint arXiv:1804.00456*, 2018.

[100] Lei Tai and Ming Liu. Towards cognitive exploration through deep reinforcement learning for mobile robots. *arXiv preprint arXiv:1610.01733*, 2016.

[101] Lei Tai, Giuseppe Paolo, and Ming Liu. Virtual-to-real deep reinforcement learning: Continuous control of mobile robots for mapless navigation. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 31–36. IEEE, 2017.

[102] Yu Fan Chen, Miao Liu, Michael Everett, and Jonathan P How. Decentralized non-communicating multiagent collision avoidance with deep reinforcement learning. In *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, pages 285–292. IEEE, 2017.

[103] Yu Fan Chen, Michael Everett, Miao Liu, and Jonathan P How. Socially aware motion planning with deep reinforcement learning. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2017.

[104] Jakob Foerster, Yannis Assael, Nando de Freitas, and Shimon Whiteson. Learning to communicate with deep multi-agent reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 2137–2145, 2016.

[105] Shayegan Omidshafiei, Jason Pazis, Christopher Amato, Jonathan P How, and John Vian. Deep decentralized multi-task multi-agent rl under partial observability. *arXiv preprint arXiv:1703.06182*, 2017.

[106] F.R. Kschischang, B.J. Frey, and H-A. Loeliger. Factor graphs and the sum-product algorithm, February 2001.

[107] F. Dellaert. Factor graphs and GTSAM: A hands-on introduction. Technical Report GT-RIM-CP&R-2012-002, Georgia Institute of Technology, September 2012.

[108] T. Bailey, S. Julier, and G. Agamennoni. On conservative fusion of information with unknown non-gaussian dependence. In *Intl. Conf. on Information Fusion, FUSION*, pages 1876 – 1883, 2012.

[109] V. Tchuiev and V. Indelman. Semantic distributed multi-robot classification, localization, and mapping with a viewpoint dependent classifier model - supplementary material. Technical report, Technion - Israel Institute of Technology, 2020.

[110] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.

[111] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. IEEE, 2009.

[112] Jonathan Huang. Maximum likelihood estimation of dirichlet distribution parameters. *CMU Technique Report*, 2005.

[113] Phil Ammirato, Patrick Poirson, Eunbyung Park, Jana Kosecka, and Alexander C. Berg. A dataset for developing and benchmarking active vision. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2017.

[114] R. Singh, B. C. Pal, and R. A. Jabr. Statistical representation of distribution system loads using gaussian mixture model. *IEEE Transactions on Power Systems*, 25(1):29–37, 2009.

[115] David Ha and Jürgen Schmidhuber. World models. *arXiv preprint arXiv:1803.10122*, 2018.

[116] E. I. Farhi and V. Indelman. ix-bsp: Belief space planning through incremental expectation. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, May 2019.

[117] T.P. Minka. Estimating a dirichlet distribution. 2003.

[118] Arjun Singh, James Sha, Karthik S Narayan, Tudor Achim, and Pieter Abbeel. Bigbird: A large-scale 3d database of object instances. In *2014 IEEE international conference on robotics and automation (ICRA)*, pages 509–516. IEEE, 2014.

לבסוף, אם ישנה יכולת להעריך אי ודאות אפיסטמית ודרוש סיווג מדויק של האובייקטים בתמונה, רובוט יכול לקבל החלטה בצורה אוטונומית לנוע לכיוון איזורים בהם האי-ודאות האפיסטמית נמוכה. לשם כך, יש להתייחס גם לתלות של נקודת המבט היחסית בין סוכן ואובייקטים בתוצאות הסיווג. בחלק הרביעי, אנו מתייחסים לשני מקורות האי ודאות בו זמנית, קודם עבור מקרה פאסיבי בו המסלול מוכתב לרובוט, ואז מקרה אקטיבי בו הרובוט צריך לקבל החלטה. כמו בחלק השלישי, המסווג מספק מספר וקטורי הסתברות לכל תמונה שמייצגים אי ודאות אפיסטמית עבור אותה תמונה. בחלק הרביעי אנו מציעים שתי גישות SLAM מבוסס אובייקטים המטפלות בשני המקרים. עבודה זו העבודה הראשונה שמשלבת בין סיווג עם התייחסות לאי ודאות אפיסטמית של המסווג בתוך תשתית SLAM שגם מחשבת מיקום של המצלמה והאובייקטים בסביבה. הגישה הראשונה שאנו מציגים היא ,MH בה מתחזקות מספר אמונות היברידיות במקביל כאשר כל אחת מהן מקבלת קלט של וקטור הסתברות יחיד שונה לאובייקט בתמונה. שילוב כל האמונות מאפשר התייחסות לאי ודאות אפיסטמית משוכללת של כל אובייקט. למרות היותה מדויקת, גישה זו מאוד כבדה חישובית. אנו מציעים גישה אלטרנטיבית הנקראת ,JLP אשר בקיום הנחות על מודל המסווג מתחזקת אמונה רציפה יחידה ומאפשרת שיערוך פרמטרים משמעותית יותר יעיל מ-MH, תוך כדי כך שהיא מאפשרת גישה לאי ודאות אפיסטמית. את שני הגישות אנו מפתחים למקרה הפאסיבי תחילה, ואז מרחיבים למקרה האקטיבי, תוך כדי התייחסות ליצירת מדידות עתידית לקבלת החלטות ושימוש בפונקצית פרס המאפשרת תכנון למזעור אי ודאות אפיסטמית. עבור השיטות אנו משתמשים במודל אי-ודאות מסווג מאפשר לנו לדגום מדידות עתידיות במקום לחזות ולייצר תמונות עתידיות. בסימולציה וניסוי אנו מראים שיפור ביצועים בסיווג לעומת שיטות שלא מתחשבות באי-ודאות אפיסטמית, ומציגים יתרון בביצועים לתכנון עם פונקצית פרס שמתייחסת לאי ודאות אפיסטמית לעומת כזו שלא.

לסיכום, הצגנו גישות המתמודדות עם שני מקורות אי ודאות בסיווג: תלות במיקום יחסי בין מצלמה לאובייקט, ואי ודאות אפיסטמית של המסווג. הצגנו גישות מבוססות אמונה היברידית להתמודדות עם בעיית המיקום היחסי, הצגנו שיטת סיווג סדרתית לחישוב אי ודאות איפסטמית מצטברת, והצגנו שיטות לטיפול בשני מקורות אי הודאות ביחד. כיוונים אפשריים למחקר עתידי עשויים לכלול שיפור היעילות החישובית של שיטות מבוססות אמונה היברידית עם גזימה חכמה של אמונות רציפות מותנות, או שימוש בהנחות המאפשרות חישוב מהיר יותר בלי לפגוע משמעותית בשיערוך.

תקציר

סיווג אובייקטים היינה בעיה חשובה בישומי רובוטיקה ואווירונאוטיקה. פתרונות סיווג אובייקטים משמשים למגוון רחב של ישומים, כגון כלי רכב אוטונומיים, ניווט מל"טים, חיפוש והצלה, וכו'. בשנים האחרונות חלה התקדמות משמעותית בתחום של סיווג אובייקטים עם התפתחות הלמידה העמוקה, כאשר במהרה הביצועים של רשתות ניורונים עולים על ביצועים של בני אדם. אף על פי כן, סיווג אמין נשארת בעיה פתוחה, שכן היא בעיה מורכבת. אי דיוקים בסיווג יכולים לנבוע מתאורה לקויה, הסתרות, רזולוציה נמוכה, תלות של תוצאות סיווג בנקודת מבט בה לא ניתן להבחין במחלקה שלו, והגבלות של סט האימון שידוע כחוסר ודאות אפיסטמית, או חוסר ודאות במודל. בעבודה זו אנו מפתחים גישות להתמודדות עם השניים האחרונים. עבודה זו מחולקת לארבעה חלקים, כאשר השניים הראשונים עוסקים בהתמודדות עם אי דיוקים עקב תלות בנקודת מבט יחסית, החלק השלישי מתמודד עם אי ודאות אפיסטמית, והאחרון מתמודד עם שניהם בו-זמנית.

בפרט, אובייקטים ממחלקות שונות יכולים להראות דומה מנקודות מבט מסויימות, ומסווגים לא יהיו מסוגלים להבחין בין שתי המחלקות, בעיה זו נקראת כפילות בסיווג. לכן, לקבלת תוצאת סיווג מדויקת, נדרש מידע נוסף מתמונות נוספות וגם ממידע על מיקום היחסי בין המצלמה לאובייקט. מסווגים מודרניים מבוססי למידה-עמוקה מקבלים תמונה כקלט ומספקים וקטור של הסתברויות סיווג בין מספר מחלקות מועמדות כפלט. הוקטור הסתברות הזה משתנה בין נקודות מבט יחסיות בין אובייקט למצלמה, ואם אנו יכולים למדל את אותו יחס אנו יכולים להשתמש בזה לשיפור תוצאות סיווג; אותו מודל הוא מודל מסווג תלוי נקודת מבט. בנוסף, אם המבט היחסי אינו ידוע, אנו יכולים לשלב את המודל הזה בתוך פיתרון של לוקליזציה ומיפוי סימולטני (SLAM) מבוסס אובייקטים למען שיפור ביצועי לוקיזציה, מיפוי ושיוך מידע. עד כה, עבודות קיימות שמתייחסות לסיווג ולקוליזציה ביחד משתמשות בפלט מסווג של המחלקה הסבירה ביותר, או משתמשות במודל מסווג תלוי נקודת מבט במצב בו המיקום והמפנה של המצלמה ידועה מראש. החלק הראשון של העבודה מציע שיטה לשילוב אותו מודל המסווג ב-SLAM מבוסס אובייקטים. אנו מפתחים שיטה לתחזוק אמונה היברידית המשלבת משתנים אקראיים רציפים (מיקום ואוריינטציה של המצלמה והאובייקטים), ובדידים (מחלקות סיווג של האובייקטים). האמונה ההיברידית ניתנת לחלוקה לאמונה רציפה מותנית על מיקום ומפנה שהיא פונקצית צפיפות הסתברות, ואמונה בדידה על מימוש מחלקות של אובייקטים ושיוך מידע שהיא המשקל של אותה אמונה רציפה. בשני סוגי האמונות אנו משתמשים במודל מסווג. אנו מראים שימוש במודל המסווג משפר ביצועים בלוקליזציה ועוזר בפתרון בעיית שיוך מידע ב-SLAM, בהשוואה לגישות שלא משתמשות באותו מודל.

לעיתים, מצלמה יחידה אינה מסוגלת לסווג את האובייקטים בתמונה בצורה מספיק מדויקת, עבודה שמספר רובוטים או כלי טייס מסוגלים לעשות תוך כדי שילוב מידע. ישנן 3 תצורות עיקריות של מערכות מרובות רובוטים: ממורכזות, בה כל הסוכנים מתקשרים עם יחידת עיבוד מרכזית; מערכות בהן הסוכנים מתקשרים עם מספר יחידות עיבוד; ומבוזרות, בהן כל סוכן הוא עצמאי ומעבד בעצמו מידע שהוא קיבל ממדידו שלו עצמו ומסוכנים אחרים. מערכות מבוזרות הינן איתנות יותר שכן אינן תלויות במספר קטן של יחידות עיבוד, אך תכנונן מורכב יותר. אתגר שספציפי למערכות מבוזרות הינו עקביות בשיערוך; כל סוכן חייב להתחשב בעיבוד המידע בכל מדידה פעם אחת בלבד, אחרת הוא מסתכן בשיערוך שגוי ויותר מדי בטוח בעצמו (עם איזור אי ודאות קטן מדי). עבודות קיימות עבור SLAM מרובה רובוטים מתייסות למשתנים רציפים בלבד, לכן גם התייחסותן לבעית עקביות השיערוך נוגע למשתנים רציפים בלבד. לעומתן בחלק השני, בהנחת שיוך מידע פתור, אנו מרחיבים את הגישה שפותחה בחלק הראשון למערכת מרובת רובוטים מבוזרת כאשר אנו מתמודדים עם בעית כפילות הסיווג. אנו מתחזקים בנפרד שני אמונות היברידיות: אחת אינדיבידואלית עבור מידע שהרובוט בעצמו מקבל, ושנייה משותפת שהיא שילוב של כל המידע שהרובוט מקבל מעצמו ומאחרים. הסוכנים מתקשרים ביניהם ,משתפים אמונות אינדיבידואליות, ושומרים אותן בערמה שמשתמשים בה לתחזוק האמונה המשותפת. כך הם גם שומרים על שיערוך עקבי עבור משתנים רציפים ובייחוד בדידים שעד כה לא נעשה קודם. אני מראים שגישתנו משפרת ביצועים בהשוואה לשימוש ברובוט יחיד וכאשר לא מתחשבים בבעיית עקביות השיערוך.

מסווגים מאומנים על ידי סט אימון הכולל דוגמאות מתויגות של אובייקטים. לישומים בפועל סט האימון יכול להיות מוגבל ולא לייצג היטב את הקלט אותו המסווג עשוי לקבל. עקב כך, הפלט של המסווג עשוי להיות אקראי ולא ניתן להשען עליו לישומים המחייבים סיווג מדויק, כגון נהיגה אוטונומית של כלי רכב. אם הקלט שהתקבל במסווג אינו תואם לדוגמאות בסט האימון, האי ודאות בתוצאות הסיווג גבוהה והיא נקראת אי ודאות אפיסטמית. במהלך השנים פותחו גישות לחישוב אי ודאות אפיסטמית עבור תמונה יחידה, אך לא בסיווג סדרתי. בחלק השלישי בעבודה זו, אנו מגשרים על הפער ומציעים שיטה לסיווג סדרתי המחשב אי ודאות אפיסטמית בסט האימון. אני משתמשים במסווג המספק סדרה של וקטורי הסתברות שהפיזור שלהם מתאר אי ודאות אפיסטמית בתמונה יחידה של אובייקט, ומאחדים את המידע המתקבל ממספר תמונות כדי לקבל סיווג ואי ודאות אפיסטמית משוקללת. אנו מראים בניסוי שיפור בביצועי סיווג לעומת גישות שמתחשבות רק בוקטור סיווג משוכלל, בלי להתחשב באי ודאות אפיסטמית.

המחקר בוצע בהנחייתו של פרופסור חבר ואדים אינדלמן, בפקולטה להנדסת אווירונאוטיקה וחלל.

חלק מן התוצאות בחיבור זה פורסמו כמאמרים מאת המחבר ושותפיו למחקר בכנסים ובכתבי-עת במהלך תקופת מחקר הדוקטורט של המחבר, אשר גרסאותיהם העדכניות ביותר הינן:

V. Tchuiev and V. Indelman. Inference Over Distribution of Posterior Class Probabilities For Reliable Bayesian Classification and Object-Level Perception. *IEEE Robotics and Automation Letters (RA-L)*, 3(4):4329–4336, 2018.

V. Tchuiev, Y. Feldman, V. Indelman. Data Association Aware Semantic Mapping and Localization via a Viewpoint-Dependent Classifier Model. classificators. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*. 7742–7749, IEEE, October 2019.

V. Tchuiev and V. Indelman. Distributed Consistent Multi-Robot Semantic Localization and Mapping. *IEEE Robotics and Automation Letters (RA-L)*, 5(3):4649–4656, 2020.

V. Tchuiev and V. Indelman. Epistemic Uncertainty Aware Localization And Mapping For Inference and Belief Space Planning. *Submitted to Artificial Inteligence Journal (AIJ)*, 2021.

תודות

אני רוצה להודות למנחה שלי פרופ. חבר ואדים אינדלמן על הנחייתו ותמיכתו המקצועית במהלך 4 שנות מחקר לדוקטורט עם הידע ותובנותיו, ליורי פלדמן על תרומתו למאמר שהוצג בכנס IROS 2019, לפקולטה להנדסת אווירונאוטיקה וחלל מהטכניון על תרומתם הכספית במשך כל הזמן הזה, לקבוצת ANPL על דיונים פורים שתרמו רבות למחקר. לבסוף, אני רוצה להודות להוריי אולגה ויבגני צ'וייב עבור תמיכה מורלית מתמשכת במשך עבודתי לקראת תואר דוקטור.

הכרת תודה מסורה לטכניון על מימון מחקר זה.

סיווג אוטונומי מרובה רובוטים תחת אי ודאות

חיבור על מחקר

לשם מילוי חלקי של הדרישות לקבלת התואר
דוקטור לפילוסופיה

ולדימיר צ'וייב

הוגש לסנט הטכניון — מכון טכנולוגי לישראל
אייר התשפ"א חיפה אפריל 2021

סיווג אוטונומי מרובה רובוטים תחת אי ודאות

ולדימיר צ'וייב