Technische Universität München
TUM School of Computation, Information and Technology

Spatial State-Space Models:

Dense Probabilistic Prediction and Inference in 3D

Atanas Georgiev Mirchev

Vollständiger Abdruck der von der TUM School of Computation, Information and Technology der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

Vorsitz:		Prof. Dr. Stephan Günnemann
Prüfer der Dissertation:	1.	Prof. Dr. Daniel Cremers
	2.	Prof. Dr. Herke van Hoof
	3.	apl. Prof. Dr. Georg Groh

Die Dissertation wurde am 18.09.2023 bei der Technischen Universität München eingereicht und durch die TUM School of Computation, Information and Technology am 21.03.2024 angenommen.

Abstract

This work is about a spatial state-space model, also seen as a *world* model, that supports the control of mobile agents by simulating agent movement, RGB-D image generation and a dense 3D map. The main contribution relative to prior art is in reconciling a) prediction and inference in a state-space model (for control) with b) volumetric dense 3D maps and direct image generation.[1] The thesis is cumulative with three core publications.

The first is VSSM-LM, a model that defines the predictive state space. It combines rendering in 3D (emission) with control-driven 6-DoF dynamics (transition). VSSM-LM can be queried with control inputs, for which it predicts the agent future (states, images, map) with uncertainty estimates. This way VSSM-LM complies with the model-based control framework.

In parallel, inference in the model is addressed with a variational inference (VI) smoother. Compared to traditional maximum a-posteriori (MAP) optimisation, VI enables a wider range of approximate posteriors. Variational smoothing works reliably for indoor flight trajectories, but is limited to off-line use on current hardware.

The second publication, PRISM, addresses Bayesian real-time inference under VSSM-LM. Specific to rendering in state-space models, PRISM makes the point that filtering is one path to real-time. The recursion of the Bayes filter is used to limit computations, relying on closed-form updates where applicable. PRISM carefully reveals all Bayesian approximations, so that they can be lifted in the future as hardware improves. It runs in real-time on commodity GPUs with comparable accuracy to the current SotA, on a variety of indoor hand-held and flight trajectories.

The final core contribution is TNP-SM, which explores the applicability of VSSM-LM for navigation in complex 3D mazes. It is both shown that VSSM-LM reveals sufficient collision information and that state estimation with the model is fast enough for real-time control, under partial observability and noisy agent dynamics in the execution environment.

[1] State-space models are fundamental for model-based control, and dense assumptions increase the information for making decisions.

Acknowledgments

This thesis would not be were it not for the following people.

Dr. Justin Bayer, who taught me how to do research and always had my back.

Prof. Patrick van der Smagt, who put a start to all of this and trusted me enough to let me follow my own path.

Prof. Daniel Cremers, who agreed to supervise me and whose feedback always resonated with me.

Special thanks to my awesome lab mates and students: Adnan, Ahmed, Alexej, Alex, Amna, Baris, Begum, Botond, Djalel, Ebraheem, Elie, Faruk, Felix, Felix, Grady, Justin, Karolina, Laura, Liesbeth, Marvin, Max, Max, Michelle, Nutan, Ole, Patrick, Philip, Richard, Salem, Simon, Tai, Xingyuan, Ziqing.

Baris, thanks for soldiering through this with me, I learned a lot from you.

Knowing so many bright and humble people is a privilege.

Mom, Dad, Nikola, Teddy – thank you for being by my side. You make me truly happy.

Contents

Abbreviations

Notation	Description	Notation	Description
ELBO	evidence lower bound	KL	Kullback-Leiber divergence
MAP	maximum a-posteriori	MC	Monte Carlo
MDP	Markov decision process	MI	mutual information
POMDP	partially observable Markov decision process	PRISM	core paper, see appendix A.2
RL	reinforcement learning	RNN	recurrent neural network
SGD	stochastic gradient descent	SLAM	simultaneous localisation and mapping
SSM	state-space model	TNP-SM	core paper, see appendix A.3
VI	variational inference	VSSM-LM	core paper, see appendix A.1

Nomenclature

Notation	Description
x	a scalar value $x \in \mathbb{R}$
$\mathbf{x}$	a vector $\mathbf{x} \in \mathbb{R}^n$
$\mathbf{X}$	a matrix $\mathbf{X} \in \mathbb{R}^{n \times m}$
$\mathbf{I}$	the identity matrix
t	time index $t \in [T]$
$\mathbf{x}_{1:T}$	a sequence of T variables $\mathbf{x}_1, \ldots, \mathbf{x}_T$
$\mathbf{x}_{>t}$	future variables $\mathbf{x}_{t+1}, \mathbf{x}_{t+2}, \ldots$, up to T or ∞
$\mathbf{x}_{\leqslant t}$	past variables $\mathbf{x}_1, \ldots, \mathbf{x}_t$
$\mathcal{D}$	previously collected data, basis for inference
$p(\mathbf{x})$	a marginal distribution, $\mathbf{x}$ is random
$p(\mathbf{x} \mid \mathbf{u})$	a conditional distribution, $\mathbf{x}$ is random, $\mathbf{u}$ is given
$p_\theta(\mathbf{x})$	θ are distribution parameters
$\mathbf{x} \sim p(\mathbf{x})$	a sample $\mathbf{x}$ is drawn from $p(\mathbf{x})$
$\mathbb{E}_{\mathbf{x} \sim p(\mathbf{x})}[f(\mathbf{x})]$	expectation of $f(\mathbf{x})$ under $p(\mathbf{x})$
$q(\mathbf{z})$	(in inference) an approximate distribution
$\mathbb{H}(\mathbf{x})$	differentiable entropy of the random variable $\mathbf{x}$
$\mathbb{H}(\mathbf{x}_1 \mid \mathbf{x}_2)$	conditional entropy (not the entropy of $\mathbf{x}_1 \mid \mathbf{x}_2$)
$\mathrm{I}(\mathbf{x}_1; \mathbf{x}_2 \mid \mathbf{x}_3)$	mutual information between $\mathbf{x}_1$ and $\mathbf{x}_2$, given $\mathbf{x}_3$
$\mathbf{x}$	(in the models) an observation (an image)
$\mathbf{z}$	(in the models) an agent state
$\mathbf{u}$	(in the models) a control input
$\mathfrak{M}$	(in the models) a map

Introduction

Moving robots that can make decisions have piqued the interest of researchers for decades. From drones, to cars, to turtlebots, to legged quadrupeds, even robot arms – as a community we quest after mobile agent autonomy. And with good reason, as the applications are many: drone delivery fleets, factory and building inspection, environment reconstruction, transportation, object manipulation, the list goes on.

In this regard, over nearly half a century the synonymous fields of structure from motion (Hartley and Zisserman, 2006) and Simultaneous Localisation and Mapping (SLAM) (Cadena et al., 2016) have made major progress in letting moving robots know their place in space and know what is around them. Simultaneously, in the past decades we have also solidified our understanding of how to control agents by planning ahead (Bertsekas, 2005), and nowadays model-based optimal control methods like dynamic programming and Model-Predictive Control (MPC) are widely used to enable autonomy both in research and in industry. This same understanding of control has sculpted the modern field of model-based Reinforcement Learning (RL) (Sutton and Barto, 2018), leading to the concept of *world models*, data-driven differentiable *simulators* which are used to learn parametric control policies, also aimed at autonomy (e. g. Dreamer by Hafner et al. (2020) and SLAC by A. X. Lee et al. (2020)).

Unifying these pillars – SLAM and model-based control, into a *spatial world model* is desirable. Further research is needed, specifically for *dense* models of raw RGB-D image data, because the assumptions of the Markovian models used for control are not completely aligned with the typical modelling in modern dense volumetric SLAM, for example because control-driven state transitions are not always considered, or for efficiency reasons odometry estimation is isolated from dense map estimation, or posterior and predictive distributions are not explicitly spelled out (see the surveys of Zollhöfer et al. (2018), Civera and S. H. Lee (2020), and Tosi et al. (2024) for an overview). Building this bridge is the main goal of the thesis.

This cohesion, between state estimation and control, is a prerequisite for controllers that are aware of potential state estimation errors in the future

(i. e. proper control under partial observability). While such control per se is out of scope for the thesis, it is a motivation nonetheless. The other argument in favour of unity is the natural cross-pollination of methods.

The thesis gives one possible answer to each of the following questions: How to define a spatial state-space model well-suited for control such that it generates dense images directly? How to formulate inference for it? Can everything be framed in a Bayesian way, in order to obtain uncertainty? How to infer such a model from data in real-time, and what compromises does this imply? How to use such a model to solve navigation control and autonomous exploration?

Possible solutions are derived using the Bayesian framework, and then evaluated with both simulated and real-world data.

A Note on Jargon and Notation

The following terms appear very often in the thesis and are key for understanding the contributions.

State-space model is used throughout, and is synonymous with a hidden Markov model (i. e. specific connections between variables are assumed).

Prediction is reserved for generating any model variables either in the future of the agent, or from novel regions in the state space. *Simulation* is used in the same way in the context of graphical models.

Inference is reserved for the estimation of unknown (latent) model variables from past data.

Point estimation refers to inferring a single value for a latent variable, as opposed to *fully probabilistic* or *full-posterior* inference which gives a full distribution.

Cost-to-go refers to the sum of future control costs; it is the same as the negative *value* (sum of rewards), or *return*.

Also, the notation in the thesis follows standard conventions from the field of machine learning. Please consult the preceding pages for a list of abbreviations and nomenclature.

How to Read the Thesis?

The thesis is publication-based, organised in three parts.

Part I has the most important theoretical background. It can be read end-to-end and shows the connections to control. For convenience, each section of part II also lists the most relevant background sections.

Part II is the thesis core, it details the central problem and purpose, positions them in the scientific context and then presents the contributions. This part has the overarching narrative, but it is painted in broad strokes. The relevant background and the papers have the details.

Finally, the core publications are included as appendices in part III. This is the only part that contains all the math and experiment details.

Publication List

This publication-based thesis is made of three core publications, VSSM-LM (Mirchev et al., 2021), PRISM (Mirchev et al., 2022) (oral presentation) and TNP-SM (oral presentation, shared first author, Kayalibay et al. (2022)). They are discussed in part II and the respective papers are attached in appendices A.1 to A.3.

Core Publications

Mirchev, Atanas, Baris Kayalibay, Patrick van der Smagt, and Justin Bayer (2021). "Variational State-Space Models for Localisation and Dense 3D Mapping in 6 DoF." In: *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net. URL: https://openreview.net/forum?id=XAS3uKeFWj.

Mirchev, Atanas, Baris Kayalibay, Ahmed Agha, Patrick van der Smagt, Daniel Cremers, and Justin Bayer (2022). "PRISM: Probabilistic Real-Time Inference in Spatial World Models." In: *Conference on Robot Learning, CoRL 2022, 14-18 December 2022, Auckland, New Zealand*. Vol. 205. Proceedings of Machine Learning Research. PMLR, pp. 161–174. URL: https://proceedings.mlr.press/v205/mirchev23a.html.

Kayalibay, Baris, Atanas Mirchev, Patrick van der Smagt, and Justin Bayer (2022). "Tracking and Planning with Spatial World Models." In: *Learning for Dynamics and Control Conference, L4DC 2022, 23-24 June 2022, Stanford University, Stanford, CA, USA*. Vol. 168. Proceedings of Machine Learning Research. PMLR, pp. 124–137.

During the doctorate, the author was also a co-author in three non-core publications, tangent to the topic of the thesis and therefore not included.

The first identifies a problem with suboptimal conditioning of amortised posteriors, coined a *conditioning gap*, in the context of variational state-space models (Bayer et al., 2021). The second studies the impact of this same conditioning gap on model-based reinforcement learning (Kayalibay et al.,

2021). The last makes MPC control aware of the estimation errors in the future of an agent, in line with notions in this thesis (Kayalibay et al., 2023).

Non-Core Publications

Bayer, Justin, Maximilian Soelch, Atanas Mirchev, Baris Kayalibay, and Patrick van der Smagt (2021). "Mind the Gap when Conditioning Amortised Inference in Sequential Latent-Variable Models." In: *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net. URL: https://openreview.net/forum?id=a2gqxKDvYys.

Kayalibay, Baris, Atanas Mirchev, Patrick van der Smagt, and Justin Bayer (2021). "Less Suboptimal Learning and Control in Variational POMDPs." In: *Self-Supervision for Reinforcement Learning Workshop - ICLR 2021*. URL: https://openreview.net/forum?id=oe4q7ZiXwkL.

Kayalibay, Baris, Atanas Mirchev, Ahmed Agha, Patrick van der Smagt, and Justin Bayer (2023). "Filter-Aware Model-Predictive Control." In: *Learning for Dynamics and Control Conference, L4DC 2023, 15-16 June 2023, Philadelphia, PA, USA*. Vol. 211. Proceedings of Machine Learning Research. PMLR, pp. 1441–1454. URL: https://proceedings.mlr.press/v211/kayalibay23a.html.

Lastly, the two publications (Kayalibay et al., 2018) and (Mirchev et al., 2019) appeared right when the doctoral project began. They are aimed at the same topics, but are admittedly a more naive attempt at modelling. They *do not* count towards the dissertation contributions (they partially overlap with the author's master's thesis).

Prior Work (Before Doctorate)

Kayalibay, Baris, Atanas Mirchev, Maximilian Soelch, Patrick van der Smagt, and Justin Bayer (2018). *Navigation and planning in latent maps*. URL: http://reinforcement-learning.ml/papers/pgmrl2018_kayalibay.pdf.

Mirchev, Atanas, Baris Kayalibay, Maximilian Soelch, Patrick van der Smagt, and Justin Bayer (2019). "Approximate Bayesian Inference in Spatial Environments." In: *Robotics: Science and Systems XV, University of Freiburg, Freiburg im Breisgau, Germany, June 22-26, 2019*. DOI: 10.15607/RSS.2019.XV.083. URL: https://doi.org/10.15607/RSS.2019.XV.083.

Part I.

Background

The presented background is based on the following sources: Bishop (2007) and Koller and Friedman (2009) on graphical models and general inference, Särkkä (2013) on state-space models, smoothing and filtering, Thrun et al. (2005) on probabilistic robotics and kinematics, Hartley and Zisserman (2006) on computer vision, and Bertsekas (2005) on control.

Chapter 1 introduces the basic tools of probabilistic modelling, followed by a discussion of state-space models and a short section on model-based control. Chapter 2 presents the necessary spatial domain knowledge, it covers 6-DoF kinematics, cameras and rendering.

Probabilistic Models and Control

1

This thesis builds a *spatial world model*, a *probabilistic simulator* that is tailored to control requirements. More specifically, the outcome is a variation of a *state-space model* (a hidden Markov model) (Särkkä, 2013).

The methods use the toolkit of directed probabilistic graphs (Koller and Friedman, 2009)[1]. The rest of the chapter first covers the basic tools:

- Monto-Carlo evaluations (section 1.1).
- Parameterising individual graph factors (section 1.2).
- Inference from data (section 1.3).

Then it delves into the state-space assumptions and indicates how they fit in a control context:

- State-space models (section 1.4).
- Model-based control (section 1.5).

1.1. Monte Carlo

The presented methods will often rely on Monte Carlo, *sampling ancestrally* (Bishop, 2007) from distributions following their factorisation.

For example, for the factorised joint distribution

$$\mathrm{p}(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3) = \mathrm{p}(\mathbf{x}_1)\mathrm{p}(\mathbf{x}_2 \mid \mathbf{x}_1)\mathrm{p}(\mathbf{x}_3 \mid \mathbf{x}_1) \tag{1.1}$$

we would first sample $\mathbf{x}_1^k \sim \mathrm{p}(\mathbf{x}_1)$, followed by $\mathbf{x}_2^k \sim \mathrm{p}\left(\mathbf{x}_2 \mid \mathbf{x}_1^k\right)$ and $\mathbf{x}_3^k \sim \mathrm{p}\left(\mathbf{x}_3 \mid \mathbf{x}_1^k\right)$. Repeating this gives a set of samples $\{\mathbf{x}_1^k, \mathbf{x}_2^k, \mathbf{x}_3^k\}_{[\mathrm{K}]}$ from $\mathrm{p}(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3)$.

Sampling is used for predictive simulation (e. g. when predicting agent rollouts) and also in stochastic optimisation.

[1] The graph perspective is most useful as a conceptual basis for probabilistic programming (e. g. Edward by Tran et al. (2018)) and implementations in auto-differentiable frameworks (e. g. JAX by Bradbury et al. (2018)), which is what the thesis uses throughout.

1.1.1. Monte-Carlo Estimation

Both in simulation and in derivations we will also encounter expectations

$$\mathbb{E}_{\mathbf{x}\sim p(\mathbf{x})}[f(\mathbf{x})] = \int p(\mathbf{x}) f(\mathbf{x}) \, \mathrm{d}\mathbf{x}, \tag{1.2}$$

where $f(\mathbf{x})$ is some utility function. Expectation integrals are generally intractable for arbitrary $f(\cdot)$. To work around this, we will often use *Monte-Carlo* (MC) estimation

$$\frac{1}{K}\sum_{k=1}^{K} f(\mathbf{x}_k) \approx \mathbb{E}_{\mathbf{x}\sim p(\mathbf{x})}[f(\mathbf{x})], \quad \mathbf{x}_k \sim p(\mathbf{x}), \tag{1.3}$$

where the estimate is the average of $f(\cdot)$ across K samples. MC estimation is unbiased.

1.2. Parameterisation

The joint distributions in this thesis are almost always implemented as a product of factors (e. g. like in eq. (1.1)). For the individual factors, the thesis relies on closed-form *parametric distributions*. Particle-based *empirical* distributions are an alternative covered in appendix C.4, but they appear only in prior work.

1.2.1. Parametric Distributions

Parametric distributions are constructed in two steps. First, we select a distribution family in which deterministic *distribution parameters* $\boldsymbol{\psi}$ define the probability density function. The commonplace example are Gaussians $\mathcal{N}(\mathbf{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma})$ with parameters $\boldsymbol{\psi} = (\boldsymbol{\mu}, \boldsymbol{\Sigma})$. Second, we specify a deterministic function that maps deterministic *model parameters* $\boldsymbol{\theta}$ into $\boldsymbol{\psi}$, for example $f(\boldsymbol{\theta}) = \boldsymbol{\mu}, \boldsymbol{\Sigma}$ for a Gaussian.[2] We denote the PDF of the resulting distribution with $p_{\boldsymbol{\theta}}(\mathbf{x})$, as it is a function of $\boldsymbol{\theta}$. Gradient-based optimisation of $\boldsymbol{\theta}$, and thus the whole distribution, is then straightforward when $\nabla_{\boldsymbol{\theta}} f(\boldsymbol{\theta})$ and $\nabla_{\boldsymbol{\psi}} p(\mathbf{x})$ are well-defined.

This construction applies to both marginal and conditional distributions:

$$p_{\boldsymbol{\theta}}(\mathbf{x}), \quad \boldsymbol{\psi} = f(\boldsymbol{\theta}) \tag{1.4}$$

$$p_{\boldsymbol{\theta}}(\mathbf{x} \mid \mathbf{z}), \quad \boldsymbol{\psi} = f(\boldsymbol{\theta}, \mathbf{z}), \tag{1.5}$$

[2] In the trivial case of $f(\boldsymbol{\theta}) = \boldsymbol{\theta}$ we have $\boldsymbol{\psi} = \boldsymbol{\theta}$, then we model the distribution parameters directly.

where for conditionals it is more natural to think of $f(\theta, \mathbf{z})$ as a parametric function $f_\theta(\mathbf{z})$ (e. g. a neural net or physical equations with parameters θ). Notation-wise, we will often use one θ to denote the combined parameters of all factors in a joint distribution, e. g. $p_\theta(\mathbf{x}, \mathbf{z}) = p_\theta(\mathbf{x} \mid \mathbf{z}) p_\theta(\mathbf{z})$.

Parametric distributions are chosen for two reasons. First, they allow us to incorporate inductive biases through $f(\cdot)$ – in the thesis this will get as complex as a 3D renderer. And second, their PDFs are well-defined, which is needed in posing objectives.

1.2.2. A Note on Gaussians

Parametric Gaussian factors of the form $p_\theta(\mathbf{z})$ and $p_\theta(\mathbf{x} \mid \mathbf{z})$ are prevalent in this work. Here we will highlight three of their properties.

Non-Linear Gaussian Systems

A joint $p_\theta(\mathbf{x}, \mathbf{z}) = p_\theta(\mathbf{x} \mid \mathbf{z}) p_\theta(\mathbf{z})$ made of only Gaussian factors can still be complex and non-Gaussian, as long as the Gaussian conditional $p_\theta(\mathbf{x} \mid \mathbf{z})$ is *non-linear* in $\mathbf{z}$. Individual Gaussian factors are thus not as limiting as assuming the whole joint $p_\theta(\mathbf{x}, \mathbf{z})$ is Gaussian, which implicitly carries a linear assumption. A comparison is given in fig. 1.1b and fig. 1.1c. Linearisation is avoided in the predictive models, to remain expressive.

Linear Gaussian Systems

The aforementioned linearity of joint Gaussians is easiest to see from the perspective of *linear Gaussian systems*. A Gaussian marginal $\mathcal{N}(\mathbf{z} \mid \boldsymbol{\mu}_\mathbf{z}, \boldsymbol{\Sigma}_\mathbf{z})$ and a linear Gaussian conditional $\mathcal{N}(\mathbf{x} \mid \mathbf{A}\mathbf{z} + \mathbf{a}, \boldsymbol{\Sigma}_\mathbf{x})$ always combine into a joint Gaussian $\mathcal{N}(\mathbf{x}, \mathbf{z} \mid \boldsymbol{\mu}^*, \boldsymbol{\Sigma}^*)$. Similarly, a Gaussian joint $\mathcal{N}(\mathbf{x}, \mathbf{z} \mid \boldsymbol{\mu}^*, \boldsymbol{\Sigma}^*)$ can always be decomposed into a Gaussian marginal and a linear Gaussian conditional, in both directions $p(\mathbf{x} \mid \mathbf{z})$ and $p(\mathbf{z} \mid \mathbf{x})$. These operations happen in closed form, which makes them *fast*. The details are in appendix C.1.

This is the basis for both Kalman filters and variable marginalisation in Gauss-Newton optimisation (e. g. see Dellaert and Kaess (2017)); the latter is typical for modern SLAM. In this thesis linear Gaussian systems are used much more sparingly, in only one building block in state estimation (not prediction) when real-time operation is needed (publication in appendix A.2).

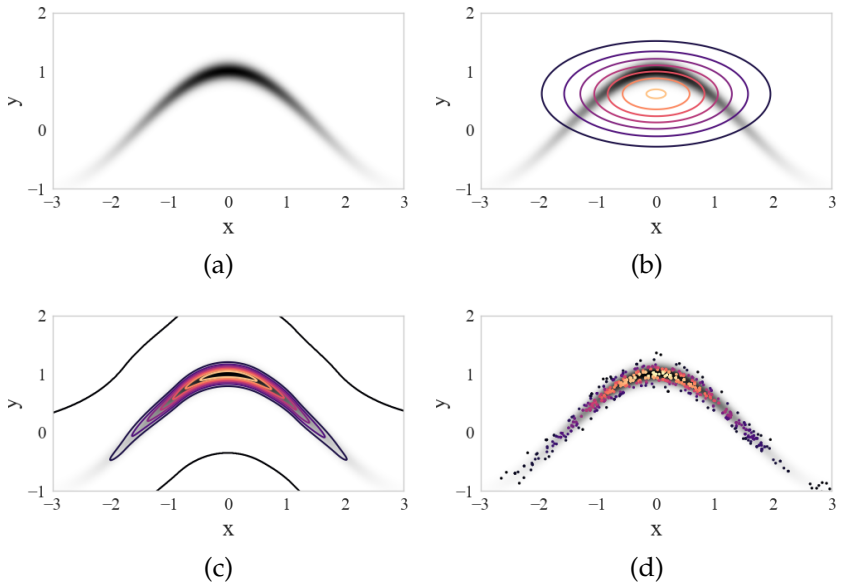


Figure 1.1.: Various options for representing distributions. (a) A target bivariate distribution $p_{\text{target}}(x, y)$ over $\mathbb{R}^2$ with a non-linear support. (b) Contour plot of a joint 2D Gaussian $p_\theta(x, y)$. It cannot capture the non-linear manifold. (c) Contour plot of a *factorised* joint $p_\theta(y \mid x)p_\theta(x)$ with Gaussian factors. The conditional is given by a neural net. It fits the non-linear manifold. (d) Particles obtained via importance sampling (appendix C.4) fit the target accurately (coloured by weight).

Gaussian Products

Products of Gaussians over the *same* variable happen in closed form:

$$\mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}^*, \boldsymbol{\Sigma}^*) \propto \mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1)\mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2) \tag{1.6}$$

$$\boldsymbol{\Sigma}^* = (\boldsymbol{\Sigma}_1^{-1} + \boldsymbol{\Sigma}_2^{-1})^{-1} \tag{1.7}$$

$$\boldsymbol{\mu}^* = \boldsymbol{\Sigma}^*(\boldsymbol{\Sigma}_1^{-1}\boldsymbol{\mu}_1 + \boldsymbol{\Sigma}_2^{-1}\boldsymbol{\mu}_2) \tag{1.8}$$

Intuitively, the mean of the resulting Gaussian is a weighted combination of the input means[3], and the resulting Gaussian's precision is the sum of the input precisions. This result is historically linked to sensor fusion (Moravec, 1988), and combined with Bayes' law it can lead to very fast inference (it is used for fast 3D map updates in appendix A.2).

[3] In the multivariate case this weighted average does not necessarily lie on the line between the two modes. It is the point of highest density overlap, which can be offset.

1.3. Inference

So far we have talked about probabilistic models as abstract simulators, and only hinted at model inference. Let us now make the distinction clear.

The models in this thesis are *latent variable models* (LVMs), also *generative* models. They simulate both variables that can be *observed* with sensors (e. g. images), as well as variables that remain *latent*, internal to the model (e. g. agent dynamic states). The observed variables (grouped in $\mathbf{x}$) are generated from the unobserved variables (grouped in $\mathbf{z}$):

$$p_\theta(\mathbf{x}, \mathbf{z}) = p_\theta(\mathbf{x} \mid \mathbf{z}) p_\theta(\mathbf{z}). \tag{1.9}$$

The conditional $p_\theta(\mathbf{x} \mid \mathbf{z})$ is known as a *likelihood*, while $p_\theta(\mathbf{z})$ simulates the unobserved $\mathbf{z}$ and holds our *prior* knowledge.

When we talk about simulation or prediction, we refer to the generative model $p_\theta(\mathbf{x}, \mathbf{z})$ – e. g. sampling from it or evaluating expectations. When we talk about inference, we mean inverting generation to infer either the simulated unknown variables $\mathbf{z}$ (section 1.3.1) or the parameters θ of the simulator $p_\theta(\mathbf{x}, \mathbf{z})$ itself (section 1.3.2).

1.3.1. Posterior Inference

Inferring the latents $\mathbf{z}$ means finding their *posterior* with Bayes' law

$$p_\theta(\mathbf{z} \mid \mathbf{x}) = \frac{p_\theta(\mathbf{z}, \mathbf{x})}{p_\theta(\mathbf{x})} = \frac{p_\theta(\mathbf{x} \mid \mathbf{z}) p_\theta(\mathbf{z})}{p_\theta(\mathbf{x})}. \tag{1.10}$$

The integral in the *normaliser* $p_\theta(\mathbf{x}) = \int p_\theta(\mathbf{x}, \mathbf{z}) \, \mathrm{d}\mathbf{z}$ is intractable in general. However, because the normaliser $p_\theta(\mathbf{x})$ is constant in $\mathbf{z}$, the posterior is proportional to the joint, which itself is a tractable starting point:

$$p_\theta(\mathbf{z} \mid \mathbf{x}) \propto p_\theta(\mathbf{z}, \mathbf{x}) = p_\theta(\mathbf{x} \mid \mathbf{z}) p_\theta(\mathbf{z}). \tag{1.11}$$

Point Estimation

The simplest posterior approximation is a point mass, given by the posterior-mode-seeking *maximum a-posteriori* (MAP) optimisation[4]

$$\mathbf{z}^* = \arg\max_{\mathbf{z}} \log p_\theta(\mathbf{x} \mid \mathbf{z}) + \log p_\theta(\mathbf{z}). \tag{1.12}$$

This is principled because $p_\theta(\mathbf{z} \mid \mathbf{x}) \propto p_\theta(\mathbf{x} \mid \mathbf{z}) p_\theta(\mathbf{z})$ (from eq. (1.11)).

[4] MLE approaches that ignore the prior and optimise only $p(\mathbf{x} \mid \mathbf{z})$ can be seen as a special case of MAP with an uninformative prior assumption.

Laplace Approximation

An MAP point-estimate $\mathbf{z}^*$ can be extended with uncertainty via a *Laplace approximation* by fitting a joint Gaussian to the joint PDF curvature of $p_\theta(\mathbf{z}, \mathbf{x})$ around $\mathbf{z}^*$. [5] This is justified because of eq. (1.11). We have

$$\boldsymbol{\Lambda} = -\underbrace{\nabla^2_{\mathbf{z}} \log p_\theta(\mathbf{z}, \mathbf{x})}_{\text{log-joint Hessian}}\Big|_{\mathbf{z}=\mathbf{z}^*} \tag{1.13}$$

$$q(\mathbf{z}) = \mathcal{N}\big(\mathbf{z} \;\big|\; \mathbf{z}^*, \boldsymbol{\Lambda}^{-1}\big) \approx p_\theta(\mathbf{z} \mid \mathbf{x}). \tag{1.14}$$

where $\boldsymbol{\Lambda} = \boldsymbol{\Sigma}^{-1}$ is the fitted Gaussian precision. For the Gaussian approximation it is required that $\boldsymbol{\Lambda}$ is positive definite, which can be ensured by approximating the Hessian in eq. (1.13) with the squared Jacobian, often done in conjunction with Gauss-Newton optimisation (appendix C.2 explains the connection, but it is tangent to the research).

Variational Inference

The combination of MAP and a Laplace approximation is limited to a joint Gaussian over all latent variables in $\mathbf{z}$, and we already know this implies linearity. One way to relax this is with *variational inference* (VI), which finds a parametric *approximate posterior* $q_\phi(\mathbf{z})$ by minimising the KL divergence

$$\boldsymbol{\phi}^* = \arg\min_{\boldsymbol{\phi}} \mathrm{KL}\big(q_{\boldsymbol{\phi}}(\mathbf{z}) \;\big||\; p_\theta(\mathbf{z} \mid \mathbf{x})\big). \tag{1.15}$$

Since $p_\theta(\mathbf{z} \mid \mathbf{x})$ is unknown, Bayes' law is used to flip conditionals and introduce the known joint factorisation $p_\theta(\mathbf{z}, \mathbf{x}) = p_\theta(\mathbf{z}) p_\theta(\mathbf{x} \mid \mathbf{z})$:

$$\begin{aligned}
&\mathrm{KL}\big(q_{\boldsymbol{\phi}}(\mathbf{z}) \;\big||\; p_\theta(\mathbf{z} \mid \mathbf{x})\big) \\
&\quad = \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z})}\big[\log q_{\boldsymbol{\phi}}(\mathbf{z}) - \log p_\theta(\mathbf{z} \mid \mathbf{x})\big] \\
&\quad = \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z})}\big[\log q_{\boldsymbol{\phi}}(\mathbf{z}) - \log p_\theta(\mathbf{z}) - \log p_\theta(\mathbf{x} \mid \mathbf{z}) + \log p_\theta(\mathbf{x})\big] \\
&\quad = \mathrm{KL}\big(q_{\boldsymbol{\phi}}(\mathbf{z}) \;\big||\; p_\theta(\mathbf{z})\big) - \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z})}[\log p_\theta(\mathbf{x} \mid \mathbf{z})] + \log p_\theta(\mathbf{x}).
\end{aligned} \tag{1.16}$$

Considering that $p_\theta(\mathbf{x})$ is constant in $\boldsymbol{\phi}$, we get the objective

$$\boldsymbol{\phi}^* = \arg\min_{\boldsymbol{\phi}} \mathrm{KL}\big(q_{\boldsymbol{\phi}}(\mathbf{z}) \;\big||\; p_\theta(\mathbf{z})\big) - \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z})}[\log p_\theta(\mathbf{x} \mid \mathbf{z})]. \tag{1.17}$$

[5] Laplace approximations work for any $p(\mathbf{z}) \propto f(\mathbf{z})$, using a maximum and the curvature of $f(\mathbf{z})$. The discussion is intentionally limited to posteriors for simplicity.

Note that $\mathbf{x}$ here is the *data* we infer from, i. e. the known observations in the condition of the target posterior $p_\theta(\mathbf{z} \mid \mathbf{x})$. The term $\log p_\theta(\mathbf{x} \mid \mathbf{z})$ is known as a *reconstruction* log-likelihood, and maximising it finds a $\mathbf{z}$ that improves the generation of observed data. The KL divergence $\mathrm{KL}\big(q_\phi(\mathbf{z}) \,\big|\big|\, p_\theta(\mathbf{z})\big)$ is between the approximate posterior and the latent prior, respecting all prior assumptions we've made about the unobserved $\mathbf{z}$.

In practice, expectations in eq. (1.17) are usually MC-estimated using samples from $q_\phi(\mathbf{z})$, and ϕ is optimised with variants of stochastic gradient descent. Gradients also propagate through the MC sampling, via a reparameterisation trick. This is explained in more detail in appendix C.3.

Contrary to MAP followed by a Laplace approximation, $q_\phi(\mathbf{z})$ is flexible and can factorise into non-linear, even non-Gaussian, factors in the sense of section 1.2 that can be as complex as normalising flows (Rezende and Mohamed, 2015).

1.3.2. Model Parameter Inference

Next to inferring simulated variables $\mathbf{z}$ there is also the inference of deterministic model parameters θ in the parametric generative model $p_\theta(\mathbf{x}, \mathbf{z}) = p_\theta(\mathbf{x} \mid \mathbf{z}) p_\theta(\mathbf{z})$. For this we would want to maximise the log-likelihood of the observed data given θ, but $\log p_\theta(\mathbf{x}) = \log \int p_\theta(\mathbf{x}, \mathbf{z}) \, \mathrm{d}\mathbf{z}$ is not tractable due to the integral.

Fortunately, if we rearrange eq. (1.16) we see that

$$\begin{aligned} \mathcal{L}_{\text{elbo}}(\phi, \theta) &:= \mathbb{E}_{q_\phi(\mathbf{z})}[\log p_\theta(\mathbf{x} \mid \mathbf{z})] - \mathrm{KL}\big(q_\phi(\mathbf{z}) \,\big|\big|\, p_\theta(\mathbf{z})\big) \\ &= \log p_\theta(\mathbf{x}) - \mathrm{KL}\big(q_\phi(\mathbf{z}) \,\big|\big|\, p_\theta(\mathbf{z} \mid \mathbf{x})\big) \\ &\leqslant \log p_\theta(\mathbf{x}). \end{aligned} \tag{1.18}$$

The objective $\mathcal{L}_{\text{elbo}}(\phi, \theta)$ is known as the *evidence lower bound (ELBO)*, since it is smaller or equal to the marginal log-likelihood $\log p_\theta(\mathbf{x})$ (evidence in Bayesian terminology). It is thus a principled surrogate objective:

$$\theta^* = \arg\max_\theta \mathcal{L}_{\text{elbo}}(\phi, \theta) \approx \arg\max_\theta \log p_\theta(\mathbf{x}). \tag{1.19}$$

Note that this is the same objective as in eq. (1.17). Optimisation of θ proceeds analogously to how we optimised $q_\phi(\mathbf{z})$ before.

Equations (1.17) and (1.18) show the ELBO has a dual purpose: it is used to find variational posteriors and thus estimate $\mathbf{z}$, but it can also be used to optimise the predictive model $p_\theta(\mathbf{x}, \mathbf{z})$ itself. Both can be done in parallel, and the lower bound is tight when $q_\phi(\mathbf{z}) = p_\theta(\mathbf{z} \mid \mathbf{x})$.

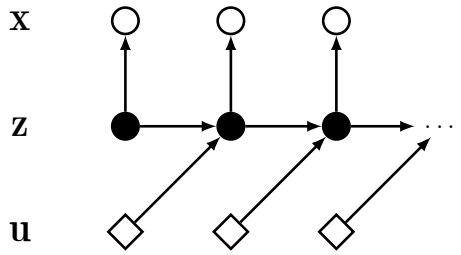


Figure 1.2.: A state-space model with observations $\mathbf{x}_t$ (white circles), latents $\mathbf{z}_t$ (black circles) and controls $\mathbf{u}_t$ (white diamonds).

1.4. State-Space Models

We now turn our attention to *state-space models* (SSM) that simulate a sequence of observations $\mathbf{x}_{1:T}$ (e. g. images) and latents $\mathbf{z}_{1:T}$ (e. g. dynamic states) over discrete time steps $t = 1, \dots, T$, conditional on controls $\mathbf{u}_{1:T-1}$. An SSM has the following factorisation:

$$\begin{aligned} &p_\theta(\mathbf{z}_{1:T}, \mathbf{x}_{1:T} \mid \mathbf{u}_{1:T-1}) \\ &\quad = p_\theta(\mathbf{z}_1) p_\theta(\mathbf{x}_1 \mid \mathbf{z}_1) \prod_{t=1}^{T-1} p_\theta(\mathbf{x}_{t+1} \mid \mathbf{z}_{t+1}) p_\theta(\mathbf{z}_{t+1} \mid \mathbf{z}_t, \mathbf{u}_t). \end{aligned} \tag{1.20}$$

For brevity, we will omit controls $\mathbf{u}_t$ from $p_\theta(\mathbf{z}_{t+1} \mid \mathbf{z}_t, \mathbf{u}_t)$ in the following sections, they are always implied. An SSM depiction is given in fig. 1.2.

SSMs have two main properties. First, each *emission*[6] factor $p_\theta(\mathbf{x}_t \mid \mathbf{z}_t)$ is independent from all past and future variables (i. e. $\mathbf{z}_{<t}, \mathbf{x}_{<t}, \mathbf{x}_{>t}, \mathbf{z}_{>t}$). Second, the *transition*[7] factors $p_\theta(\mathbf{z}_{t+1} \mid \mathbf{z}_t)$ form a Markov chain, which means knowing $\mathbf{z}_t$ alone is sufficient to predict the future.

As we will see shortly both of these properties a) streamline the recursion of the Bayes filter and b) tightly align with model-based control (Bertsekas, 2005). This is why the thesis adheres to an SSM factorisation.

1.4.1. Prediction in State-Space Models

Like any other generative model, state-space models are also interpreted as simulators. One way to simulate with them is to sample ancestrally over

[6] Also measurement model.
[7] Also dynamics model.

the Markov chain. A *rollout* of samples $\{\mathbf{z}_{1:T}^k, \mathbf{x}_{1:T}^k\}_{[K]}$ can be obtained with

$$\mathbf{z}_t^k \sim p_\theta\left(\mathbf{z}_t \mid \mathbf{z}_{t-1}^k\right), \quad \mathbf{x}_t^k \sim p_\theta\left(\mathbf{x}_t \mid \mathbf{z}_t^k\right). \tag{1.21}$$

Rollout expectations can be estimated with Monte Carlo as well

$$\mathbb{E}_{p_\theta(\mathbf{z}_{1:T}, \mathbf{x}_{1:T})}(f(\mathbf{z}_{1:T}, \mathbf{x}_{1:T})) \approx \frac{1}{K} \sum_{k=1}^{K} f(\mathbf{z}_{1:T}^k, \mathbf{x}_{1:T}^k). \tag{1.22}$$

Rollouts are needed for control, when planning ahead.

1.4.2. Inference in State-Space Models

In terms of SSM inference, all of the techniques from section 1.3 apply. We will briefly cover how *smoothing* and *filtering* are approached in the thesis.

Smoothing

The biggest posterior to infer in an SSM is the *smoother* (Särkkä, 2013)

$$p_\theta(\mathbf{z}_{1:T} \mid \mathbf{x}_{1:T}) \propto p_\theta(\mathbf{z}_{1:T}, \mathbf{x}_{1:T}), \tag{1.23}$$

where every $\mathbf{z}_t$ is inferred from all observed data, before and after t. The thesis applies only variational inference to smoothing, as a more general alternative to MAP. Alternatives such as forward-backward (two-filter) smoothers are left out of scope (Särkkä, 2013).

Assuming an approximate posterior $q_\phi(\mathbf{z}_{1:T})$, the ELBO for an SSM is[8]

$$\begin{aligned}
\mathcal{L}_{\text{elbo}}(\theta, \phi) &= \mathbb{E}_{q_\phi(\mathbf{z}_{1:T})}\left[\log p_\theta(\mathbf{z}_{1:T}, \mathbf{x}_{1:T}) - \log q_\phi(\mathbf{z}_{1:T})\right] \\
&= \mathbb{E}_{q_\phi(\mathbf{z}_{1:T})}\left[\sum_{t=1}^{T} \log p_\theta(\mathbf{x}_t \mid \mathbf{z}_t)\right] \\
&\quad + \mathbb{E}_{q_\phi(\mathbf{z}_{1:T})}\left[\sum_{t=1}^{T} \log p_\theta(\mathbf{z}_t \mid \mathbf{z}_{t-1}) - \log q_\phi(\mathbf{z}_{1:T})\right].
\end{aligned} \tag{1.24}$$

In the thesis, a variant of the SSM ELBO is used to infer a smoothing posterior $q_\phi(\cdot)$ over all latents in the proposed spatial model (agent states, a 3D map). In a separate model, the SSM ELBO is also used to learn generative transition parameters θ, to improve the SSM's predictive rollouts. Both are done with stochastic optimisation, as described in section 1.3.1.

[8] For brevity, in the sums it is implied that $p_\theta(\mathbf{z}_1 \mid \mathbf{z}_0) = p_\theta(\mathbf{z}_1)$.

Filtering

When smoothing is too expensive (e. g. on a real-time budget), one can infer the *filtering* posterior instead, which is a sub-problem:

$$p_\theta(\mathbf{z}_T \mid \mathbf{x}_{1:T}) = \int p_\theta(\mathbf{z}_{1:T} \mid \mathbf{x}_{1:T}) \, \mathrm{d}\mathbf{z}_{1:T-1}. \tag{1.25}$$

SSM filters admit a specific recursion, that of the *Bayes filter* (Särkkä, 2013):

$$\begin{aligned} p_\theta(\mathbf{z}_T \mid \mathbf{x}_{1:T}) &\propto p_\theta(\mathbf{x}_T \mid \mathbf{z}_T) p_\theta(\mathbf{z}_T \mid \mathbf{x}_{1:T-1}) && (1.26)\\ &= p_\theta(\mathbf{x}_T \mid \mathbf{z}_T) \int p_\theta(\mathbf{z}_T, \mathbf{z}_{T-1} \mid \mathbf{x}_{1:T-1}) \, \mathrm{d}\mathbf{z}_{T-1} && (1.27)\\ &= p_\theta(\mathbf{x}_T \mid \mathbf{z}_T) \int p_\theta(\mathbf{z}_T \mid \mathbf{z}_{T-1}) \underbrace{p_\theta(\mathbf{z}_{T-1} \mid \mathbf{x}_{1:T-1})}_{\text{prev. filter}} \, \mathrm{d}\mathbf{z}_{T-1}. && (1.28) \end{aligned}$$

The Markovian independence of SSMs makes the recursion particularly streamlined: the integral in eq. (1.28) needs to only propagate the previous filter through the transition $p_\theta(\mathbf{z}_T \mid \mathbf{z}_{T-1})$ and the emission $p_\theta(\mathbf{x}_T \mid \mathbf{z}_T)$ remains on the outside. Because the developed spatial models target dense image generation, which implies an expensive emission, the above significantly simplifies the design of approximate non-linear filters and is used to develop a real-time solution. In essense, the SSM independences are seen as a gateway to more flexible Bayesian inference.

In general, solving the recursion can be approached in different ways. If the whole system is linearised, the linear Gaussian equations from section 1.2.2 make the recursion analytical and lead to (Extended) Kalman filters (detailed in appendix C.5). Alternatively, one can also pose the recursion as an optimisation problem, treating the integral in eq. (1.28) as a prior (detailed in appendix C.6), which is more flexible. The solution chosen in this thesis uses a custom derivation tailored to dense spatial modelling, mixing different approaches (MAP optimisation, Laplace approximation, partial linearisation of only the transition, sensor fusion).

The main caveat of approximate filtering is that the recursion can compound approximation errors, and the past cannot be easily corrected.

For the rest of the thesis, we will drop θ from $p_\theta(\cdot)$, and only use them if they are contextually relevant.

1.5. Model-Based Control

We will now look at control, i. e. finding control inputs that minimise a *cost-to-go*[9], a sum of costs c_τ over the uncertain agent future starting at agent state $\mathbf{z}_t$:

$$J(\mathbf{z}_t) = \mathbb{E}_p\left[\sum_{\tau=t}^{\infty} \gamma^{\tau-t} c_\tau\right], \tag{1.29}$$

where $\gamma < 1.0$ is a discount factor (to bound $J(\mathbf{z}_t)$). This thesis targets *model-based control*, where a model of $p(\cdot)$ enables simulation of eq. (1.29).[10]

1.5.1. Full State Information

When the agent states $\mathbf{z}_t$ are observable, the standard model for control is a Markov Decision Process (MDP), where $p(\mathbf{z}_{\geqslant t}, c_{\geqslant t}, \mathbf{u}_{\geqslant t})$ is given by *the following terms*:

$p(\mathbf{z}_t)$	initial state
$p(\mathbf{z}_\tau \mid \mathbf{z}_{\tau-1}, \mathbf{u}_{\tau-1})$	transition
$\pi(\mathbf{u}_\tau \mid \mathbf{z}_\tau)$	control policy
$c(\mathbf{z}_\tau, \mathbf{u}_\tau)$	cost function.

Note how this is only the transition chain of an SSM, extended with a policy and a cost function. See fig. 1.3a for a diagram. Because of the Markovian independences the MDP cost-to-go is recursive

$$J^\pi(\mathbf{z}_t) = \mathbb{E}_{\pi(\mathbf{u}_t|\mathbf{z}_t)}[c(\mathbf{z}_t, \mathbf{u}_t)] + \gamma \mathbb{E}_{p(\mathbf{z}_{t+1}|\mathbf{z}_t)}[J^\pi(\mathbf{z}_{t+1})]. \tag{1.30}$$

The optimal cost-to-go is then given by

$$J^*(\mathbf{z}_t) = \min_{\mathbf{u}_t} c(\mathbf{z}_t, \mathbf{u}_t) + \gamma \mathbb{E}_{p(\mathbf{z}_{t+1}|\mathbf{z}_t, \mathbf{u}_t)}[J^*(\mathbf{z}_{t+1})], \tag{1.31}$$

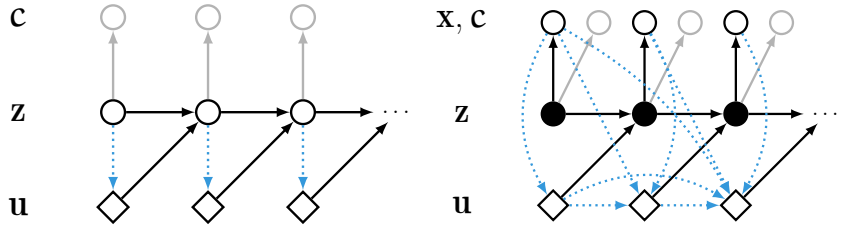
for a deterministic policy $\mathbf{u}_t = \pi^*(\mathbf{z}_t)$ that minimises every recursive step.

When states and controls are discrete, expectations become tractable vector and matrix products and we can rewrite eq. (1.31) as

$$J^* = T^* J^*. \tag{1.32}$$

[9] Same as negative *value*, in RL terminology.

[10] Up to a finite horizon, which is sufficient under a terminal cost-to-go approximation.



(a) MDP. (b) POMDP.

Figure 1.3.: Graphical models of an MDP and a POMDP. Observed variables are white circles ($\mathbf{z}_t$ is observed in an MDP), latent variables are black circles, controls are white diamonds. The cost function $c(\mathbf{z}_t)$ and cost outputs are in gray, for clarity we omit edges from controls to costs. The policy edges (observations to control) are dotted in blue, noting the MDP vs. POMDP difference (depending on observability).

where now J^* is a cost-to-go vector for all states and T^* applies eq. (1.31) elementwise to it. T^* is known as the *Bellman optimality operator* and the fixed-point iteration in eq. (1.32) is guaranteed to converge to the optimal cost-to-go from any initial J vector (see Bertsekas (2005)). This is known as *value iteration*. After convergence, the optimal policy is given by eq. (1.31).

Value iteration can be applied to continuous problems by first discretising the state and action spaces and the MDP transition, which is known as *aggregation* (Bertsekas, 2005). This is how it is used in this thesis. Actor-critic methods with a parametric neural policy and critic (i. e. cost-to-go approximation) are left out of scope (see Sutton and Barto (2018)).

1.5.2. Partial Observability

In practice, spatial agents cannot observe their state directly, the problem is *partially observable*. The standard formalism to account for this is a *Partially Observable Markov Decision Process* (POMDP). This is exactly why we discussed state-space models – POMDPs follow the same Markovian state-space assumptions, adding a state-to-observation emission $p(\mathbf{x}_t \mid \mathbf{z}_t)$ to the MDP components. See fig. 1.3b for a diagram.

Because the state $\mathbf{z}_t$ is not observed, an optimal POMDP policy needs all past observed data $\mathbf{x}_{\leqslant t}, \mathbf{u}_{<t}$ as input (Bertsekas, 2005) [11]. Since past data keeps growing this is unwieldy, and one workaround is to condition the

[11] Due to the dependences in the underlying SSM, the future depends on all past observed data if the latent states are not given.

policy on the *distribution parameters* $\boldsymbol{\psi}_t$ (e. g. mean and covariance) of the filtering posterior $p_{\boldsymbol{\psi}_t}(\mathbf{z}_t) = p(\mathbf{z}_t \mid \mathbf{x}_{\leqslant t}, \mathbf{u}_{<t})$ of the underlying SSM. The filter parameters $\boldsymbol{\psi}_t$ are sufficient statistics for $\mathbf{x}_{\leqslant t}, \mathbf{u}_{<t}$.

This results in a "lifted" MDP with states $\boldsymbol{\psi}_t$ and new cost and transition conditionals derived from the original POMDP. Its transition is given by

$$p(\boldsymbol{\psi}_t \mid \boldsymbol{\psi}_{t-1}, \mathbf{u}_{t-1}) = \iiint \underbrace{\delta\big(\boldsymbol{\psi}_t = f_{\text{update}}(\boldsymbol{\psi}_{t-1}, \mathbf{x}_t, \mathbf{u}_{t-1})\big)}_{\text{filter param. update}} \underbrace{p(\mathbf{x}_t \mid \mathbf{z}_t) p(\mathbf{z}_t \mid \mathbf{z}_{t-1}, \mathbf{u}_{t-1})}_{\text{SSM transition and emission}} \underbrace{p_{\boldsymbol{\psi}_{t-1}}(\mathbf{z}_{t-1})}_{\text{prev. filter, given by } \boldsymbol{\psi}_{t-1}} \mathrm{d}\mathbf{z}_{t-1} \, \mathrm{d}\mathbf{z}_t \, \mathrm{d}\mathbf{x}_t. \tag{1.33}$$

The crux here is that this new MDP can be solved with the regular MDP control toolkit, which like before means infinite-horizon planning via cost-to-go estimation is possible. However, this is also expensive. In addition to the regular SSM transition, it requires imagining what the agent would see with $p(\mathbf{x}_t \mid \mathbf{z}_t)$ and imagining how that would influence the filter parameters with $f_{\text{update}}(\boldsymbol{\psi}_{t-1}, \mathbf{x}_t, \mathbf{u}_{t-1})$. The lifted states $\boldsymbol{\psi}_t$ are referred to as *beliefs* and planning in the new MDP as *belief-space planning*, as opposed to *state-space planning* in a regular fully-observed MDP with states $\mathbf{z}_t$. Please see Bertsekas (2005) for a thorough treatment and Placed et al. (2023) for a link to SLAM methods.

The above is the price for being aware of state-estimation errors when planning. Note that for it to work, state estimation uncertainties should be well-calibrated, to inform the controller of potential failures. While hard to execute, it seems desirable that spatial models should strive to support such estimator-aware control in the long run. This is the ideal from the perspective of the thesis, hence the Bayesian standpoint and the active pursuit of non-linear inference methods.

In practice, because belief-space planning is costly on current commodity hardware, a common compromise is to use the filter $p(\mathbf{z}_t \mid \mathbf{x}_{\leqslant t}, \mathbf{u}_{<t})$ as an initial state distribution $p(\mathbf{z}_t)$ and proceed with state-space planning, reverting to the regular MDP assumptions and ignoring the effect of state estimation in future rollouts.

Regardless of the chosen method, these considerations motivate the use of SSMs, as they unlock both belief- and state-space planning in the POMDP / MDP framework, and ensure prediction and state estimation are consistent with each other and come with uncertainty.

Spatial Domain Knowledge

2

With the SSM framework in place, we will now focus on the spatial modelling of the individual graph factors $p(\mathbf{z}_{t+1} \mid \mathbf{z}_t, \mathbf{u}_t)$ and $p(\mathbf{x}_t \mid \mathbf{z}_t)$.

2.1. Rigid-Body Movement

We assume the agent is a moving rigid body. This section is about how to represent its 6-DoF dynamic states $\mathbf{z}_t$, using rotations, translations, angular and linear velocities, such that we can still express distributions $p(\mathbf{z}_t)$.

2.1.1. Rotations

The group of 3D rotations $\mathbf{R} \in \mathrm{SO}(3)$[1] is a Lie group, not isomorphic to any vector space $\mathbb{R}^n$. However, an additive representation in $\mathbb{R}^n$ is needed both for gradient-based optimisation and for distributions. Rotations are hence *parameterised* with vectors $\theta \in \mathbb{R}^n$ that are mapped to $\mathrm{SO}(3)$:

$$f_{\mathrm{rot}}(\theta) = \mathbf{R}, \quad \theta \in \mathbb{R}^n,\ \mathbf{R} \in \mathrm{SO}(3). \tag{2.1}$$

Note that θ here is not the same as the one in $p_\theta(\cdot)$, notation overlaps. Distributions in the thesis are on the rotation parameters θ directly, i. e. θ is a part of the $\mathbf{z}_t$ states. Following are two different options for θ and f_{rot}.

Lie-Algebra Parameters

One option is to put rotation parameters in the *Lie algebra* $\mathfrak{so}(3)$, which is the linear space of 3D skew-symmetric matrices

$$[\theta]_\times = \begin{bmatrix} 0 & -\theta_z & \theta_y \\ \theta_z & 0 & -\theta_x \\ -\theta_y & \theta_x & 0 \end{bmatrix}.$$

[1] Rotation matrices $\mathbf{R}$ as treated as direct elements of $\mathrm{SO}(3)$ for brevity, see appendix C.8.

Skew-symmetric matrices are isomorphic to the space of 3D vectors $\theta \in \mathbb{R}^3$ in terms of cross products, in the sense that

$$\forall \mathbf{p} \in \mathbb{R}^3 : \quad \left[\theta\right]_\times \mathbf{p} = \theta \times \mathbf{p}. \tag{2.2}$$

We can thus use the vector $\theta \in \mathbb{R}^3$ directly, the parameterisation is minimal. The mapping of $\theta \in \mathbb{R}^3$ to a rotation $\mathbf{R} \in \mathrm{SO}(3)$ is given by

$$\mathrm{f}_{\mathrm{rot}}(\theta) = \exp\Big(\left[\theta\right]_\times\Big) \tag{2.3}$$

$$= \mathbf{I} + \left[\frac{\theta}{\|\theta\|}\right]_\times \sin\|\theta\| + \left[\frac{\theta}{\|\theta\|}\right]_\times^2 (1 - \cos\|\theta\|). \tag{2.4}$$

This is known as the *exponential map* of $\mathrm{SO}(3)$ and also as the *Euler-Rodrigues formula* (Dai, 2015). The mapping is differentiable, and thus suitable for gradient-based optimisation.

The vectors θ associated with the Lie-algebra are also referred to as *axis-angle* or *rotation vectors*. Their magnitude $\|\theta\|$ gives the rotation angle around the normalised axis $\frac{\theta}{\|\theta\|}$. The mapping is thus many-to-one, and parameters are equivalent in a repeating pattern with a period of 2π, i. e.:

$$\mathrm{f}_{\mathrm{rot}}(\theta) = \mathrm{f}_{\mathrm{rot}}((\|\theta\| + 2k\pi)\frac{\theta}{\|\theta\|}), \quad k \in \mathbb{Z}. \tag{2.5}$$

This is a short summary, appendix C.9 has more details on $\mathfrak{so}(3)$.

Quaternions

An alternative rotation representation are *unit quaternions*:

$$\mathbf{q} = (\cos\frac{\alpha}{2}, \sin\frac{\alpha}{2}\mathbf{v}) \in \mathbb{R}^4 \tag{2.6}$$

$$\text{where} \quad \mathbf{v} \in \mathbb{R}^3, \|\mathbf{v}\| = 1. \tag{2.7}$$

Intuitively, $\mathbf{v}$ is a unit 3D axis around which we rotate by an angle α. Unit quaternions are thus related to Lie-algebra parameters, as $\mathbf{v}$ is collinear with a respective Lie-algebra vector θ, and $\alpha = \|\theta\|$ (modulo 2π). Respectively, a unit quaternion $\mathbf{q}$ is mapped onto $\mathrm{SO}(3)$ via

$$\mathrm{g}_{\mathrm{rot}}(q) = \mathbf{I} + [\mathbf{v}]_\times \sin\alpha + [\mathbf{v}]_\times^2 (1 - \cos\alpha), \tag{2.8}$$

where $[\mathbf{v}]_\times$ denotes the skew-symmetric matrix parameterised by the unit vector $\mathbf{v} \in \mathbb{R}^3$. The mapping is similar to the one for Lie-algebra vectors in

eq. (2.4). With that, any non-zero vector $\theta \in \mathbb{R}^4$ can be first normalised and then projected to $SO(3)$, which gives the parameterisation mapping:

$$f_{rot}(\theta) = g_{rot}(\frac{\theta}{\|\theta\|}). \tag{2.9}$$

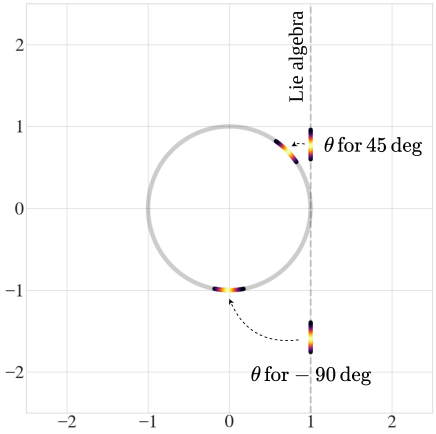
This is also many-to-one, collinear vectors map to the same rotation:

$$f_{rot}(\theta) = f_{rot}(\eta\theta), \quad \eta \in \mathbb{R}/\{0\}. \tag{2.10}$$

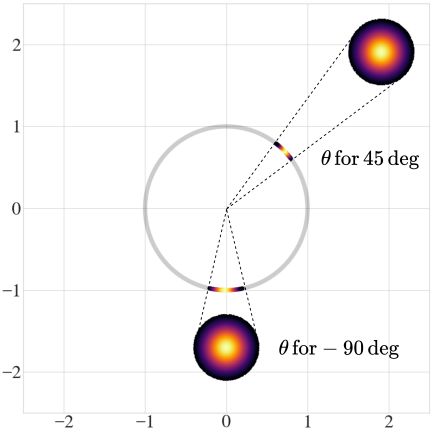
For a proper treatment of quaternions, see Gallier (2011).

Comparison

The difference between the two parameterisations is illustrated intuitively for 1D rotations in fig. 2.1. The mappings f_{rot} for both Lie-algebra and quaternion parameterisations are many-to-one, with topological differences in the translation of uncertainty from parameter to rotation space.



(a) Lie-algebra for 1D rotations.



(b) "Quaternion" equivalent for 1D rotations.

Figure 2.1.: Analogues of Lie-algebra parameters and quaternions for 1D rotations. Lie-algebra parameters are in $\mathbb{R}$, and are mapped via the exponential map, wrapping around the rotation manifold. Unnormalised "quaternion"-analogue parameters are in $\mathbb{R}^2$, and are projected onto the unit circle manifold S^1. The plots show how whole distributions over parameters are mapped, with the distribution modes in orange.

Because quaternions need to be projected to the unit sphere, the uncertainty over unnormalised quaternion parameters θ shrinks in terms of $\mathrm{SO}(3)$ when $\|\theta\|$ increases. It is therefore advisable that e. g. mean parameters for Gaussians over quaternions are kept close to unit.

Overall, while unnormalised quaternion vectors can serve as parameters, they are less elegant than parameters on the Lie-algebra. The only reason they are discussed here is because they appear in some of the publications due to legacy implementation code, before Lie-algebra vectors were appreciated and adopted by the author.

2.1.2. Rigid-Body Transformations

Rotations combined with translations form the special Euclidean group $\mathrm{SE}(3)$ of *rigid-body transformations*. $\mathrm{SE}(3)$ elements represent both relative movement, as well as absolute *poses* relative to a reference frame. They are matrices $\mathbf{T} \in \mathbb{R}^{4\times4}$ of the form

$$\mathbf{T} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ 0 & 1 \end{bmatrix}, \quad \mathbf{t} \in \mathbb{R}^3, \mathbf{R} \in \mathrm{SO}(3). \tag{2.11}$$

$\mathrm{SE}(3)$ transformations are composed through matrix multiplication $\mathbf{T}_1\mathbf{T}_2$, and $\mathbf{Tp}$ rotates and translates a *homogeneous* point $\mathbf{p} \in \mathbb{R}^4$.

Because of the rotation component, $\mathrm{SE}(3)$ is not closed under addition. We again work with parameters $\xi \in \mathbb{R}^n$ and a mapping $\mathsf{f}_{\text{pose}} : \mathbb{R}^n \to \mathrm{SE}(3)$ from parameters to group elements.

Lie-Algebra Parameters

$\mathrm{SE}(3)$ is a Lie group, and like $\mathrm{SO}(3)$ it has its own Lie algebra $\mathfrak{se}(3)$. Parameterising in $\mathfrak{se}(3)$ proceeds analogously to parameterising in $\mathfrak{so}(3)$.

The Lie-algebra elements have the form

$$\hat{\xi} = \begin{bmatrix} [\theta]_\times & \boldsymbol{\tau} \\ \mathbf{0} & 1 \end{bmatrix}, \quad \theta \in \mathbb{R}^3, \boldsymbol{\tau} \in \mathbb{R}^3. \tag{2.12}$$

They have six degrees of freedom (6 DoF) and the algebra is isomorphic to $\mathbb{R}^6$. In practice we represent them as the corresponding vectors $\xi \in \mathbb{R}^6$, where $\xi = (\theta, \boldsymbol{\tau})$. Using f_{rot} from eq. (2.4), the parameter mapping is

$$\mathsf{f}_{\text{pose}}(\xi) = \begin{bmatrix} \mathsf{f}_{\text{rot}}(\theta) & g(\theta)\boldsymbol{\tau} \\ \mathbf{0} & 1 \end{bmatrix} \tag{2.13}$$

$$g(\theta) = \mathbf{I} + [\theta]_\times \frac{1 - \cos\|\theta\|}{\|\theta\|^2} + [\theta]_\times^2 \frac{\|\theta\| - \sin\|\theta\|}{\|\theta\|^3}. \tag{2.14}$$

Parameterising with Quaternions

If quaternions are used to represent the rotation component, the rigid-body transformation parameters become

$$\boldsymbol{\xi} = (\boldsymbol{\theta}, \mathbf{t}), \quad \boldsymbol{\theta} \in \mathbb{R}^4, \mathbf{t} \in \mathbb{R}^3. \tag{2.15}$$

With $\mathrm{f}_{\mathrm{rot}}$ from eq. (2.9), the mapping to $\mathrm{SE}(3)$ is then

$$\mathrm{f}_{\mathrm{pose}}(\boldsymbol{\xi}) = \begin{bmatrix} \mathrm{f}_{\mathrm{rot}}(\boldsymbol{\theta}) & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix}.$$

2.1.3. Basic Kinematics

Next comes the parameterisation of angular and linear velocities and basic kinematic equations used in transition models $\mathrm{p}(\mathbf{z}_{t+1} \mid \mathbf{z}_t, \mathbf{u}_t)$, with $\mathbf{u}_t$ acceleration control inputs. Velocities are expressed in the world frame, as vectors

$$\mathbf{z}^{\mathrm{vel}} = (\boldsymbol{\omega}, \mathbf{v}), \quad \boldsymbol{\omega} \in \mathbb{R}^3, \mathbf{v} \in \mathbb{R}^3.$$

The vector $\boldsymbol{\omega}$ is a Lie-algebra axis-angle vector for the angular velocity. The Lie algebra appears here again because it is a tangent space whose elements are related to derivatives (details in eq. (C.58)). The vector $\mathbf{v}$ is a linear velocity in world frame. The basic rigid-body kinematic equation is then

$$\begin{pmatrix} \mathbf{z}^{\mathrm{pose}}_{t+1} \\ \mathbf{z}^{\mathrm{vel}}_{t+1} \end{pmatrix} = \begin{pmatrix} \mathbf{R}_{t+1} \\ \mathbf{t}_{t+1} \\ \boldsymbol{\omega}_{t+1} \\ \mathbf{v}_{t+1} \end{pmatrix} = \begin{pmatrix} \exp(\boldsymbol{\omega}_t \Delta t)\mathbf{R}_t \\ \mathbf{t}_t + \mathbf{v}_t \Delta t \\ \boldsymbol{\omega}_t + \mathbf{u}^{\mathrm{ang}}_t (\Delta t)^2 \\ \mathbf{v}_t + \mathbf{u}^{\mathrm{lin}}_t (\Delta t)^2 \end{pmatrix},$$

where $\mathbf{u}^{\mathrm{ang}}_t$ and $\mathbf{u}^{\mathrm{lin}}_t$ are respectively angular and linear acceleration control inputs in world frame, and Δt is the integration time step. In the above, $\exp(\boldsymbol{\omega}_t \Delta_t)$ is the exponential map of the rotation Lie group and integrates the angular velocity $\boldsymbol{\omega}_t$ over time Δt.

The publication in appendix A.1 explores learning a corrective dynamics model on top of engineered kinematics, to account for external influences on the dynamics and bias in the control inputs.

2.2. Cameras and Rendering

This section has the relevant background on cameras and rendering, needed for emissions $\mathrm{p}(\mathbf{x}_t \mid \mathbf{z}_t)$ that directly generate images.

2.2.1. Camera Geometry

A model of image generation can simulate the physics of real-world cameras. In this thesis this is done via a projective pinhole camera model, aligned with standard computer vision assumptions (Hartley and Zisserman, 2006).

About notation: in this section 3D points are represented with homogeneous coordinates in $\mathbb{R}^4$. Also, the following symbols are used:

- $\mathbf{p}_w \in \mathbb{R}^4$ is a homogeneous point in world frame;
- $\mathbf{p}_c \in \mathbb{R}^4$ is a homogeneous point in camera frame;
- $\mathbf{T}_a^b \in \mathbb{R}^4$ is an $\mathrm{SE}(3)$ pose from frame a to frame b;
- (x, y) are coordinates in the image plane;
- d is the depth for a given image coordinate.

With that, let us define

$$\mathbf{K} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}, \quad \mathbf{P} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}. \tag{2.16}$$

Here, $\mathbf{K}$ is the *intrinsic camera matrix* (Hartley and Zisserman, 2006). It holds the camera focal lengths f_x and f_y, as well as the principal offset c_x, c_y of the camera center relative to the origin of the image plane. The matrix $\mathbf{P}$ can remove the trailing 1 in homogeneous points below.[2]

We define the projection function

$$\pi : \mathbb{R}^4 \to \Omega \times \mathbb{R},$$

which maps a homogeneous 3D point in the camera frame to a coordinate in the image plane $\Omega \subset \mathbb{R}^2$ and a depth in $\mathbb{R}$. It is given by

$$\pi(\mathbf{p}_c) = (x, y, d), \quad \text{where} \quad d \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \mathbf{K}\mathbf{P}\mathbf{p}_c. \tag{2.17}$$

The multiplication with $\mathbf{K}$ captures the projection geometry.

The projection can also be inverted

$$\pi^{-1}(x, y, d) = \begin{bmatrix} d\mathbf{K}^{-1}[x, y, 1]^T \\ 1 \end{bmatrix} = \mathbf{p}_c. \tag{2.18}$$

[2] Technically $\mathbf{P}$ is also a standard projective matrix, see Hartley and Zisserman (2006).

$\pi^{-1}(\cdot)$ returns a homogeneous 3D point in camera frame, given a pixel coordinate (x, y) and a depth d.

Across the thesis, it is assumed that cameras are calibrated and images are undistorted; respectively, the camera intrinsics do not include a distortion model.

Projecting World Points

We are often interested in projecting world points $\mathbf{p}_w$ into the image plane and vice versa, for example in rendering. For this, first the inverse camera pose $\mathbf{T}_w^c$ is used to move the world points to camera frame, after which they are projected:

$$\pi(\mathbf{T}_w^c \mathbf{p}_w) = (x, y, d). \tag{2.19}$$

Respectively, finding a 3D world point $\mathbf{p}_w$ that corresponds to a given pixel (x, y) with depth d is given by

$$\mathbf{T}_c^w \pi^{-1}(x, y, d) = \mathbf{p}_w. \tag{2.20}$$

Projection Between Images

Given two camera poses $\mathbf{T}_{c1}^w$ and $\mathbf{T}_{c2}^w$, it is straightforward to re-project an image pixel (x_1, y_1) and depth d_1 from one image into another

$$(x_2, y_2, d_2) = \pi(\mathbf{T}_w^{c2} \cdot \mathbf{T}_{c1}^w \cdot \pi^{-1}(x_1, y_1, d_1)). \tag{2.21}$$

This image coordinate (pixel) association is the basis for photometric constraints (i. e. colours in different images must agree) as well as geometric constraints (i. e. the depth in RGB-D images must agree). It is differentiable and enables the optimisation of poses $\mathbf{T}_{c1}^w, \mathbf{T}_{c2}^w$ and depth d_1.

Such reprojection is at the heart of direct image alignment (e. g. Steinbrücker et al. (2011)) and iterative closest point (ICP) algorithms (Chen and Medioni, 1992). It is also used to formulate a surrogate optimisation objective for the core publications in appendices A.2 and A.3.

2.2.2. Rendering

Rendering images requires reasoning about the direction of light rays that enter the camera. Based on the pinhole model, the ray through the pixel coordinate (x, y) is expressed in world coordinates with the line equation

$$r_{xy}(d) = \mathbf{T}_c^w \pi^{-1}(x, y, d) \in \mathbb{R}^3. \tag{2.22}$$

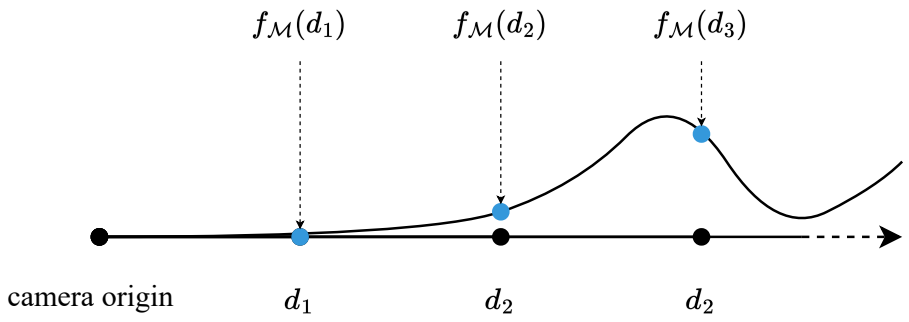


Figure 2.2.: Raycasting diagram, evaluating $f_{\mathcal{M}}$ along a single ray. The outputs of $f_{\mathcal{M}}$ are multidimensional, the depiction is in 1D for simplicity.

Here $\mathbf{T}_c^w$ is a camera-to-world pose and we implicitly drop the homogeneous part of $r_{xy}(d)$. Note that the inverse projection π^{-1} is linear in d (see eq. (2.18)). Increasing d slides along the ray.

This thesis uses *volumetric rendering* (Curless and Levoy, 1996; Parker et al., 1998). First, a scene function $f_{\mathcal{M}}$ ($\mathcal{M}$ stands for map) is evaluated for discrete points along the ray for different $d_k, k = 1, \ldots, K$:

$$\begin{aligned} &f_{\mathcal{M}} : \mathbb{R}^3 \to \mathbb{R}^4 \\ &f_{\mathcal{M}}(r_{xy}(d_k)) = (\mathbf{c}_{xy,k}, o_{xy,k}). \end{aligned} \tag{2.23}$$

This is shown in fig. 2.2. The function $f_{\mathcal{M}}$ returns a color $\mathbf{c}_{xy,k} \in \mathbb{R}^3$ and a signed distance $o_{xy,k} \in \mathbb{R}$ to the nearest surface, for all queried points along the ray. The renderer then searches for the first d_k that is a surface crossing and returns the depth and color at that point. Note that this limits the model to only objects with crisp solid boundaries, e. g. smoke or transparency are not modelled. More advanced rendering via light aggregation, like with neural radiance fields (Mildenhall et al., 2020), is left for future work.

The core publication in appendix A.1 further extends the above by making rendering differentiable and framing it as a conditional emission $p(\mathbf{x}_t \mid \mathbf{z}_t, \mathcal{M})$. The map can be seen as emission parameters.

2.2.3. Scene Representation

The next question is how to parameterise the continuous $\mathbb{R}^3$ field of $f_{\mathcal{M}}$.

Grid Representations

One option is to organise scene parameters in a 3D grid $\mathcal{M}$ with cells $\mathcal{M}_{ijk} \in \mathbb{R}^4$. To ensure $f_{\mathcal{M}}$ is differentiable, one can *trilinearly interpolate* grid parameters

$$f_{\mathcal{M}}(\mathbf{p}) = \sum_{ijk} \alpha_{ijk}(\mathbf{p}) \mathcal{M}_{ijk}. \tag{2.24}$$

The indices ijk run over the eight neighbour voxels around the continuous point $\mathbf{p}$. The weight coefficients $\alpha_{ijk}(\mathbf{p})$ are formed by linearly interpolating along one dimension at a time, combining the previously interpolated results. The equations can be found in the survey of Canelhas et al. (2018).

The main advantages of grids is that they are fast to evaluate (as parameters are queried locally) and that they can be updated in closed form (e. g. with signed-distance function updates (Curless and Levoy, 1996)). The main caveat is that their memory grows cubically, which can be relaxed with octrees (Meagher, 1982) and voxel hashing (Nießner et al., 2013), effectively pruning grid cells in free space.

Volumetric grid representations are used in all core publications (appendices A.1 to A.3), and they are also extended with uncertainty. In appendix A.2, the possibility to update voxel grids in closed form leads to a method that runs in real-time.

Neural Fields

A *coordinate neural network* can also define a continuous field over $\mathbb{R}^3$:

$$f_{\mathcal{M}}(\mathbf{p}) = \mathrm{NN}(\mathbf{p}). \tag{2.25}$$

$\mathcal{M}$ would then be the weights of the neural net. Such networks have been shown to reach high fidelity in recent years (e. g. Mildenhall et al. (2020)).

Neural nets have the potential for better memory efficiency than grids. They can also be conditioned on additional inputs, leading to scene models that change over time (e. g. Z. Li et al. (2021)) or distributions of scene priors (e. g. Kosiorek et al. (2021)). Their main disadvantage is that they are expensive, as the whole network is executed for every query point. This is why the community is steering towards spatially decomposed networks, with the aim to combine the benefits of both grids and neural nets (e. g. Reiser et al. (2021), Lombardi et al. (2021), and Müller et al. (2022)).

Because vanilla grids are generally faster to evaluate and are easy to augment with uncertainty and update in closed form, they were prioritised over neural nets in the thesis.

Part II.

Thesis Core

The Problem

3

This thesis proposes a spatial model that *simulates* how a robot moves in a static scene, how the scene looks and how the robot visually perceives it. However, model parameters are unknown and need to be obtained from online sensor data. We assume the robot only has an on-board RGB-D camera and access to past acceleration control inputs.

On a high level, from past image-and-control data $\mathcal{D}_{\leqslant t}$ we seek both to uncover a scene map $\mathfrak{M}$ and to predict future dynamic agent states $\mathbf{z}_{>t}$ and future to-be-observed images $\mathbf{x}_{>t}$ for hypothetical future controls $\mathbf{u}_{\geqslant t}$,

$$(\mathbf{x}_{>t}, \mathbf{z}_{>t}, \mathfrak{M}) = f_{\text{model}}(\mathbf{u}_{\geqslant t}, \mathcal{D}_{\leqslant t}). \tag{3.1}$$

In other words, we seek a model that both *reveals* (unknown map & dynamic states) and *predicts* (system evolution given controls).

In reality, there are modelling errors, system and data noise that one cannot correct for. To be robust toward them we need uncertainty. Hence the deterministic function above evolves into the conditional distribution

$$p(\mathbf{x}_{>t}, \mathbf{z}_{>t}, \mathfrak{M} \mid \mathbf{u}_{\geqslant t}, \mathcal{D}_{\leqslant t}). \tag{3.2}$$

The developed model enables this conditional. One way to predict with it is to sample many rollouts and maps $\{\mathbf{x}^k_{>t}, \mathbf{z}^k_{>t}, \mathfrak{M}^k\}_{k\in[K]}$, where

$$\mathbf{x}^k_{>t}, \mathbf{z}^k_{>t}, \mathfrak{M}^k \sim p(\mathbf{x}_{>t}, \mathbf{z}_{>t}, \mathfrak{M} \mid \mathbf{u}_{\geqslant t}, \mathcal{D}_{\leqslant t}). \tag{3.3}$$

The main goal of the thesis is to approximate this distribution in a way that scales to dense 3D assumptions. The next sections highlight the overall purpose and justify the modelling choices throughout the thesis.

3.1. Purpose

The described predictive model is intended as a simulator for making decisions. Based on simulations, we can define control costs $c(\cdot)$ for future

time steps $\tau = t + 1, \dots$ that say what an agent should do through a cost-to-go objective

$$\mathbb{E}\left[\sum_{\tau} c(\mathcal{M}, \mathbf{z}_\tau, \mathbf{x}_\tau)\right]. \tag{3.4}$$

Here the expectation is over stochastic predictions like in eq. (3.3), and their explicit dependence on $\mathbf{u}_\tau$ is what enables control optimisation.

The predictive models of this thesis make it easy to specify a variety of control tasks. For example, explicitly uncovering the dense scene geometry in $\mathcal{M}$ enables the definition of collision costs $c(\mathcal{M}, \mathbf{z}_t)$ so that agents can avoid obstacles; revealing the dynamic states $\mathbf{z}_t$ allows for velocity costs $c(\mathbf{z}_t)$ so that agents can emulate different velocity profiles; predicting future images allows for costs $c(\mathbf{x}_t)$ so that agents can navigate to a shown image, and so on. The more the model *reveals* and the more faithfully it *predicts* reality, the easier it is to specify how an agent should act.

3.2. High-Level Attack Angle

The thesis approaches the target distribution $p(\mathbf{x}_{>t}, \mathbf{z}_{>t}, \mathcal{M} \mid \mathbf{u}_{\geqslant t}, \mathcal{D}_{\leqslant t})$ from a Bayesian perspective. Specifically, first the joint model of all past and future variables is formulated (with controls as conditions)

$$p(\mathbf{x}_{1:T}, \mathbf{z}_{1:T}, \mathcal{M} \mid \mathbf{u}_{1:T-1}), \tag{3.5}$$

where the current time t that appears in e. g. $\leqslant t$ is an intermediate time step in $1, \dots, T$. Under the assumed joint, $p(\mathbf{x}_{>t}, \mathbf{z}_{>t}, \mathcal{M} \mid \mathbf{u}_{\geqslant t}, \mathcal{D}_{\leqslant t})$ is then a posterior predictive distribution, noting that $\mathcal{D}_{\leqslant t} = (\mathbf{x}_{\leqslant t}, \mathbf{u}_{<t})$. The following section is about the design choices for eq. (3.5), and will reveal the connection to the targeted posterior predictive towards the end.

3.3. Motivating Design Choices

As previously mentioned in the introduction, the main goal of the research is to combine the spatial modelling typical for dense visual SLAM with the probabilistic state-space fundamentals of model-based control.

To reflect this, the perspectives behind this work are grouped accordingly. Section 3.3.1 is primarily focused on spatial desiderata, and section 3.3.2 on probabilistic principles.

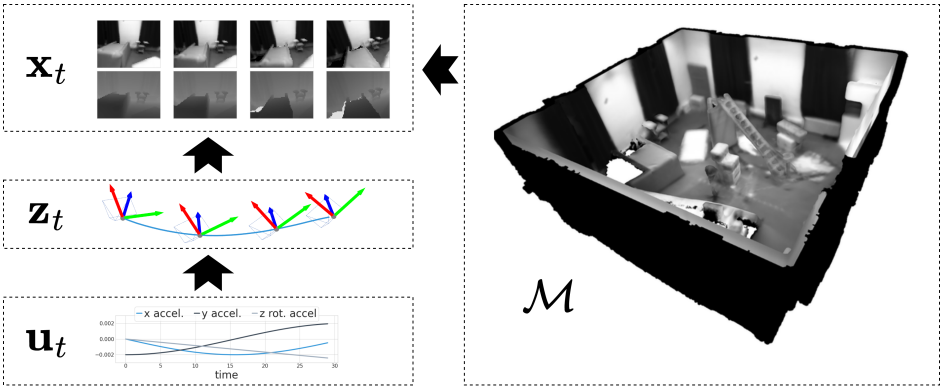


Figure 3.1.: White-box models reveal variables useful for downstream tasks. $\mathbf{x}_t$ are RGB-D images (generated), $\mathbf{z}_t$ are dynamic agent states (latent, including velocity), $\mathcal{M}$ is a 3D volumetric map (latent) and $\mathbf{u}_t$ are acceleration control inputs (conditions).

3.3.1. White-Box Spatial Modelling

The proposed methods rely on spatial domain knowledge, like rendering or kinematics, instead of black-box neural nets. This white-box approach is illustrated in fig. 3.1. Note that the overall model is still a differentiable world model, in the RL sense. Here is why inductive biases are preferred.

Identification First, domain knowledge identifies latent variables with real-world physics (e. g. $\mathrm{SE}(3)$ poses in $\mathbf{z}_t$, 3D occupancy in $\mathcal{M}$, etc.). Control costs $c(\cdot)$ are thus easy to design. Without the 3D map and 6-DoF agent states long-term collision avoidance or spatial exploration would be difficult to solve well. This is specific to spatial agents.

Generalisation Second, domain knowledge means the latents $\mathbf{z}_t$ and $\mathcal{M}$ can be inferred consistently across scenes, unsupervised from image data, because inductive biases like rendering or kinematics generalise. This is particularly important given how much scenes can vary.

On-line inference Third, given the above an agent can be placed in a new scene and controlled without pre-training, solely based on the data observed on-line. This contrasts end-to-end RL, where models are pre-trained either by exhaustively probing the target environment (expensive)

or in man-made simulators (sim-to-real can be an issue).

The price of white-box models is the design complexity and the potential for bias.

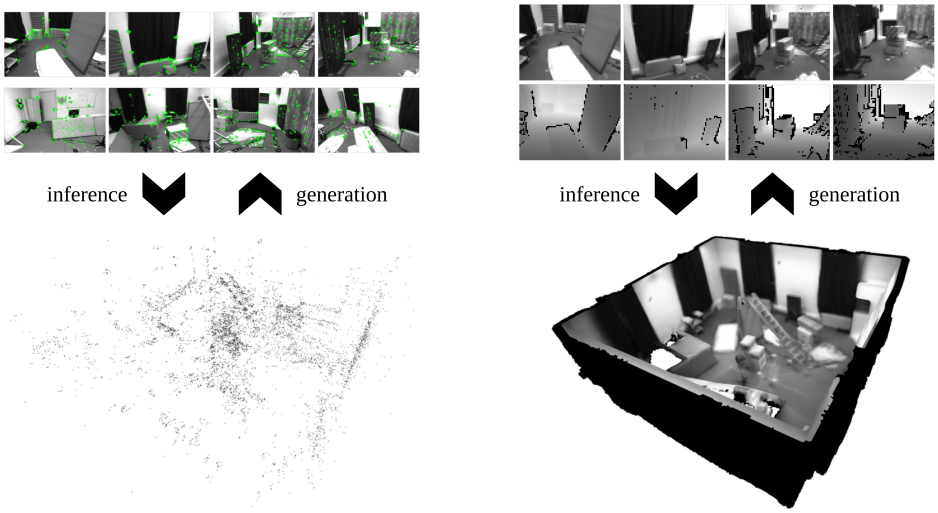
Dense Direct Models

The models in this thesis are *dense* and *direct* (e. g. see Engel (2017)).

Dense A dense model predicts complete images, capturing all observed information in a dense map $\mathfrak{M}$. In this thesis $\mathfrak{M}$ parameterises a dense field over $\mathbb{R}^3$. In contrast, a sparse model captures only a sparse set of world points (or patches) in $\mathfrak{M}$, which are then predicted in the observed images. Even though sparse models are cheaper and can be inferred more accurately on a budget, the focus is on dense models as they are more informative for downstream control. For example, a dense model can generate complete images from novel view points, which can enable image-goal navigation costs $c(\mathbf{x}_t)$. The difference is illustrated in fig. 3.2.

Direct Direct models predict the raw observed images directly. In contrast, indirect models predict different observations distilled from the images, usually a set of sparse 2D coordinates of image features tracked over time (see Cadena et al. (2016)). The purpose of such preprocessing is to establish correspondences between image features and give them as conditions to the model, which simplifies inference. Direct models are chosen over this approach for a) a streamlined end-to-end model, b) to avoid the bias of preprocessing and c) to allow the model to predict smooth, featureless surfaces which are problematic for feature-based methods.

One trade-off here is that direct prediction generally makes latent inference harder. Because image content is highly non-linear (thinking of images as functions over $\mathbb{R}^2$), log-likelihood optimisation terms in inference objectives are also non-linear. Convergence is therefore not guaranteed and requires very good initialisation. Rephrased in probabilistic terms, this means posteriors over $\mathfrak{M}, \mathbf{z}_t$ are highly multi-modal and capturing even one good mode requires a very accurate initial guess.



(a) Sparse generative model.

(b) Dense generative model.

Figure 3.2.: Left: sparse models have a sparse map and only generate sparse image points (in green); the example is from ORB-SLAM3 (Campos et al., 2021). Right: dense models generate full images (here RGB-D) from a dense map; the example is from the core publications.

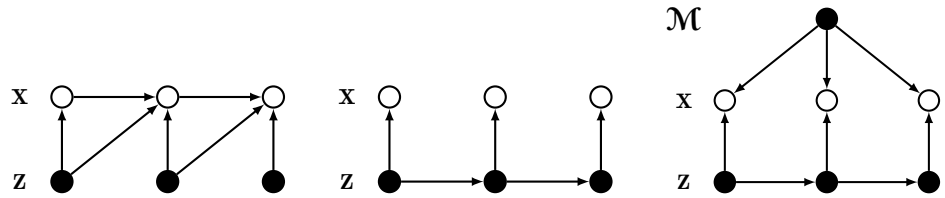
3.3.2. Probabilistic Considerations

The joint distribution $p(\mathbf{x}_{1:T}, \mathbf{z}_{1:T}, \mathfrak{M} \mid \mathbf{u}_{1:T-1})$ is a directed graphical model, which enables straightforward prediction with ancestral sampling. In terms of its design, the main choices are a) to use a *state-space* factorisation and b) to seek compatible *full-posterior* approximations for it.

State-Space Assumptions

Direct SLAM methods often use autoregressive image-to-image connections $\mathbf{x}_{t-1} \to \mathbf{x}_t$ like in fig. 3.3a to project information from one image into another in optimisation objectives (e. g. Steinbrücker et al. (2013) and Engel (2017)). This is an inexpensive and well-established assumption that is well-suited for inference. Still, the thesis deviates from it and prioritises state-space assumptions as in fig. 3.3b and extends them with a map as in fig. 3.3c, here is why.

Markov assumptions As already mentioned in the background, the most convenient foundation for model-based control is a state-space model.



(a) Autoregressive. (b) State-space. (c) Extra static map.

Figure 3.3.: Three examples of directed graphical models relevant for the sequential spatial setting. Observed variables in white, latent variables in black. Control inputs are omitted for brevity, but remain important to this work.

Ideally a spatial world model should comply, for a clean integration in the control loop. The Bayes filter then has a streamlined recursion , which opens the path to POMDP control (background in section 1.5.2). These are the main reasons for an SSM.

Also, previously recoded images $\mathbf{x}_{t-1}$ may come with noise or biases (due to motion blur, reflections, etc.). The latent-to-observation bottleneck of an SSM can be useful in that regard. Translating past images into a latent map $\mathcal{M}$ provides a mechanism to resolve errors and estimate uncertainty before predicting a new image $\mathbf{x}_t$ (see fig. 3.3c).

Two caveats are that the latent-to-observation bottleneck in SSMs comes with approximation errors (e. g. due to limited map resolution) and increases computation (compared to autoregressive connections).

A Persistent Static Map An SSM emission without extra image-to-image connections is possible because of the added map $\mathcal{M}$.[1] The map is a latent random variable over a 3D parameter grid, and needs to be inferred from past data. Map parameters are also *never* marginalised out (marginalisation is common in SLAM and visual odometry, see Leutenegger et al. (2013)). This is done for the sake of retaining maximum information for downstream control, with the caveat of a bigger memory footprint.

For simplicity, the map also does not change over time. Dynamically changing maps are left for future work.

Control-Driven Transitions The SSM *control-driven* transition between consecutive latent states $\mathbf{z}_t, \mathbf{z}_{t+1}$ given a control input $\mathbf{u}_t$ (cf. edges in

[1] The map can be seen as emission parameters.

figs. 3.3b and 3.3c) is important. The dependence of transition rollouts on the control inputs $\mathbf{u}_t$ is what enables control. Across all presented methods, the transition naturally appears both in prediction (as a link in ancestral sampling) and in inference (as a term in the optimisation objectives).

Unity in Fully Probabilistic Prediction and Inference

Recall that previously we established $p(\mathbf{x}_{>t}, \mathbf{z}_{>t}, \mathcal{M} \mid \mathbf{u}_{\geqslant t}, \mathcal{D}_{\leqslant t})$ as a target distribution. When the joint $p(\mathbf{x}_{1:T}, \mathbf{z}_{1:T}, \mathcal{M} \mid \mathbf{u}_{1:T-1})$ is an SSM, this target distribution breaks down into a prediction conditional and an inference conditional:

$$p(\mathbf{x}_{>t}, \mathbf{z}_{>t}, \mathcal{M} \mid \mathbf{u}_{\geqslant t}, \mathcal{D}_{\leqslant t}) = \int \underbrace{p(\mathbf{x}_{>t}, \mathbf{z}_{>t} \mid \mathbf{u}_{\geqslant t}, \mathcal{M}, \mathbf{z}_t)}_{\text{prediction}} \underbrace{p(\mathcal{M}, \mathbf{z}_{\leqslant t} \mid \mathcal{D}_{\leqslant t})}_{\text{inference}} \, \mathrm{d}\mathbf{z}_{\leqslant t}. \tag{3.6}$$

In this regard, the thesis asserts two points.

Unity To satisfy Equation (3.6), $p(\mathcal{M}, \mathbf{z}_{\leqslant t} \mid \mathcal{D}_{\leqslant t})$ should be a posterior distribution derived from the same state-space assumptions that govern $p(\mathbf{z}_{>t}, \mathbf{x}_{>t} \mid \mathbf{u}_{\geqslant t}, \mathcal{M}, \mathbf{z}_t)$ (cf. section 1.3 and Koller and Friedman (2009)). This is worth highlighting because inference of the map and 6-DoF agent states could also be done under a different graphical model (e. g. using an out-of-band SLAM), but then the synergy between the two terms would not be guaranteed.

Complete distributions The aim is to have both $p(\mathbf{z}_{>t}, \mathbf{x}_{>t} \mid \mathbf{u}_{\geqslant t}, \mathcal{M}, \mathbf{z}_t)$ (prediction) and $p(\mathcal{M}, \mathbf{z}_{\leqslant t} \mid \mathcal{D}_{\leqslant t})$ (posterior) as full distributions and not point masses. For this, the thesis explores non-linear inference algorithms that go beyond maximum a-posteriori (MAP) estimation and return a full posterior over the map and agent states. This way uncertainty from the posterior mixes with the generative uncertainty of future predictions, as per eq. (3.6), and the combined uncertainty can be used for decision making.

The caveat of this approach is that full-posterior estimation is more expensive. When added to the cost of dense rendering, this means that some accuracy may be sacrificed due to computational limitations. This is accepted for the sake of uncertainty in a holistic model.

Managing Tractability

The tight integration of all aforementioned choices is computationally more expensive than the alternatives. The cost of dense assumptions is due to the sheer volume of data. The cost of state-space assumptions is in translating past data into a fused map. The cost of full-posterior estimation is generally due to stochastic gradients in optimisation.

The second core publication is dedicated to reconciling all of the above under real-time constraints. Of course, this requires certain compromises in the Bayesian approximation and these are carefully signposted.

Related Work

4

To recap the previous chapters, the thesis seeks predictive spatial models for control. This asks for a (PO)MDP, and in turn a state-space (i. e. Markovian) model with a latent transition and a latent-to-observation emission (background in section 1.4). A preference for a POMDP is also highlighted in the very recent active SLAM survey of Placed et al. (2023).

This puts the work at the intersection of machine learning, robotics and computer vision. The next three sections outline the most related prior work from each field. The chapter then ends with a section about positioning.

4.1. The Machine Learning Perspective

Neural state-space models are proliferous in modern control and deep reinforcement learning. They are seen as *world* models (Ha and Schmidhuber, 2018), because they are *differentiable simulators* of how the observed data (e. g. images) come to be. They have neural net transitions and emissions that have to be learned for the specific use case. The main purpose of such simulators is enabling control, and this work aligns with that objective.

Overview

Prominent examples are Planet (Hafner et al., 2019), Dreamer (Hafner et al., 2020; Hafner et al., 2023) or SLAC (A. X. Lee et al., 2020). Such state-space models have enabled drones learning to fly (Becker-Ehmck et al., 2020), agents playing Atari games (Kaiser et al., 2020) and robotic object manipulation from images (Wu et al., 2022). They are often inferred from data using variational inference (Kingma and Welling, 2014), using the evidence lower bound (ELBO) to optimise generative parameters. Background on this is available in section 1.3.2. Their appeal is their generality – they are formulated without domain knowledge, and used to simulate different types of raw sensory data.

Open Questions

Despite all recent progress, agents moving freely in space with only egocentric observations (e. g. images or inertial) still pose a challenge for neural state-space models. The main problem is that neural nets can struggle with generalisation from images, which change a lot when an agent moves.

To address this, there have been numerous past attempts that mix rudimentary inductive biases in the otherwise neural state space (e. g. Fraccaro et al. (2018), Chaplot et al. (2018), Corneil et al. (2018), Gregor et al. (2019), Mirchev et al. (2019), and Chaplot et al. (2020)). These generally involve some notion of a map, with *abstract* content arranged either in a topological graph or in 2D. However, this strategy still does not scale reliably to free movement in 3D.

4.2. The Robotics Perspective

On the flip side, *engineered* state-space models for spatial agents are well-studied in robotics (e. g. see Thrun et al. (2005)). State-space assumptions were central to SLAM algorithms from the classical era (H. Durrant-Whyte and Bailey, 2006) and are still used in robotics today. Early methods of that type relied on Kalman filtering for state estimation (Kalman, 1960). In such models transitions are defined through known dynamics equations, and similarly emissions are defined through geometry. Engineering usually implies more complex design and possibly modelling bias, but also improved generalisation.

Overview

Seminal works of the late 1980s, such as those by e. g. Chatila and Laumond (1985), R. C. Smith and Cheeseman (1986), H. F. Durrant-Whyte (1988), Ayache and Faugeras (1988) and Crowley (1989), paved the way towards Markovian assumptions in the context of EKF estimation (R. Smith et al., 1990). Since then, state-space assumptions have prevailed in many spatial filters, be it EKFs (e. g. Thrun et al. (2002), Davison et al. (2007), Anastasios I. Mourikis and Roumeliotis (2007), M. Li and Anastasios I Mourikis (2013), Liu et al. (2020), and Yang et al. (2022)) or Rao-Blackwellised particle filters (e. g. Murphy (1999), Thrun et al. (2000), Montemerlo et al. (2002), Grisetti et al. (2007), and Q. Huang and Leonard (2023)). To not mischaracterise the robotics field, it should be noted that the aforementioned assumptions are not used solely in EKFs and RBPFs, but also in more general factor graphs

for MAP smoothing (e. g. see Cadena et al. (2016) and Dellaert and Kaess (2017)), and in modern days SLAM in robotics and SLAM in computer vision are intertwined (see next section).

Open Questions

The aforementioned SLAM state-space models comply with the POMDP framework (Thrun et al., 2005). However, one limitation is that most such models do not predict dense RGB(-D) images directly. Instead, often the emission predicts a low-dimensional, often preprocessed version of the raw 3D data.

For example, in the simplest case a model can assume observations are 6-DoF poses obtained out-of-band, e. g. by running a visual odometry algorithm on the side. The latent model states then also contain a 6-DoF pose, and the observations are seen as a noisy version of the latents that needs to be filtered. This forms a *pose graph*, with no explicit 3D map.

Another common alternative is that observations are 2D xy-coordinates of tracked image features[1]. For that, the latent states $\mathbf{z}_t$ are generally a combination of the latest agent 6-DoF state and 3D world points which make a sparse map. The association between these 3D points and the 2D image features is normally determined a-priori, by tracking salient image points in the video stream over time (e. g. see SIFT (Lowe, 1999), SURF (Bay et al., 2006), FAST (Rosten et al., 2010) or ORB (Rublee et al., 2011)).

Both of the aforementioned observation variants require preprocessing of the raw image data, which a state-space model (and associated inference) would not be aware of. The reason for preprocessed observations is efficiency, often motivated by the scalability properties of EKFs (dense quadratic covariances), PFs (many particles) or factor-graph inference (number of optimised variables).

In comparison, state-space models with direct RGB(-D) observations are much rarer. One example is the early EKF work of Pietzsch (2008), where the map is limited to a set of planes. There is also a line of dense SLAM methods that use forward kinematics loss terms (Wagner et al., 2014; Klingensmith et al., 2016; Scona et al., 2017), very similar to an SSM transition. The focus of these latter approaches is on MAP estimation, whereas this thesis also highlights fully-Bayesian inference and the link to model-based control. Also, 3D information-theoretic active mapping approaches (e. g. Charrow et al. (2015), Zhang et al. (2020), and Asgharivaskasi et al. (2022)), which

[1] This assumption was the result of cross-pollination between the fields of robotics and computer vision in the early days of SLAM.

are spiritual successors of early 2D methods (Stachniss, 2009), often use probabilistic maps, RGB-D images and SSM assumptions for planning, but not for SLAM inference.

On the whole, state-space models that generate complete images, and are tightly coupled with inference, deserve further attention.

4.3. The Computer Vision Perspective

The computer vision subfield of 3D structure and motion (Hartley and Zisserman, 2006) is also very relevant to this thesis. Below we will discuss two general directions: *indirect* methods, which predict tracked 2D features, and *direct* methods, which predict raw image data directly.

The work of Lucas and Kanade (1981) and Tomasi and Kanade (1991) has been very influential for both indirect and direct approaches. Its core idea is to seek alignment of the content of two images, i. e. to require *photometric consistency* between them. In direct methods, this can be used to find a pose transformation between two views. In indirect methods, it can be used to track image features by minimising the photometric consistency error of pixel patches around them.[2]

Indirect Methods

Indirect methods have their roots in *bundle adjustment* (Brown, 1976), which aligns a bundle of rays from multiple cameras. They are classified as indirect because they model the geometry of rays and associated image coordinates, not the image content directly. This requires that the correspondence between rays is known, which is given by tracking 2D image features associated with each ray. Note that the same assumptions were mentioned in the previous section on SLAM in robotics, due to cross-pollination between the two fields.

In modern SLAM and visual odometry, bundle adjustment is seen as a special case of maximum a-posteriori (MAP) optimisation (Cadena et al., 2016) in a graphical model that predicts 2D image coordinates from latent camera poses and 3D map points. MAP has overtaken filtering approaches in the realm of sparse methods, due to seminal works like PTAM (Klein and Murray, 2007) and later Strasdat et al. (2012) empirically demonstrating that it is a more accurate solution on a small computational budget. The

[2]Kanade-Lucas-Tomasi (KLT) trackers were later superseded by searching for global descriptors like ORB (Rublee et al., 2011), but the idea remains influential.

result is a plethora of indirect large-scale SLAM systems, like ORB-SLAM (Mur-Artal et al., 2015; Campos et al., 2021), or VINS-MONO (Qin et al., 2018), or the pose optimisation back end of Kimera (Rosinol et al., 2020).

Direct Methods

On the flip side, the early Lukas-Kanade image alignment has inspired direct formulations as well. Most such approaches reproject (or warp) the content from one image into another based on latent camera poses and latent depth (or 3D points), and then optimise for photometric consistency. Progress by Audras et al. (2011) and Steinbrücker et al. (2011) has shown this as a reliable method for odometry estimation. This has led to a number of stable sparse semi-direct and direct methods, such as SVO (Forster et al., 2014), LSD-SLAM (Engel et al., 2014) and DSO (Engel et al., 2018), in which the direct assumptions are framed as MAP optimisation.

In parallel, direct assumptions are also dominant in dense RGB-D registration, a good example of which is the RGB-D SLAM method of Steinbrücker et al. (2013). Notably, the direct alignment of depth images is very close to iterative closest point (ICP) methods, as introduced by Chen and Medioni (1992). Direct depth-only or RGB-D alignment is also used in KinectFusion (Newcombe et al., 2011a), VoxelHashing (Nießner et al., 2013) and ElasticFusion (Whelan et al., 2015).

In modern days, there is also the emerging paradigm of mapping via differentiable rendering (Lombardi et al., 2019; Mildenhall et al., 2020), in which dense neural maps are learned by gradient descent through a renderer that predicts images directly. SLAM systems based on this approach have started to emerge only recently, for example iMAP (Sucar et al., 2021), NICE-SLAM & NICER-SLAM (Zhu et al., 2022; Zhu et al., 2023) and NERF-SLAM (Rosinol et al., 2022).

Open Questions

This thesis subscribes to *dense direct* models. Indirect models are efficient and can be accurate, but they retain less information (tracked features are normally sparse), they can be sensitive to errors in feature tracking (cannot be corrected easily) and they can struggle with images with smooth gradients (few features to track) (Cadena et al., 2016; Engel, 2017).

Next, recall that the thesis specifically seeks spatial *state-space* models for control, i. e. with Markovian assumptions. There are two aspects not covered by prior art in this regard.

First, to the best of the author's knowledge mainstream dense SLAM methods usually do not spell out SSM predictive and posterior distributions, even if their assumptions are not far away from the SSM independences (e. g. BAD-SLAM (Schöps et al., 2019)). Sometimes this is simply because a control-driven transition is missing, but it could also be due to incompatible autoregressive connections when images are aligned frame-to-frame and not frame-to-model (e. g. see section 2.3.1 of Zollhöfer et al. (2018)). A re-examination of dense methods from the SSM perspective is thus desirable, especially for emergent paradigms like differentiable rendering.

Second, the de-facto inference method in nearly all modern visual odometry and SLAM is maximum a-posteriori optimisation, which returns point-estimates that can be extended with Gaussian uncertainty via a Laplace approximation (Laplace, 1986). MAP is currently necessary when efficiency is a top priority. But from a research standpoint, it also leaves room for exploring alternative inference methods that would return a full posterior directly.

4.4. Positioning

In summary, the thesis seeks to unify concepts from all three fields and explore some of the aforementioned research gaps.

The developed spatial model predicts raw RGB-D observations, just like the discussed world models from machine learning predict raw data. It is end-to-end differentiable and avoids observation preprocessing. Variational inference from the field of ML is also explored as a relaxed alternative to MAP smoothing in computer vision and robotics.

At the same time, generalisation is addressed by incorporating domain knowledge, like in all listed examples from robotics and computer vision. The developed model features a dense map and a renderer, which puts it close to the aforementioned dense direct methods.

Finally, the proposed model has a well-defined state space and can be easily extended into a POMDP. Generative distributions and posteriors are made explicit. On the whole, this adds one more perspective to the existing body of work on dense visual odometry and SLAM.

The main price for these advances are sacrifices in terms of efficiency. Because of the increased computational footprint, the developed methods are applicable in moderate-scale indoor scenes. This also implies a potential drop in accuracy on a tight inference budget, which is accepted for the sake of more informative predictions for control.

Methods and Findings

5

This chapter summarises the main outcomes of the research. The discussion remains from a bird's-eye perspective, highlighting only core methods and takeaways. Each section lists relevant background from part I and one relevant core paper from part III as prerequisites.

The core publications are:

1. *VSSM-LM* (Mirchev et al., 2021), for narrative purposes it is split between section 5.1 (the SSM), section 5.2 (smoothing), section 5.4 (dynamics learning).
2. *PRISM* (Mirchev et al., 2022), in section 5.3 (real-time filtering).
3. *TNP-SM* (Kayalibay et al., 2022)[1], in section 5.5 (navigation).

In addition, section 5.6 has unpublished, very recent work on further control with VSSM-LM. The overall research contributions are listed at the end of the chapter, after all method sections.

The actual papers are attached in appendices A.1 to A.3. The personal involvement of the author is explained in the beginning of each paper appendix.

Appendix B.1 also holds a *non-core prior-work* publication with results from the author's master's thesis, which sparked the research that followed. It is included only for context, as it manifests similar ideas, albeit in a rather naive implementation. It does not count towards the dissertation.

[1] Shared lead authorship.

5.1. Dense Spatial State-Space Models

This section is about the spatial SSM, VSSM-LM for short (paper in appendix A.1), which is central to all core publications. It requires background on state-space models (section 1.4) and a basic understanding of rigid-body poses (section 2.1) and rendering (section 2.2).

VSSM-LM is the state-space model proposed in this thesis, it is

$$\begin{aligned} &p_\theta(\mathbf{x}_{1:T}, \mathbf{z}_{1:T}, \mathcal{M} \mid \mathbf{u}_{1:T-1}) = \\ &\quad p_\theta(\mathcal{M}) p_\theta(\mathbf{z}_1) \underbrace{\prod_{t=2}^{T} p_\theta(\mathbf{z}_t \mid \mathbf{z}_{t-1}, \mathbf{u}_{t-1})}_{\text{dynamics}} \underbrace{\prod_{t=1}^{T} p_\theta(\mathbf{x}_t \mid \mathbf{z}_t, \mathcal{M})}_{\text{rendering}} . \end{aligned} \tag{5.1}$$

Given controls $\mathbf{u}_{1:T-1}$, the model simulates dynamic agent states $\mathbf{z}_{1:T}$, and with them generates RGB-D images $\mathbf{x}_{1:T}$ from a 3D map $\mathcal{M}$. For this, the factors $p_\theta(\mathcal{M})$, $p_\theta(\mathbf{x}_t \mid \mathbf{z}_t, \mathcal{M})$ and $p_\theta(\mathbf{z}_t \mid \mathbf{z}_{t-1}, \mathbf{u}_{t-1})$ are implemented with domain knowledge.

The Map $p_\theta(\mathcal{M})$ is a Gaussian distribution that factorises over a 3D grid. Each cell holds a signed-distance (SDF, can be seen as occupancy) and a colour value. The grid serves as the Bayesian parameters for a continuous 3D field $f_\mathcal{M} : \mathbb{R}^3 \to \mathbb{R}^4$, obtained by interpolating the grid voxels. The zero-level set of the SDF field implicitly defines surfaces.

The Emission $p_\theta(\mathbf{x}_t \mid \mathbf{z}_t, \mathcal{M})$ is a *differentiable* RGB-D renderer. It assumes a known camera intrinsic matrix $\mathbf{K}$ (part of θ) and predicts RGB-D images by volumetric rendering. Details about its differentiability are in the paper. Epistemic map uncertainty flows through it and into the generated images.

The Transition $p_\theta(\mathbf{z}_t \mid \mathbf{z}_{t-1}, \mathbf{u}_{t-1})$ predicts the 6-DoF dynamics of the agent. VSSM-LM offers two versions for it, one with engineered Euler kinematics and one learned from data (discussed later). Respectively, the latents $\mathbf{z}_t$ hold an $\mathrm{SE}(3)$ pose, velocity and potentially other information the transition needs.

VSSM-LM is a differentiable SSM, a simulator for model-based control, and thus a *world model* (Ha and Schmidhuber, 2018). With an added cost function, it becomes a POMDP. It is thus a bridge between SSMs for model-based control, like the ones in most classical SLAMs (H. Durrant-Whyte

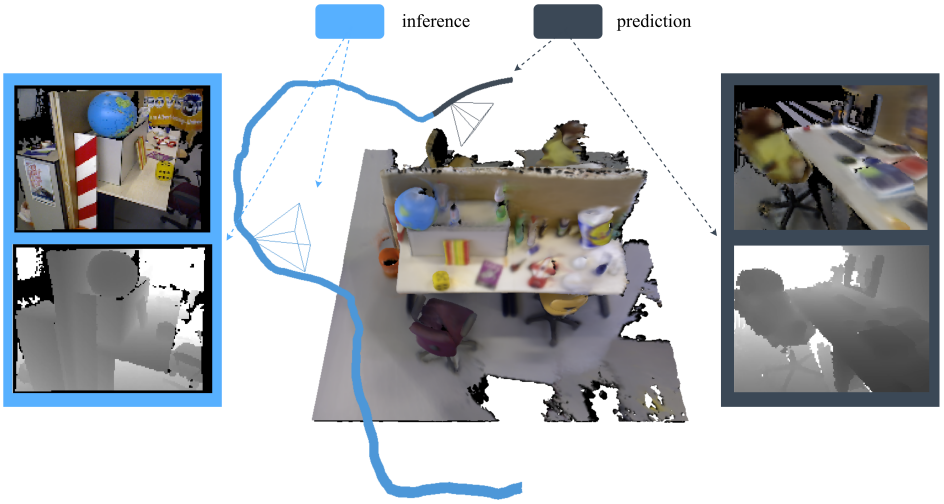


Figure 5.1.: The panel shows the mean of an inferred trajectory (**blue**) and an inferred map (center) from past RGB-D observations (left) and controls. Then, a predictive rollout (**dark gray**) is produced into the future for hypothetical controls, again showing the mean prediction. The mode of a rendered RGB-D prediction, a novel view from the rollout, is shown on the right. Data is from the TUM-RGBD data set (Sturm et al., 2012).

and Bailey, 2006), and the assumptions of dense SLAM (e. g. Newcombe et al. (2011a), Steinbrücker et al. (2013), and Schöps et al. (2019)).

Defining an SSM is possible because of differentiable rendering. It makes it possible to use Markovian emissions, as opposed to $\mathbf{x}_{t-1} \to \mathbf{x}_t$ predictions, and it makes it possible to generate raw data. It also makes the whole joint of VSSM-LM end-to-end differentiable. This last point matters for inference and for optimising $\mathbf{u}_t$ through the model. The thesis sticks to volumetric rendering, but note that the above would hold for other renderers too (e. g. the surfel-splatter of BAD-SLAM (Schöps et al., 2019) would work).

This is the big picture, for the rest of this section we will zoom in on details about how VSSM-LM works.

5.1.1. Control-Driven Probabilistic Predictions

VSSM-LM focuses on prediction for control. An example of latent inference of $\mathfrak{M}, \mathbf{z}_{\leqslant t}$ followed by a predictive rollout is given in figure fig. 5.1.

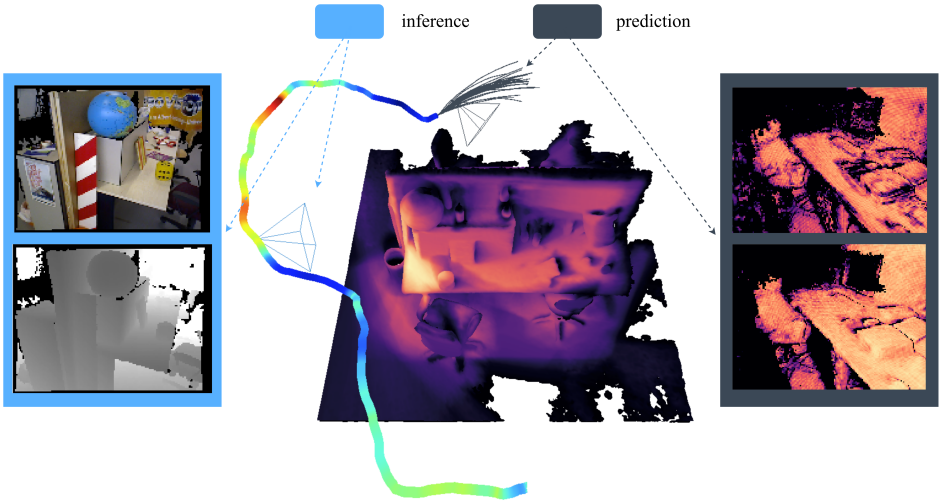
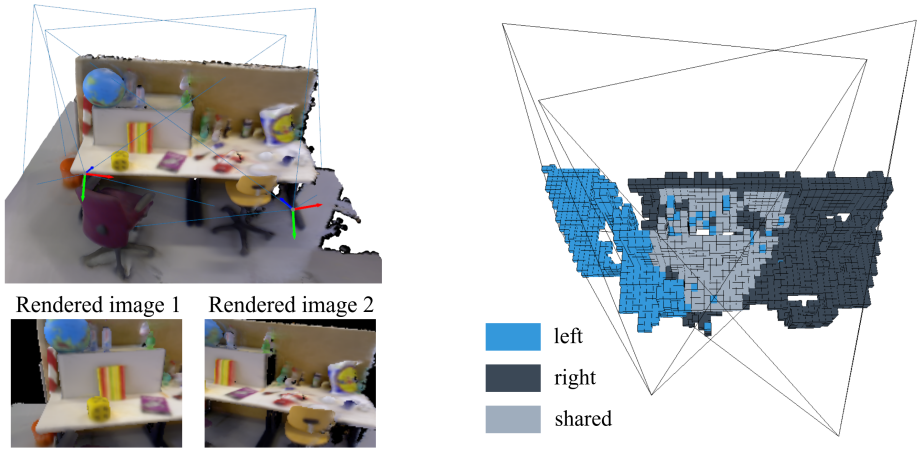


Figure 5.2.: Same panel as the previous, but now showing the uncertainty in all variables. For the map and the predicted RGB-D images on the right, **orange** means the estimate is with higher certainty. For the inferred trajectory (jet colorscheme) **blue** indicates certainty. The predicted segment is again in **grey**, but this time multiple sampled rollouts are shown.

The transition $p_\theta(\mathbf{z}_t \mid \mathbf{z}_{t-1}, \mathbf{u}_{t-1})$ is necessary for prediction, as it causally links the controls $\mathbf{u}_{t-1}$ to the future rollouts $\mathbf{z}_{t:T}, \mathbf{x}_{t:T}$. Related to this, VSSM-LM specifically targets autonomous agents, where control inputs are available, and less so hand-held cameras. Also, note that controlled transitions are orthogonal to maximising the likelihood of inertial measurements in visual-inertial odometry (e. g. VINS Qin et al. (2019)), and the two can complement each other.

VSSM-LM's predictions come with uncertainty. This is shown in fig. 5.2, noting that all variables are random. The factors $p_\theta(\mathcal{M})$, $p_\theta(\mathbf{x}_t \mid \mathbf{z}_t, \mathcal{M})$ and $p_\theta(\mathbf{z}_t \mid \mathbf{z}_{t-1}, \mathbf{u}_{t-1})$ are all parametric distributions with closed-form PDFs. MC sampling and estimation of expectations $\mathbb{E}_p[f(\mathbf{z}_{1:T}, \mathbf{x}_{1:T}, \mathcal{M})]$ are thus straightforward. Of course, the speed caveats of MC apply.

Uncertainty matters for three reasons. First, it is necessary for any sort of marginalisation or filtering (we will later see this is needed for real-time inference). Second, it can inform POMDP control and make it robust to estimation errors (out of scope for the thesis, see background in section 1.5.2). Third, it enables information-theoretic active inference, e. g. for map exploration (will be demonstrated for VSSM-LM later).



(a) Rendering in 3D.

(b) Parameter sharing.

Figure 5.3.: Map parameters are automatically shared when rendering. Left: rendering from two overlapping viewpoints. Right: the grid parameters in $\mathcal{M}$ are **blue** when only used for the left view, **light gray** when shared for both views and **dark gray** when only used for the right view.

5.1.2. Dense Rendering

Next we will turn our attention to rendering again, as it has a few more interesting properties.

Rendering generates data densely, which makes VSSM-LM more informative for downstream tasks. For example, the dense map $\mathcal{M}$ will be useful when reasoning about collisions in navigation. Also, rendering generalises well, because of geometry (compared to neural nets).

A more intricate point is that rendering implicitly associates map parameters to observed images, based on the projective camera frustums. This is shown in fig. 5.3, noting that when frustums overlap parameters are shared. The map $\mathcal{M}$ is therefore non-redundant, and grows with the size of the scene and not the amount of observed data. This implicit parameter association and sharing allows us *not to marginalise map parameters* out of the system as new data keeps coming in, which is a choice that contrasts many visual odometry methods. For downstream planning, it is desirable that the whole map remains. The price for this is memory.

At the same time, rendering brings in some more controversial points. For example, rendering implies that in inference new images $\mathbf{x}_t$ have to align with the existing map $\mathcal{M}$, not previous observations $\mathbf{x}_{t-1}$ like in many

alternative SLAM methods. This can be beneficial when the observed RGB-D data is noisy, but it can also lead to localisation inaccuracy if the map estimate is biased. Two further caveats are that rendering is expensive and that the rendering emission $p(\mathbf{x}_t \mid \mathfrak{M}, \mathbf{z}_t)$ is highly non-linear in $\mathbf{z}_t$, which makes optimisation through it very dependent on good initialisation.

5.1.3. A Note on the Markov Assumptions

Since Markovian assumptions are actively targeted by VSSM-LM, it should be noted that when $\mathfrak{M}$ is marginalised out, as in

$$p(\mathbf{z}_{1:T}, \mathbf{x}_{1:T} \mid \mathbf{u}_{1:T-1}) = \int p(\mathbf{z}_{1:T}, \mathbf{x}_{1:T}, \mathfrak{M} \mid \mathbf{u}_{1:T-1}) \, \mathrm{d}\mathfrak{M}, \tag{5.2}$$

the remaining marginal over $\mathbf{z}_{1:T}, \mathbf{x}_{1:T}$ is not Markovian. The observations become correlated.

However, given a map sample $\mathfrak{M}$, the conditional $p(\mathbf{z}_{1:T}, \mathbf{x}_{1:T} \mid \mathfrak{M}, \mathbf{u}_{1:T-1})$ is again a vanilla state-space model. This means that in a control context, VSSM-LM asks for evaluating a cost-to-go (see section 1.5) in expectation over the map, i. e.

$$\mathbb{E}_{\mathfrak{M} \sim p(\mathfrak{M})} \left[\sum_{\tau} c_\tau \right], \tag{5.3}$$

where the costs c_τ are over SSM rollouts. In practice one can always take the map mean on a budget, or use an MC-estimate. Generally, MC-estimates w. r. t. the map distribution can vary in cost, and are most expensive if they have to render inside the expectation. Note that the individual costs $c(\cdot)$ are free to peek directly at the respective $\mathfrak{M}$ sample (e. g. for collision avoidance), since it is given as a condition inside the expectation in eq. (5.3).

5.2. Smoothing

> *This section is about the smoothing inference method introduced with VSSM-LM (appendix A.1). It requires background on variational inference (section 1.3.1) and state-space models (section 1.4), and a basic understanding of rendering (section 2.2).*

Consider using VSSM-LM to inform the decisions of a moving agent. To do so, first we need to infer a belief over the map (i. e. mapping) and the current agent state (i. e. localisation) from past observed data $\mathcal{D}_{\leqslant t} = (\mathbf{x}_{\leqslant t}, \mathbf{u}_{<t})$, so

that predictive rollouts into the future make sense. Such inference aligns with SLAM. The thesis explores two approaches for this

- a variational smoother (this section);
- and a real-time filter (next section).

The research gap addressed here is fully-Bayesian posterior inference under dense rendering, derived from the generative factorisation of VSSM-LM. The aim is to minimise additional generative assumptions, in order to keep inference aligned with prediction.

5.2.1. SMOOTHING VIA VARIATIONAL INFERENCE

The smoothing posterior $p(\mathbf{z}_{\leqslant t}, \mathcal{M} \mid \mathbf{x}_{\leqslant t}, \mathbf{u}_{<t})$ can correct the past, as old states are inferred from all data until the current time. It is the biggest posterior one can phrase up to time t, revealing all latents at once.

The paper in appendix A.1 approximates $p(\mathbf{z}_{\leqslant t}, \mathcal{M} \mid \mathbf{x}_{\leqslant t}, \mathbf{u}_{<t})$ through *variational inference* (VI). An approximate distribution $q_{\boldsymbol{\phi}}(\mathbf{z}_{\leqslant t}, \mathcal{M})$ with parameters $\boldsymbol{\phi}$ is fit to it by minimising the $\mathrm{KL}(q \parallel p)$ divergence:

$$q_{\boldsymbol{\phi}^*}(\mathbf{z}_{\leqslant t}, \mathcal{M}) \approx p(\mathbf{z}_{\leqslant t}, \mathcal{M} \mid \mathbf{x}_{\leqslant t}, \mathbf{u}_{<t}) \tag{5.4}$$

$$\boldsymbol{\phi}^* \approx \arg\min_{\boldsymbol{\phi}} \mathrm{KL}\big(q_{\boldsymbol{\phi}}(\mathbf{z}_{\leqslant t}, \mathcal{M}) \,\big|\big|\, p(\mathbf{z}_{\leqslant t}, \mathcal{M} \mid \mathbf{x}_{\leqslant t}, \mathbf{u}_{<t})\big). \tag{5.5}$$

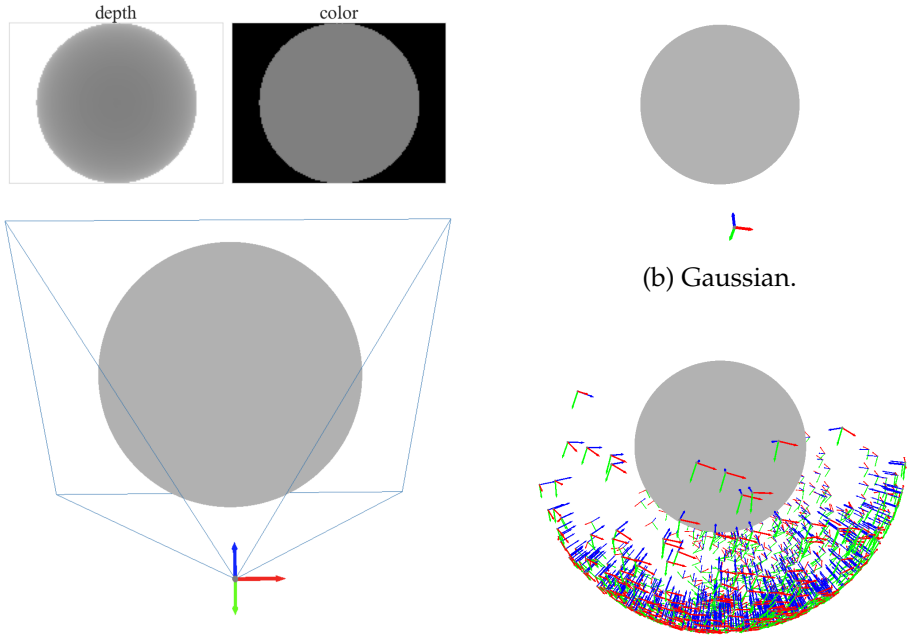
The approximation q is non-amortised, obtained via stochastic optimisation.[2] Please consult the included paper for the detailed derivation. The background on VI is in sections 1.3.1 and 1.4.2.

5.2.2. FLEXIBILITY

In modern SLAM and odometry estimation the de-facto standard is MAP optimisation via Gauss-Newton, optionally followed by a Laplace approximation (explained in detail in appendix C.2, also see Cadena et al. (2016)). This approach is restricted to either point-mass estimates or a joint Gaussian over $\mathbf{z}_{\leqslant t}, \mathcal{M}$ obtained under linearisation of the whole predictive graph (i. e. here VSSM-LM).

In contrast, the VI approximation $q_{\boldsymbol{\phi}}(\mathbf{z}_{\leqslant t}, \mathcal{M})$ is a full distribution by design. It is free to factorise into complex non-linear, potentially non-Gaussian terms. A motivating example is given in fig. 5.4.

[2] For better generalisation across new scenes, q is not predicted by a neural network like in a variational auto-encoder (Kingma and Welling, 2014).



(a) Rendering with perceptual aliasing.

(b) Gaussian.

(c) Normalising flow.

Figure 5.4.: In this simplified setup, an agent is uncertain about the perspective from which it observes a ball (*perceptual aliasing*). A Gaussian posterior can only provide a uni-modal estimate of the agent pose in space. A variational inference posterior (here a normalising flow) can distribute uncertainty more accurately over the manifold of possible poses, in this case a sphere wrapping around the observed ball.

The thesis demonstrates the first successful application of VI for dense SLAM smoothing. While in the experiments $q_\phi(\mathbf{z}_{\leqslant t}, \mathcal{M})$ is a simple mean-field[3] approximation (product of Gaussians), it is a proof of concept that should open the path for more intricate approximate posteriors in the future, without further algorithmic changes. This is the main reason why VI was explored.

Variational inference was chosen over other relaxed methods, such as *Markov-chain Monte Carlo* or *sequential importance sampling* (SIS) because of scalability (the dense map has millions of parameters).

[3]This means $q_\phi(\mathcal{M}, \mathbf{z}_{\leqslant t}) = q_\phi(\mathcal{M}) \prod_{\tau=1}^{t} q_\phi(\mathbf{z}_\tau)$. The terms $q_\phi(\mathbf{z}_\tau)$ and $q_\phi(\mathcal{M})$ also break down over independent dimensions. This posterior does not capture correlations.

5.2.3. About Optimisation

VI methods require stochastic optimisation. This is because the KL in eq. (5.5) needs to be estimated via Monte Carlo. One implication is that second-order optimisers, like Gauss-Newton, cannot be applied directly because any Hessian estimate would not be stable over optimisation steps. Because of this the smoother relies on Adam (Kingma and Ba, 2015), a variant of first-order SGD optimisation.

Also, and this is detailed in the paper, the objective in eq. (5.5) naturally breaks down into terms that involve the emission and the transition of VSSM-LM. This is where differentiable rendering comes into play– gradients through the emission are necessary for optimisation. As already mentioned, the non-linearity of $p(\mathbf{x}_t \mid \mathcal{M}, \mathbf{z}_t)$ also leads to local minima, and optimisation depends on good initialisation.

5.2.4. Limitations

Experimentally it was found that variational smoothing is limited by its runtime; on modern commodity hardware it is circa 10-15 times slower than real-time, even when approximation accuracy is traded for a smaller budget. This is because the optimisation objective includes a sum of rendering reconstruction terms over the whole history $1, \ldots, t$

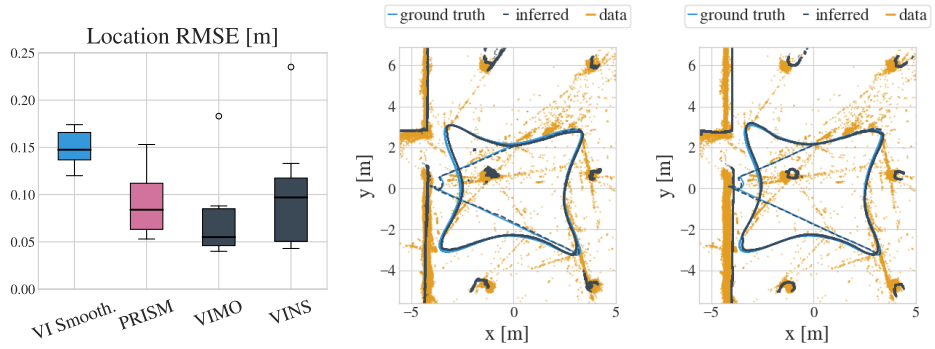
$$\mathbb{E}_q \left[\sum_{\tau=1}^{t} -\log \underbrace{p(\mathbf{x}_\tau \mid \mathbf{z}_\tau, \mathcal{M})}_{\text{renderer}} \right]. \tag{5.6}$$

Computing even a subset of these repeatedly is expensive.[4] Moreover, they are in expectation over q, which leads to either increased gradient variance or the need for repeated evaluations for a tighter Monte-Carlo estimate. Unsurprisingly, this means that the posterior approximation is suboptimal on a reasonable budget.

As a side note, it is understood that odometry estimation is efficient and stable with only sparse data (e. g. Engel (2017)), and this could be used to initialise the pose estimates and improve the runtime. This remained out of scope as the publication only addressed what can be derived directly from the streamlined VSSM-LM assumptions.

Despite the computational challenges, dense variational smoothing works consistently for flight trajectories of up to 4 m/s, lagging slightly

[4] In practice, gradients are evaluated for random subsets of pixels in each step.



(a) Localisation summary. (b) Smoother, 3 m/s. (c) PRISM, 3 m/s.

Figure 5.5.: The VI smoother and PRISM have similar accuracy to the SotA. (a) Localisation comparison to two SotA visual odometry methods, VIMO (Nisar et al., 2019) and VINS-MONO (Qin et al., 2018), results are on Blackbird data (Antonini et al., 2020) based on the core publications. (b) A run of the VI smoother. (c) A run of PRISM. Both are top-down views. Inferred map and trajectory are in black, true trajectory is in blue. Both filter out the noise in the observed depth data (in orange).

behind the SotA. The paper has the experiment details, a glimpse at the results is given in fig. 5.5. On-line use may become viable as hardware improves, until then the smoother is most useful for optimising generative parameters through the evidence lower bound, such as camera intrinsics or transition parameters, as per the background in section 1.3.2. These are usually off-line tasks done once per robot.

5.3. Real-Time Filtering

> *This section is about the real-time filtering method introduced with PRISM (appendix A.2). It requires background on closed-form Gaussian updates (section 1.2.2), inference methods like MAP and Laplace (section 1.3), filtering in state-space models (section 1.4), and a basic understanding of rendering (section 2.2).*

The off-line variational smoother leaves room for a real-time solution. The main challenge is how to work around the cost of rendering in VSSM-LM. PRISM gives one solution to this problem.

5.3.1. Divide and Conquer

PRISM scales to real-time by approximating the two marginal filters[5]

$$q_t^{\Phi}(\mathbf{z}_t) \approx p(\mathbf{z}_t \mid \mathbf{x}_{\leqslant t}, \mathbf{u}_{<t}) \tag{5.7}$$

$$q_t^{\Phi}(\mathcal{M}) \approx p(\mathcal{M} \mid \mathbf{x}_{\leqslant t}, \mathbf{u}_{<t}). \tag{5.8}$$

Specifically, they are both derived from the recursion of the Bayes filter

$$p(\mathcal{M}, \mathbf{z}_t \mid \mathbf{x}_{\leqslant t}, \mathbf{u}_{<t}) \propto p(\mathbf{x}_t \mid \mathbf{z}_t, \mathcal{M}) \int p(\mathbf{z}_t \mid \mathbf{z}_{t-1}, \mathbf{u}_{t-1}) \cdot \underbrace{p(\mathcal{M}, \mathbf{z}_{t-1} \mid \mathbf{x}_{\leqslant t-1}, \mathbf{u}_{<t-1})}_{\text{prev. filter}} \, \mathrm{d}\mathbf{z}_{t-1}. \tag{5.9}$$

This recursion is possible because of the SSM assumptions of VSSM-LM.

Filtering is motivated by the high cost of rendering. The main insight is that recursive updates can be solved one at a time (i. e. divide and conquer), and each recursive step can evaluate the renderer only *once*. The problem is smaller than smoothing, as it estimates only the last state and map.

In essence, this is a marginalisation strategy similar to what is done in visual odometry and SLAM (e. g. Leutenegger et al. (2013)) to bound computation. But whereas in traditional methods marginalisation is based on sparsity and linearisation of the complete system, here the filters are derived from the rendering and 6-DoF dynamics assumptions of VSSM-LM.

On the whole, this strategy leads to a Bayesian posterior that is similar in accuracy to the preceding VI smoother and the current visual SotA in indoor environments, on trajectories of up to 4 m/s. This is summarised in fig. 5.5, the detailed results are in the paper. At the same time, it runs in real-time and complies with the dense rendering of VSSM-LM.

5.3.2. Approximations and Compromises

The path to real-time is in the details of the Bayesian approximation, which is rather complex. The remaining discussion will benefit from familiarity with the derivations of PRISM in appendix A.2, it is recommended to read these first. Following are only the final outcomes and their implications.

[5] The joint filter $p(\mathbf{z}_t, \mathcal{M} \mid \mathbf{x}_{\leqslant t}, \mathbf{u}_{<t})$ was not targeted because capturing the map-state correlations is expensive for the dense 3D map.

State filter PRISM updates the state filter $q_t^{\Phi}(\mathbf{z}_t)$ in three stages.

- An $\mathrm{SE}(3)$ pose is found by MAP gradient-based optimisation.
- The pose is extended with uncertainty via a Laplace approximation.
- Velocity, with uncertainty, completes the state estimate.

These steps naturally involve the emission and transition of VSSM-LM. Both appear as terms in the MAP optimisation which looks like this

$$\arg\max_{\mathbf{z}_t^{\text{pose}}} \log p\left(\mathbf{x}_t \mid \mathbf{z}_t, \hat{\mathcal{M}}\right) + \text{dynamics prior}, \tag{5.10}$$

where the dynamics prior is obtained by linearising the transition and propagating uncertainty through it in closed form via LGS equations (background in section 1.2.2).

The main speed-up is from replacing any optimisation through the rendering term, i. e. $\log p\left(\mathbf{x}_t \mid \mathbf{z}_t, \hat{\mathcal{M}}\right)$, with a surrogate image-alignment objective that only renders *once*, which introduces approximation bias. This surrogate is a version of the standard photometric consistency (e. g. Audras et al. (2011) and Steinbrücker et al. (2011)) and point-to-plane ICP (Chen and Medioni, 1992) equations in direct odometry methods. It is one piece of the state filter and will be detailed in the next section, as it emerged in the context of the another core publication (TNP-SM).

The other compromise for speed is the aforementioned Laplace approximation, which limits uncertainty quality due to its implicit linear Gaussian assumption. The MAP and Laplace combination can be replaced by variational inference (see appendix C.6 if interested), but this was not explored.

The overall state update is depicted in fig. 5.6.

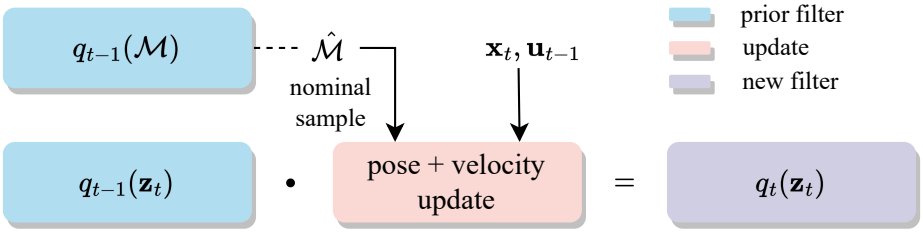


Figure 5.6.: State filter diagram. $\hat{\mathcal{M}}$ is from the previous map filter.

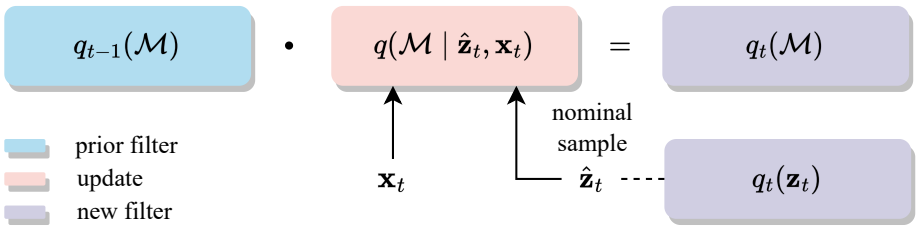


Figure 5.7.: Map filter diagram. $\hat{\mathbf{z}}_t$ is from the new state filter.[6]

Map filter PRISM updates the map filter $q_t^{\Phi}(\mathcal{M})$ in closed-form, via

$$q_t^{\Phi}(\mathcal{M}) \propto q_{t-1}^{\Phi}(\mathcal{M}) \times q(\mathcal{M} \mid \mathbf{x}_t, \hat{\mathbf{z}}_t). \tag{5.11}$$

The update $q(\mathcal{M} \mid \mathbf{x}_t, \hat{\mathbf{z}}_t)$ approximately inverts rendering and costs roughly as much as rendering once. This is the *biggest* speed factor for PRISM, the paper has the runtime details. One restriction is that the map and the update are Gaussian, for fast Gaussian-product sensor fusion (Moravec, 1988). The other caveat is that the update conditional $q(\mathcal{M} \mid \mathbf{x}_t, \hat{\mathbf{z}}_t)$ needs to be predefined (engineered or learned), which can introduce bias. The update is depicted in fig. 5.7.

5.3.3. About the Bayesian Map and Marginalisation

An important point is that PRISM would not be possible without a Bayesian treatment of the map $\mathcal{M}$. This is necessary for the map updates in eq. (5.11) to work. Intuitively, the previous map belief $q_{t-1}^{\Phi}(\mathcal{M})$ is a fused proxy that reconciles all past image data. If the map was just a point estimate instead, one would need to account for all past images at once through the renderer (e. g. like in iMAP (Sucar et al., 2021)), which is inefficient.

Also, PRISM only marginalises out past poses (in the state filter), but not map parameters. Besides having the map for downstream tasks, one advantage due to this is that there is an *implicit loop-closure* effect whenever the agent revisits a map region.

Note that PRISM does not rely on linearisation of the full VSSM-LM graph in the filter derivations (i. e. it is not an EKF). This is because a) a full Gaussian covariance would be prohibitive for the dense VSSM-LM model anyway and b) it is well-known that linearisation in EKFs is one of the main sources of errors (S. Huang and Dissanayake, 2016).

[6] This is approximate, technically $\hat{\mathbf{z}}_t$ should come from $p(\mathbf{z}_t \mid \mathbf{x}_{<t}, \mathbf{u}_{<t-1})$, i. e. it should be predicted by the transition model, but $q_t^{\Phi}(\mathbf{z}_t)$ is more accurate on a budget.

5.3.4. Limitations

In terms of uncertainty, PRISM makes two main compromises for speed. These stem from a point in the derivation where each marginal filter marginalises out the sibling variable from the recursion:

$$q_t^{\Phi}(\mathbf{z}_t) \approx p(\mathbf{z}_t \mid \mathbf{x}_{\leqslant t}, \mathbf{u}_{<t}) = \int p(\mathcal{M}, \mathbf{z}_t \mid \mathbf{x}_{\leqslant t}, \mathbf{u}_{<t})\, d\mathcal{M}, \tag{5.12}$$

$$q_t^{\Phi}(\mathcal{M}) \approx p(\mathcal{M} \mid \mathbf{x}_{\leqslant t}, \mathbf{u}_{<t}) = \int p(\mathcal{M}, \mathbf{z}_t \mid \mathbf{x}_{\leqslant t}, \mathbf{u}_{<t})\, d\mathbf{z}_t. \tag{5.13}$$

1. The marginalisation integrals in eqs. (5.12) and (5.13) are MC-estimated with a single nominal sample from the sibling filter, as shown in figs. 5.6 and 5.7. This means map uncertainty does not propagate into the state estimates, and vice versa. Multi-sample MC estimation could address this, but would linearly increase the runtime. For example, this is where pose uncertainty lateral to the viewing direction could be accounted for in the map updates.

2. Correlations between $\mathbf{z}_t$ and $\mathcal{M}$ are not tracked, neither are correlations between individual map parameters. The main disadvantage is that the filtering updates cannot correct errors in map regions far away from the current step (H. Durrant-Whyte and Bailey, 2006). Sparse correlation estimation remained out of scope for PRISM, but would be a logical next step.

PRISM weaves in well-established computer vision and Bayesian techniques in the approximate filters (Kalman, 1960; Moravec, 1988; Chen and Medioni, 1992; Curless and Levoy, 1996; Steinbrücker et al., 2011). None of the individual pieces are entirely new on their own. The contribution of PRISM is rather in deriving a holistic posterior solution from the dense state-space model of VSSM-LM that runs in real-time, and this is novel. The result is a state estimator that matches the assumptions of any POMDP defined based on VSSM-LM.

A lot of care is taken to signpost exactly what approximations are currently necessary for speed. The hope is that this characterisation of the Bayesian approximation gap will facilitate future research, particularly in terms of uncertainty estimation in dense models.

5.4. Dynamics Identification

This section covers the learning of transition models, introduced together with VSSM-LM (Mirchev et al., 2021), see section 4 of the paper in appendix A.1. It requires background on variational inference (section 1.3.1), state-space models (section 1.4) and rigid-body transformations (section 2.1).

The predictive rollouts of VSSM-LM depend on a good transition. For model-based control, $p(\mathbf{z}_t \mid \mathbf{z}_{t-1}, \mathbf{u}_{t-1})$ needs to predict multiple time steps into the future. This section is about learning such transition models from data, such that their long-term drift is minimised.

5.4.1. Transition Learning from Pose Data

VSSM-LM identifies 6-DoF poses as part of the latent states $\mathbf{z}_t$. This assumption enables the pre-training of transitions on the outside.

Assume a data set of noisy poses $\hat{\mathbf{z}}_{1:T}$ and associated control inputs $\mathbf{u}_{1:T-1}$ is given. For example, the poses might be obtained from a MOCAP system or SLAM inference (through VSSM-LM or not). For transition learning, the recorded poses $\hat{\mathbf{z}}_t$ are treated as a noisy version of the true latent poses, i. e.

$$\hat{\mathbf{z}}_t = \mathbf{z}_t + \epsilon, \quad \epsilon \sim \mathcal{N}(\cdot \mid \mathbf{0}, \boldsymbol{\Sigma}). \tag{5.14}$$

This leads to a state-space model with an emission as in eq. (5.14) and a transition between the "true" latents $\mathbf{z}_t$:

$$p_\theta(\hat{\mathbf{z}}_{1:T}, \mathbf{z}_{1:T} \mid \mathbf{u}_{1:T-1}) = p(\mathbf{z}_1) \prod_{t=2}^{T} \underbrace{p_\theta(\mathbf{z}_t \mid \mathbf{z}_{t-1}, \mathbf{u}_{t-1})}_{\text{learned transition}} \prod_{t=1}^{T} \underbrace{p(\hat{\mathbf{z}}_t \mid \mathbf{z}_t)}_{\text{emission}}. \tag{5.15}$$

This SSM is very similar to the pose graphs used for loop closure in SLAM, but here it is used for transition learning. Transition parameters θ in $p_\theta(\mathbf{z}_t \mid \mathbf{z}_{t-1}, \mathbf{u}_{t-1})$ are optimised with variational inference in this model, through the evidence lower bound (ELBO)

$$\theta^* = \arg\max_\theta \mathcal{L}_{\text{elbo}}(\theta) \tag{5.16}$$

$$= \arg\max_\theta \mathbb{E}_q \left[\log p_\theta(\hat{\mathbf{z}}_{1:T}, \mathbf{z}_{1:T} \mid \mathbf{u}_{1:T-1}) - \log q_\phi(\mathbf{z}_{1:T} \mid \hat{\mathbf{z}}_{1:T}, \mathbf{u}_{1:T-1})\right]. \tag{5.17}$$

This asks for an approximate posterior $q_\phi(\mathbf{z}_{1:T} \mid \hat{\mathbf{z}}_{1:T}, \mathbf{u}_{1:T-1})$, which is implemented with a bidirectional recurrent neural network. The details

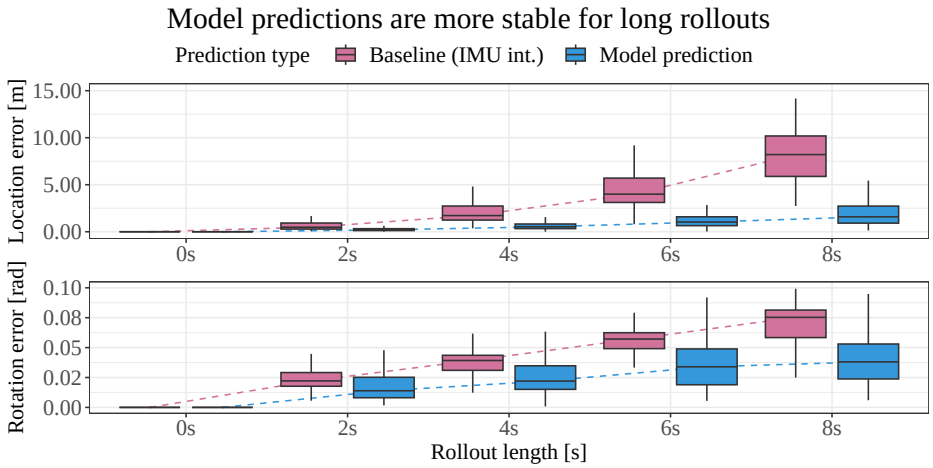


Figure 5.8.: Location and orientation absolute errors of predictions from the trained transition (blue) grow much slower than the errors of simple control integration (red). Evaluated on 3 m/s flight data from the Blackbird data set (Antonini et al., 2020).

are in the paper, and background on optimising generative parameters through the ELBO is available in section 1.3.2.

5.4.2. Mixing Inductive Biases

The mean of the transition $p_\theta(\mathbf{z}_t \mid \mathbf{z}_{t-1}, \mathbf{u}_{t-1})$ is the sum of kinematics integration (Euler's method) and a correction predicted by a neural net:

$$\mathbf{z}_{t+1} = f_{\text{kin}}(\mathbf{z}_t, \mathbf{u}_t) + \text{MLP}(\mathbf{z}_t, \mathbf{u}_t). \tag{5.18}$$

This is done to both ease learning and promote generalisation. The learned transition outperforms simple control integration, particularly in long-term prediction. This is summarised in fig. 5.8. The transition can be seamlessly integrated in VSSM-LM and leads to more stable predictive rollouts.

5.4.3. Limitations

Long-term prediction is important for control when there is no terminal cost (i. e. critic) (Bertsekas, 2005). But when there is one, it is likely that short-term prediction becomes more important, as rollouts can be terminated early. This needs to be checked empirically in future work.

Also, for the lack of a better alternative in the used data, the readings from an IMU sensor were used as a substitute for control inputs in the evaluation. While this is reasonable, it would have been ideal if the controls were the intended accelerations of the agent, rather than sensory readings.

5.5. NAVIGATION

> *This section is about downstream navigation with VSSM-LM, explored by TNP-SM (appendix A.3). Leading authorship of the paper is shared with Baris Kayalibay (the individual contributions are in the appendix). The section requires background on state-space models (section 1.4) and camera geometry and scene representations (section 2.2).*

After prediction and inference comes the applicability to control tasks. TNP-SM explores real-time navigation with VSSM-LM in complex simulated environments.

5.5.1. NAVIGATION UNDER PARTIAL OBSERVABILITY

MAP INFERENCE TNP-SM assumes a data set of prerecorded poses and images $(\mathbf{z}^{\text{pose}}_{1:k}, \mathbf{x}_{1:k})$ is given. It then reconstructs a map $\mathfrak{M}$ from this data, via gradient descent through the renderer of VSSM-LM (maximum likelihood). The agent must then navigate to a target position.

STATE ESTIMATION When deployed, the agent observes only RGB-D images, so its state must be estimated in real-time and plans must be reevaluated. Open-loop planning is not possible because the agent's real-world dynamics are noisy and cannot be predicted perfectly.

HIGH-LEVEL PLANNING & CONTROL The shortest obstacle-free navigation paths are found using dynamic programming (A* search) in a discretised 2D state space (see Bertsekas (2005) for DP methods).[7] The A* planner is informed about collisions through a cost function $c(\mathbf{z}_t, \mathfrak{M})$ based on the map occupancy, which is also used to determine whether 2D states are connected in terms of the A* graph. This is possible because occupancy can be evaluated everywhere, it is a field over $\mathbb{R}^3$. The above assumes the agent is *holonomic*, which is true in the experiments. A plan is then

[7] A* uses the Euclidean distance to the goal in this case, which is an admissible heuristic, guaranteeing optimality.

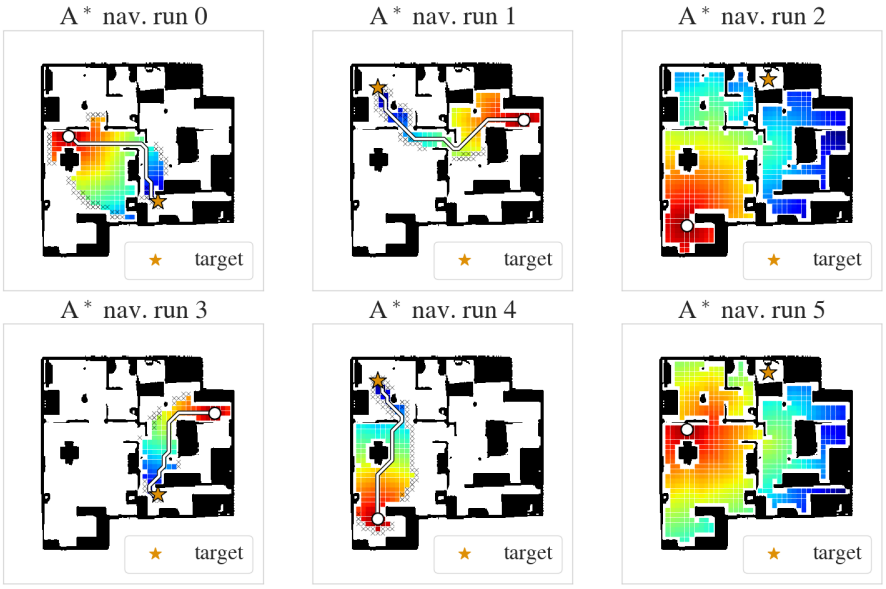
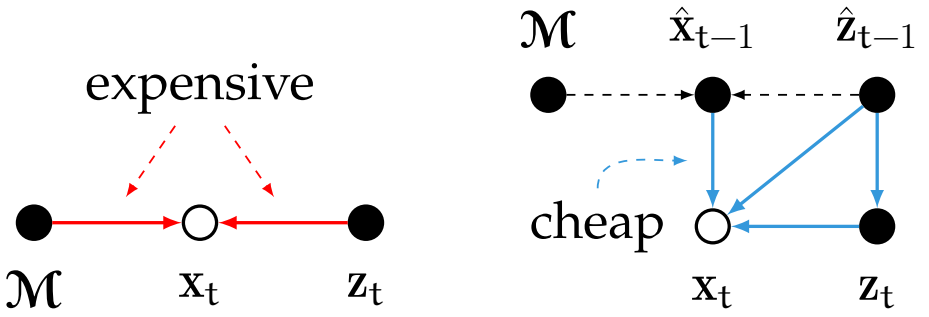


Figure 5.9.: In the figure *star* is the navigation target and *dot* is the starting point. It shows navigation plans in a reconstructed map $\mathfrak{M}$ for a 3D apartment from the ProcTHOR simulator (Deitke et al., 2022), along with the explored A* nodes colored by their cumulative cost (blue means low cost-to-go). The dense occupancy is sufficient to plan a path to the target from any starting point (first two columns), and also to identify when no path to the target exists (last column).

treated as a sequence of waypoints and traversed with a simple low-level controller. Example navigation plans are given in fig. 5.9. Note that A* covers only a part of the state space, the necessary minimum to find the shortest path.

The solution works in real-time across different multi-room simulated environments, reaching up to 0.92 SPL (success weighted by path length) under noisy agent movement and partial observability. Please consult the paper for the details and the evaluation. The remainder of this section focuses on the real-time state estimator, which is the main contribution by the author. Respectively, the high-level and low-level control laws and the navigation evaluation are a core part of the dissertation of Baris Kayalibay (to be submitted in 2023/2024).



(a) Rendering is expensive.

(b) Reprojection is cheap.

Figure 5.10.: Left: graphical model for one rendering step, showing that forward evaluation and gradient backpropagation through the renderer is expensive. Right: a previous-step RGB-D prediction $\hat{\mathbf{x}}_{t-1}$ is rendered *once* and is a proxy for a local segment of the map surface. The newest pose in $\mathbf{z}_t$ can then be optimised with gradients through the reprojection between $\hat{\mathbf{x}}_{t-1}$ and $\mathbf{x}_t$ (RGB-D image alignment), which is much faster.

5.5.2. FAST STATE ESTIMATION

One point of TNP-SM is that to achieve real-time, both for navigation and in general, the state estimator must avoid gradient descent through VSSM-LM's renderer, at least on current commodity GPUs. For this, TNP-SM derives an approximate MAP filter as such

$$\begin{aligned}
&\arg\max_{\mathbf{z}_t} p(\mathbf{z}_t \mid \mathbf{x}_{\leqslant t}, \mathbf{u}_{<t}, \mathcal{M}) \\
&\quad\approx \arg\max_{\mathbf{z}_t} \underbrace{\log p(\mathbf{x}_t \mid \mathbf{z}_t, \mathcal{M})}_{\text{rendering term}} + \underbrace{\log p(\mathbf{z}_t \mid \hat{\mathbf{z}}_{t-1}, \mathbf{u}_{t-1})}_{\text{transition term}} \\
&\quad\approx \arg\min_{\mathbf{z}_t} \underbrace{\mathcal{L}(\mathbf{x}_t, \mathbf{z}_t, \hat{\mathbf{z}}_{t-1}, \hat{\mathbf{x}}_{t-1})}_{\text{surrogate for rendering}} - \log p(\mathbf{z}_t \mid \hat{\mathbf{z}}_{t-1}, \mathbf{u}_{t-1}).
\end{aligned} \tag{5.19}$$

As usual, the details are in the included paper in appendix A.3. In the above, $\hat{\mathbf{z}}_{t-1}$ is the mean of the previous filtering step, and $\hat{\mathbf{x}}_{t-1}$ is a rendered RGB-D image for that viewpoint. The term $\mathcal{L}(\mathbf{x}_t, \mathbf{z}_t, \hat{\mathbf{z}}_{t-1}, \hat{\mathbf{x}}_{t-1})$ is a surrogate image alignment objective, mixing point-to-plane geometric alignment and photometric alignment. Minimising it finds the pose of the newest RGB-D image by aligning it to the previous $\hat{\mathbf{x}}_{t-1}$ rendered from the map. The surrogate is illustrated in fig. 5.10.

This technique is typical for dense odometry estimation (e. g. Newcombe

et al. (2011b) and Nießner et al. (2013)). The insight here is that it also applies to the VSSM-LM maps obtained via differentiable rendering. The overall objective also includes a dynamics term, which is a novel addition. What makes it fast is that it requires rendering from the map only *once*.

TNP-SM has experiments that show this surrogate leads to a four-fold decrease in estimation runtime, without loss in tracking accuracy. The same message concerning the path to real-time was later reiterated in PRISM, using $\mathcal{L}(\mathbf{x}_t, \mathbf{z}_t, \hat{\mathbf{z}}_{t-1}, \hat{\mathbf{x}}_{t-1})$ as one building block in its MAP pose optimisation and extending it with uncertainty and velocity to define the full state filter.

5.5.3. Limitations

TNP-SM shows that navigation works with VSSM-LM, which implies dense spatial SSMs are a viable control foundation. However, it does not cover the full control spectrum.

One limitation is that TNP-SM plans in a discrete 2D state space and assumes the agent is holonomic. This ignores any potential movement limitations of the dynamics. We will discuss a relaxation of this shortly. Also, TNP-SM does not make use of VSSM-LM's uncertainty.

Another limitation is that the maps are learned off-line and that the state estimator is deterministic. PRISM could have been used to address both of these issues, but chronologically it was published after TNP-SM. The integration of PRISM with an MPC controller is a contribution of the dissertation of Baris Kayalibay (to be submitted in 2023/2024), and it works on small- and medium-sized apartments from ProcTHOR (Deitke et al., 2022) (the current results indicate success rates of ca. 89% and ca. 73%, respectively). Note that this is a much harder task than the navigation in TNP-SM, as the map emerges gradually while the agent is navigating to a predefined xy-position in space.

Finally, proper POMDP belief-space planning that accounts for future state estimator errors was not explored. The main challenge is computational, as such an approach would require running the state estimator on multiple imagined rollouts. The non-core publication by Kayalibay et al. (2023) presents a middle ground between state-space (i. e. MDP) and belief-space (i. e. POMDP) planning, by training a neural net to predict the estimator errors across the whole state space, based on prerecorded data from the environment.

(a) Quanser QCar. (b) QCar arena.

Figure 5.11.: The QCar navigation setup. PRISM is applied to the scene on the right, using only RGB-D data. After this, the markers on the QCar are used to obtain fully-observable states for MDP control.

5.6. Further Control (Unpublished, Joint Work)

> *This section contains methods enabled by VSSM-LM. The presented work remains to be published.*

Following are two more example downstream applications: navigation with a robot car and spatial exploration.

The car navigation in section 5.6.1 is joint work by the author and Baris Kayalibay, Ziqing Zhao, Ahmed Agha, Ole Jonas Wenzel, Patrick van der Smagt and Justin Bayer. The author was involved in implementing the car control loop, obtaining the maps for the car, conceptualising and implementing value iteration, modelling the car dynamics and running experiments – all done together with coauthors. The exploration method in section 5.6.2 is based on the master's thesis of Ziqing Zhao (2023), which was supervised by the author. The author conceived the basic exploration approach and guided the research.

5.6.1. Navigation with a Robot Car

We will first consider navigation on a real, *non-holonomic* robot car. A Quanser QCar, shown in fig. 5.11a, is placed in a confined space cluttered with objects, shown in fig. 5.11b. The car is equipped with a Realsense 435 RGB-D camera. First, a map of the scene is obtained with PRISM given only RGB-D data collected by a human operator. The map is then fixed, and

the QCar is repeatedly asked to navigate autonomously between any two points. For simplicity, the QCar is given access to its own pose at this point (provided by an OptiTrack MOCAP oracle), isolating the control problem from state estimation. This leads to MDP instead of POMDP control.

Navigating with the QCar poses two challenges. First, the QCar is non-holonomic (it cannot move sideways), which requires sufficient *foresight* to plan maneuvers. Second, the QCar dynamics $p(\mathbf{z}_t \mid \mathbf{z}_{t-1}, \mathbf{u}_{t-1})$ are modelled with a simple bicycle kinematics model, which is imperfect (e. g. because of friction and changes in battery voltage). This can be compensated for by *replanning* as new state observations arrive.

Fortunately, both of these challenges are covered by the standard MDP control toolkit.

Formulating a Critic

The foresight aspect is addressed by value iteration. It is used to approximate the navigation cost-to-go

$$J(\mathbf{z}_t) = \mathbb{E}\left[\sum_{\tau=t}^{\inf} \gamma^{\tau-t} c(\mathbf{z}_\tau, \mathfrak{M})\right], \tag{5.20}$$

where γ is a discount factor. The per-step cost $c(\mathbf{z}_t, \mathfrak{M})$ has two values, high if $\mathbf{z}_t$ collides with an obstacle (based on the map $\mathfrak{M}$), and low when the agent is at the target. Because value iteration solves the infinite-horizon problem, it automatically grants the necessary foresight to the agent.[8]

Aggregation is used to make value iteration tractable, discretising the agent state space and the aforementioned bicycle transition model (Bertsekas, 2005). Contrary to TNP-SM, the discretisation is now three-dimensional, taking into account both the 2D position and *the orientation* of the agent. This means the cost-to-go grid is a 3D volume, where each slice corresponds to one specific orientation. This is depicted in fig. 5.12. In practice, this makes the agent aware that its approach angle w. r. t. the target should be different depending on its xy-position, to satisfy its movement constraints.

Constrained Model Predictive Control

One way to bridge the discrete cost-to-go and the continuous reality of the agent is with constrained *Model Predictive Control* (MPC). This is a

[8] This is a general statement for any agent equipped with a cost-to-go (critic).

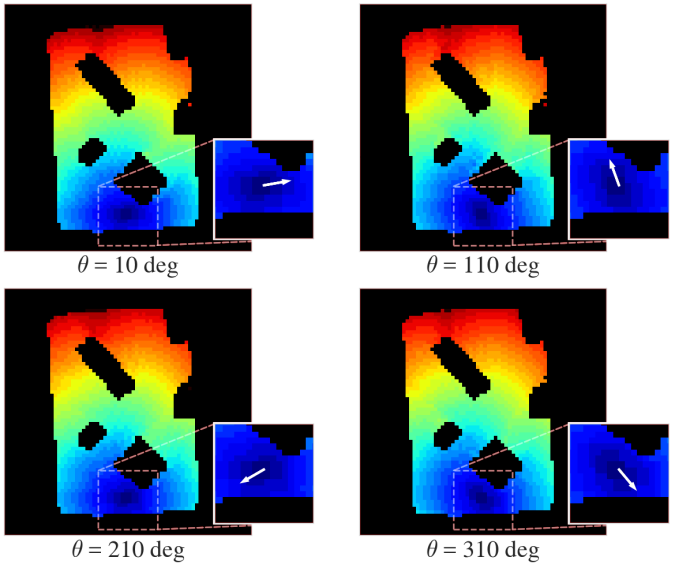


Figure 5.12.: Four slices of the 3D cost-to-go volume, low cost-to-go in blue (this is where the navigation goal is). The orientation for each slice is shown with an arrow. Note how there is a clear preference for xy-positions from which the car can move forward or backward into the target, reflecting the car's non-holonomic constraints.

contribution from the dissertation of Baris Kayalibay (to be submitted in 2023/2024), the following is only a rehash. Constrained MPC uses the computed cost-to-go as a terminal cost:

$$\mathbf{u}^*_{\geqslant t} = \arg\min_{\mathbf{u}_{\geqslant t}} \mathbb{E}_{\mathbf{z}_{t+1:T}\sim p(\mathbf{z}_{t+1:T}|\mathbf{z}_t,\mathbf{u}_{\geqslant t})} \left[\underbrace{\gamma^{T-t} J(\mathbf{z}_T)}_{\text{terminal cost}} + \sum_{\tau=t}^{T-1} \gamma^{\tau-t} c(\mathbf{z}_\tau, \mathcal{M}) \right]$$
$$\text{s.t.} \quad \alpha(\mathbf{z}_\tau, \mathcal{M}) \leqslant 0, \tau = t, \dots, T. \tag{5.21}$$

The current agent state here is $\mathbf{z}_t$. The constraints $\alpha(\mathbf{z}_\tau, \mathcal{M})$ punish any future QCar state that is too close to an obstacle. The rollout expectation in eq. (5.21) is MC-estimated[9] and the objective is minimised by a) finding initial admissible controls with random search and b) refining these controls with Lagrangian optimisation. MPC control then executes only the first $\mathbf{u}^*_t$, after which the procedure repeats – this satisfies the need for replanning

[9] This is in general, in the experiments the QCar has a deterministic transition.

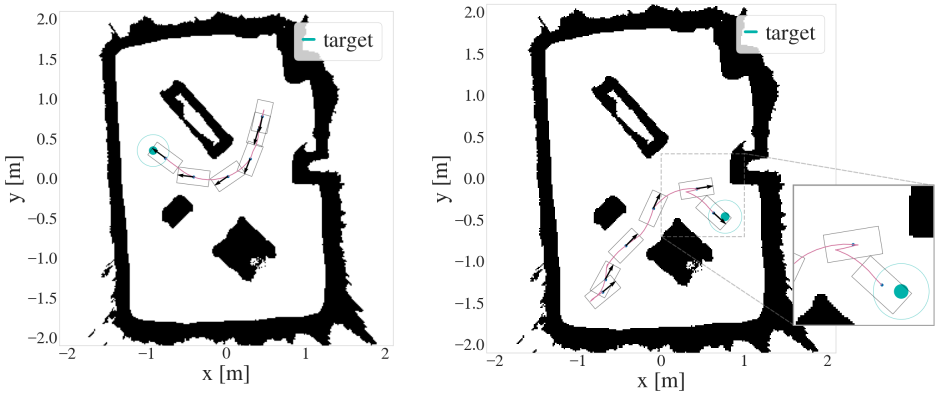


Figure 5.13.: Example QCar navigation runs. Note how in the second case the car is aware that it cannot complete the initially planned maneuver, therefore it backtracks and completes the run.

due to the suboptimal transition. As the cost-to-go has already done the heavy lifting, the MPC horizon is kept short (e. g. up to 25 steps, or ca. 3 s).

Results

Intuitively, the MPC objective makes the QCar drive towards states of low cost-to-go (accounting for orientation), while at the same time avoiding obstacles at all costs. This also makes it aware that it sometimes needs to backtrack, in order to approach the target from a safe angle. This is shown in the example runs in fig. 5.13. In one round of experiments done by the author, the car was asked to navigate to 100 consecutive random targets. It reached each target every time, touching an obstacle on the way in 8% of the runs. Note that these results may evolve until the work gets published.

Limitations

The limitations of TNP-SM apply here as well, with two exceptions. First, the non-holonomic dynamics of the agent are accounted for, both in value iteration and in MPC control. And second, MPC control with a terminal cost makes for a tighter, more transparent control loop compared to the disjoint high-level planning and low-level control in TNP-SM.

A unique limitation of value iteration using aggregation is that the size of the cost-to-go grid scales poorly. Evaluation is still fast for a

three-dimensional state space, parallelised on a GPU, but adding further dimensions (e. g. for 6-DoF navigation) would require a different approach.

5.6.2. Autonomous Exploration

Next we consider how VSSM-LM can be used for autonomous exploration. This is joint work produced during the master's thesis of Ziqing Zhao (2023), supervised by the author. The author conceived the high-level mathematical derivations and guided the work. The implementation, the further development of the method to improve its efficiency and the experiments were done by Ziqing Zhao. Note that similar equations and results will appear in her thesis, along with more details. This is only a summary.

We focus on exploration in isolation, and assume full observability of the agent states $\mathbf{z}_t$. The agent is placed in an unknown scene, and has to reconstruct its map as fast as possible. The uncertainty in the approximate map posterior $q(\mathcal{M})$ is what enables this, using information theory. Specifically, the mutual-information (MI) objective

$$\mathbf{u}^*_{\geqslant t} = \arg\max_{\mathbf{u}_{\geqslant t}} I(\mathcal{M}; \mathbf{x}_{>t} \mid \mathbf{u}_{\geqslant t}, \mathbf{z}_t) \tag{5.22}$$

promotes finding future controls $\mathbf{u}^*_{\geqslant t}$ that will lead to a sequence of images $\mathbf{x}_{>t}$ maximally informative about the map $\mathcal{M}$ (starting from the current state $\mathbf{z}_t$, for brevity past data $\mathcal{D}_{\leqslant t}$ is omitted from conditions but implied). This is equivalent to minimising the map entropy $\mathbb{H}(\mathcal{M})$ after observing $\mathbf{x}_{>t}$, because $I(\mathcal{M}; \mathbf{x}_{>t}) = \mathbb{H}(\mathcal{M}) - \mathbb{H}(\mathcal{M} \mid \mathbf{x}_{>t})$.[10]

For simplicity, in the following experiment we will shift to high-level planning and optimise the MI w. r. t. the future states $\mathbf{z}_{>t}$ instead of $\mathbf{u}_{\geqslant t}$. This will lead to a trajectory of waypoints that should be followed. It is conceptually equivalent to assuming a deterministic initial state and transition. The optimisation becomes:

$$\mathbf{z}^*_{>t} = \arg\max_{\mathbf{z}_{>t}} I(\mathcal{M}; \mathbf{x}_{>t} \mid \mathbf{z}_{>t}). \tag{5.23}$$

[10] Here $\mathbf{u}_{\geqslant t}$ and $\mathbf{z}_t$ are omitted to avoid confusion between *conditional entropy* vs. *entropy of a conditional distribution*.

Evaluation is made tractable using the MI decomposition

$$\begin{aligned} &\mathrm{I}(\mathfrak{M}; \mathbf{x}_{>t} \mid \mathbf{z}_{>t}) \\ &\quad = \mathbb{H}(\mathfrak{M}) - \mathbb{E}_{\mathrm{p}(\mathbf{x}_{>t} \mid \mathbf{z}_{>t})}\left[\mathbb{E}_{\mathrm{p}(\mathfrak{M} \mid \mathbf{x}_{>t}, \mathbf{z}_{>t})}[-\log \mathrm{p}(\mathfrak{M} \mid \mathbf{x}_{>t}, \mathbf{z}_{>t})]\right] \qquad (5.24) \\ &\quad \approx \mathbb{H}(\mathfrak{M}) - \mathbb{E}_{\mathrm{p}(\mathbf{x}_{>t} \mid \mathbf{z}_{>t})}\left[\underbrace{\mathbb{E}_{\mathrm{q}(\mathfrak{M} \mid \mathbf{x}_{>t}, \mathbf{z}_{>t})}[-\log \mathrm{q}(\mathfrak{M} \mid \mathbf{x}_{>t}, \mathbf{z}_{>t})]}_{\text{entropy of } \mathrm{q}(\mathfrak{M} \mid \mathbf{x}_{>t}, \mathbf{z}_{>t})}\right], \qquad (5.25) \end{aligned}$$

where eq. (5.24) follows from the definition of MI. Equation (5.25) computes the expected entropy of a posterior approximation $\mathrm{q}(\mathfrak{M} \mid \mathbf{x}_{>t}, \mathbf{z}_{>t})$. This is the crux of the method, as we can use PRISM's map updates to obtain this approximate posterior quickly in closed form:

$$\mathrm{q}(\mathfrak{M} \mid \mathbf{x}_{>t}, \mathbf{z}_{>t}) \propto \mathrm{q}_t^{\Phi}(\mathfrak{M}) \prod_{\tau=t+1}^{T} \mathrm{q}(\mathfrak{M} \mid \mathbf{z}_\tau, \mathbf{x}_\tau). \qquad (5.26)$$

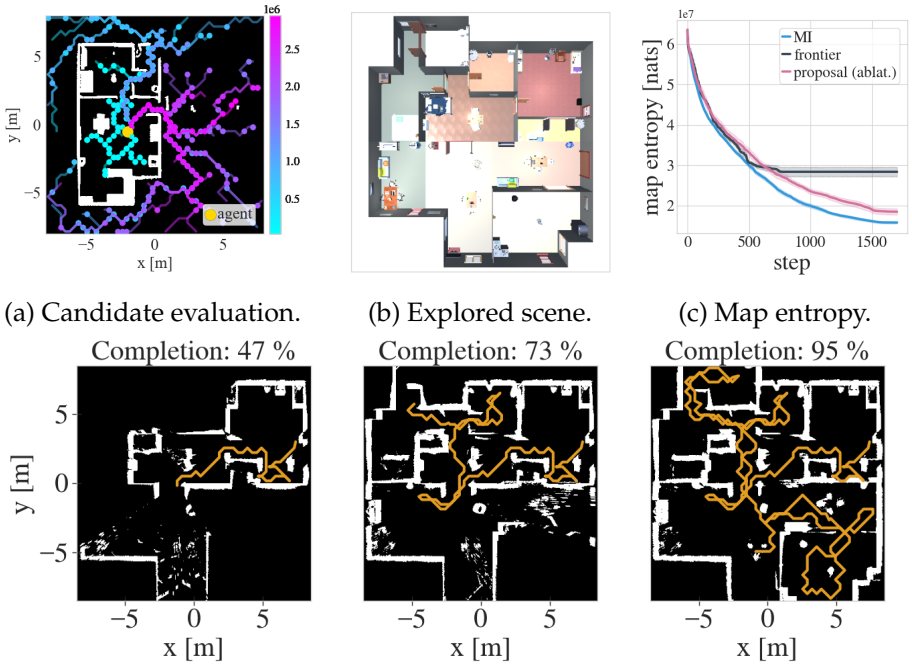
The observations $\mathbf{x}_{>t}$ in eq. (5.25) are *imagined* by rendering from the current map belief $\mathrm{q}_t^{\Phi}(\mathfrak{M})$, they are ancestrally sampled from the predictive distribution

$$\mathrm{p}(\mathbf{x}_{>t} \mid \mathbf{z}_{>t}) = \mathbb{E}_{\mathfrak{M} \sim \mathrm{q}_t^{\Phi}(\mathfrak{M})}\left[\prod_{\tau=t+1}^{T} \mathrm{p}(\mathbf{x}_\tau \mid \mathfrak{M}, \mathbf{z}_\tau)\right], \qquad (5.27)$$

and used to MC-estimate the outer expectation in eq. (5.25). Intuitively, the agent imagines the impact of these hypothetical observations on the map.

The evaluation of one MC sample takes ca. 30 ms per rollout time step on a commodity GPU. The algorithm can be sped up even further using a *sparse* set of rendered pixels and respective updates, the runtime scales linearly in the pixel count. The overall method is inspired from early prior work on infogain exploration (e. g. Stachniss and Burgard (2003)), the novelty here is in basing it on VSSM-LM's predictive rendering and PRISM's map filter.

Optimisation is done with random search. First, candidate trajectories $\{\mathbf{z}_{>t}^k\}_{[K]}$ are proposed via heuristic breadth-first search that spans the open regions in the map. Free regions are determined via the occupancy in the current map belief $\mathrm{q}_t^{\Phi}(\mathfrak{M})$, which reflects only what was explored so far. The best candidate trajectory $\mathbf{z}_{>t}^*$ is then selected based on the MI objective. An example candidate evaluation is depicted in fig. 5.14a. The agent then traverses the chosen waypoints and reconstructs the scene until an unforeseen obstacle is too close (based on a safety threshold), at which point replanning occurs.



(a) Candidate evaluation. (b) Explored scene. (c) Map entropy.

(d) Example exploration run.

Figure 5.14.: (a) Candidate trajectories, colored by their MI score (magenta is high). The empty map space on the right is preferred. (b) The explored 3D scene, simulated with PocTHOR (Deitke et al., 2022). (c) Comparison of the proposed approach, frontier-based exploration and an ablation that only uses the candidate generation heuristic. (d) Three stages of an exploration run, occupancy in white, traversed trajectory in orange.

Results

The method is used to explore the apartment scene depicted in fig. 5.14b. An example exploration run is shown in fig. 5.14d. The agent reaches all apartment rooms. In fig. 5.14c, the algorithm is compared to a traditional frontier-based exploration baseline and an ablation that uses only the candidate generation heuristic. The proposed method leads to moderately faster reduction of map entropy aggregated over 25 runs, indicating faster exploration overall. Notably, frontier-based exploration stops early when all walls have been seen once, regardless of the quality of the map estimate. In contrast, the information-theoretic approach continues to actively refine the map based on the remaining uncertainty.

Limitations

One limitation of the method is that it does not consider the agent dynamics, as it optimises for future poses directly. Agents with non-holonomic constraints can then struggle to follow the returned waypoints. An optimal solution would optimise the MI w. r. t. future controls, but this would be expensive.

The other main caveat is the candidate generation heuristic, which can bias exploration. The root problem here is that MI evaluations are rather expensive even under the current approximations, which makes exhaustive search prohibitive. Overall, this leads to suboptimal exploration, but as the experiments show it is still competitive with standard frontier-based approaches.

5.7. Contributions

This concludes the methods chapter. In summary, the contributions are:

1. A new dense *spatial state-space* model is formulated. It is direct, it predicts raw RBG-D images and agent movement through a rendering emission and a 6-DoF dynamics transition. It is control-driven and can be extended into a POMDP. The model bridges the assumptions of model-based control and dense SLAM (section 5.1).

2. All variables come with *uncertainty*, both in prediction and in inference from data (section 5.1.1). Uncertainty can enable robust control in the future, and map uncertainty already enables information-theoretic exploration (section 5.6.2).

3. The renderer is *differentiable* which enables optimisation through the model (sections 5.1 and 5.1.2).

4. A smoothing posterior for the model is obtained via *variational inference*. This is the first application of VI for dense SLAM smoothing. VI is non-linear, opens the door to flexible uncertainty estimation and explores new ground compared to traditional MAP or EKF SLAM inference, but it currently runs off-line (section 5.2).

5. Real-time inference is addressed by a *Bayes filter* approximation (section 5.3). This Bayes filter can be used as an approximate state estimator in a POMDP. It is shown that one path to real-time is to

limit rendering in the filter (e. g. by using closed-form updates for the map). The approximations necessary for speed are carefully signposted.

6. It is shown how to *learn transitions* for the model from prerecorded trajectory data (section 5.4).

7. Overall, it is shown the developed model enables standard control algorithms, like DP, value iteration and MPC control (sections 5.5 and 5.6.1). These are used for *real-time navigation* with a holonomic agent with only RGB-D observations (TNP-SM) and a non-holonomic agent with observed states (QCar). Collision avoidance is enabled by the dense occupancy in the inferred 3D map.

The presented methods are suitable for agents operating in moderate-scale indoor scenes.

Discussion

6

Early on in chapter 3 the thesis established some general desiderata:

- models targeting control,
- white-box assumptions,
- dense direct assumptions,
- state-space assumptions,
- full-posterior probabilistic inference.

These were for the sake of aligning dense SLAM and model-based control. The presented research is one perspective on how to fulfill them. The following section 6.1 summarises the trade-offs of the chosen path, grouping them based on the bullets above. The chapter then ends with an outlook in section 6.2.

6.1. Compromises & Trade-Offs

Models for Control

First, and this is important to clarify, spatial control can be approached with any SLAM. For example, one can use most off-the-shelf visual odometry or SLAM methods to obtain agent poses from a raw video stream. State estimation can then be seen as a black box, and one can use the pose (and possibly map) estimates as *observations* in a separate MDP, with its own dynamics. In fact, this is structurally not that different from the control scenarios (TNP-SM, QCar) explored in this thesis. Of course, what can be controlled will differ – e. g. not all SLAMs return dense maps, but the overall scheme works conceptually.

The point of the research is that cohesive state estimation and planning will eventually be needed for POMDP control, when the state estimator

needs to be accounted for in the planned rollouts, and this is currently not straightforward with *dense direct spatial models*. VSSM-LM and PRISM manage to bridge a part of this gap, by being POMDP-compliant. The trade-off for this is that the thesis has to deviate from the well-trodden paths for dense SLAM.

The presented navigation experiments do not go as far as POMDP control (i. e. belief-space planning) because running PRISM's filter over simulated MC-rollouts is expensive on current hardware (see Kayalibay et al. (2023) for a middle ground). Only the information-theoretic exploration comes close, because it evaluates the effect of map updates into the agent future. In general, there is a long list of control tasks that should be explored in the future, such as making better use of VSSM-LM's uncertainty estimates or using more of the map content (e. g. for semantics). The control results so far are an indication that dense SSMs like VSSM-LM are a reasonable foundation for spatial control.

White-Box Assumptions

The heavy use of domain knowledge in VSSM-LM narrows down the applicability of the model to only the spatial domain. This loses some of the appeal of deep *world models*, which are meant to be generic simulators that work with any data and any agent, e. g. Dreamer (Hafner et al., 2023). In that sense, spatial domain knowledge is accepted as the price of generalisation. It remains to be seen whether large-scale learning initiatives, along the lines of Gato (Reed et al., 2022) or RT-1/2 (Brohan et al., 2023), will be able to shift the balance in favour of black-box models in the future.

Note that just like any other world model VSSM-LM can be used for the training of parametric policies and critics. The positive is that the generative model does not need to be learned, it generalises by design.

Dense, Direct Assumptions

Dense modelling makes everything more expensive, due to the sheer amount of modelled parameters and data. It also limits the presented models to indoor scenes, not bigger than large multi-room apartments (this could be improved, see next section). At the same time, it makes simulation more informative. For example, this is why collision avoidance is easily possible.

Related to this, the direct prediction of RGB-D data via $p(\mathbf{x}_t \mid \mathfrak{M}, \mathbf{z}_t)$ is highly non-linear in $\mathbf{z}_t$. Posteriors are thus multi-modal, and captur-

ing them faithfully would require non-Gaussian assumptions (hence the interest in variational inference). This also means unimodal posterior approximations are dependent on careful initialisation, in a limited convergence basin. Directly predicting images also means that inference is sensitive to the quality of the RGB-D data.

In the thesis state initialisation is addressed by using the control-driven transition (both in the smoother and in PRISM).

SSM Assumptions

As mentioned before, the renderer $p(\mathbf{x}_t \mid \mathcal{M}, \mathbf{z}_t)$ is what makes it possible to define a dense SSM. As such, it is currently seen as a necessary component. Its computational cost in the current implementation is $\mathcal{O}(whr)$, where $w \times h$ is an image size and r a ray resolution. Improving this runtime would automatically improve inference (more optimisation on a budget) and enable more interesting control (cheaper simulation).

As for the transition, one thing to note is that VSSM-LM assumes control inputs are given. One limitation of the public robot data used for evaluation was that IMU data had to be used as a substitute for intended acceleration controls.

Finally, the SSM assumes observations are RGB-D images. While VSSM-LM could be extended to monocular data, by treating the rendered depth as latent, this was not explored. This direction would additionally require very careful initialisation of the depth latents & map.

Full-Posterior Inference

In terms of inference, the main limitations stem from the expense of the dense assumptions and rendering.

For the variational smoother, this limits convergence accuracy on a budget. The result is that the method runs off-line, and even then its localisation and uncertainty estimates are approximate (there is a small gap in localisation accuracy compared to the SotA). Also, the thesis explores only a unimodal product of Gaussians for the posterior, a proof of concept for applying VI. Still, the fact that VI works at all matters, as it paves the way to multi-modal posterior approximations in the future.

Similarly, for PRISM's filter the expense of rendering means uncertainty is quite approximate, because uncertainty propagation through the filtering recursion becomes difficult on a budget. Also, in both PRISM and in the smoother correlations between the latents $\mathcal{M}$ and $\mathbf{z}_t$ are not modelled,

because of the size of the dense 3D map, which is another uncertainty limitation.

Despite these caveats, the two solutions deliver full posteriors approximations, which is not common for spatial models based on differentiable rendering (e. g. see iMAP (Sucar et al., 2021) or NICE-SLAM (Zhu et al., 2022)). The usefulness of the map uncertainty is shown in the MI exploration experiment. For the state uncertainty, there is an uncertainty calibration analysis of PRISM compared to the smoother at the end of appendix A.2, but its usefulness remains to be tested in a control context.

6.2. Outlook

The above compromises lead to a number of interesting directions for future research.

Addressing Scalability

First, the easiest point to address would be the scalability of the dense maps. For example, an octree (Meagher, 1982) can directly replace the voxel grid in $\mathfrak{M}$, and respectively reduce the memory footprint and speed up rendering. This is expected to improve performance across the board.

Advanced Maps & Advanced Rendering

From an impact perspective, the most interesting direction appears to be that of dynamic maps, as this is very relevant for real-world control. Because VSSM-LM models raw RGB-D images directly, reasoning about objects in the scene should be possible. Segmentation methods in 3D can be considered for the extraction of individual rigid bodies (e. g. see the very recent Cen et al. (2023)). Such a direction would require restructuring the single voxel grid map into a set of posed topological map nodes, somewhat similar to the notion of volumetric primitives (Lombardi et al., 2021) or 3D Gaussian primitives (Luiten et al., 2023). This would then require reasoning about the dynamics of multiple objects in the scene, which could still be framed as a state-space model and remain compatible with the POMDP paradigm.

In terms of the map fidelity, one can also consider different rendering formulations for the SSM emission. For example, both surfel splatting (e. g. Schöps et al. (2019)) and radiance aggregation (e. g. Mildenhall et al. (2020)) are differentiable and would fit the rest of the model.

More Flexible Filtering

If more computation is available, PRISM's uncertainty propagation could be improved next. The single-sample MC approximations for the expectations of the filter can be replaced by multi-sample MC uncertainty propagation. Even with a small number of samples uncertainty will start flowing between the map and state estimates.

Another point is that the MAP pose optimisation and Laplace approximation in PRISM's state filter should be replaced with variational inference, as that can improve the uncertainty calibration of the estimated poses. However, convergence is expected to require longer than the current solution.

Lastly, PRISM currently does not model any correlations between the map and state parameters. A sparse correlation tracking scheme (e. g. between only a subset of the map parameters and the states) appears as a feasible middle-ground that should improve the uncertainty of the filter further.

More Flexible Smoothing

Another interesting research avenue would be using more advanced approximate posterior families in the variational smoother, along the lines of normalising flows (Rezende and Mohamed, 2015). Variational inference directly supports this. These are generally more expensive to optimise than Gaussians, so this would require either a significant reduction in the rendering runtime (e. g. see previous note on scalability) or better GPU hardware.

From a methods standpoint, one could also consider a keyframe approach, using pose marginalisation to limit the scope of the smoother (Leutenegger et al., 2013). Effectively, this could turn the full smoother into a fixed-lag smoother, moving more towards the divide-and-conquer strategy of PRISM. This will focus the optimisation budget on fewer time steps, but would possibly require modelling dense correlations between map parameters and the agent states, which can be challenging.

Advanced Control

As already noted, more research is needed to tap into VSSM-LM's potential for control.

The information in the map and respectively the rendered images can be used for tasks that go beyond point-to-point navigation, like finding objects

based on visual cues. This can be formulated as a task that maximises the likelihood of a certain image observation under the model.

Also, more experiments are needed to verify if VSSM-LM's state uncertainty is useful in a control context. In regular MDP planning, this can be done by accounting for the model uncertainty both in value iteration and in MPC rollouts. If computation allows it, one could also consider using PRISM for POMDP belief-space planning, where the uncertainty of the state estimator will also contribute.

Part III.

Included Publications

Core Publications

A.1. Variational SSMs for Localisation and Dense 3D Mapping

Paper Summary

VSSM-LM introduces a novel dense state-space model. The model combines a differentiable renderer (emission) with a 6-DoF dynamics model (transition). The state-space model predicts raw RGB-D data directly, from a dense 3D grid map. It is shown that the resulting probabilistic predictions are suitable for control.

The model is end-to-end differentiable and its parameters can be inferred with variational inference, a more relaxed fully-Bayesian method compared to traditional MAP optimisation. For that, a novel variational smoother is derived using the ELBO, and then shown to reach localisation performance close to that of SotA odometry and SLAM methods. The experiments are performed on quadcopter indoor flight data. The VI smoother is computationally expensive and runs off-line.

Finally, it is shown that agent dynamics models can be learned from prerecorded trajectory data. Such models seamlessly integrate into VSSM-LM and improve its predictive performance.

Author Contributions

problem definition	significant
literature review	significant
theoretical derivation	significant
implementation	significant
experiments	significant
writing	significant

VARIATIONAL STATE-SPACE MODELS FOR LOCALISATION AND DENSE 3D MAPPING IN 6 DOF

Atanas Mirchev **Baris Kayalibay** **Patrick van der Smagt** **Justin Bayer**

Machine Learning Research Lab, Volkswagen Group, Munich, Germany
{atanas.mirchev,bkayalibay,bayerj}@argmax.ai

ABSTRACT

We solve the problem of 6-DoF localisation and 3D dense reconstruction in spatial environments as approximate Bayesian inference in a deep state-space model. Our approach leverages both learning and domain knowledge from multiple-view geometry and rigid-body dynamics. This results in an expressive predictive model of the world, often missing in current state-of-the-art visual SLAM solutions. The combination of variational inference, neural networks and a differentiable raycaster ensures that our model is amenable to end-to-end gradient-based optimisation. We evaluate our approach on realistic unmanned aerial vehicle flight data, nearing the performance of state-of-the-art visual-inertial odometry systems. We demonstrate the applicability of the model to generative prediction and planning.

1 INTRODUCTION

We address the problem of learning representations of spatial environments, perceived through RGB-D and inertial sensors, such as in mobile robots, vehicles or drones. Deep sequential generative models are appealing, as a wide range of inference techniques such as state estimation, system identification, uncertainty quantification and prediction is offered under the same framework (Curi et al., 2020; Karl et al., 2017a; Chung et al., 2015). They can serve as so-called *world models* or environment simulators (Chiappa et al., 2017; Ha & Schmidhuber, 2018), which have shown impressive performance on a variety of simulated control tasks due to their predictive capability. Nonetheless, learning such models from realistic spatial data and dynamics has not been demonstrated. Existing spatial generative representations are limited to simulated 2D and 2.5D environments (Fraccaro et al., 2018).

On the other hand, the state estimation problem in spatial environments—SLAM—has been solved in a variety of real-world settings, including cases with real-time constraints and on embedded hardware (Cadena et al., 2016; Engel et al., 2018; Qin et al., 2018; Mur-Artal & Tardós, 2017). While modern visual SLAM systems provide high inference accuracy, they lack a predictive distribution, which is a prerequisite for downstream perception–control loops.

Our approach scales the above deep sequential generative models to real-world spatial environments. To that end, we integrate assumptions from multiple-view geometry and rigid-body dynamics commonly used in modern SLAM systems. With that, our model maintains the favourable properties of generative modelling and enables prediction. We use the recently published approach of Mirchev et al. (2019) as a starting point, in which a variational state-space model, called DVBF-LM, is extended with a spatial map and an attention mechanism. Our contributions are as follows:

- We use multiple-view geometry to formulate and integrate a differentiable raycaster, an attention model and a volumetric map.
- We show how to integrate rigid-body dynamics into the learning of the model.
- We demonstrate the successful use of variational inference for solving direct dense SLAM for the first time, obtaining performance close to that of state-of-the-art localisation methods.
- We demonstrate strong predictive performance using the learned model, by generating spatially-consistent real-world drone-flight data enriched with realistic visuals.
- We demonstrate the model's applicability to downstream control tasks by estimating the cost-to-go for a collision scenario.

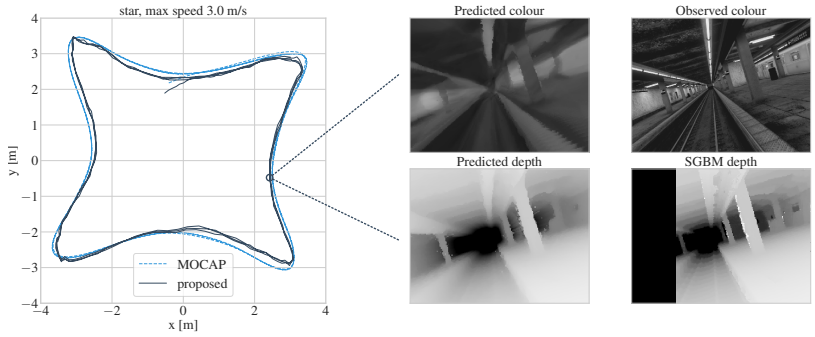


Figure 1: Illustration of the proposed quadcopter localisation and dense mapping. Left: top-down view of the localisaton estimate. Right: generative depth and colour reconstructions for one time step.

The contributions allow the reformulated model to tackle realistic RGB-D scenarios with 6 DoF.

2 Related work

Generative models for spatial environments GTM-SM (Fraccaro et al., 2018) focuses on long-term predictions with a non-metric deterministic external memory. Chaplot et al. (2018) formulate an end-to-end learning model for active global localisation, filtering with a likelihood update predicted by a neural network. The agent can turn in four directions and move on a plane, perceiving images of the environment. VAST (Corneil et al., 2018) assumes a discrete state space for a generative model applied to the 2.5D Vizdoom environment. Whittington et al. (2018) model agents moving on a 2D grid with latent neurologically-inspired grid and place cells. Other works propose end-to-end learnable generative scene models (Eslami et al., 2018; Engelcke et al., 2020), without considering the agent dynamics. Like in the above, we put major emphasis on the generative predictive distribution of our model. With it, the agent can imagine the consequences of its future actions, a prerequisite for data-efficient model-based control (Chua et al., 2018; Hafner et al., 2019a;b; Becker-Ehmck et al., 2020). However, the aforementioned deep generative spatial models have only been applied on simulated 2D, 2.5D (movement restricted to a plane) and very simplified 3D environments.

A major challenge when scaling to the real world is to ensure that the learned components, and in turn the generative predictions, generalise to observed but yet unvisited places. Gregor et al. (2019) highlight another problem, that of long-term consistency when predicting ahead, and address it by learning with overshooting. In contrast, our method resolves these issues by injecting a sufficient amount of domain knowledge, without limiting the flexibility w. r. t. learning. To this end, we begin by sharing the probabilistic factorisation of DVBF-LM (Mirchev et al., 2019), a deep generative model that addresses the tasks of localisation, mapping, navigation and exploration in 2D. We then redefine the map, the attention, the states, the generation of observations and the overall inference, allowing for real-world 3D modelling and priming our method for data-efficient online inference in the future. We discuss why these changes are necessary more thoroughly in appendix A.

Combining learning and spatial domain knowledge Fully-learned spatial models with an explicit memory component have been studied by Parisotto & Salakhutdinov (2018); Zhang et al. (2017); Oh et al. (2016). Further relying on geometric knowledge, Tang & Tan (2019) propose learning through the whole bundle adjustment optimisation, formulated on CNN feature maps of the observed images. Czarnowski et al. (2020) define a SLAM system based on learned latent feature codes of depth images, a continuation of the works by Zhi et al. (2019); Bloesch et al. (2018). Factor-graph maximum a posteriori optimisation is then conducted, substituting the observations for their respective low-dimensional codes, leading to point estimates of the individual geometry of N keyframes and the agent poses over time. Wei et al. (2020) maintain cost volumes (Newcombe et al., 2011) for discretised poses and depth, and let a 3D CNN learn how to predict the correct geometry and pose estimates from them. Depth cost volumes are also used by Zhou et al. (2018) in learning to predict depth and odometry with neural networks. In the work by Yang et al. (2020), networks that predict odometry and depth are combined with DSO, leading to a SLAM system that utilises learning to its

advantage. Jatavallabhula et al. (2019) investigate differentiable SLAM, treating odometry estimation and mapping separately. The considered rendering gradients in that method are from the fused map to the observations, which is the opposite of the gradient paths used for learning in our work.

As in our approach, the geometric assumptions in the majority of these works allow the systems to generalise more easily to unseen cases and real-world data. What distinguishes our method is an explicit generative model, able to predict the agent movement and observations in the future. Additionally, our approach is fully-probabilistic, maintaining complete distributions over the variables of interest, whereas the aforementioned approaches are not. Our method is also end-to-end differentiable and can be implemented in auto-diff frameworks, welcoming learned components. We are bridging the gap between probabilistic generative models, learning and spatial domain knowledge.

Depth estimation and differentiable rendering Recent promising approaches combine learning and non-parametric categorical distributions for depth estimation (Laidlow et al., 2020; Liu et al., 2019), fusing likelihood terms into a consistent depth estimate. Such depth estimation is compatible with our system and can be used to formulate priors, but for now we rely on a traditional method as a first step (Hirschmuller, 2007). Inferring whole scenes parameterised with neural networks by backpropagation through realistic differentiable rendering has also become a prevalent direction of research (Bi et al., 2020; Mildenhall et al., 2020; Sitzmann et al., 2020). In our method the occupancy and colour map are inferred in a similar way, but the raycasting scheme we follow is simple, meant only to illustrate the framework as a whole. We note that the current inference times of e.g. Bi et al. (2020) amount to days (see appendix of that work), which is hard to scale to online inference. Extending our approach with a more advanced rendering method is the subject of future work.

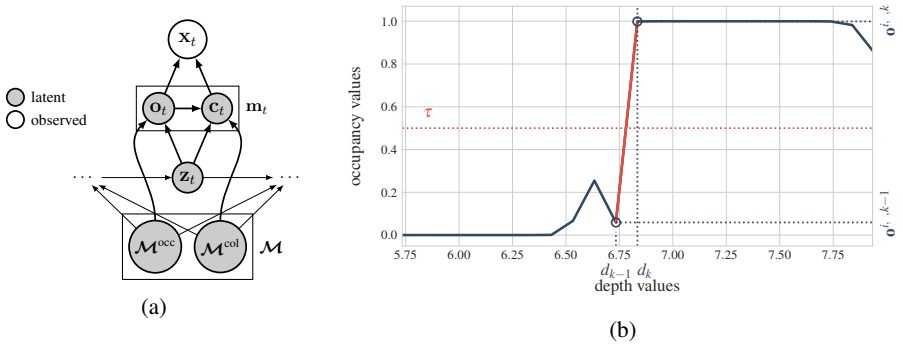
Bayesian SLAM inference To keep exposition brief, we refer to (Cadena et al., 2016) for an overview of modern SLAM inference and focus only on approaches that have applied fully-Bayesian methods to SLAM. The inference in this work can be categorised as probabilistic SLAM, other prominent examples of which are FastSLAM (Montemerlo et al., 2002) and RBPF SLAM (Grisetti et al., 2005). What distinguishes our method is the application of variational inference with SGVB (Kingma & Welling, 2014). Our model does not restrict the used distributions and allows any differentiable functional form, which enables us to use neural networks. The contribution by Murphy (Murphy, 1999) is one of the first to infer a global map with Bayesian methods. Bayesian Hilbert maps (Senanayake & Ramos, 2017) focus on a fully Bayesian treatment of Hilbert maps for long-term mapping in dynamic environments. Stochastic variational inference is used to infer agent poses from observed 2D image features in (Jiang et al., 2017; Jiang et al., 2019). DVBF-LM (Mirchev et al., 2019) uses Bayes by Backprop (Blundell et al., 2015) for the inference of the global map variable.

3 METHOD

Background We adhere to the graphical model of DVBF-LM (Mirchev et al., 2019), but we introduce novel design choices for every model component and implement the overall inference differently, to allow for real-world 3D modelling. In the following, we will first describe the assumed factorisation and then explain the introduced modifications. The assumed joint distribution of all variables is:

$$
\begin{aligned}
&p(\mathbf{x}_{1:T}, \mathbf{z}_{1:T}, \mathbf{m}_{1:T}, \boldsymbol{\mathcal{M}} \mid \mathbf{u}_{1:T-1}) \\
&\quad = p(\boldsymbol{\mathcal{M}})\rho(\mathbf{z}_1) \prod_{t=1}^{T} p(\mathbf{x}_t \mid \mathbf{m}_t) p(\mathbf{m}_t \mid \mathbf{z}_t, \boldsymbol{\mathcal{M}}) \prod_{t=1}^{T-1} p(\mathbf{z}_{t+1} \mid \mathbf{z}_t, \mathbf{u}_t),
\end{aligned} \tag{1}
$$

where $\mathbf{x}_{1:T}$ are observations, $\mathbf{z}_{1:T}$ agent states, $\mathbf{m}_{1:T}$ map charts and $\mathbf{u}_{1:T-1}$ conditional inputs (controls). The factorisation defines a traditional state-space model extended with a global map variable $\boldsymbol{\mathcal{M}}$. For a single step t, an observation $\mathbf{x}_t$ is generated from a map chart $\mathbf{m}_t$—the relevant extract from the global $\boldsymbol{\mathcal{M}}$ around the current agent pose $\mathbf{z}_t$ (cf. fig. 2a). Chart extraction is given by $p(\mathbf{m}_t \mid \mathbf{z}_t, \boldsymbol{\mathcal{M}})$, which can be seen as an attention mechanism. In this graphical model, SLAM is equivalent to inference of the agent states $\mathbf{z}_{1:T}$ and the map $\boldsymbol{\mathcal{M}}$. For the remainder of this work, we assume all observations $\mathbf{x}_t \in \mathbb{R}^{w \times h \times 4}$ are RGB-D images. Next, we will describe the functional forms of the map $\boldsymbol{\mathcal{M}}$, the attention $p(\mathbf{m}_t \mid \mathbf{z}_t, \boldsymbol{\mathcal{M}})$, the emission $p(\mathbf{x}_t \mid \mathbf{m}_t)$ and the states $\mathbf{z}_{1:T}$.



(a) (b)

Figure 2: (a) One time step of the proposed probabilistic graphical model. (b) Linear interpolation during ray casting for a single ray in the emission model. d_k is the depth corresponding to the first ray value that exceeds τ. The output depth d is formed by linearly interpolating between d_{k-1} and d_k based on the occupancy values $\mathbf{o}^{i,j,k-1}$ and $\mathbf{o}^{i,j,k}$.

Geometric map The map random variable $\boldsymbol{\mathcal{M}} = (\boldsymbol{\mathcal{M}}^{\text{occ}}, \boldsymbol{\mathcal{M}}^{\text{col}})$ consists of two components. $\boldsymbol{\mathcal{M}}^{\text{occ}} \in \mathbb{R}^{l \times m \times n}$ is a spatially arranged 3D grid of scalar values that represent occupancy. $\boldsymbol{\mathcal{M}}^{\text{col}}$ represents the parameters of a feed-forward neural network $f_{\boldsymbol{\mathcal{M}}^{\text{col}}} : \mathbb{R}^3 \to [0, 255]^3$. The network assigns an RGB colour value to each point in space. In this work, the network weights are deterministic and point-estimated via maximum likelihood, the fully-Bayesian treatment of the colour map is left for future work. The prior and approximate posterior distributions over the occupancy map are:

$$p(\boldsymbol{\mathcal{M}}^{\text{occ}}) = \prod_{i,j,k} \mathcal{N}(\boldsymbol{\mathcal{M}}^{\text{occ}}_{i,j,k} \mid 0, 1), \quad q_{\boldsymbol{\phi}}(\boldsymbol{\mathcal{M}}^{\text{occ}}) = \prod_{i,j,k} \mathcal{N}(\boldsymbol{\mathcal{M}}^{\text{occ}}_{i,j,k} \mid \mu_{i,j,k}, \sigma^2_{i,j,k}).$$

Here and for the rest of this work $q_{\boldsymbol{\phi}}$ will denote a variational approximate posterior distribution, with all its optimisable parameters summarised in $\boldsymbol{\phi}$. We assume $p(\boldsymbol{\mathcal{M}}_{\text{occ}})$ and $q_{\boldsymbol{\phi}}(\boldsymbol{\mathcal{M}}_{\text{occ}})$ factorise over grid cells. The variational parameters $\mu_{i,j,k}, \sigma_{i,j,k}$ are optimised with Bayes by Backprop (Blundell et al., 2015).

Attention In the proposed model, the composition of the attention $p(\mathbf{m}_t \mid \mathbf{z}_t, \boldsymbol{\mathcal{M}})$ and the emission $p(\mathbf{x}_t \mid \mathbf{m}_t)$ implements volumetric raycasting. We engineer them based on our understanding of geometry to ensure generalisation across unseen environments. The attention $p(\mathbf{m}_t \mid \mathbf{z}_t, \boldsymbol{\mathcal{M}})$ forms latent charts $\mathbf{m}_t$, which correspond to extracts from the map $\boldsymbol{\mathcal{M}}$ around $\mathbf{z}_t$. We identify $\mathbf{m}_t$ with the part of the map contained in the frustum of the current camera view. To attend to that region, first the intrinsic camera matrix $\mathbf{K}$ (assumed to be known) and the agent pose $\mathbf{z}_t$ are used to cast a ray for any pixel $[i, j]^T$ in the reconstructed observation. The ray is then discretised equidistantly along the depth dimension into r-many points, resulting into a collection of 3D world coordinates $\mathbf{p}_t \in \mathbb{R}^{w \times h \times r \times 3}$. Depth candidate values $d \in \{k\epsilon\}_{1 \leq k \leq r}$ are associated with each point along a ray, where ϵ is a resolution hyperparameter. The latent chart $\mathbf{m}_t = (\mathbf{o}_t, \mathbf{c}_t)$ factorises into an occupancy chart $\mathbf{o}_t \in \mathbb{R}^{w \times h \times r}$ and a colour chart $\mathbf{c}_t \in \mathbb{R}^{w \times h \times r \times 3}$. Let $p_t^{ijk} \in \mathbb{R}^3$ be a 3D point in the spanned camera frustum. To form the occupancy chart $\mathbf{o}_t$, cells from the map $\boldsymbol{\mathcal{M}}^{occ}$ around p_t^{ijk} are combined with a weighted kernel $o_t^{ijk} = \sum_{l,h,s} \boldsymbol{\mathcal{M}}^{\text{occ}}_{l,h,s} \alpha_{l,h,s}(p^{ijk})$. Note that here l, h, s are indices of the occupancy map voxels. We choose a trilinear interpolation kernel for α, merging only eight map cells per point. This makes the attention fast and differentiable w.r.t $\mathbf{z}_t$. The colour chart $\mathbf{c}_t = f_{\boldsymbol{\mathcal{M}}^{\text{col}}}(\mathbf{p}_t)$ is formed by applying $f_{\boldsymbol{\mathcal{M}}^{\text{col}}}$, the colour neural network, *point-wise* to each 3D point. In this work, we keep the chart $\mathbf{m}_t$ deterministic. The full attention procedure can be described as:

$$p(\mathbf{m}_t \mid \mathbf{z}_t, \boldsymbol{\mathcal{M}}) = \prod_{ijk} \delta(\mathbf{m}_t^{ijk} = f_A(\boldsymbol{\mathcal{M}}, p^{ijk})), \quad p^{ijk} = \mathbf{T}(\mathbf{z}_t)\mathbf{K}^{-1}[i, j, 1]^T \underbrace{d}_{:=k\epsilon}.$$

Here $\mathbf{T}(\mathbf{z}_t) \in \mathbb{SE}(3)$ denotes the rigid camera transformation defined by the current agent state $\mathbf{z}_t$ and i, j, k index the points lying inside the attended camera frustum.

Emission through ray casting The emission model factorises over the observed pixels:

$$p(\mathbf{x}_t \mid \mathbf{m}_t) = \prod_{ij} p(\mathbf{x}_t^{ij} \mid \mathbf{m}_t), \quad p(\mathbf{x}_t^{ij} \mid \mathbf{m}_t) = p(d_t^{ij}, \tilde{\mathbf{c}}_t^{ij} \mid \mathbf{o}_t, \mathbf{c}_t).$$

It operates on the extracted chart $\mathbf{m}_t = (\mathbf{o}_t, \mathbf{c}_t)$. Here $\mathbf{x}_t^{ij} \in \mathbb{R}^4$ denotes an RGB-D pixel value, i.e. for each pixel $[i, j]^T$ we reconstruct a depth d_t^{ij} and a colour value $\tilde{\mathbf{c}}_t^{ij}$. The mean of the depth value d_t^{ij} is formed by a function f_E:

$$f_E(\mathbf{o}_t)^{ij} = \epsilon \cdot \min_{k \in [r]} k \quad \text{s.t.} \quad \mathbf{o}_t^{ijk} > \tau.$$

f_E traces the ray for pixel $[i, j]^T$, searching for the minimum depth $d = \epsilon k$ for which the occupancy value $\mathbf{o}_t^{ijk}$ exceeds a threshold τ (a hyperparameter).[1] Since the above $\min$ operation is not differentiable in $\mathbf{o}_t$, we linearly interpolate between the depth value for the first ray hit and its predecessor to form the mean of the emitted depth (cf. fig. 2b):

$$\mu_{d_t}^{ij} = \alpha f_E(\mathbf{o}_t)^{ij} + (1 - \alpha)(f_E(\mathbf{o}_t)^{ij} - \epsilon), \quad \alpha = \frac{\tau - \mathbf{o}_t^{i,j,k-1}}{\mathbf{o}_t^{i,j,k} - \mathbf{o}_t^{i,j,k-1}}.$$

The mean of the emitted colour $\boldsymbol{\mu}_{\tilde{\mathbf{c}}_t}^{ij} = \mathbf{c}_t^{ijk}$ directly corresponds to the k-th element of the attended colour values, where k is the index of the first hit from raycasting above. A heteroscedastic Laplace distribution is assumed for both the emitted depth and colour values:

$$p(\mathbf{x}_t^{ij} \mid \mathbf{m}_t) = \text{Laplace}(\mathbf{x}_t^{ij}; (\mu_{d_t}^{ij}, \boldsymbol{\mu}_{\tilde{\mathbf{c}}_t}^{ij}), \text{diag}(\boldsymbol{\sigma}_E^{ij})).$$

Agent states All agent states are represented as vectors $\mathbf{z}_t = (\boldsymbol{\lambda}_t, \boldsymbol{\omega}_t, \mathbf{z}_t^{\text{rest}}) \in \mathbb{R}^{d_{\mathbf{z}}}$. $\boldsymbol{\lambda}_t \in \mathbb{R}^3$ is the agent location in space. $\boldsymbol{\omega}_t \in \mathbb{H}^4$ is the agent orientation, represented as a quaternion. $\mathbf{z}_t^{\text{rest}} \in \mathbb{R}^{d_{\mathbf{z}}-7}$ is a remainder. Depending on the used transition model, $\mathbf{z}_t^{\text{rest}}$ can be $\dot{\boldsymbol{\lambda}}_t$ alone or it can contain an abstract latent portion not explicitly matching physical quantities. The approximate posterior variational family over the agent states factorises over time:

$$q_{\boldsymbol{\phi}}(\mathbf{z}_{1:T}) = \prod_t q_{\boldsymbol{\phi}}(\mathbf{z}_t) = \prod_t \mathcal{N}(\mathbf{z}_t \mid \boldsymbol{\mu}_t^{\mathbf{z}}, \text{diag}(\boldsymbol{\sigma}_t^{\mathbf{z}})^2).$$

Here $\boldsymbol{\mu}_t^{\mathbf{z}} \in \mathbb{R}^{d_{\mathbf{z}}}$ and $\boldsymbol{\sigma}_t^{\mathbf{z}} \in \mathbb{R}^{d_{\mathbf{z}}}$ are free variables for each latent state and are optimised with SGVB (Kingma & Welling, 2014). Notably, the above factorisation over states bears similarity to pose-graph optimisation. One can see the individual terms $q_{\boldsymbol{\phi}}(\mathbf{z}_t)$ as graph nodes, and the loss terms induced by the transition and emission in the objective presented next as the edge constraints.

Overall objective The elements described so far, together with the transition $p(\mathbf{z}_{t+1} \mid \mathbf{z}_t, \mathbf{u}_t)$ discussed in the next section, form the probabilistic graphical model in eq. (1). The assumed variational approximate posterior is

$$q_{\boldsymbol{\phi}}(\mathbf{z}_{1:T}) q_{\boldsymbol{\phi}}(\boldsymbol{\mathcal{M}}) \approx p(\mathbf{z}_{1:T}, \boldsymbol{\mathcal{M}} \mid \mathbf{x}_{1:T}, \mathbf{u}_{1:T-1}).$$

For the optimisation objective we use the negative *evidence lower bound* (ELBO) (Jordan et al., 1999), given as

$$\begin{aligned} \mathcal{L}_{\text{elbo}} = &-\mathbb{E}_q \left[\sum_{t=1}^{T} \log p(\mathbf{x}_t \mid \mathbf{m}_t) \right] \\ &+ \text{KL}(q_{\boldsymbol{\phi}}(\boldsymbol{\mathcal{M}}) \,||\, p(\boldsymbol{\mathcal{M}})) + \mathbb{E}_q \left[\sum_{t=2}^{T} \text{KL}(q_{\boldsymbol{\phi}}(\mathbf{z}_t) \,||\, p(\mathbf{z}_t \mid \mathbf{z}_{t-1}, \mathbf{u}_{t-1})) \right]. \end{aligned} \tag{2}$$

We employ the approximate particle optimisation scheme from (Mirchev et al., 2019) to deal with long data sequences. The only optimised parameters are $\boldsymbol{\phi}$, containing the parameters of the map and the agent states.

Making image reconstruction tractable Using the full observations during inference is not feasible, as raycasting for all pixels is too computationally demanding. To ensure tractability of the inference method we therefore use reconstruction sampling (Dauphin et al., 2011), emitting a random part of $\mathbf{x}_t$ at a time, by randomly selecting c-many pixel coordinates $[i, j]^T$ for every gradient step. Here c is a constant much smaller than the image size wh, speeding up gradient updates by a few orders of magnitude. Note that this results in an unbiased, faster and more memory-efficient Monte Carlo approximation of the original objective, avoiding loss of information due to subsampling or sparse feature selection.

[1] $\mathbf{o}_t^{ijk}$ is set to 0 for $k \leq 1$ and $f_E(\mathbf{o}_t) = r\epsilon$ if no value exceeds τ along the ray.

4 LEARNING RIGID-BODY DYNAMICS

The introduced model factorisation includes a transition $p(\mathbf{z}_{t+1} \mid \mathbf{z}_t, \mathbf{u}_t)$, which allows the natural inclusion of agent movement priors. This is reflected in the corresponding KL terms in eq. (2). Note that using variational inference lets us integrate any differentiable transition model as-is, without additional linearisation. In the following, we assume the agent has an inertial measurement unit (IMU) providing readings $\ddot{\boldsymbol{\lambda}}_t^{\text{imu}}$ (linear acceleration) and $\dot{\boldsymbol{\omega}}_t^{\text{imu}}$ (angular velocity) over time, which we choose to treat as conditional inputs $\mathbf{u}_t = (\ddot{\boldsymbol{\lambda}}_t^{\text{imu}}, \dot{\boldsymbol{\omega}}_t^{\text{imu}})$.

Engineering rigid-body dynamics In the absence of learning, one can use an engineered transition prior that integrates the IMU sensor readings over time. The latent state $\mathbf{z}_t = (\boldsymbol{\lambda}_t, \boldsymbol{\omega}_t, \dot{\boldsymbol{\lambda}}_t)$ then contains the location, orientation and linear velocity of the agent at every time step. The transition is defined as:

$$p(\mathbf{z}_{t+1} \mid \mathbf{z}_t, \mathbf{u}_t) = \mathcal{N}(\mathbf{z}_{t+1} \mid f_T(\mathbf{z}_t, \mathbf{u}_t), \operatorname{diag}(\boldsymbol{\sigma}_T)^2).$$

The state update f_T implements standard rigid-body dynamics using Euler integration (see appendix D.3). This engineered model will serve as a counterpart for the learned transition model presented next.

Learning a dynamics model Engineered models of the agent movement are often imperfect or not available. We therefore provide a method for learning a fully-probabilistic transition model from streams of prerecorded controls and agent pose observations, which we can then seamlessly include as a prior in the full model. We do not learn the transition with per-step, fully-supervised regression. Instead we formulate a generative sequence model for T time steps. This allows us to separate the aleatoric uncertainty in the observed agent states from the uncertainty in the transition itself. We follow the literature on variational state-space models (Fraccaro et al., 2016; Karl et al., 2017a). We assume we have a sequence of locations $\hat{\boldsymbol{\lambda}}_{1:T}$ and orientations $\hat{\boldsymbol{\omega}}_{1:T}$ as observations, and a sequence of IMU readings, as well as per-rotor revolutions per minute (RPM) and pulse-width modulation (PWM) signals, as conditional inputs $\mathbf{u}_{1:T-1} = (\ddot{\boldsymbol{\lambda}}_{1:T-1}^{\text{imu}}, \dot{\boldsymbol{\omega}}_{1:T-1}^{\text{imu}}, \mathbf{u}_{1:T-1}^{\text{rpm}}, \mathbf{u}_{1:T-1}^{\text{pwm}})$. We define the generative state-space model:

$$p(\hat{\boldsymbol{\lambda}}_{1:T}, \hat{\boldsymbol{\omega}}_{1:T}, \mathbf{z}_{1:T} \mid \mathbf{u}_{1:T-1}) = \delta(\mathbf{z}_1) p(\hat{\boldsymbol{\lambda}}_1, \hat{\boldsymbol{\omega}}_1 \mid \mathbf{z}_1) \prod_{t=1}^{T-1} p_{\boldsymbol{\theta}_T}(\mathbf{z}_{t+1} \mid \mathbf{z}_t, \mathbf{u}_t) p(\hat{\boldsymbol{\lambda}}_{t+1}, \hat{\boldsymbol{\omega}}_{t+1} \mid \mathbf{z}_{t+1}).$$

The objective is to learn generative transition parameters $\boldsymbol{\theta}_T$, such that the marginal likelihood of observed agent poses $p_{\boldsymbol{\theta}_T}(\hat{\boldsymbol{\lambda}}_{1:T}, \hat{\boldsymbol{\omega}}_{1:T} \mid \mathbf{u}_{1:T-1})$ is maximised. The latent state is $\mathbf{z}_t = (\boldsymbol{\lambda}_t, \boldsymbol{\omega}_t, \dot{\boldsymbol{\lambda}}_t, \mathbf{z}_t^{\text{rest}})$, identifying its first three components with location, orientation and linear velocity. The remainder $\mathbf{z}_t^{\text{rest}}$ acts as an abstract state part. Its role is to absorb any quantities that might affect the transition, for example higher moments of the dynamics or sensor biases accumulated over previous time steps. The transition is implemented as a residual neural network on top of Euler integration:

$$p_{\boldsymbol{\theta}_T}(\mathbf{z}_{t+1} \mid \mathbf{z}_t, \mathbf{u}_t) = \mathcal{N}(\mathbf{z}_{t+1} \mid \boldsymbol{\mu}_{t+1}, \operatorname{diag}(\boldsymbol{\sigma}_{t+1})^2)$$
$$\boldsymbol{\mu}_{t+1} = \begin{bmatrix} f_T(\mathbf{z}_t, \mathbf{u}_t) \\ \mathbf{0} \end{bmatrix} + \text{MLP}_{\boldsymbol{\mu}}(\mathbf{z}_t, \mathbf{u}_t), \quad \boldsymbol{\sigma}_{t+1} = \text{MLP}_{\boldsymbol{\sigma}}(\mathbf{z}_t, \mathbf{u}_t),$$

where f_T is the engineered Euler integration from the previous section and the abstract remainder of the latent state is formed entirely by the network (MLP). This strong inductive bias shapes the transition to resemble regular integration in the beginning of training, exploiting engineering knowledge, while still allowing the MLP to eventually take over and correct biases as necessary.

The emission isolates the location and orientation from the latent state as its mean:

$$p(\hat{\boldsymbol{\lambda}}_t, \hat{\boldsymbol{\omega}}_t \mid \mathbf{z}_t) = \mathcal{N}(\hat{\boldsymbol{\lambda}}_t, \hat{\boldsymbol{\omega}}_t \mid (\boldsymbol{\lambda}_t, \boldsymbol{\omega}_t), \operatorname{diag}(\boldsymbol{\sigma})^2).$$

The inference over the latent states uses Gaussian fusion as per (Karl et al., 2017b) and the necessary inverse emission is given by a bidirectional RNN that looks into all observations and conditions:

$$\hat{q}(\mathbf{z}_t \mid \hat{\boldsymbol{\lambda}}_{1:T}, \hat{\boldsymbol{\omega}}_{1:T}, \mathbf{u}_{1:T-1}) = \mathcal{N}(\mathbf{z}_t \mid \text{RNN}(\hat{\boldsymbol{\lambda}}_{1:T}, \hat{\boldsymbol{\omega}}_{1:T}, \mathbf{u}_{1:T-1})).$$

We minimise the negative ELBO w.r.t. $\boldsymbol{\theta}_T$, omitting the conditions in q for brevity:

$$\mathcal{L}(\boldsymbol{\theta}_T) = -\mathbb{E}_q \left[\sum_{t=1}^{T} \log p(\hat{\boldsymbol{\lambda}}_t, \hat{\boldsymbol{\omega}}_t \mid \mathbf{z}_t) \right] + \mathbb{E}_q \left[\sum_{t=2}^{T} \mathrm{KL}(q(\mathbf{z}_t) \mid\mid p_{\boldsymbol{\theta}_T}(\mathbf{z}_t \mid \mathbf{z}_{t-1}, \mathbf{u}_{t-1})) \right].$$

5 Experiments

The experiments are designed to validate three model aspects—the usefulness of the reconstructed 3D world maps, multi-step prediction given future controls and the localisation quality.

For evaluation, we use the Blackbird data set (Antonini et al., 2020). It consists of over ten hours of real quadcopter flight data. The ground truth poses are recorded by a motion capture (MOCAP) system. For each trajectory, Blackbird contains realistic simulated stereo images. We obtain depth from these images using OpenCV's Semi-Global Block Matching (SGBM) (Hirschmuller, 2007) and treat the left RGB camera image and the estimated depth as an observation $\mathbf{x}_t$. We evaluate our method on the test trajectories used in (Nisar et al., 2019). Other trajectories with no overlap are used for training the learned dynamics model and model selection. All model hyperparameters are fixed to the same values for all evaluations. More details can be found in appendices C and D.

5.1 Dense geometric mapping

A fused dense map, obtained as an approximate variational posterior $q_{\boldsymbol{\phi}}(\boldsymbol{\mathcal{M}})$, allows us to simulate (emit) the environment from any novel view point. We take the *NYC subway station* Blackbird environment as an example, in which the test set trajectories take place. Figure 3a shows image reconstructions, generated along an example trajectory segment. The model can successfully generate both colour and depth based on $q_{\boldsymbol{\phi}}(\boldsymbol{\mathcal{M}})$. Note that the true observations are not needed for this, as all of the information is recorded in $\boldsymbol{\mathcal{M}}$ through gradient descent. Even though we use reconstruction sampling during training, on average all image pixels contribute to learning the map. This leads to dense predictions of the agent's sensors. The inferred map correctly filters out wrong observations, as can be seen in the top-down point-cloud comparison in fig. 4d, noting the subway station columns.

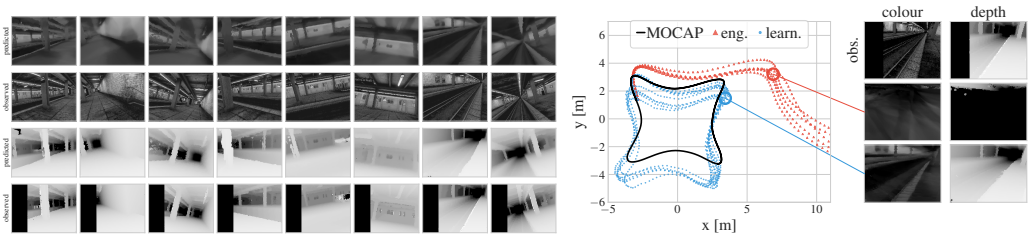
5.2 Using maps for downstream tasks

Besides parameterising predictive emissions, a map can be used to define downstream navigation and exploration tasks. Since the map is modelled as a random variable, the approximate posterior $q_{\boldsymbol{\phi}}(\boldsymbol{\mathcal{M}})$ also gives us an uncertainty estimate, in this work only applying to $\boldsymbol{\mathcal{M}}^{occ}$. Figure 4a shows a horizontal slice from the occupancy grid, along which the map uncertainty is evaluated. The uncertainty is low inside the subway station, and high on the outside where the agent has not visited. Meaningful map uncertainty is useful for information-theoretic exploration (Mirchev et al., 2019).

The occupancy map can also be used to construct navigation plans, by computing a collision cost-to-go $J(\mathbf{z}_1) = \mathbb{E}_{\mathbf{u}_{1:T}, \mathbf{z}_{1:T} \sim p(\cdot)}[\sum_t c(\mathbf{u}_t, \mathbf{z}_t)]$ (Bertsekas, 2005). For example, the cost $c(\mathbf{u}, \mathbf{z})$ can be the occupancy value in the map at $\mathbf{z}_t$, defining a collision cost. To simulate a control policy, we use an empirical distribution of randomly picked 40-step sequences of IMU readings $\mathbf{u}_{1:T}$ from the test data. Then we generatively sample future states $\mathbf{z}_{1:T}$ for these controls and evaluate the respective costs. Figure 4c shows $J(\mathbf{z})$ evaluated for all states along the considered 2D slice of the map. The cost-to-go is high near the walls and columns, and gradually drops off in the free space.

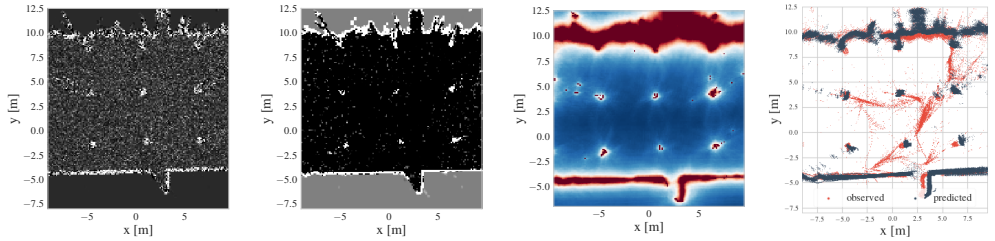
5.3 Generating future predictions

The proposed model can predict the agent movement and observations for a sequence of future conditional inputs, i.e. $p(\mathbf{z}_{2:T}, \mathbf{x}_{1:T} \mid \mathbf{u}_{1:T-1}, \mathbf{z}_1)$. Such predictions are crucial for model-based control, and typically not readily available from modern visual SLAM systems. Figure 3b shows predictions of the full spatial model for 200 steps in the future, comparing both the engineered and learned transition priors. Long-term prediction is significantly better with the learned transition, indicating that it corrects biases present in the agent sensors. Conversely, with the engineered transition localisation drift is higher and wrong map regions are queried for generation, translating into wrong depth and colour predictions. We refer to the supplementary video for more examples using the predictive model. Evaluated on 200 test set trajectories with 100 steps, the learned transition



(a) (b)

Figure 3: (a) Emissions at different trajectory points, sampled at a one second interval. Top to bottom: predicted colour, observed colour, predicted depth, observed depth. (b) Generative predictions using the engineered transition vs. the learned transition in the complete model. Left: top-down view of 200-step location predictions. Right: predicted colour and depth for the same step for both models.



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

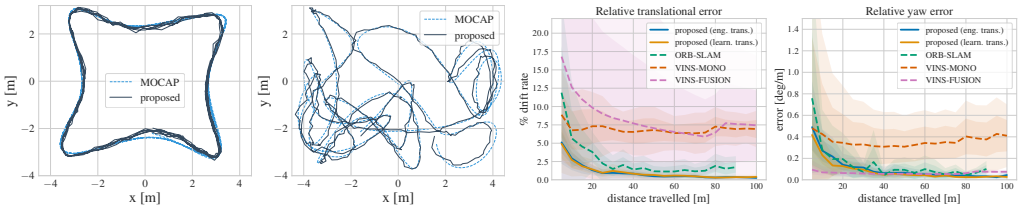
Figure 4: An illustration of a reconstructed dense map for the *NYC subway station* environment. Note that columns from the subway are captured in the reconstructed map. The figures show a horizontal map slice. (a) Occupancy map uncertainty (white means low uncertainty). (b) Occupancy map mean (white means occupied). (c) Collision cost-to-go (red means high cost). (d) Generated point cloud (black, for inferred agent poses) vs. data point cloud (red, for ground truth MOCAP poses).

leads to better predictions than its engineered counterpart, with an average translational RMSE of 1.156 and rotational RMSE of 0.096, compared to 13.768 and 0.111 for the engineered model. Similarly, the pixel-wise log-likelihood of the observation predictions is -1.23 when the learned model is used, and -1.85 for the engineered model, averaged over 1000 images. This evaluation clearly illustrates the positive effect of the learned transition on the model's predictive performance.

5.4 Agent Localisation

Finally, we evaluate localisation performance during SLAM inference. Figure 5 shows estimates for the fastest trajectories in the test set (up to $4m/s$). We refer to appendix F and the supplementary video for more inference examples in different environments. Table 1 summarises the average absolute RMSE for all test trajectories, evaluated following Zhang & Scaramuzza (2018). Here we compare to the results reported by Nisar et al. (2019), including VIMO (Nisar et al., 2019) and VINS-MONO (Qin et al., 2018)—two state-of-the art visual odometry methods. In this case both systems are carefully tuned to the environment. While the RMSE of our method is not as low as that of the baselines, we note that these are absolute values. The online translational error of our method does not exceed $0.4m$, which is practical considering the average $232m$ trajectory length. Our method also succeeds on the fastest *star* test trajectory, for which both baselines have been reported to fail without specific retuning. We note that the two systems run in real-time, whereas currently our method does not—we will tackle real-time inference in our future work (see appendix E).

We also compare to the system benchmarks provided by Antonini et al. (2020), including results for VINS-MONO, VINS-Fusion (Qin et al., 2019) and ORB-SLAM2 (Mur-Artal & Tardós, 2017). We use the same *star* and *picasso* trajectories reported in table 1, and refer to Antonini et al. (2020) for the exact evaluation assumptions. We also explicitly restate a note made by the authors: the benchmarked systems are not carefully tuned to the Blackbird environment and their loop-closure



(a) (b) (c)

Figure 5: (a,b) Top-down view of localisation for *star, forward yaw, max speed* $4.0m/s$ and *picasso, constant yaw, max speed* $4.0m/s$. (c) Localisation with our method compared to the benchmark results reported by Antonini et al. (2020) for *picasso, constant yaw* and *star, forward yaw*, 1 to $4m/s$.

Table 1: Absolute localisation RMSE for each test trajectory. See appendix B for a discussion of the two noticeable translation RMSE outliers with the learned transition.

		Translational RMSE [m]				**Rotational RMSE [rad]**			
		proposed (eng. trans.)	proposed (learn. trans)	VIMO	VINS	proposed (eng. trans.)	proposed (learn. trans.)	VIMO	VINS
picasso	1 $^{m}/_{s}$	0.139	0.143	0.055	0.097	0.053	0.052	0.013	0.011
	2 $^{m}/_{s}$	0.136	0.131	0.040	0.043	0.069	0.064	0.007	0.008
	3 $^{m}/_{s}$	0.120	0.122	0.043	0.045	0.073	0.070	0.005	0.005
	4 $^{m}/_{s}$	0.174	0.368	0.049	0.056	0.124	0.149	0.009	0.011
star	1 $^{m}/_{s}$	0.137	0.133	0.088	0.102	0.057	0.056	0.008	0.008
	2 $^{m}/_{s}$	0.163	0.626	0.082	0.133	0.061	0.157	0.010	0.011
	3 $^{m}/_{s}$	0.281	0.187	0.183	0.235	0.080	0.059	0.015	0.016
	4 $^{m}/_{s}$	0.156	0.160	-	-	0.065	0.059	-	-

modules are disabled, which might not be reflective of their best possible performance. Therefore, these baselines represent the performance of an off-the-shelf visual odometry system without changing its hyperparameters. Figure 5c summarises the comparison, reporting error statistics for segments of different length (x-axis) divided by the distance travelled. Localisation with our method is robust and drift does not compound, which we attribute to the global map variable $\mathcal{M}$ serving as an anchor.

Overall, localisation is successful for all test trajectories and its accuracy is practical and close to that of state-of-the-art systems, while all merits of deep probabilistic generative modelling are retained.

6 Conclusion

This work is the first to show that learning a dense 3D map and 6-DoF localisation can be accomplished in a deep generative probabilistic framework using variational inference. The proposed spatial model features an expressive predictive distribution suitable for downstream control tasks, it is fully-differentiable and can be optimised end-to-end with SGD. We further propose a probabilistic method for learning agent dynamics from prerecorded data, which significantly boosts predictive performance when incorporated in the full model. The proposed framework was used to model quadcopter flight data, exhibiting performance close to that of state-of-the-art visual SLAM systems and bearing promise for real-world applications. In the future, we will address the current model's speed limitations and move towards downstream applications based on the learned representation.

References

Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*, pp. 265–283, 2016.

Amado Antonini, Winter Guerra, Varun Murali, Thomas Sayre-McCord, and Sertac Karaman. The blackbird uav dataset. *The International Journal of Robotics Research*, 0(0):0278364920908331, 2020. doi: 10.1177/0278364920908331.

Philip Becker-Ehmck, Maximilian Karl, Jan Peters, and Patrick van der Smagt. Learning to fly via deep model-based reinforcement learning, 2020.

Dimitri P. Bertsekas. *Dynamic Programming and Optimal Control*, volume I. Athena Scientific, Belmont, MA, USA, 3rd edition, 2005.

S. Bi, Zexiang Xu, Pratul P. Srinivasan, Ben Mildenhall, Kalyan Sunkavalli, Milovs Havsan, Yannick Hold-Geoffroy, D. Kriegman, and R. Ramamoorthi. Neural reflectance fields for appearance acquisition. *ArXiv*, abs/2008.03824, 2020.

Michael Bloesch, Jan Czarnowski, Ronald Clark, Stefan Leutenegger, and Andrew J. Davison. Codeslam - learning a compact, optimisable representation for dense visual slam. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2560–2568, 2018.

Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural networks. *CoRR*, abs/1505.05424, 2015.

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

Devendra Singh Chaplot, Emilio Parisotto, and Ruslan Salakhutdinov. Active neural localization. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=ry6-G_66b.

Silvia Chiappa, Sébastien Racanière, Daan Wierstra, and Shakir Mohamed. Recurrent environment simulators. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. URL https://openreview.net/forum?id=B1s6xvqlx.

Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. In *Advances in Neural Information Processing Systems*, pp. 4754–4765, 2018.

Junyoung Chung, Kyle Kastner, Laurent Dinh, Kratarth Goel, Aaron C. Courville, and Yoshua Bengio. A recurrent latent variable model for sequential data. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pp. 2980–2988, 2015.

Dane Corneil, Wulfram Gerstner, and Johanni Brea. Efficient model-based deep reinforcement learning with variational state tabulation. volume 80 of *Proceedings of Machine Learning Research*, pp. 1049–1058, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR.

Sebastian Curi, Silvan Melchior, Felix Berkenkamp, and Andreas Krause. Structured variational inference in partially observable unstablegaussian process state space models. volume 120 of *Proceedings of Machine Learning Research*, pp. 147–157, The Cloud, 10–11 Jun 2020. PMLR.

J Czarnowski, T Laidlow, R Clark, and AJ Davison. Deepfactors: Real-time probabilistic dense monocular slam. *IEEE Robotics and Automation Letters*, 5:721–728, 2020. doi: 10.1109/lra.2020.2965415. URL http://dx.doi.org/10.1109/lra.2020.2965415.

Yann N. Dauphin, Xavier Glorot, and Yoshua Bengio. Large-scale learning of embeddings with reconstruction sampling. In *Proceedings of the 28th International Conference on Machine Learning, ICML 2011, Bellevue, Washington, USA, June 28 - July 2, 2011*, pp. 945–952, 2011.

J. Engel, V. Koltun, and D. Cremers. Direct sparse odometry. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, March 2018.

Martin Engelcke, Adam R. Kosiorek, Oiwi Parker Jones, and Ingmar Posner. Genesis: Generative scene inference and sampling with object-centric latent representations. In *International Conference on Learning Representations*, 2020.

S. M. Ali Eslami, Danilo Jimenez Rezende, Frederic Besse, Fabio Viola, Ari S. Morcos, Marta Garnelo, Avraham Ruderman, Andrei A. Rusu, Ivo Danihelka, Karol Gregor, David P. Reichert, Lars Buesing, Theophane Weber, Oriol Vinyals, Dan Rosenbaum, Neil Rabinowitz, Helen King, Chloe Hillier, Matt Botvinick, Daan Wierstra, Koray Kavukcuoglu, and Demis Hassabis. Neural scene representation and rendering. *Science*, 360(6394):1204–1210, 2018. doi: 10.1126/science.aar6170. URL https://science.sciencemag.org/content/360/6394/1204.

Marco Fraccaro, Søren Kaae Sønderby, Ulrich Paquet, and Ole Winther. Sequential neural models with stochastic layers. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pp. 2199–2207, 2016.

Marco Fraccaro, Danilo Jimenez Rezende, Yori Zwols, Alexander Pritzel, S. M. Ali Eslami, and Fabio Viola. Generative temporal models with spatial memory for partially observed environments. *CoRR*, abs/1804.09401, 2018.

Karol Gregor, Danilo Jimenez Rezende, Frederic Besse, Yan Wu, Hamza Merzic, and Aaron van den Oord. Shaping belief states with generative environment models for rl. In *Advances in Neural Information Processing Systems*, pp. 13475–13487, 2019.

Giorgio Grisetti, Cyrill Stachniss, and Wolfram Burgard. Improving grid-based slam with rao-blackwellized particle filters by adaptive proposals and selective resampling. *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, pp. 2432–2437, 2005.

David Ha and Jürgen Schmidhuber. Recurrent world models facilitate policy evolution. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 31*, pp. 2450–2462. Curran Associates, Inc., 2018.

Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. *arXiv preprint arXiv:1912.01603*, 2019a.

Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In *International Conference on Machine Learning*, pp. 2555–2565. PMLR, 2019b.

Heiko Hirschmuller. Stereo processing by semiglobal matching and mutual information. *IEEE Transactions on pattern analysis and machine intelligence*, 30(2):328–341, 2007.

Krishna Murthy Jatavallabhula, Ganesh Iyer, and Liam Paull. gradslam: Dense slam meets automatic differentiation, 2019.

X. Jiang, M. Hoy, H. Yu, and J. Dauwels. Linear-complexity stochastic variational bayes inference for slam. In *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, pp. 1–6, 2017.

Xiaoyue Jiang, Hang Yu, Michael Hoy, and Justin Dauwels. Robust linear-complexity approach to full slam problems: Stochastic variational bayes inference. In *2019 IEEE 90th Vehicular Technology Conference (VTC2019-Fall)*, pp. 1–5. IEEE, 2019.

Michael I Jordan, Zoubin Ghahramani, Tommi S Jaakkola, and Lawrence K Saul. An introduction to variational methods for graphical models. *Machine learning*, 37(2):183–233, 1999.

Maximilian Karl, Maximilian Soelch, Justin Bayer, and Patrick van der Smagt. Deep variational bayes filters: Unsupervised learning of state space models from raw data. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017a.

Maximilian Karl, Maximilian Soelch, Philip Becker-Ehmck, Djalel Benbouzid, Patrick van der Smagt, and Justin Bayer. Unsupervised real-time control through variational empowerment. *arXiv preprint arXiv:1710.05101*, 2017b.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. In *Proceedings of the 2nd International Conference on Learning Representations (ICLR)*, 2014.

T. Laidlow, J. Czarnowski, A. Nicastro, R. Clark, and S. Leutenegger. Towards the probabilistic fusion of learned priors into standard pipelines for 3d reconstruction. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 7373–7379, 2020. doi: 10.1109/ICRA40945.2020.9197001.

Chao Liu, Jinwei Gu, Kihwan Kim, Srinivasa G Narasimhan, and Jan Kautz. Neural rgb (r) d sensing: Depth and uncertainty from a video camera. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 10986–10995, 2019.

Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *ECCV*, 2020.

Atanas Mirchev, Baris Kayalibay, Maximilian Soelch, Patrick van der Smagt, and Justin Bayer. Approximate bayesian inference in spatial environments. In *Proceedings of Robotics: Science and Systems*, FreiburgimBreisgau, Germany, June 2019.

Michael Montemerlo, Sebastian Thrun, Daphne Koller, and Ben Wegbreit. Fastslam: A factored solution to the simultaneous localization and mapping problem. In *AAAI/IAAI*, 2002.

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

Kevin P. Murphy. Bayesian map learning in dynamic environments. In *Advances in Neural Information Processing Systems 12, [NIPS Conference, Denver, Colorado, USA, November 29 - December 4, 1999]*, pp. 1015–1021, 1999.

Richard A. Newcombe, Steven Lovegrove, and Andrew J. Davison. Dtam: Dense tracking and mapping in real-time. *2011 International Conference on Computer Vision*, pp. 2320–2327, 2011.

Barza Nisar, Philipp Foehn, Davide Falanga, and Davide Scaramuzza. Vimo: Simultaneous visual inertial model-based odometry and force estimation. In *Proceedings of Robotics: Science and Systems*, FreiburgimBreisgau, Germany, June 2019.

Junhyuk Oh, Valliappa Chockalingam, Satinder P. Singh, and Honglak Lee. Control of memory, active perception, and action in minecraft. In *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, pp. 2790–2799, 2016. URL http://jmlr.org/proceedings/papers/v48/oh16.html.

Emilio Parisotto and Ruslan Salakhutdinov. Neural map: Structured memory for deep reinforcement learning. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=Bk9zbyZCZ.

Tong Qin, Peiliang Li, and Shaojie Shen. Vins-mono: A robust and versatile monocular visual-inertial state estimator. *IEEE Transactions on Robotics*, 34(4):1004–1020, 2018.

Tong Qin, Jie Pan, Shaozu Cao, and Shaojie Shen. A general optimization-based framework for local odometry estimation with multiple sensors. *arXiv preprint arXiv:1901.03638*, 2019.

Ransalu Senanayake and Fabio Ramos. Bayesian hilbert maps for dynamic continuous occupancy mapping. In *Proceedings of the 1st Annual Conference on Robot Learning*, volume 78 of *Proceedings of Machine Learning Research*, pp. 458–471. PMLR, 13–15 Nov 2017.

Vincent Sitzmann, Julien Martel, Alexander Bergman, David Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. *Advances in Neural Information Processing Systems*, 33, 2020.

Chengzhou Tang and Ping Tan. BA-net: Dense bundle adjustment networks. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=B1gabhRcYX.

Dustin Tran, Matthew D. Hoffman, Dave Moore, Christopher Suter, Srinivas Vasudevan, Alexey Radul, Matthew Johnson, and Rif A. Saurous. Simple, distributed, and accelerated probabilistic programming. In *Neural Information Processing Systems*, 2018.

Xingkui Wei, Yinda Zhang, Zhuwen Li, Yanwei Fu, and Xiangyang Xue. Deepsfm: Structure from motion via deep bundle adjustment. In *ECCV*, 2020.

James Whittington, Timothy Muller, Shirely Mark, Caswell Barry, and Tim Behrens. Generalisation of structural knowledge in the hippocampal-entorhinal system. In *Advances in neural information processing systems*, pp. 8484–8495, 2018.

Nan Yang, Lukas von Stumberg, Rui Wang, and Daniel Cremers. D3VO: Deep depth, deep pose and deep uncertainty for monocular visual odometry. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

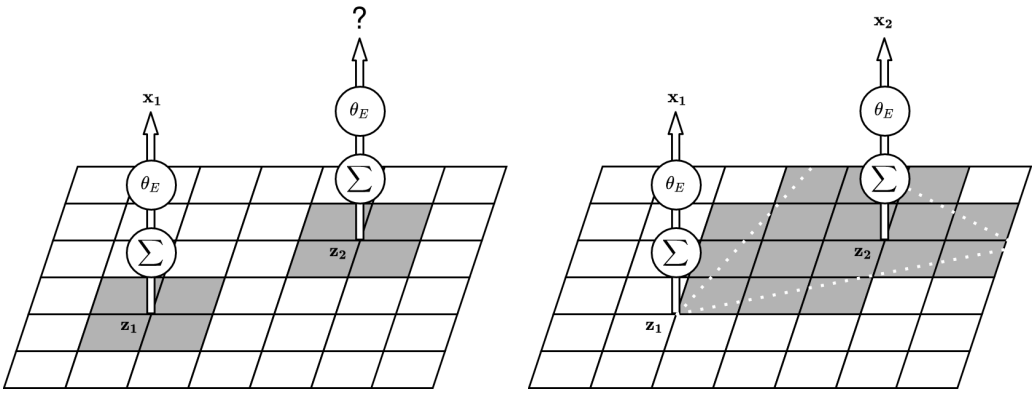
Jingwei Zhang, Lei Tai, Joschka Boedecker, Wolfram Burgard, and Ming Liu. Neural slam: Learning to explore with external memory. *arXiv preprint arXiv:1706.09520*, 2017.

Zichao Zhang and Davide Scaramuzza. A tutorial on quantitative trajectory evaluation for visual(-inertial) odometry. In *IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*, 2018.

Shuaifeng Zhi, Michael Bloesch, Stefan Leutenegger, and Andrew J Davison. Scenecode: Monocular dense semantic reconstruction using learned encoded scene representations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 11776–11785, 2019.

H. Zhou, B. Ummenhofer, and T. Brox. Deeptam: Deep tracking and mapping. In *European Conference on Computer Vision (ECCV)*, 2018.

A USE OF DOMAIN KNOWLEDGE WHEN DESIGNING LEARNED SPATIAL MODELS



(a) When the attention to the map is too narrow, emitting from observed but yet unvisited places suffers from non-local generalisation issues in an online setting.

(b) A 2D illustration of attending to the whole field of view of the agent. Note that in our method the occupancy map is a 3D volume, and the attended camera frustum resembles a pyramid.

Figure 6: Simplified 2D examples of the conceptual difference the chosen attention can have on generating observations.

What distinguishes our method from DVBF-LM (Mirchev et al., 2019) and other previous 2D and 2.5D deep generative sequence models is that we introduce geometric inductive biases (multiple-view geometry, rigid-body dynamics) in a deep generative sequence model (inferred via the ELBO). We hope the following overview of DVBF-LM's methodology will exemplify why such assumptions could be necessary and motivate our design choices.

DVBF-LM models the world as a 2D chessboard where the content of each grid cell describes a 360° horizontal panoramic view around the z-axis, centered at the agent 2D location. Observation predictions are cropped from that view, relative to the way the agent is facing in 2D (and transformed by an MLP). This already consitutes a set of basic geometric assumptions, but they can be limiting, as explained below. First of all, the agent movement is restricted to a plane, with rotations around one axis (no 6-DoF modelling). The main problem, however, is that the map is updated very locally, only at the 2D location of the agent, because the attention of DVBF-LM only considers four map cells directly underneath. For example, if the agent is standing 5m in front of a wall and facing it, DVBF-LM only stores this information at that location in the map. If the agent moves 2m closer to the wall, it is now accessing different cells, whose content is completely arbitrary (see fig. 6a). These cells will have to learn the presence of the wall again. The map is thus unnecessarily redundant and the agent would have to visit virtually all map cells to infill everything. This means observation predictions from cells which have not been visited before will not be meaningful, even when they were already in the field of view (FoV) of the agent. Localisation also suffers, as the map is more easily allowed to establish multiple modes for the pose posterior because of the redundancy (same content can be stored at different places by mistake), further complicating the perceptual aliasing problem of spatial environments.

By contrast, in our method we use raycasting and attend to the whole camera frustum, accessing all map content in the field of view of the agent, as shown in fig. 6b. Note that the depiction is intentionally simplified to 2D for the sake of clarity, while the actual attention in our approach operates in 3D. In the above example, the wall and the empty space in front of the agent would be captured as soon as the agent sees the wall for the first time, and can be predicted from all regions in the FoV. This constitutes a strong geometric inductive bias that will generalise across different environments.

In general, the lack of geometric assumptions in previous deep spatial generative models (e.g. the ones described in the first paragraph of section 2) is attractive, as it lessens the need for domain knowledge, instead attempting to learn the whole system end-to-end. However, learning everything

becomes problematic (not just in terms of runtime) when the inference needs to happen on-the-fly and data is scarce, as is the case for autonomous agents. When the agent cannot afford to exhaustively observe the whole environment, insufficient data for learning from scratch can lead to problems with consistency in the map, consistency in the long-term predictions and ambiguous agent localisation. One option is to address this through learning certain components in advance (e.g. as we do with the dynamics model, see section 4). Once image data is involved, however, the generalisation of learned components is much harder to ensure because of the curse of dimensionality. Observations can vary greatly from one scene to the next, and free 6-DoF movement of the agent exacerbates the problem further. Instead of collecting data exhaustively, in an attempt to pretrain a network to learn how to render, we rely on the geometric knowledge that is already available to us to design the map, the attention and the emission in section 3. Such assumptions are also at the core of many of the models discussed in the second paragraph in the related work (section 2). In contrast to these models, however, in our work we weave the aforementioned inductive biases into the deep generative framework, which is fully-probabilistic, implies end-to-end differentiability and eases the introduction of new learned components when needed. Therefore, our goal is to maintain an expressive predictive distribution $p(\mathbf{z}_{2:T}, \mathbf{x}_{1:T} \mid \mathbf{u}_{1:T-1}, \mathbf{z}_1)$ that lets us predict the future, precisely aligning with the performed inference based on past data at the same time. We stress that maintaining a full probabilistic predictive model is needed to quantify uncertainty when planning for future actions of the agent.

B CASE STUDY: DRONE LANDING

In terms of the integrated transition $p(\mathbf{z}_{t+1} \mid \mathbf{z}_t, \mathbf{u}_t)$, we find that the learned transition performs on par with its engineered counterpart when used for inference in the full model. However, when inspecting the absolute RMSE in table 1, there appear to be two outliers—*picasso, max speed 4.0 m/s* and *star, max speed 2.0 m/s*, when using the learned transition. A closer examination showed that this is entirely due to a large localisation error during landing at the end of the trajectories. In both cases the agent fails to land correctly or falls over, leading to out-of-distribution conditions $\mathbf{u}_t$ from the IMU sensor (cf. fig. 7), for which the learned model does not generalise. Such behaviour is not unexpected when it comes to neural networks. Localisation beforehand is stable along the whole trajectory, the landing tracking failure happens only during the last 2 seconds of movement.

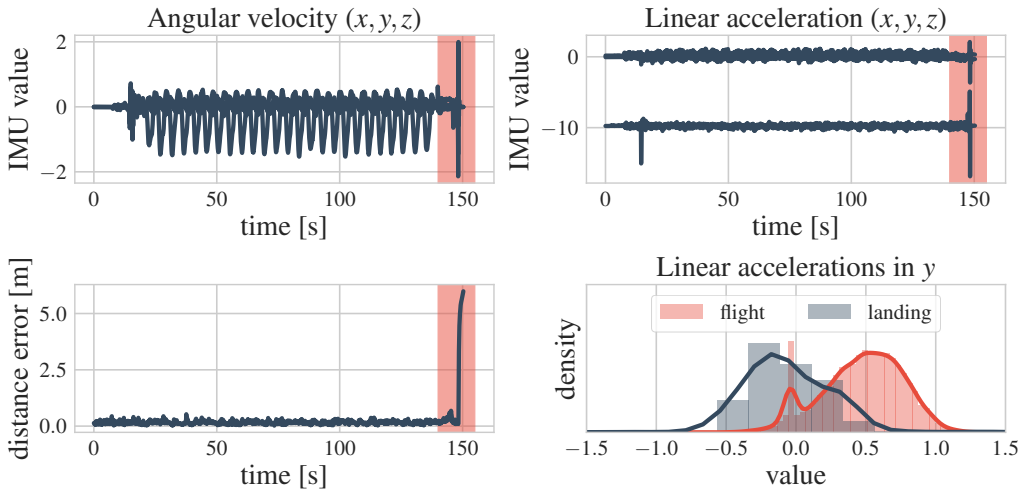


Figure 7: Illustration of the discussed failure landing of the quadcopter at the end of the two test set trajectories, taking *star, max speed 2.0 m/s* as an example. The problematic segment of the trajectory is marked with a red vertical band. Note the outlier controls at the end. Top left: angular velocity IMU readings. Top right: linear acceleration IMU readings. Bottom left: absolute Euclidean distance error w.r.t. the ground truth MOCAP locations. Note how the error is large only at the end of the trajectory, and directly coincides with the outlier controls. Bottom right: comparison of the distribution of linear accelerations in y during flight (red) vs. during landing (gray). The controls during landing are out of distribution for the learned transition model.

C DATA DETAILS AND OVERALL SETUP

For all presented experiments we used the Blackbird data set (Antonini et al., 2020), which can be found here: https://github.com/mit-fast/Blackbird-Dataset. In the following we describe the exact pre-processing of the data.

C.1 DATA PARTITIONING AND USAGE

The data was partitioned into a training, a validation and a test set. The trajectory shapes (e.g. *star*, *picasso*, *patrick*, etc.) in every split were different. This was done to prevent accidental overfitting of the learned transition model to any particular trajectory shape. In particular, the test set contains the trajectories *star, forward yaw* and *picasso, constant yaw*, traversed at different speeds, everything else is used for training and validation. This is the same test setup as the one used by Nisar et al. (2019) (considered as a baseline). Appendix G lists the exact trajectories used for each split, along with the number of steps in each trajectory after pre-processing (which includes subsampling).

The training set was used only to train the learned transition model $p_{\boldsymbol{\theta}_T}(\mathbf{z}_{t+1} \mid \mathbf{z}_t, \mathbf{u}_t)$, following the method described in section 4. The learned transition model is trained separately before it is used as a prior in the full spatial model. Pretraining the model on multiple trajectories beforehand, as opposed to training it from scratch during SLAM inference, ensures that the transition really captures the agent dynamics and does not overfit the currently explored environment. The training trajectories were further randomly rotated around the z axis and linearly translated in space, to promote generalisation.

The validation set was used for checkpointing and selecting the best weights $\boldsymbol{\theta}_T$ of the neural network in the learned transition model, based on the ELBO defined in section 4. The validation set was also used for hyperparameter selection for the engineered transition model, the learned transition model and the full spatial model. The best hyperparameters for all models (cf. appendix D) were selected with random search. You can find details for the search ranges in appendix E.

The test set was used for evaluation only—all results reported in the experimental section of the paper were done on the test data. The full spatial model was tested with the forward yaw *star* trajectories, speeds 1.0 m/s to 4.0 m/s and with the constant yaw *picasso* trajectories, speeds 1.0 m/s to 4.0 m/s, to match the evaluation by Nisar et al. (2019). These trajectories take place in the *NYC subway station* environment.

C.2 DATA PREPROCESSING

Each trajectory contains IMU, RPM and PWM readings, MOCAP ground truth pose observations, as well as simulated grayscale images from a forward-facing stereo pair and a downward-facing camera.

We pre-processed every trajectory in the following way:

- Ground truth MOCAP state readings were extracted from the provided ROS bags.
- Downward-facing images were ignored.
- The remaining data was subsampled to 10 Hz, using nearest neighbour subsampling based on the provided time stamps.
- Depth was then precomputed from the left and right images using OpenCV's SGBM (Hirschmuller, 2007).
- For every time step, the right colour image was then ignored, while the left image and the depth estimate were together treated as an RGB-D observation $\mathbf{x}_t \in \mathbb{R}^{w \times h \times 4}$.
- The provided IMU readings are measured in the coordinate frame of the IMU sensor. Therefore, they had to be rotated to the body frame of the agent, using the respective quaternion provided with the data set: $\mathbf{q} = [0.707, 0.005, -0.004, 0.706]^T$.

The intrinsic parameters of the camera, specifying the intrinsic camera matrix $\mathbf{K}$, and the stereo baseline were fixed to the values provided with the data set:

- Stereo baseline: 0.1m.
- Image size: 1024×768.

- Focal length, x: 665.1.
- Focal length, y: 665.1.
- Principal point offset, x: 511.5.
- Principal point offset, y: 383.5.

Depth was computed based on the disparity values produced by SGBM. We used the following parameters, keeping the default values for everything else:

- Block size: 5.
- Min. disparity: 0.
- Max. disparity: 256.

We did not filter the images and the produced depth values in any way and treated them as direct observations, to limit the pre-processing steps as much as possible and rely on the defined spatial model instead. Note that due to the simplicity of the method used for depth estimation, the depth observations contain significant amounts of noise (e.g. fig. 4d). The pixel values for the color images were normalized to be in the range $[0, 1]$.

D MODEL DETAILS

This section lists all model and optimisation hyperparameters used for generating the results reported in the paper.

D.1 FULL SPATIAL MODEL

The full spatial model uses either the learned or the engineered transition, see appendix D.2 and appendix D.3 for their respective hyperparameters.

D.1.1 OPTIMISATION

After subsampling, the trajectories in the Blackbird dataset can still contain thousands of time steps. To deal with this, during inference the proposed model follows the approximate particle optimisation scheme introduced by Mirchev et al. (2019), using chunks of 5 time steps for every gradient update, using 50 approximating state particles and refreshing the respective particles after every gradient step.

Adam (Kingma & Ba, 2014) is used to optimise the parameters ϕ for the approximate posterior distribution $q_{\boldsymbol{\phi}}(\mathbf{z}_{1:T}, \boldsymbol{\mathcal{M}})$. Table 2b lists the used optimiser hyperparameters. Note that due to the employed attention $p(\mathbf{m}_t \mid \mathbf{z}_t, \boldsymbol{\mathcal{M}})$, due to the reconstruction sampling in the emission and due to the approximate optimisation scheme mentioned above (which uses only a chunk of the trajectory at a time), only a part of the parameters for the occupancy map and the full trajectory of agent states is used for a gradient step. In other words, gradient updates happen locally in the spatially arranged occupancy map parameters, based on the currently selected local chunk of the agent trajectory. Because of this, momentum is disabled in Adam for both the occupancy map and the agent states, to avoid accidental drift in regions currently not covered by the optimisation.

For every variable in question, the same optimiser is used to optimise the mean and scale of the assumed distribution (where applicable). For any scale distribution parameters (e.g. standard deviations for the assumed Gaussian distributions), the optimisation is performed in log-space to satisfy the positive value constraint.

D.1.2 INITIALISATION OF NEW POSES

While performing SLAM, data is added incrementally to the spatial model, to emulate online usage of the method. A new time step t is explored every 500 gradient steps, effectively adding a new observation $\mathbf{x}_t$ and a new conditional input $\mathbf{u}_t$ to the data. Whenever a new time step is added, the parameters of the corresponding state $\mathbf{z}_t$ are unlocked for optimisation and initialized according to table 2a.

Table 2: Model details.

(a) Hyperparameters of the individual spatial model components.

Component	Parameter	Value
$\mathbf{z}_t$	init. value for $\boldsymbol{\mu}_t$	mean of $p_{\boldsymbol{\theta}_T}(\mathbf{z}_t \mid \boldsymbol{\mu}_{t-1}, \mathbf{u}_t)$
	init. value for $\boldsymbol{\sigma}_t$	$0.01 \times \mathbf{1}$
$\boldsymbol{\mathcal{M}}^{occ}$	grid size	$200 \times 200 \times 200$
	init. value for $\mu_{i,j,k}$	-0.5
	init. value for $\sigma_{i,j,k}$	0.1
$\boldsymbol{\mathcal{M}}^{col}$	# hidden layers	5
	# hidden units	256
	activation	softsign
	residual connections	true
$p(\mathbf{m}_t \mid \mathbf{z}_t, \boldsymbol{\mathcal{M}})$	ϵ (ray resolution)	0.1m
	max depth ($k\epsilon$)	20m
	# reconstr. pixels	200

(b) Optimisation hyperparameters for the full spatial model.

Variables	Parameter	Value
$\mathbf{z}_{1:T}$	Adam, learning rate	0.001
	Adam, β_1	0.0
	Adam, β_2	0.999
$\boldsymbol{\mathcal{M}}^{occ}$	Adam, learning rate	0.05
	Adam, β_1	0.0
	Adam, β_2	0.999
$\boldsymbol{\mathcal{M}}^{col}$	Adam, learning rate	0.001
	Adam, β_1	0.9
	Adam, β_2	0.999

(c) Hyperparameters of the learned transition model.

Component	Parameter	Value
$p_{\boldsymbol{\theta}_T}(\mathbf{z}_{t+1} \mid \mathbf{z}_t, \mathbf{u}_t)$	# hidden layers	5
	# hidden units	64
	activation	relu
	residual connections	true
	size of $\mathbf{z}^{\text{rest}}$	8
$\hat{q}(\mathbf{z}_t \mid \hat{\boldsymbol{\lambda}}_{1:T}, \hat{\boldsymbol{\omega}}_{1:T}, \mathbf{u}_{1:T})$	RNN type	LSTM
	# RNN units	64

D.1.3 Choice of approximate posterior over states

In (Mirchev et al., 2019) a bootstrap particle filter is used to implement the variational posterior over agent states. The increased model complexity due to the presented raycasting model makes the application of particle filters with sufficiently many particles costly. In section 3 we therefore introduce an approximate posterior over the agent states $q_{\boldsymbol{\phi}}(\mathbf{z}_{1:T})$, with free state parameters $\boldsymbol{\phi}$, that factorises over time. The chosen variational states approximation represents a shift in perspective—from filtering towards a method more similar to pose-graph optimisation.

D.1.4 Handling orientations

The portions $\boldsymbol{\omega}_t$ of the sampled latent states $\mathbf{z}_t$ are identified with quaternions and are explicitly normalised to unit quaternions after gradient updates before they are used in the rest of the model. This means that the assumed Gaussian distribution is not directly expressed on the manifold $\mathbb{SO}(3)$, but we found that this parameterisation works well enough.

D.1.5 Component parameters

The rest of the hyperparameters for the full spatial model, including initial values for the optimised variational parameters, are specific to the individual model components and are listed in table 2a.

The scale $\boldsymbol{\sigma}_E$ for the heteroscedastic Laplace emission $p(\mathbf{x}_t \mid \mathbf{m}_t)$ is learned with gradient descent (applied in log-space). The colour part of $\boldsymbol{\sigma}_E$ is forced to be 10 times smaller than that for the depth part, to account for the different scales of the observed values (colour range is $[0, 1]$, depth range is $[0, 20]$).

D.2 LEARNED TRANSITION

The learned transition is obtained by training the generative parameters $\boldsymbol{\theta}_T$ in the context of the model defined in section 4. When reconstructing orientations in that model, we do not reconstruct quaternions directly because of the ambiguity $\mathbf{q} = -\mathbf{q}$. Instead we reconstruct the rotation matrix corresponding to the observed orientation $\hat{\boldsymbol{\omega}}_t$, i.e. the mean of the emission $p(\hat{\boldsymbol{\omega}}_t \mid \mathbf{z}_t)$ is a rotation matrix constructed from the quaternion $\boldsymbol{\omega}_t$ that's part of the latent state $\mathbf{z}_t$. When used in the full spatial model, the weights of the transition neural network $\boldsymbol{\theta}_T$ are fixed to their pretrained values. This is done to avoid accidental overfitting of the transition parameters to the current trajectory and current environment for which SLAM is performed.

The selected hyperparameters of the learned transition are listed in table 2c.

D.3 ENGINEERED TRANSITION

The mean of the engineered transition prior $p(\mathbf{z}_{t+1} \mid \mathbf{z}_t, \mathbf{u}_t)$ from section 4 is formed by a function f_T, which implements the rigid body dynamics

$$f_T(\mathbf{z}_t, \mathbf{u}_t) = \begin{bmatrix} \boldsymbol{\lambda}_{t+1} \\ \boldsymbol{\omega}_{t+1} \\ \dot{\boldsymbol{\lambda}}_{t+1} \end{bmatrix} = \begin{bmatrix} \boldsymbol{\lambda}_t + \dot{\boldsymbol{\lambda}}_t \Delta t \\ \boldsymbol{\omega}_t \oplus \mathbf{R}(\boldsymbol{\omega}_t) \dot{\boldsymbol{\omega}}_t^{\text{imu}} \Delta t \\ \dot{\boldsymbol{\lambda}}_t + \mathbf{R}(\boldsymbol{\omega}_t) \ddot{\boldsymbol{\lambda}}_t^{\text{imu}} (\Delta t)^2 \end{bmatrix}.$$

Here $\oplus$ denotes standard quaternion integration. The standard deviation $\boldsymbol{\sigma}_T$ of $p(\mathbf{z}_{t+1} \mid \mathbf{z}_t, \mathbf{u}_t)$ (a Gaussian) is a hyperparameter, estimated via search on the validation set. Its diagonal entries are 0.01 for the location state dimensions, 0.001 for the orientation state dimensions and 0.001 for the velocity state dimensions. The gravitational force $\mathbf{g}$ is subtracted from the IMU readings when applying the Euler integration given by f_T. The time delta for the integration is set to $\Delta t = 0.1$, to match the 10Hz data subsampling.

E HYPERPARAMETER SEARCH AND EXECUTION DETAILS

All models are implemented in python using TensorFlow (Abadi et al., 2016), making use of automatic differentiation to optimise model parameters. This allows for the easy integration of neural networks and makes end-to-end optimisation straightforward. In TensorFlow 1.15, one gradient step of the model takes 0.0607s on average, without XLA compilation. In TensorFlow 2.3 one gradient step of the model takes 0.0073s on average, with XLA compilation enabled. The inference experiments in the paper currently assume 500 iterations per added data point (once every 0.1s for a 10Hz stream), which amounts to 36.5s of inference runtime per 1s of real-time movement in the newer TensorFlow version. As already mentioned in the main text, we have not yet optimised the model for real-time inference, and we will address this in our future work. The large speed-up achieved by simply enabling XLA and switching to a higher version of TensorFlow indicates that a lot of the computations in the model can be improved further (e.g. by moving to a C++ runtime or writing dedicated CUDA kernels). We also anticipate that better initialisation and careful tuning of the model hyperparameters will let the system reach interactive operation rates.

All probabilistic modelling aspects were implemented using Edward (Tran et al., 2018). Hyperparameter search (HPS) experiments were executed in a cluster with 8 Tesla V100 GPUs and 40 Intel Xeon E5-2698 CPU cores. Because of the large amount of Blackbird data (4.7 TB) and the current model's speed limitations, the search was performed for a subset of all possible hyperparameters. The performed model selection is therefore not exhaustive, possibly leaving potential for improvement. In the following we list the considered hyperparameter search ranges and the number of trials for each model.

E.1 HPS: FULL SPATIAL MODEL

The hyperparameters for the full spatial model were selected based on localisation RMSE for 500-step segments of trajectories in the validation set. A total of 200 experiments were conducted, randomly picking a trajectory segment on which SLAM inference is performed. The considered parameter ranges are listed in table 3a.

Table 3: HPS details.

(a) HPS ranges for the full spatial model.

Parameter	Range
reconstr. pixel count	$[100, 200, 500]$
$q(\boldsymbol{\mathcal{M}}^{occ})$ init. value stddev	$[0.01, 0.1, 1.0]$
$p(\boldsymbol{\mathcal{M}}^{occ})$ stddev	$[0.1, 1.0, 10.0]$
$\boldsymbol{\mathcal{M}}^{occ}$ grid side	$[50, 100, 200]$
$\boldsymbol{\mathcal{M}}^{occ}$ learning rate	$[0.01, 0.05, 0.001]$
$\boldsymbol{\mathcal{M}}^{col}$ learning rate	$[0.01, 0.05, 0.001]$
$\mathbf{z}_{1:T}$ learning rate	$[0.01, 0.05, 0.001]$

(b) HPS ranges for the learned transition model.

Parameter	Range
learning rate	$[0.0001, 0.0005, 0.001, 0.005]$
$\mathbf{z}^{\text{rest}}$	$[8, 16]$
# hidden units	$[32, 64, 128, 256]$
# layers	$[2, 3, 4, 5]$
activation	[softsign, relu]
# RNN units	$[32, 64, 128, 256]$

(c) HPS ranges for the engineered transition model.

Parameter	Range
$\boldsymbol{\lambda}$ stddev	$[0.0001, 0.001, 0.01]$
$\boldsymbol{\omega}$ stddev	$[0.0001, 0.001, 0.01]$
$\dot{\boldsymbol{\lambda}}$ stddev	$[0.0001, 0.001, 0.01]$

E.2 HPS: LEARNED TRANSITION MODEL

The learned transition hyperparameters were selected based on the validation set ELBO value from the model discussed in section 4. A total of 400 experiments were conducted, using all of the blackbird training and validation data. The considered parameter ranges are listed in table 3b.

E.3 HPS: ENGINEERED TRANSITION MODEL

The engineered transition hyperparameters were selected in the same HPS for the full model discussed above, based on localisation RMSE for 500-step segments of trajectories in the validation set. The considered parameter ranges are listed in table 3c.

F FURTHER INFERENCE EXAMPLES

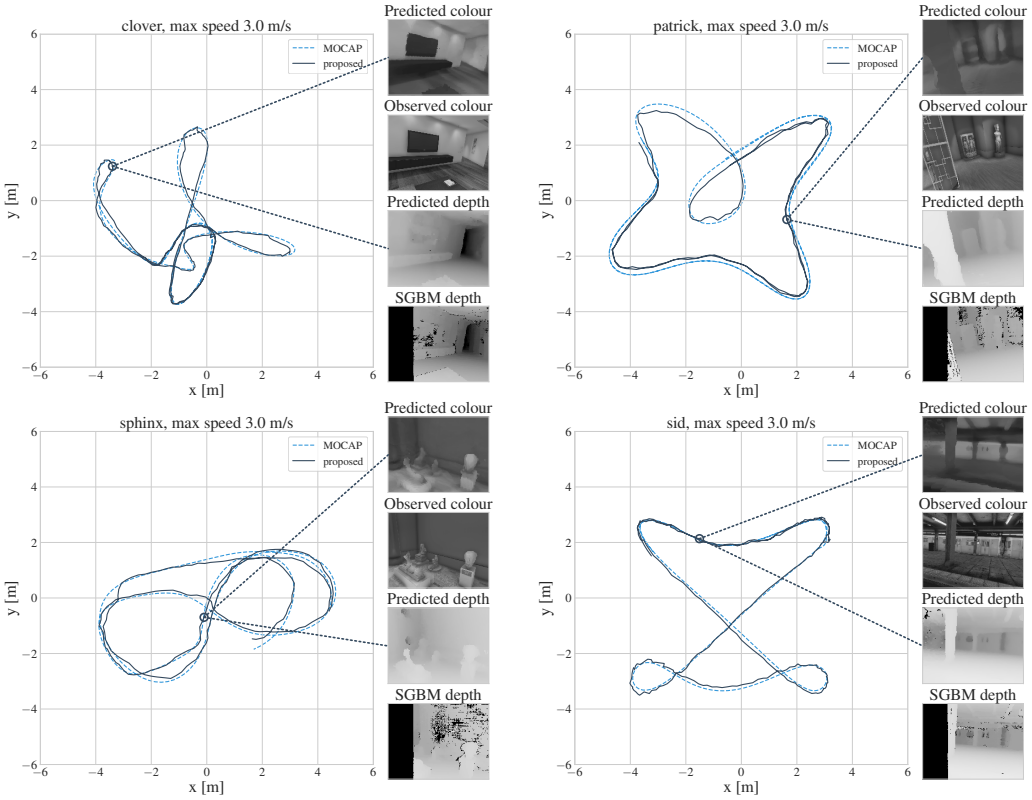


Figure 8: Examples of inference in the full spatial model, expressed in localisation and mapping for segments of the following trajectories: *clover, forward yaw, max speed 3.0 m/s*, *patrick, forward yaw, max speed 3.0 m/s*, *sphinx, forward yaw, max speed 3.0 m/s* and *sid, forward yaw, max speed 3.0 m/s*. Left on every subfigure: top-down view of the localisaton estimate. Right on every subfigure: depth and colour reconstructions from the generative model for one time step.

G DATA SET SPLITS

Training set

trajectory	# steps	duration [s]
3dFigure8, constant yaw, 1.0 m/s	2087	208.7
3dFigure8, constant yaw, 2.0 m/s	2130	213.0
3dFigure8, constant yaw, 3.0 m/s	2091	209.1
3dFigure8, constant yaw, 4.0 m/s	2136	213.6
3dFigure8, constant yaw, 5.0 m/s	2240	224.0
ampersand, constant yaw, 1.0 m/s	2086	208.6
ampersand, constant yaw, 2.0 m/s	2061	206.1
ampersand, constant yaw, 3.0 m/s	833	83.3
clover, constant yaw, 1.0 m/s	2575	257.5
clover, constant yaw, 2.0 m/s	2605	260.5
clover, constant yaw, 3.0 m/s	833	83.3
clover, constant yaw, 4.0 m/s	2631	263.1
clover, constant yaw, 5.0 m/s	2607	260.7
dice, constant yaw, 2.0 m/s	2605	260.5
dice, constant yaw, 3.0 m/s	833	83.3
dice, constant yaw, 4.0 m/s	2614	261.4
figure8, constant yaw, 1.0 m/s	1513	151.3
figure8, constant yaw, 2.0 m/s	2052	205.2
figure8, constant yaw, 5.0 m/s	1999	199.9
mouse, constant yaw, 1.0 m/s	2641	264.1
mouse, constant yaw, 2.0 m/s	2674	267.4
mouse, constant yaw, 3.0 m/s	833	83.3
mouse, constant yaw, 4.0 m/s	2620	262.0
mouse, constant yaw, 5.0 m/s	2655	265.5
oval, constant yaw, 2.0 m/s	2033	203.3
oval, constant yaw, 3.0 m/s	833	83.3
oval, constant yaw, 4.0 m/s	2035	203.5
thrice, constant yaw, 1.0 m/s	2726	272.6
thrice, constant yaw, 2.0 m/s	2658	265.8
thrice, constant yaw, 3.0 m/s	2656	265.6
thrice, constant yaw, 4.0 m/s	2656	265.6
thrice, constant yaw, 5.0 m/s	2621	262.1
tiltedThrice, constant yaw, 1.0 m/s	2624	262.4
tiltedThrice, constant yaw, 2.0 m/s	2624	262.4
tiltedThrice, constant yaw, 3.0 m/s	833	83.3
tiltedThrice, constant yaw, 4.0 m/s	2602	260.2
tiltedThrice, constant yaw, 5.0 m/s	2574	257.4
winter, constant yaw, 1.0 m/s	2625	262.5
winter, constant yaw, 2.0 m/s	2570	257.0
winter, constant yaw, 3.0 m/s	833	83.3
winter, constant yaw, 4.0 m/s	2569	256.9
winter, constant yaw, 5.0 m/s	2620	262.0
ampersand, forward yaw, 1.0 m/s	1100	110.0
ampersand, forward yaw, 2.0 m/s	893	89.3
clover, forward yaw, 1.0 m/s	1088	108.8
clover, forward yaw, 2.0 m/s	1088	108.8
clover, forward yaw, 3.0 m/s	833	83.3
clover, forward yaw, 4.0 m/s	1101	110.1
clover, forward yaw, 5.0 m/s	1091	109.1
dice, forward yaw, 1.0 m/s	1096	109.6
dice, forward yaw, 2.0 m/s	1099	109.9
dice, forward yaw, 3.0 m/s	833	83.3
mouse, forward yaw, 1.0 m/s	1110	111.0
mouse, forward yaw, 2.0 m/s	1107	110.7
mouse, forward yaw, 3.0 m/s	833	83.3
mouse, forward yaw, 4.0 m/s	1108	110.8
mouse, forward yaw, 5.0 m/s	1107	110.7
oval, forward yaw, 1.0 m/s	1086	108.6
oval, forward yaw, 2.0 m/s	892	89.2
oval, forward yaw, 3.0 m/s	833	83.3
oval, forward yaw, 4.0 m/s	1086	108.6
thrice, forward yaw, 1.0 m/s	1088	108.8
thrice, forward yaw, 2.0 m/s	1088	108.8
thrice, forward yaw, 3.0 m/s	833	83.3
thrice, forward yaw, 4.0 m/s	1088	108.8
thrice, forward yaw, 5.0 m/s	1088	108.8
tiltedThrice, forward yaw, 1.0 m/s	898	89.8
tiltedThrice, forward yaw, 2.0 m/s	1088	108.8
tiltedThrice, forward yaw, 3.0 m/s	833	83.3
tiltedThrice, forward yaw, 4.0 m/s	1088	108.8
tiltedThrice, forward yaw, 5.0 m/s	1087	108.7
winter, forward yaw, 2.0 m/s	1089	108.9
winter, forward yaw, 3.0 m/s	833	83.3
winter, forward yaw, 4.0 m/s	1091	109.1

Test set

trajectory	# steps	duration [s]
star, constant yaw, 1.0 m/s	2680	268.0
star, constant yaw, 2.0 m/s	2635	263.5
star, constant yaw, 3.0 m/s	2640	264.0
star, constant yaw, 4.0 m/s	2667	266.7
star, constant yaw, 5.0 m/s	2611	261.1
star, forward yaw, 1.0 m/s	1491	149.1
star, forward yaw, 2.0 m/s	1503	150.3
star, forward yaw, 3.0 m/s	1523	152.3
star, forward yaw, 4.0 m/s	1627	162.7
star, forward yaw, 5.0 m/s	1108	110.8
picasso, constant yaw, 1.0 m/s	2040	204.0
picasso, constant yaw, 2.0 m/s	2071	207.1
picasso, constant yaw, 3.0 m/s	2072	207.2
picasso, constant yaw, 4.0 m/s	2109	210.9
picasso, constant yaw, 5.0 m/s	2626	262.6
picasso, forward yaw, 1.0 m/s	833	83.3
picasso, forward yaw, 3.0 m/s	2083	208.3
picasso, forward yaw, 4.0 m/s	2057	205.7
picasso, forward yaw, 5.0 m/s	891	89.1

Validation set

trajectory	# steps	duration [s]
sid, constant yaw, 1.0 m/s	2688	268.8
sid, constant yaw, 2.0 m/s	2677	267.7
sid, constant yaw, 3.0 m/s	833	83.3
sid, constant yaw, 4.0 m/s	2668	266.8
sid, constant yaw, 5.0 m/s	2661	266.1
sid, forward yaw, 1.0 m/s	897	89.7
sid, forward yaw, 2.0 m/s	1109	110.9
sid, forward yaw, 3.0 m/s	833	83.3
sid, forward yaw, 4.0 m/s	1109	110.9
sid, forward yaw, 5.0 m/s	1110	111.0
sphinx, constant yaw, 1.0 m/s	2612	261.2
sphinx, constant yaw, 2.0 m/s	2601	260.1
sphinx, constant yaw, 3.0 m/s	833	83.3
sphinx, constant yaw, 4.0 m/s	2560	256.0
sphinx, forward yaw, 1.0 m/s	1088	108.8
sphinx, forward yaw, 2.0 m/s	1087	108.7
sphinx, forward yaw, 3.0 m/s	833	83.3
sphinx, forward yaw, 4.0 m/s	1086	108.6
bentDice, constant yaw, 1.0 m/s	2624	262.4
bentDice, constant yaw, 2.0 m/s	2698	269.8
bentDice, constant yaw, 3.0 m/s	417	41.7
bentDice, constant yaw, 4.0 m/s	2632	263.2
bentDice, forward yaw, 1.0 m/s	1088	108.8
bentDice, forward yaw, 2.0 m/s	1088	108.8
bentDice, forward yaw, 3.0 m/s	833	83.3
patrick, constant yaw, 1.0 m/s	2598	259.8
patrick, constant yaw, 2.0 m/s	2584	258.4
patrick, constant yaw, 3.0 m/s	417	41.7
patrick, constant yaw, 4.0 m/s	2611	261.1
patrick, constant yaw, 5.0 m/s	2580	258.0
patrick, forward yaw, 1.0 m/s	1089	108.9
patrick, forward yaw, 2.0 m/s	1089	108.9
patrick, forward yaw, 3.0 m/s	833	83.3
patrick, forward yaw, 4.0 m/s	1091	109.1

A.2. PRISM: Probabilistic Real-Time Inference in Spatial World Models

Paper Summary

PRISM is a method for real-time filtering in a pre-established dense state-space model (VSSM-LM, appendix A.1). The filters are derived from the generative assumptions in a principled way, respecting the dense image generation, and all approximations are carefully signposted in the process.

Filtering is identified as a necessary compromise for dense state-space models (compared to smoothing), in order to limit the size of the inference problem. Real-time is reached by incorporating traditional techniques from computer vision and robotics, with emphasis on closed-form updates where possible.

PRISM manages to reconcile dense rendering, dynamics and a fully probabilistic treatment on a tight budget. Its accuracy is comparable to SotA visual odometry methods, tested on indoor data across three data sets. The price is compromises in the uncertainty approximations, that could be addressed in the future as hardware improves.

Errata

Eq. 3 and 10 should say $\cdots \propto q_t^{\Phi}(\mathcal{M})$ and $\cdots \propto \mathcal{N}\big(\mathcal{M} \mid \mu_{t+1}^{\mathcal{M}}, \operatorname{diag}(\sigma_{t+1}^{\mathcal{M}})^2\big)$ $:= q_{t+1}^{\Phi}(\mathcal{M})$, respectively ($\propto$ instead of $=$ because Gaussian products are proportional to a Gaussian). Alternatively, $q(\mathcal{M} \mid x_t, z_t)$ can be defined as an unnormalised Gaussian in appendix D (then $=$ instead of $\propto$ holds). This is only to make notation precise, the method remains the same.

Author Contributions

problem definition	significant
literature review	significant
theoretical derivation	significant
implementation	significant
experiments	significant
writing	significant

PRISM: Probabilistic Real-Time Inference in Spatial World Models

Atanas Mirchev[1], **Baris Kayalibay**[1], **Ahmed Agha**[1],
Patrick van der Smagt[1], **Daniel Cremers**[2], **Justin Bayer**[1]

[1]Machine Learning Research Lab, Volkswagen Group, [2]Technical University of Munich
atanas.mirchev@argmax.ai

Abstract: We introduce PRISM, a method for real-time filtering in a probabilistic generative model of agent motion and visual perception. Previous approaches either lack uncertainty estimates for the map and agent state, do not run in real-time, do not have a dense scene representation or do not model agent dynamics. Our solution reconciles all of these aspects. We start from a predefined state-space model which combines differentiable rendering and 6-DoF dynamics. Probabilistic inference in this model amounts to simultaneous localisation and mapping (SLAM) and is intractable. We use a series of approximations to Bayesian inference to arrive at probabilistic map and state estimates. We take advantage of well-established methods and closed-form updates, preserving accuracy and enabling real-time capability. The proposed solution runs at 10Hz real-time and is similarly accurate to state-of-the-art SLAM in small to medium-sized indoor environments, with high-speed UAV and handheld camera agents (Blackbird, EuRoC and TUM-RGBD).

Keywords: generative model, SLAM, Bayes filter, uncertainty, diff. rendering

1 Introduction

Moving agents perceive streams of information, typically a mix of RGB images, depth and inertial measurements. Probabilistic generative models [1] are a principled way to formalise the *synthesis* of this data, and from these models inference can be derived through Bayes' rule. We focus on exactly such inference and target the agent states and the scene map, a problem known as simultaneous localisation and mapping (SLAM). We treat it as a posterior approximation for a given state-space model, such that the combination is useful for model-based control: the posterior inference serves as a state estimator and the predictive state-space model as a simulator with which to plan ahead [2].

To pave the way towards decision making, we believe an inference method should have:

- a compatible predictive model for both RGB-D images and 6-DoF dynamics;
- principled state and map uncertainty;
- real-time performance on commodity hardware;
- state-of-the-art localisation accuracy.

We motivate these requirements further in appendix J. Prominent methods like LSD-SLAM [3], ORB-SLAM [4], DSO [5] have propelled visual SLAM forward, with heavy focus on large-scale localisation. The core of modern large-scale SLAM is maximum a-posteriori (MAP) smoothing in a probabilistic factor graph [6, 7]. At present this demands sparsity assumptions for computational feasibility, which obstructs the tight integration of dense maps and rendering. Nonetheless, for smaller scenes the recent popularity of neural models (e.g. NERF [8]) has sparked interest in inference through a renderer (e.g. [9, 10, 11]), but dynamics modelling and uncertainty have remained out of scope. Conversely, classical filtering comes with dynamics and uncertainty in real-time (e.g. [12, 13, 14]),

6th Conference on Robot Learning (CoRL 2022), Auckland, New Zealand.

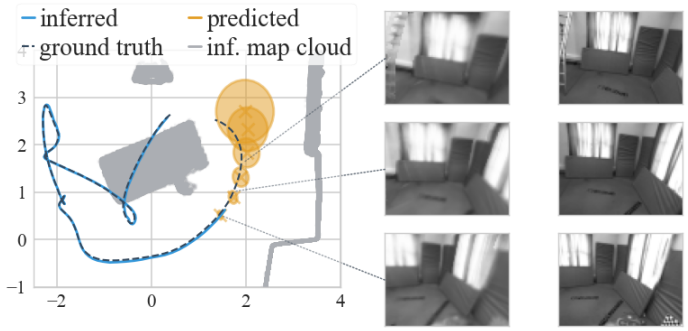


Figure 1: Inference is tailored to the depicted predictive model. Predicting future rollouts, as shown, is required for optimal control. Ground-truth trajectory in *black*, inferred trajectory from past data in *blue*. In *orange*, we see uncertainty envelopes for the predicted future states. On the right, we see predicted and ground-truth future images. Visualised in 2D for clarity, our method operates in 3D.

but over time has given way to large-scale smoothing [6] and to our knowledge has not been well explored for the integration of dense differentiable rendering and dynamics on a moderate scale.

Overall, we find there is a need for a cohesive inference solution that satisfies our requirements. We thus contribute by meeting all the above goals, emphasising the link to a predictive model (fig. 1).

We start from the generative model of Mirchev et al. [15], who combine differentiable rendering and agent dynamics in a probabilistic framework. The authors considered stochastic variational inference for this model, applying it off-line with runtime orders of magnitude too long for on-line use. We pursue an alternative route for real-time inference: from the generative assumptions we derive approximations to the true marginal filters over the last state and map [16]. By focusing on recursive filtering updates, we identify where established probabilistic inference and computer vision techniques can be used, putting emphasis on fast closed-form updates. We find this divide-and-conquer strategy is a good compromise for achieving the aforementioned objectives under computational constraints.

We evaluate the proposed solution on two unmanned aerial vehicle (UAV) data sets [17, 18] and on TUM-RGBD [19]. Our method PRISM runs at 10 Hz real-time with similar localisation accuracy to state-of-the-art SLAM in moderately-sized indoor environments. It provides uncertainty estimates and features a predictive distribution that can both render images and forecast the agent's movement.

2 Related Work

Generative models Generative state-space models simulate the formation of observed data over time in a Markov chain [1, 12, 20, 21, 22, 23], serving as *world* models [24, 25]. With their agent dynamics and state-to-observation emission models we can imagine future rollouts for planning [2, 26, 27, 28, 29, 30, 31, 32, 33]. We abide by this framework and design a posterior inference for a *spatial* state-space model, to enable on-line control. Among such models (e.g. [34, 35, 36, 37, 38, 39, 40]), we tailor our inference to the model of Mirchev et al. [15]. It scales to 3D with rendering and 6-DoF dynamics. We contribute a real-time inference that fits its probabilistic formulation.

SLAM through image synthesis The assumed generative model renders RGB-D images, which is related to SLAM through full-image synthesis. Traditional methods feature varied maps, from volumetric to surfels (e.g. [41, 42, 43, 44, 45, 46, 47, 48, 49]), and commonly estimate new camera poses by aligning new observations to a rendered image with variants of point-to-plane ICP with photometric consistency [50, 51, 52, 53, 44]. We extend this optimisation with dynamics in our approximate state filter [54]. A recent trend is to use implicit scene representations like NERF (e.g. [8, 55, 56]) with high rendering fidelity. Gradient-based pose inference through NERF-like rendering has received attention [57, 58], with iMAP [11] and NICE-SLAM [9] being two real-time

solutions. The mapping runtime of such methods is weighed down by optimisation through the renderer. Rendering can be sped up by decomposing parameters over space, e.g. by using voxels or primitives [59, 60, 61, 62, 63], but how to update neural maps in closed form remains unclear. Therefore, we rely on vanilla voxel grid maps [15, 64], as their probabilistic treatment and closed-form updates are straightforward, leaving implicit representations for future work. We note that none of the aforementioned methods incorporate dynamics and uncertainty, which distinguishes our approach.

Probabilistic SLAM inference SLAM filters are thoroughly explored for flat 2D modelling [65, 12, 66, 13, 67, 68, 69], but have been superseded by MAP smoothing in modern visual SLAM (e.g. [3, 4, 5, 70, 71, 72, 73]), primarily due to scalability concerns [6, 74]. However, as of now smoothing is not computationally feasible without sparsity assumptions. We therefore reexamine filtering for differentiable rendering, as we aim to obtain a dense map posterior with uncertainty in real-time (see appendix J for further motivation). Filters may benefit from the dense modelling of observations [74], which aligns with our objective, and we will demonstrate they can be a feasible solution for moderately-sized indoor environments. For the states, we use a Laplace approximation [75] and velocity updates similar to those in extended Kalman filters [12]. For the map, occupancy grids are a common probabilistic choice [64, 66, 76] and closed-form mapping has been used in that context [69]. To enable rendering we provide a similar derivation, but for a signed distance function (SDF), which is related. Probabilistic SDF mapping dates back to Curless and Levoy [41], and SDF updates have a well-known probabilistic interpretation [77, 78]. We use these approximations to arrive at a holistic probabilistic solution that scales to dense 3D modelling in real-time.

3 Overview

We approach on-line SLAM inference with two aims in mind. First, we want to harmonise our map and state estimation with a predictive model. Second, we want to quantify uncertainty: estimates and predictions should account for modelling inaccuracies as well as measurement and process noise. Both are important for autonomous decision making. To achieve this, we derive a Bayesian posterior in the probabilistic model of Mirchev et al. [15], to ensure that inference matches the forward model. Before we delve into our proposed solution, we present a practical summary. At every time step:

1. we point-estimate the agent's pose using gradient descent, involving geometry and dynamics.
2. we extend the pose with a Gaussian covariance matrix through a Laplace approximation.
3. with the pose, we estimate the agent's current velocity in closed form.
4. with the pose and the current observation, we update the map in closed form.

We use well-established methods for the above. In 1. we combine assumed density filtering [79], point-to-plane ICP [50] and photometric alignment [51, 52]. In 2. we use a Laplace approximation [75, 80]. In 3. we use linear-Gaussian updates, akin to Kalman filters [12]. In 4. we first derive generic closed-form map updates, which boil down to SDF updates [41] for our generative assumptions.

We contribute by deriving a holistic Bayesian inference from the generative model we started with. In doing so, we identify where traditional techniques are applicable to make a practical algorithm.

4 Methods

In the following we will denote generative distributions, true posterior distributions and conditionals with $p(\cdot)$. Respectively, approximate distributions will be denoted with $q(\cdot)$. Approximation steps will be indicated by $\approx$ in equations. We use $q^{\boldsymbol{\phi}}(\cdot)$ to subsume estimated distribution parameters into $\boldsymbol{\phi}$. A subscript $\cdot_t$ indicates that a variable or a distribution is different at every time step.

4.1 Background

We start with an overview of the generative model of Mirchev et al. [15] from which we will derive the inference. We assume a sequence of RGB-D observations $\mathbf{x}_{1:T}$ and a sequence of agent states

$\mathbf{z}_{1:T}$ driven by controls $\mathbf{u}_{1:T-1}$ form a Markovian state-space model. Each observation is constructed from a respective state with a rendering emission model $p(\mathbf{x}_t \mid \boldsymbol{\mathcal{M}}, \mathbf{z}_t)$, where $\boldsymbol{\mathcal{M}}$ is a global latent random variable for a dense map. A transition model $p(\mathbf{z}_t \mid \mathbf{z}_{t-1}, \mathbf{u}_{t-1})$ accounts for the agent dynamics, where $\mathbf{u}_t$ are known acceleration controls. Assuming $\mathbf{z}_1$ is given, the joint distribution is:

$$p(\boldsymbol{\mathcal{M}}, \mathbf{z}_{2:T}, \mathbf{x}_{1:T} \mid \mathbf{u}_{1:T-1}, \mathbf{z}_1) = p(\boldsymbol{\mathcal{M}})p(\mathbf{x}_1 \mid \boldsymbol{\mathcal{M}}, \mathbf{z}_1) \prod_{t=2}^{T} p(\mathbf{z}_t \mid \mathbf{z}_{t-1}, \mathbf{u}_{t-1})p(\mathbf{x}_t \mid \mathbf{z}_t, \boldsymbol{\mathcal{M}}).$$

The map is a 3D voxel grid of occupancy and color–each cell contains four values. The emission is fully-differentiable and performs volumetric raymarching, searching for a unique hit position at a surface along each ray [81]. The transition performs Euler integration, using the acceleration controls and maintained velocity from the latent state. Appendix B and the original paper have the details.

4.2 Posterior Choice

First we need to choose which posterior to approximate. For example, Mirchev et al. [15] approximate the full posterior over the map and *all* states $p(\boldsymbol{\mathcal{M}}, \mathbf{z}_{2:T} \mid \mathbf{x}_{1:T}, \mathbf{u}_{1:T-1}, \mathbf{z}_1)$ with variational inference [82]. While generic, this approach is slowed down by rendering at every optimisation step [54], and the inevitable stochastic optimisation demands multiple steps until convergence. In addition, estimating the posterior over all states scatters the optimisation budget across the whole trajectory.

To enable real-time inference we target an alternative posterior, the filter $p(\boldsymbol{\mathcal{M}}, \mathbf{z}_t \mid \mathbf{x}_{1:t}, \mathbf{u}_{1:t-1}, \mathbf{z}_1)$, as the last state belief is enough for planning ahead [2]. Since filters can be updated recursively [16, 80], we can use closed-form updates for fast inference. Still, maintaining the joint distribution is too costly because of the large dense 3D map $\boldsymbol{\mathcal{M}}$.[1] Instead, we approximate the two marginal filters:

$$q_t^{\boldsymbol{\phi}}(\boldsymbol{\mathcal{M}}) \approx p(\boldsymbol{\mathcal{M}} \mid H_t) = p(\boldsymbol{\mathcal{M}} \mid \mathbf{x}_{1:t}, \mathbf{u}_{1:t-1}, \mathbf{z}_1)$$
$$q_t^{\boldsymbol{\phi}}(\mathbf{z}_t) \approx p(\mathbf{z}_t \mid H_t) = p(\mathbf{z}_t \mid \mathbf{x}_{1:t}, \mathbf{u}_{1:t-1}, \mathbf{z}_1),$$

where $H_t = \mathbf{x}_{1:t}, \mathbf{u}_{1:t-1}, \mathbf{z}_1$. More details about this modelling choice can be found in appendix A. We draw attention to the shorthand notation $p(\ \cdot\ \mid H_t)$, which will appear again in the following.

4.3 Approximate Filtering

For both marginal filters, we will arrive at adequate approximations by reusing the following equation:

$$p(\boldsymbol{\mathcal{M}}, \mathbf{z}_t \mid H_t) \propto p(\mathbf{x}_t \mid \mathbf{z}_t, \boldsymbol{\mathcal{M}}) \int p(\mathbf{z}_t \mid \mathbf{z}_{t-1}, \mathbf{u}_{t-1})p(\boldsymbol{\mathcal{M}}, \mathbf{z}_{t-1} \mid H_{t-1})d\mathbf{z}_{t-1}, \tag{1}$$

This is a classic recursive expression of the Bayes filter [16]. Starting from each true marginal posterior, we will first expand the joint, then use eq. (1) and apply a set of approximations. Next we will discuss our final result, we defer the detailed derivation of both filters to appendices C and E.

4.3.1 Marginal Map Filter

We begin with the map approximation, starting from the true marginal Bayes filter:

$$\begin{aligned} p(\boldsymbol{\mathcal{M}} \mid H_t) &= \int p(\boldsymbol{\mathcal{M}}, \mathbf{z}_t \mid H_t)\, \mathrm{d}\mathbf{z}_t \\ &\propto \int p(\mathbf{x}_t \mid \mathbf{z}_t, \boldsymbol{\mathcal{M}}) \int p(\mathbf{z}_t \mid \mathbf{z}_{t-1}, \mathbf{u}_{t-1})p(\boldsymbol{\mathcal{M}}, \mathbf{z}_{t-1} \mid H_{t-1})\, \mathrm{d}\mathbf{z}_{t-1}\, \mathrm{d}\mathbf{z}_t \\ &\approx p(\mathbf{x}_t \mid \hat{\mathbf{z}}_t, \boldsymbol{\mathcal{M}}) \times q_{t-1}^{\boldsymbol{\phi}}(\boldsymbol{\mathcal{M}}) \qquad (2) \\ &\approx q(\boldsymbol{\mathcal{M}} \mid \mathbf{x}_t, \hat{\mathbf{z}}_t) \times q_{t-1}^{\boldsymbol{\phi}}(\boldsymbol{\mathcal{M}}) =: q_t^{\boldsymbol{\phi}}(\boldsymbol{\mathcal{M}}). \qquad (3) \end{aligned}$$

Equations (2) and (3) hide a few approximations detailed in appendix C. The resulting solution takes a nominal state sample $\hat{\mathbf{z}}_t$, with which a map update $q(\boldsymbol{\mathcal{M}} \mid \mathbf{x}_t, \hat{\mathbf{z}}_t)$ is applied to the previous map belief $q_{t-1}^{\boldsymbol{\phi}}(\boldsymbol{\mathcal{M}})$. We set $\hat{\mathbf{z}}_t$ to the mean of the current state belief $q_t^{\boldsymbol{\phi}}(\mathbf{z}_t)$. Accepting some bias, we do this for speed as it is our best guess for $\mathbf{z}_t$ without extra computation.[2] Intuitively, the map update

[1] E.g. the size of full-covariance Gaussian representations [12] or carrying multiple maps in parallel for a Rao-Blackwellised particle filter [13, 14] become prohibitive.

[2] Appendix F discusses this approximation further.

$q(\boldsymbol{\mathcal{M}} \mid \mathbf{x}_t, \hat{\mathbf{z}}_t)$ populates the map such that the observation $\mathbf{x}_t$ can be reconstructed. Our derivation of the updates is similar to the one by Grisetti et al. [69] for 2D occupancy maps, but now applied to 3D.

The above approximation is generic, agnostic to the specific map and rendering assumptions. In practice, we need a closed-form map update $q(\boldsymbol{\mathcal{M}} \mid \mathbf{x}_t, \hat{\mathbf{z}}_t)$ that is faithful to the emission $p(\mathbf{x}_t \mid \hat{\mathbf{z}}_t, \boldsymbol{\mathcal{M}})$. In this work, we follow Mirchev et al. [15] and use a Gaussian map that factorises over voxels:

$$q_t^{\boldsymbol{\phi}}(\boldsymbol{\mathcal{M}}) = \prod_{ijk} \mathcal{N}\big(\boldsymbol{\mathcal{M}}_{ijk} \mid \boldsymbol{\mu}^{\boldsymbol{\mathcal{M}}}_{ijk,t}, \operatorname{diag}((\boldsymbol{\sigma}^{\boldsymbol{\mathcal{M}}}_{ijk,t})^2)\big).$$

Here the indices ijk run over voxels in a 3D grid. For this specific representation and the assumed surface-based rendering, we identify that the map update $q(\boldsymbol{\mathcal{M}} \mid \mathbf{x}_t, \hat{\mathbf{z}}_t)$ can be implemented as a probabilistic signed distance function (SDF) update [41]. We provide the technical details in appendix D. SDF updates for voxel maps are a traditional concept in computer vision, and prior work has considered their probabilistic interpretation before [77, 78]. We contribute by identifying the place of such updates in a probabilistic filter that follows the generative model of [15]. A detailed discussion of how the above relates to classical SDF update equations can be found in appendix D.

The above approximations are motivated by the real-time constraint. For example, one could optimise eq. (2) directly with gradient descent through the renderer, but evaluating the emission is expensive and hinders accurate convergence on a budget. This is particularly true when uncertainty estimates are desirable, as optimisation would then be stochastic and gradients noisy [83]. In contrast, the derived one-shot map updates are meant to have a cost similar to emitting just once, while capturing uncertainty as well. We show some of the differences between the two approaches in section 5.3.

4.3.2 Marginal State Filter

Similarly, for the state filter we start from the true marginal and arrive at approximations via eq. (1):

$$\begin{aligned}
p(\mathbf{z}_t \mid H_t) &= \int p(\boldsymbol{\mathcal{M}}, \mathbf{z}_t \mid H_t)\, \mathrm{d}\boldsymbol{\mathcal{M}} \\
&\propto \int p(\mathbf{x}_t \mid \mathbf{z}_t, \boldsymbol{\mathcal{M}}) \int p(\mathbf{z}_t \mid \mathbf{z}_{t-1}, \mathbf{u}_{t-1}) p(\boldsymbol{\mathcal{M}}, \mathbf{z}_{t-1} \mid H_{t-1})\, \mathrm{d}\mathbf{z}_{t-1}\, \mathrm{d}\boldsymbol{\mathcal{M}} \\
&\approx p\Big(\mathbf{x}_t \mid \mathbf{z}_t^{\text{pose}}, \hat{\boldsymbol{\mathcal{M}}}\Big) q_t(\mathbf{z}_t^{\text{pose}} \mid \mathbf{u}_{t-1}, H_{t-1}) q_t(\mathbf{z}_t^{\text{vel}} \mid \mathbf{z}_t^{\text{pose}}, \mathbf{u}_{t-1}, H_{t-1}) \qquad (4) \\
&\approx q_t^{\boldsymbol{\phi}}(\mathbf{z}_t^{\text{pose}}) \times q_t(\mathbf{z}_t^{\text{vel}} \mid \mathbf{z}_t^{\text{pose}}, \mathbf{u}_{t-1}, H_{t-1}) =: \; q_t^{\boldsymbol{\phi}}(\mathbf{z}_t). \qquad (5)
\end{aligned}$$

We detail all the approximations that lead to eq. (4) in appendix E. In eq. (4) we have three terms: an image reconstruction likelihood, a Gaussian pose prior and a linear Gaussian velocity conditional given a pose. The latter two we obtain analytically with a linear approximation of the transition model and the previous Gaussian belief $q_{t-1}^{\boldsymbol{\phi}}(\mathbf{z}_{t-1})$ (c.f. appendix E). First, using the first two terms of eq. (4) we define a maximum a-posteriori (MAP) objective for pose optimisation:

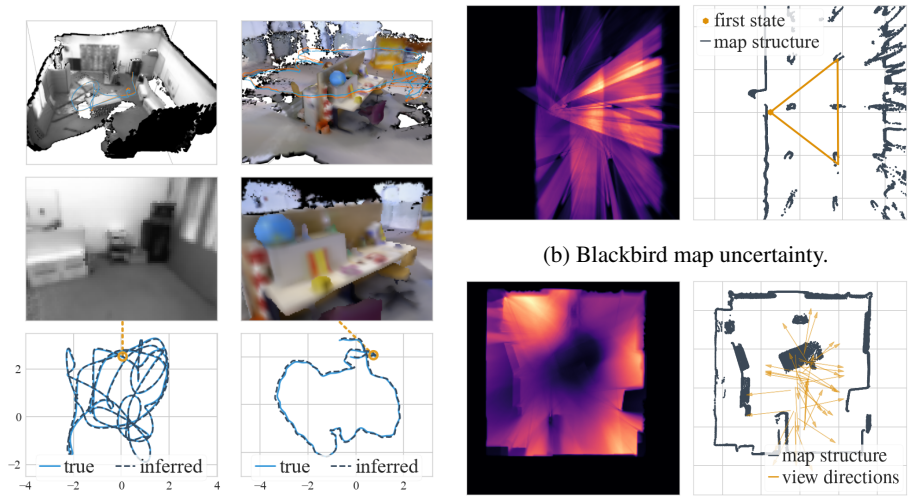
$$\arg\max_{\mathbf{z}_t^{\text{pose}}} \; \log p\Big(\mathbf{x}_t \mid \hat{\boldsymbol{\mathcal{M}}}, \mathbf{z}_t^{\text{pose}}\Big) + \log q_t(\mathbf{z}_t^{\text{pose}} \mid \mathbf{u}_{t-1}, H_{t-1}).$$

Here, $\hat{\boldsymbol{\mathcal{M}}}$ is a nominal map sample set to the mean of the previous map belief $q_{t-1}^{\boldsymbol{\phi}}(\boldsymbol{\mathcal{M}})$.[3] The term $\log q_t(\mathbf{z}_t^{\text{pose}} \mid \mathbf{u}_{t-1}, H_{t-1})$ is an approximate dynamics prior over the current pose, it makes the pose respect the transition model. The term $\log p(\mathbf{x}_t \mid \hat{\boldsymbol{\mathcal{M}}}, \mathbf{z}_t^{\text{pose}})$ represents reconstructing the current observation, optimising it for the current pose will align the observation to the map. However, evaluating this rendering term in every gradient step is inefficient. Because of this, we replace it with the prediction-to-observation objective used by Kayalibay et al. [54], Nießner et al. [45], Newcombe et al. [84]. We refer to [54] for further motivation and we list the technical details in appendix E.

The above optimisation gives us a MAP pose estimate, which we denote with $\boldsymbol{\mu}_t^{\text{pose}}$. Next, we apply a Laplace approximation [75] around it to obtain a full covariance matrix $\boldsymbol{\Sigma}_t^{\text{pose}}$ which captures the curvature of the objective. This leaves us with a full Gaussian belief over the current pose:

$$q_t^{\boldsymbol{\phi}}(\mathbf{z}_t^{\text{pose}}) = \mathcal{N}(\mathbf{z}_t^{\text{pose}} \mid \boldsymbol{\mu}_t^{\text{pose}}, \boldsymbol{\Sigma}_t^{\text{pose}}).$$

[3] Appendix F discusses this approximation further.



(a) Example mapping and localisation

(b) Blackbird map uncertainty.

(c) EuRoC map uncertainty.

Figure 2: (a) 3D reconstruction, example emission and inferred trajectory for EuRoC/V102 and TUM-RGBD fr3/office. (b) Blackbird experiment. Top-down map uncertainty on the left, *black* is uncertain, *orange* is precise. Precision is highest in a triangle around the center, which is the camera frustum where the agent remains sitting on a platform for a long time, see the orange triangle amidst the map point cloud on the right. (c) Analogous EuRoC experiment. Map uncertainty is high outside of the room, at the center and behind the two structures on the left due to occlusion. The uncertainty in the center is high because the agent primarily looks outwards (view directions in the right image).

Finally, we can combine this Gaussian with the Gaussian velocity conditional $q_t(\mathbf{z}_t^{\text{vel}} \mid \mathbf{z}_t^{\text{pose}}, \mathbf{u}_{t-1}, H_{t-1})$ (the third term in eq. (4)) into a full-state belief in closed form:

$$q_t^{\boldsymbol{\phi}}(\mathbf{z}_t) = \mathcal{N}(\mathbf{z}_t \mid \boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t) = \mathcal{N}(\mathbf{z}_t^{\text{pose}} \mid \boldsymbol{\mu}_t^{\text{pose}}, \boldsymbol{\Sigma}_t^{\text{pose}}) \mathcal{N}\left(\mathbf{z}_t^{\text{vel}} \;\middle|\; \mathbf{D}_t \mathbf{z}_t^{\text{pose}} + \mathbf{e}_t, \boldsymbol{\Sigma}_t^{\mathbf{vel}}\right).$$

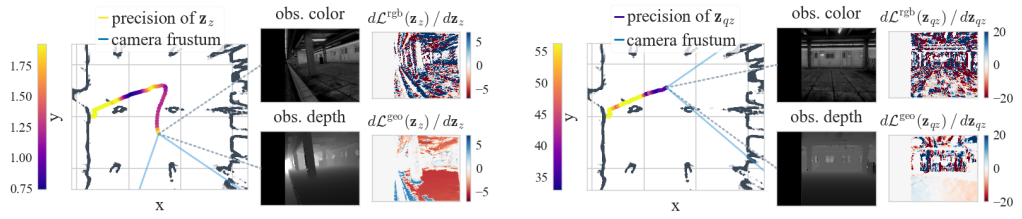
This is approximate, we do it for speed and find it does not harm localisation in practice. Appendix E describes how the linear Gaussian terms come to be in more detail.

5 Experiments

Originally we set out with a few goals: the inference method should be faithful to the generative assumptions, it should quantify uncertainty and it should run in real-time. What follows is an empirical analysis of these aspects. We evaluate on the EuRoC [17], Blackbird [18] and TUM-RGBD [19] data sets. The agent in the former two is an unmanned aerial vehicle (UAV), with speed of up to 4 m/s. For Blackbird, we use Semi-Global Block Matching (SGBM) for stereo depth estimation [85]. For EuRoC, we use the ground-truth Leica MS50 depth readings provided by [10]. We pretend the IMU readings from these data sets are our control inputs. For TUM-RGBD we do not feed in any controls and assume a constant-velocity transition. All experimental details are in appendix G.

5.1 Inference Through a Probabilistic Generative Model

First we look into the synergy between the inference and the generative assumptions. In fig. 2a we see mapping and localisation examples. The inferred scenes are consistent, with no dramatic offsets in geometry. More importantly, rendering from the inferred map using the emission $p(\mathbf{x}_t \mid \mathbf{z}_t, \boldsymbol{\mathcal{M}})$ works as expected (see middle row), indicating that map updates are consistent with the generative assumptions. This is evident from the accuracy of the inferred state trajectories as well (last row), as the pose optimisation objective from section 4.3.2 uses rendered images at every filtering step. A potential discrepancy between the inference and the generative assumptions would lead to errors that would accumulate over time, which is not the case.



(a) x location uncertainty

(b) $\mathbf{q}_z$ (yaw) orientation uncertainty

Figure 3: Inferred state uncertainty. Inferred trajectories are colored by precision (inverse uncertainty) of a certain state dimension, followed by observations, followed by columns of the tracking Jacobian for that same state dimension. (a) Here the precision in z (vertical movement) is high (*yellow*), because the z-orthogonal floor produces a consistent Jacobian (bottom right). (b) Here the precision in $\mathbf{q}_z$ orientation (yaw, azimuth) is low (*violet*), as there are no orthogonal surfaces (i.e. facing sideways). Note the low Jacobian magnitude of the horizontal floor this time (bottom right).

Map uncertainty The inferred map uncertainty is determined by the map updates. We show its interpretable effects in figs. 2b and 2c for two examples, one from Blackbird and another from EuRoC. Our map updates are akin to traditional SDF updates and the main factor that decides whether a map region is certain is how often it was observed. Regions that were occluded by objects, are behind walls or were rarely in view remain uncertain, e.g. as seen in fig. 2c. In contrast, if the agent spends a lot of time looking at a certain map region, the uncertainty there decreases, as seen in fig. 2b.

State uncertainty In fig. 3 we analyse state uncertainty by looking at the variance for individual dimensions. We notice that state uncertainty changes along the trajectory. Uncertainty is determined by what the agent currently sees, based on the geometric relationship between the agent movement and the observed scene (e.g. fig. 3a and fig. 3b). This effect can be explained if we examine the Laplace approximation used to estimate pose covariances. At any given time step, we set the covariance to $\boldsymbol{\Sigma}_t^{\text{pose}} \approx -\mathbf{H}^{-1} \approx -\left(2\mathbf{J}^T\mathbf{J}\right)^{-1}$. Here $\mathbf{H}$ is the Hessian of the tracking objective at the mean pose estimate and $\mathbf{J}$ is the Jacobian. The Jacobian connects the pose to all image pixel errors. The more consistent Jacobian entries are for a given pose dimension, the smaller the variance for that dimension will be. We refer to appendix H for more details about the map and state uncertainty quality.

5.2 Localisation Accuracy

We compare PRISM's localisation to state-of-the-art methods in moderately-sized indoor environments. We consider both baselines with dense maps (TANDEM [10], VSSM-LM [15], iMAP [11], NICE-SLAM [9], CodeVIO [86]) and sparse methods without rendering (ORB-SLAM2 [4], VINS [71], VIMO [70]). The results are in table 1. For the considered trajectories accuracy is comparable to the baselines, with differences of a few centimeters. At the same time, our inference boasts a predictive state-space model with both rendering and dynamics as well as uncertainty estimates, which is not common in the dense visual SLAM literature. Finally, in fig. 4 we see example inferred agent velocities, noting the uncertainty bands. This is possible because we model the agent dynamics.

Our localisation accuracy on Blackbird is better than the off-line variational inference results of VSSM-LM presented by Mirchev et al. [15], and at the same time our solution runs in real-time and also captures uncertainty. This shows the advantages of the proposed divide-and-conquer filtering.

5.3 Approximations for Runtime Improvement

All of our approximations are motivated by the real-time constraint, dictating the need for closed-form map updates, a Laplace approximation, linearisation assumptions and a surrogate pose optimisation objective. Figure 5 shows a runtime breakdown for different image resolutions, measured on an

[4] Last 10 s are skipped, as the drone hits the ground during landing.

Table 1: Localisation absolute error RMSE in meters on EuRoC [17], Blackbird [18] and TUM-RGBD [19].

Trajectory	Ours	Code VIO	TANDEM	ORB SLAM2
EuRoC/V101	0.041 (± 0.002)	0.05	0.09	**0.031**
EuRoC/V102	0.035 (± 0.002)	0.07	0.17	**0.02**
EuRoC/V103	**0.042 (± 0.002)**	0.07	-	0.048
EuRoC/V201	**0.037 (± 0.001)**	0.10	0.09	**0.037**
EuRoC/V202	**0.035 (± 0.003)**	0.06	0.12	**0.035**
EuRoC/V203	x	**0.275**	-	x

Trajectory	Ours	VSSM LM	VIMO	VINS
picasso, 1 m/s	0.064 (± 0.003)	0.139	**0.055**	0.097
picasso, 2 m/s	0.053 (± 0.003)	0.136	**0.040**	0.043
picasso, 3 m/s	0.061 (± 0.003)	0.120	**0.043**	0.045
picasso, 4 m/s	0.079 (± 0.005)[4]	0.174	**0.049**	0.056
star, 1 m/s	0.089 (± 0.007)[4]	0.137	**0.088**	0.102
star, 2 m/s	0.111 (± 0.009)	0.163	**0.082**	0.133
star, 3 m/s	**0.115 (± 0.012)**	0.281	0.183	0.235
star, 4 m/s	**0.153 (± 0.015)**[4]	0.156	x	x

Trajectory	Ours	iMAP	NICE SLAM	ORB SLAM2*
fr1/desk	0.053 (± 0.003)	0.049	0.027	**0.016**
fr2/xyz	0.029 (± 0.001)	0.02	**0.018**	0.04
fr3/office	0.083 (± 0.001)	0.058	0.03	**0.01**

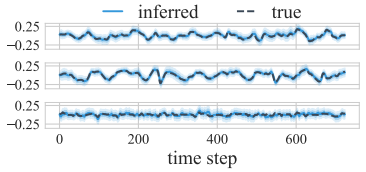


Figure 4: Inferred xyz-velocity.

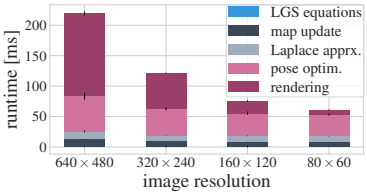


Figure 5: Runtime breakdown.

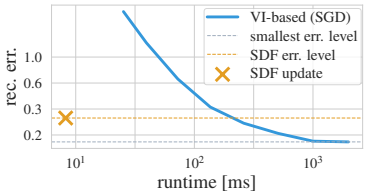


Figure 6: Mapping comparison.

NVIDIA 1080 Ti GPU and an Intel(R) Xeon(R) W-2123 CPU at 3.6 GHz. The heaviest operations are rendering and the gradient-based pose optimisation. Based on movement speed, rendering can happen periodically, whenever a new anchor image prediction for pose optimisation is needed. This leaves us with a total runtime of 10 Hz to 15 Hz, updating the map and state at every data step. In fig. 6 we also compare closed-form map updates to map inference via gradient-descent (e.g. as in [15, 8, 11, 9]). While gradient-descent is more accurate on a bigger budget, it is much more expensive. For example, to match the accuracy of the closed-form updates, which take less than 10 ms, one would need ca. 250 ms of optimisation, which is impractical. These runtimes are for a voxel grid that is significantly faster than neural representations [54], which would only exacerbate the problem.

6 Limitations and Conclusion

SDF voxel grids allow for closed-form updates, but their memory footprint limits the maximum resolution and scene size. Voxel hashing [45] or octrees [87] can directly replace them for memory efficiency. Neural maps and dynamically changing maps have remained out of our scope. Their probabilistic formulation and closed-form updates require further investigation. Our map factorises over voxels with no inter-region correlation, which could also be improved. PRISM provides interpretable uncertainty in real-time, but estimation is approximate. Obtaining perfectly calibrated uncertainty on a budget remains an open question (see appendix H). While filtering works for our generative assumptions indoors, filters cannot revisit past errors and can drift in large scenes with high levels of exploration [6]. We leave large-scale inference considerations for future work.

We have introduced PRISM, a method for probabilistic filtering in a predefined spatial state-space model. Our solution runs in real-time, provides state and map uncertainty, and infers a dense map and a 6-DoF state trajectory with velocities. It is comparably accurate to state-of-the-art SLAM in indoor environments. To the best of our knowledge this is the first real-time fully-probabilistic solution for SLAM that combines differentiable rendering and agent dynamics. We validated our method on three challenging data sets, featuring unmanned aerial vehicles and a handheld camera. The results are promising, establishing PRISM as a viable state estimator for downstream model-based control.

Acknowledgments

We thank our reviewers for the thoughtful discussion, it helped us to better position our contribution.

References

[1] D. Koller and N. Friedman. *Probabilistic Graphical Models - Principles and Techniques*. MIT Press, 2009.

[2] D. P. Bertsekas. *Dynamic programming and optimal control, 3rd Edition*. Athena Scientific, 2005. ISBN 1886529264. URL http://www.worldcat.org/oclc/314894080.

[3] J. Engel, T. Schöps, and D. Cremers. Lsd-slam: Large-scale direct monocular slam. In *European conference on computer vision*, pages 834–849. Springer, 2014.

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

[5] J. Engel, V. Koltun, and D. Cremers. Direct sparse odometry. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Mar. 2018.

[6] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. J. Leonard. Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. *IEEE Transactions on robotics*, 32(6):1309–1332, 2016.

[7] F. R. Kschischang, B. J. Frey, and H.-A. Loeliger. Factor graphs and the sum-product algorithm. *IEEE Transactions on information theory*, 47(2):498–519, 2001.

[8] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *ECCV*, 2020.

[9] Z. Zhu, S. Peng, V. Larsson, W. Xu, H. Bao, Z. Cui, M. R. Oswald, and M. Pollefeys. Nice-slam: Neural implicit scalable encoding for slam. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2022.

[10] L. Koestler, N. Yang, N. Zeller, and D. Cremers. Tandem: Tracking and dense mapping in real-time using deep multi-view stereo. In *Conference on Robot Learning (CoRL)*, 2021.

[11] E. Sucar, S. Liu, J. Ortiz, and A. J. Davison. imap: Implicit mapping and positioning in real-time, 2021.

[12] R. E. Kalman et al. A new approach to linear filtering and prediction problems [j]. *Journal of basic Engineering*, 82(1):35–45, 1960.

[13] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit. Fastslam: A factored solution to the simultaneous localization and mapping problem. In *AAAI/IAAI*, 2002.

[14] G. Grisetti, C. Stachniss, and W. Burgard. Improved techniques for grid mapping with rao-blackwellized particle filters. *IEEE transactions on Robotics*, 23(1):34–46, 2007.

[15] A. Mirchev, B. Kayalibay, P. van der Smagt, and J. Bayer. Variational state-space models for localisation and dense 3d mapping in 6 dof. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=XAS3uKeFWj.

[16] S. Särkkä. *Bayesian Filtering and Smoothing*. Number 3. Cambridge University Press, USA, 2013. ISBN 1107619289.

[17] M. Burri, J. Nikolic, P. Gohl, T. Schneider, J. Rehder, S. Omari, M. W. Achtelik, and R. Siegwart. The euroc micro aerial vehicle datasets. *The International Journal of Robotics Research*, 35 (10):1157–1163, 2016.

[18] A. Antonini, W. Guerra, V. Murali, T. Sayre-McCord, and S. Karaman. The blackbird uav dataset. *The International Journal of Robotics Research*, 0(0):0278364920908331, 2020. doi:10.1177/0278364920908331.

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

[20] L. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989. doi:10.1109/5.18626.

[21] M. Karl, M. Soelch, J. Bayer, and P. van der Smagt. Deep variational bayes filters: Unsupervised learning of state space models from raw data. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

[22] R. G. Krishnan, U. Shalit, and D. Sontag. Deep kalman filters. *arXiv preprint arXiv:1511.05121*, 2015.

[23] J. Chung, K. Kastner, L. Dinh, K. Goel, A. C. Courville, and Y. Bengio. A recurrent latent variable model for sequential data. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015. URL https://proceedings.neurips.cc/paper/2015/file/b618c3210e934362ac261db280128c22-Paper.pdf.

[24] S. Chiappa, S. Racanière, D. Wierstra, and S. Mohamed. Recurrent environment simulators. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. URL https://openreview.net/forum?id=B1s6xvqlx.

[25] D. Ha and J. Schmidhuber. Recurrent world models facilitate policy evolution. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 2450–2462. Curran Associates, Inc., 2018.

[26] M. Deisenroth and C. E. Rasmussen. Pilco: A model-based and data-efficient approach to policy search. In *Proceedings of the 28th International Conference on machine learning (ICML-11)*, pages 465–472. Citeseer, 2011.

[27] D. Hafner, T. Lillicrap, I. Fischer, R. Villegas, D. Ha, H. Lee, and J. Davidson. Learning latent dynamics for planning from pixels. In *International Conference on Machine Learning*, pages 2555–2565. PMLR, 2019.

[28] D. Hafner, T. Lillicrap, J. Ba, and M. Norouzi. Dream to control: Learning behaviors by latent imagination. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=S1lOTC4tDS.

[29] A. X. Lee, A. Nagabandi, P. Abbeel, and S. Levine. Stochastic latent actor-critic: Deep reinforcement learning with a latent variable model. *Advances in Neural Information Processing Systems*, 33:741–752, 2020.

[30] P. Becker-Ehmck, M. Karl, J. Peters, and P. van der Smagt. Learning to fly via deep model-based reinforcement learning, 2020.

[31] M. Karl, M. Soelch, P. Becker-Ehmck, D. Benbouzid, P. van der Smagt, and J. Bayer. Unsupervised real-time control through variational empowerment. *arXiv preprint arXiv:1710.05101*, 2017.

[32] D. Corneil, W. Gerstner, and J. Brea. Efficient model-based deep reinforcement learning with variational state tabulation. volume 80 of *Proceedings of Machine Learning Research*, pages 1049–1058, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR.

[33] K. Chua, R. Calandra, R. McAllister, and S. Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. In *Advances in Neural Information Processing Systems*, pages 4754–4765, 2018.

[34] M. Fraccaro, D. J. Rezende, Y. Zwols, A. Pritzel, S. M. A. Eslami, and F. Viola. Generative temporal models with spatial memory for partially observed environments. *CoRR*, abs/1804.09401, 2018.

[35] S. M. A. Eslami, D. Jimenez Rezende, F. Besse, F. Viola, A. S. Morcos, M. Garnelo, A. Ruderman, A. A. Rusu, I. Danihelka, K. Gregor, D. P. Reichert, L. Buesing, T. Weber, O. Vinyals, D. Rosenbaum, N. Rabinowitz, H. King, C. Hillier, M. Botvinick, D. Wierstra, K. Kavukcuoglu, and D. Hassabis. Neural scene representation and rendering. *Science*, 360(6394):1204–1210, 2018. doi:10.1126/science.aar6170. URL https://science.sciencemag.org/content/360/6394/1204.

[36] K. Gregor, D. J. Rezende, F. Besse, Y. Wu, H. Merzic, and A. van den Oord. Shaping belief states with generative environment models for rl. In *Advances in Neural Information Processing Systems*, pages 13475–13487, 2019.

[37] A. Mirchev, B. Kayalibay, M. Soelch, P. van der Smagt, and J. Bayer. Approximate bayesian inference in spatial environments. In *Proceedings of Robotics: Science and Systems*, Freiburgim-Breisgau, Germany, June 2019.

[38] S. Gupta, V. Tolani, J. Davidson, S. Levine, R. Sukthankar, and J. Malik. Cognitive mapping and planning for visual navigation, 2019.

[39] E. Parisotto and R. Salakhutdinov. Neural map: Structured memory for deep reinforcement learning. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=Bk9zbyZCZ.

[40] D. S. Chaplot, D. Gandhi, S. Gupta, A. Gupta, and R. Salakhutdinov. Learning to explore using active neural slam. In *International Conference on Learning Representations (ICLR)*, 2020.

[41] B. Curless and M. Levoy. A volumetric method for building complex models from range images. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 303–312, 1996.

[42] J. C. Carr, R. K. Beatson, J. B. Cherrie, T. J. Mitchell, W. R. Fright, B. C. McCallum, and T. R. Evans. Reconstruction and representation of 3d objects with radial basis functions. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 67–76, 2001.

[43] R. A. Newcombe, S. Lovegrove, and A. J. Davison. Dtam: Dense tracking and mapping in real-time. *2011 International Conference on Computer Vision*, pages 2320–2327, 2011.

[44] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohi, J. Shotton, S. Hodges, and A. Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In *2011 10th IEEE international symposium on mixed and augmented reality*, pages 127–136. IEEE, 2011.

[45] M. Nießner, M. Zollhöfer, S. Izadi, and M. Stamminger. Real-time 3d reconstruction at scale using voxel hashing. *ACM Transactions on Graphics (ToG)*, 32(6):1–11, 2013.

[46] M. Keller, D. Lefloch, M. Lambers, S. Izadi, T. Weyrich, and A. Kolb. Real-time 3d reconstruction in dynamic scenes using point-based fusion. In *2013 International Conference on 3D Vision - 3DV 2013*, pages 1–8, 2013. doi:10.1109/3DV.2013.9.

[47] T. Whelan, S. Leutenegger, R. F. Salas-Moreno, B. Glocker, and A. J. Davison. Elasticfusion: Dense slam without a pose graph. In *Robotics: Science and Systems*, 2015.

[48] Y.-P. Cao, L. P. Kobbelt, and S. Hu. Real-time high-accuracy three-dimensional reconstruction with consumer rgb-d cameras. *ACM Transactions on Graphics (TOG)*, 37:1 – 16, 2018.

[49] Y. Xu, L. Nan, L. Zhou, J. Wang, and C. C. Wang. Hrbf-fusion: Accurate 3d reconstruction from rgb-d data using on-the-fly implicits. *ACM Transactions on Graphics (TOG)*, 41(3):1–19, 2022.

[50] Y. Chen and G. Medioni. Object modelling by registration of multiple range images. *Image and vision computing*, 10(3):145–155, 1992.

[51] F. Steinbrücker, J. Sturm, and D. Cremers. Real-time visual odometry from dense rgb-d images. In *2011 IEEE international conference on computer vision workshops (ICCV Workshops)*, pages 719–722. IEEE, 2011.

[52] C. Audras, A. Comport, M. Meilland, and P. Rives. Real-time dense appearance-based slam for rgb-d sensors. In *Australasian Conf. on Robotics and Automation*, volume 2, pages 2–2, 2011.

[53] C. Kerl, J. Sturm, and D. Cremers. Dense visual slam for rgb-d cameras. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2100–2106. IEEE, 2013.

[54] B. Kayalibay, A. Mirchev, P. van der Smagt, and J. Bayer. Tracking and planning with spatial world models. *arXiv preprint arXiv:2201.10335*, 2022.

[55] V. Sitzmann, J. Martel, A. Bergman, D. Lindell, and G. Wetzstein. Implicit neural representations with periodic activation functions. *Advances in Neural Information Processing Systems*, 33, 2020.

[56] J. Ortiz, A. Clegg, J. Dong, E. Sucar, D. Novotny, M. Zollhoefer, and M. Mukadam. isdf: Real-time neural signed distance fields for robot perception. *arXiv preprint arXiv:2204.02296*, 2022.

[57] L. Yen-Chen, P. Florence, J. T. Barron, A. Rodriguez, P. Isola, and T.-Y. Lin. Inerf: Inverting neural radiance fields for pose estimation, 2021.

[58] Z. Wang, S. Wu, W. Xie, M. Chen, and V. A. Prisacariu. Nerf–: Neural radiance fields without known camera parameters, 2021.

[59] C. Reiser, S. Peng, Y. Liao, and A. Geiger. Kilonerf: Speeding up neural radiance fields with thousands of tiny mlps, 2021.

[60] S. Lombardi, T. Simon, J. Saragih, G. Schwartz, A. Lehrmann, and Y. Sheikh. Neural volumes. *ACM Transactions on Graphics*, 38(4):1–14, Jul 2019. ISSN 1557-7368. doi:10.1145/3306346.3323020. URL http://dx.doi.org/10.1145/3306346.3323020.

[61] L. Liu, J. Gu, K. Z. Lin, T.-S. Chua, and C. Theobalt. Neural sparse voxel fields. *NeurIPS*, 2020.

[62] S. Lombardi, T. Simon, G. Schwartz, M. Zollhoefer, Y. Sheikh, and J. Saragih. Mixture of volumetric primitives for efficient neural rendering. *ACM Trans. Graph.*, 40(4), jul 2021. ISSN 0730-0301. doi:10.1145/3450626.3459863. URL https://doi.org/10.1145/3450626.3459863.

[63] T. Müller, A. Evans, C. Schied, and A. Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Trans. Graph.*, 41(4):102:1–102:15, July 2022. doi:10.1145/3528223.3530127. URL https://doi.org/10.1145/3528223.3530127.

[64] H. Moravec and A. Elfes. High resolution maps from wide angle sonar. In *Proceedings. 1985 IEEE International Conference on Robotics and Automation*, volume 2, pages 116–121. IEEE, 1985.

[65] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press, 2005. ISBN 0262201623.

[66] K. P. Murphy. Bayesian map learning in dynamic environments. In *Advances in Neural Information Processing Systems 12, [NIPS Conference, Denver, Colorado, USA, November 29 - December 4, 1999]*, pages 1015–1021, 1999.

[67] D. Hahnel, W. Burgard, D. Fox, and S. Thrun. An efficient fastslam algorithm for generating maps of large-scale cyclic environments from raw laser range measurements. In *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003) (Cat. No.03CH37453)*, volume 1, pages 206–211 vol.1, 2003. doi:10.1109/IROS.2003.1250629.

[68] M. Montemerlo, S. Thrun, D. Koller, B. Wegbreit, et al. Fastslam 2.0: An improved particle filtering algorithm for simultaneous localization and mapping that provably converges. In *IJCAI*, volume 3, pages 1151–1156, 2003.

[69] G. Grisetti, C. Stachniss, and W. Burgard. Improving grid-based slam with rao-blackwellized particle filters by adaptive proposals and selective resampling. *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, pages 2432–2437, 2005.

[70] B. Nisar, P. Foehn, D. Falanga, and D. Scaramuzza. Vimo: Simultaneous visual inertial model-based odometry and force estimation. In *Proceedings of Robotics: Science and Systems*, FreiburgimBreisgau, Germany, June 2019.

[71] T. Qin, P. Li, and S. Shen. Vins-mono: A robust and versatile monocular visual-inertial state estimator. *IEEE Transactions on Robotics*, 34(4):1004–1020, 2018.

[72] A. Rosinol, T. Sattler, M. Pollefeys, and L. Carlone. Incremental visual-inertial 3d mesh generation with structural regularities. In *2019 International Conference on Robotics and Automation (ICRA)*, 2019.

[73] A. Rosinol, M. Abate, Y. Chang, and L. Carlone. Kimera: an open-source library for real-time metric-semantic localization and mapping. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2020. URL `https://github.com/MIT-SPARK/Kimera`.

[74] H. Strasdat, J. M. Montiel, and A. J. Davison. Visual slam: why filter? *Image and Vision Computing*, 30(2):65–77, 2012.

[75] P. S. Laplace. Memoir on the probability of the causes of events. *Statistical Science*, 1(3): 364–378, 1986. ISSN 08834237. URL `http://www.jstor.org/stable/2245476`.

[76] R. Senanayake and F. Ramos. Bayesian hilbert maps for dynamic continuous occupancy mapping. In *Proceedings of the 1st Annual Conference on Robot Learning*, volume 78 of *Proceedings of Machine Learning Research*, pages 458–471. PMLR, 13–15 Nov 2017.

[77] C. Hernández, G. Vogiatzis, and R. Cipolla. Probabilistic visibility for multi-view stereo. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2007.

[78] W. Dong, Q. Wang, X. Wang, and H. Zha. Psdf fusion: Probabilistic signed distance function for on-the-fly 3d data fusion and scene reconstruction. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 701–717, 2018.

[79] M. Opper and O. Winther. A bayesian approach to on-line learning. 1999.

[80] C. M. Bishop. *Pattern recognition and machine learning*. springer, 2006.

[81] S. Parker, P. Shirley, Y. Livnat, C. Hansen, and P.-P. Sloan. Interactive ray tracing for isosurface rendering. In *Proceedings Visualization'98 (Cat. No. 98CB36276)*, pages 233–238. IEEE, 1998.

[82] D. P. Kingma and M. Welling. Auto-encoding variational bayes. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014. URL http://arxiv.org/abs/1312.6114.

[83] D. M. Blei, A. Kucukelbir, and J. D. McAuliffe. Variational inference: A review for statisticians. *Journal of the American statistical Association*, 112(518):859–877, 2017.

[84] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohli, J. Shotton, S. Hodges, and A. W. Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. *2011 10th IEEE International Symposium on Mixed and Augmented Reality*, pages 127–136, 2011.

[85] H. Hirschmuller. Stereo processing by semiglobal matching and mutual information. *IEEE Transactions on pattern analysis and machine intelligence*, 30(2):328–341, 2007.

[86] X. Zuo, N. Merrill, W. Li, Y. Liu, M. Pollefeys, and G. Huang. Codevio: Visual-inertial odometry with learned optimizable dense depth. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 14382–14388. IEEE, 2021.

[87] F. Steinbrucker, C. Kerl, and D. Cremers. Large-scale multi-resolution surface reconstruction from rgb-d sequences. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3264–3271, 2013.

[88] J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne, and Q. Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL http://github.com/google/jax.

[89] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

[90] L. V. Jospin, H. Laga, F. Boussaid, W. Buntine, and M. Bennamoun. Hands-on bayesian neural networks—a tutorial for deep learning users. *IEEE Computational Intelligence Magazine*, 17 (2):29–48, 2022.

[91] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

A The Cost of Maintaining a Joint Filter

In section 4.2, we mention that Rao-Blackwellised or full-covariance Gaussian representations for the joint $p(\boldsymbol{\mathcal{M}}, \mathbf{z}_t \mid H_t)$ are difficult due to the large number of parameters in the dense 3D maps. For example, a joint Gaussian distribution of $\boldsymbol{\mathcal{M}}$ and $\mathbf{z}_t$ would require $\mathcal{O}((n_{\mathbf{z}} + n_{\boldsymbol{\mathcal{M}}})^2)$ parameters, where $n_{\mathbf{z}}$ is the size of a single state (6 degrees of freedom for the pose plus 6 degrees of freedom for the velocity) and $n_{\boldsymbol{\mathcal{M}}}$ is the size of a map (e.g. a voxel grid of size $200 \times 200 \times 200$). Similarly, a Rao-Blackwellised particle representation would require $\mathcal{O}(P(n_{\mathbf{z}} + n_{\boldsymbol{\mathcal{M}}}))$, where $P \gg 1$ is a very large number of particles, leading to billions of parameters. This is because every particle would carry its own individual map, and many particles are needed to properly cover the 6-DoF state space. In these cases both the memory and the necessary computation to process all the data are prohibitive for real-time operation.

As a workaround, we choose to approximate the individual marginal distributions. Framing the problem in this way helps with separation of concern and allows us to more easily incorporate traditional inference techniques (c.f. sections 4.3.1 and 4.3.2 and appendices C and E). It is worth noting that the product of the two marginals $q_t^{\boldsymbol{\phi}}(\boldsymbol{\mathcal{M}})$ and $q_t^{\boldsymbol{\phi}}(\mathbf{z}_t)$ is not necessarily an optimal approximation of the joint. As each approximate filter targets a marginal true posterior, their product is not directly optimised as a mean-field approximation [80]. The posterior approximation and particular derivation paths we have chosen is only one way (out of many) to frame the problem which we have found convenient to derive a practical solution.

One could attempt to sample from the joint, using one of the maintained marginal filters as a starting point. For example, consider the following joint factorisation

$$p(\boldsymbol{\mathcal{M}}, \mathbf{z}_t \mid H_t) = p(\boldsymbol{\mathcal{M}} \mid H_t) p(\mathbf{z}_t \mid \boldsymbol{\mathcal{M}}, H_t).$$

To sample from it (e.g. via importance sampling [80]), we would need to compute the term $p(\mathbf{z}_t \mid \boldsymbol{\mathcal{M}}, H_t)$ up to a normalising constant:

$$\begin{aligned}
p(\mathbf{z}_t \mid \boldsymbol{\mathcal{M}}, H_t) &\propto p(\mathbf{x}_t \mid \boldsymbol{\mathcal{M}}, \mathbf{z}_t) p(\mathbf{z}_t \mid \boldsymbol{\mathcal{M}}, \mathbf{x}_{1:t-1}, \mathbf{u}_{1:t-1}) \\
&= p(\mathbf{x}_t \mid \boldsymbol{\mathcal{M}}, \mathbf{z}_t) \int p(\mathbf{z}_t, \mathbf{z}_{t-1} \mid \boldsymbol{\mathcal{M}}, \mathbf{x}_{1:t-1}, \mathbf{u}_{1:t-1}) \, \mathrm{d}\mathbf{z}_{t-1} \\
&= p(\mathbf{x}_t \mid \boldsymbol{\mathcal{M}}, \mathbf{z}_t) \int p(\mathbf{z}_t \mid \mathbf{z}_{t-1}, \mathbf{u}_{t-1}, \boldsymbol{\mathcal{M}}, \cancel{\mathbf{x}_{1:t-1}, \mathbf{u}_{1:t-2}}) \\
&\qquad \times p(\mathbf{z}_{t-1} \mid \boldsymbol{\mathcal{M}}, \underbrace{\mathbf{x}_{1:t-1}, \mathbf{u}_{1:t-2}}_{H_{t-1}}, \cancel{\mathbf{u}_{t-1}}) \, \mathrm{d}\mathbf{z}_{t-1} \\
&= p(\mathbf{x}_t \mid \boldsymbol{\mathcal{M}}, \mathbf{z}_t) \int p(\mathbf{z}_t \mid \mathbf{z}_{t-1}, \mathbf{u}_{t-1}) p(\mathbf{z}_{t-1} \mid \boldsymbol{\mathcal{M}}, H_{t-1}) \, \mathrm{d}\mathbf{z}_{t-1}. \qquad (6)
\end{aligned}$$

The above reveals a recursive expression for the evaluation of $p(\mathbf{z}_t \mid \boldsymbol{\mathcal{M}}, H_t)$. Therefore, computing this term would require going back the Markov chain to the beginning of the sequence, which is inefficient. One could try to maintain cached approximations of $p(\mathbf{z}_t \mid \boldsymbol{\mathcal{M}}, H_t)$ at every time step (e.g. through weighted $\mathbf{z}_t$ particles, for different $\boldsymbol{\mathcal{M}}$), but this is encumbered by the large dimensionality of the map and states. The alternative filter factorisation $p(\boldsymbol{\mathcal{M}}, \mathbf{z}_t \mid H_t) = p(\mathbf{z}_t \mid H_t) p(\boldsymbol{\mathcal{M}} \mid \mathbf{z}_t, H_t)$ leads to an analogous problem. Since we target real-time inference, we opt for maintaining approximations to only the marginal filters instead. Further considerations about the joint posterior are left for future work.

B Details of the Generative Model

We follow the generative assumptions of Mirchev et al. [15]. The map prior is a 3D voxel grid of occupancy and color and factorises over voxels:

$$p(\boldsymbol{\mathcal{M}}) = \prod_{ijk} \mathcal{N}\big(\boldsymbol{\mathcal{M}}_{ijk} \mid \boldsymbol{\mu}_{ijk}, \sigma^2 \mathbf{I}\big).$$

Each map cell $\boldsymbol{\mathcal{M}}_{ijk} \in \mathbb{R}^4$ contains an SDF value and three RGB values. We assume an uninformed (very broad) prior, setting $\sigma \gg 1$. We set the very first approximate map posterior to the prior (in the absence of data) and recusively apply updates to it as new data arrives.

Rendering from the map is captured by the emission model $p(\mathbf{x} \mid \boldsymbol{\mathcal{M}}, \mathbf{z})$. First, a bundle of rays are cast inside the camera frustum, using the pose $\mathbf{z}^{\text{pose}}$ to position them in space [5]. Each ray point can be expressed as an offset from the camera center along a ray direction:

$$\mathbf{p}^{ij} = \mathbf{c} + d\mathbf{r}^{ij}.$$

Here ij runs over pixels, and d is the depth of the point and determines the length of the offset along the ray direction $\mathbf{r}^{ij}$. For every ray, a discrete set of K ray points $\{\mathbf{p}_k^{ij}\}_{[K]}$ is formed, using equidistantly spaced depth offsets $d \in \{\epsilon, 2\epsilon, 3\epsilon, \dots\}$. For all ray points in the frustum, occupancy and color are obtained by evaluating the occupancy and color field $f_{\boldsymbol{\mathcal{M}}} : \mathbb{R}^3 \to \mathbb{R}^4$ parameterised by $\boldsymbol{\mathcal{M}}$. This is done by trilinearly interpolating the cells of $\boldsymbol{\mathcal{M}}$. Next, search is performed along each ray, finding the first point $\mathbf{p}_k^{ij}$ for which the occupancy exceeds a threshold τ (in our implementation $\tau = 0$).[6] This approximately finds the first intersection with a surface along the ray, but points along the ray are discrete. To predict the depth of the surface more accurately, linear interpolation based on the occupancy values is used:

$$d^* = \alpha d_k + (1-\alpha) d_{k-1}, \qquad \alpha = \frac{\tau - f_{\text{occ}}(\mathbf{p}_{k-1}^{ij})}{f_{\text{occ}}(\mathbf{p}_k^{ij}) - f_{\text{occ}}(\mathbf{p}_{k-1}^{ij})}.$$

Here $\mathbf{p}_{k-1}^{ij}$ is the point preceding the surface, and $\mathbf{p}_k^{ij}$ the point after it (identified by the ray search).

The linear interpolation of distance to the surface based on map content matches the assumptions of signed distance function representations (SDF) [41]. If we consider a single ray in isolation, the SDF value would equal the signed distance of ray points to the surface, i.e. predicted depth along the ray would be a linear function of the SDF values as well (identity), just like in the above equation. This property of the generative renderer allows us to use closed-form map updates that are alike traditional SDF updates without sacrificing accuracy. Therefore, we treat occupancy like an SDF with a flipped sign (positive inside objects, negative outside). Appendix D provides the details of the probabilistic map updates.

Color predictions are evaluated analogously to depth for each ray. The predicted depth and color are combined into an RGB-D image mean $\boldsymbol{\mu}_{\text{rgbd}}$, which is used to parameterise a Laplace distribution:

$$p(\mathbf{x} \mid \boldsymbol{\mathcal{M}}, \mathbf{z}) = \text{Laplace}(\mathbf{x} \mid \boldsymbol{\mu}_{\text{rgbd}}, \text{diag}(\boldsymbol{\sigma}_E)).$$

The emission model is differentiable–$\boldsymbol{\mathcal{M}}$ and $\mathbf{z}$ can be optimised through it with gradient descent. However, in this work we avoid using this gradient path, as it is too expensive for real-time inference.

The transition model $p(\mathbf{z}_{t+1} \mid \mathbf{z}_t, \mathbf{u}_t)$ is defined as Euler integration of acceleration and velocity:

$$\begin{aligned} p(\mathbf{z}_{t+1} \mid \mathbf{z}_t, \mathbf{u}_t) = \mathcal{N}\big(\mathbf{z}_{t+1}^{\text{vel}} \;\big|\; f_{\text{vel}}(\mathbf{z}_t^{\text{vel}}, \mathbf{u}_t), \text{diag}(\boldsymbol{\sigma}^{\text{vel}})^2\big) \\ \times \mathcal{N}\big(\mathbf{z}_{t+1}^{\text{pose}} \;\big|\; f_{\text{pose}}(\mathbf{z}_t^{\text{pose}}, \mathbf{z}_{t+1}^{\text{vel}}), \text{diag}(\boldsymbol{\sigma}^{\text{pose}})^2\big). \end{aligned}$$

Here f_{vel} denotes acceleration integration and f_{pose} denotes velocity integration. We deviate from the formulation in the original paper, and use the velocity of step t instead of $t-1$ to obtain the pose as step t. We find this leads to a more convenient implementation when inferring velocity in a filtering setup. We discuss this factorisation further in appendix E.1, where we introduce the assumed linearisation of the transition.

We refer to the original paper for further details concerning the generative model.

[5] $\mathbf{x}$ is conditionally independent of $\mathbf{z}^{\text{vel}}$ given $\boldsymbol{\mathcal{M}}, \mathbf{z}^{\text{pose}}$.

[6] We assume occupancy is continuous, deviating from the traditional $\{0, 1\}$ Bernoulli definition.

C The Marginal Map Filter

Here we derive the approximation of the true marginal map filter:

$$
\begin{aligned}
p(\boldsymbol{\mathcal{M}} \mid H_t) &= \int p(\boldsymbol{\mathcal{M}}, \mathbf{z}_t \mid H_t)\, \mathrm{d}\mathbf{z}_t \\
&\propto \int p(\mathbf{x}_t \mid \mathbf{z}_t, \boldsymbol{\mathcal{M}}) \int p(\mathbf{z}_t \mid \mathbf{z}_{t-1}, \mathbf{u}_{t-1}) p(\boldsymbol{\mathcal{M}}, \mathbf{z}_{t-1} \mid H_{t-1})\, \mathrm{d}\mathbf{z}_{t-1}\, \mathrm{d}\mathbf{z}_t \\
&\approx \int p(\mathbf{x}_t \mid \mathbf{z}_t, \boldsymbol{\mathcal{M}}) \int p(\mathbf{z}_t \mid \mathbf{z}_{t-1}, \mathbf{u}_{t-1}) q^{\boldsymbol{\phi}}_{t-1}(\boldsymbol{\mathcal{M}}) q^{\boldsymbol{\phi}}_{t-1}(\mathbf{z}_{t-1})\, \mathrm{d}\mathbf{z}_{t-1}\, \mathrm{d}\mathbf{z}_t \qquad (7) \\
&\approx p(\mathbf{x}_t \mid \hat{\mathbf{z}}_t, \boldsymbol{\mathcal{M}}) \times q^{\boldsymbol{\phi}}_{t-1}(\boldsymbol{\mathcal{M}}) \qquad (8) \\
&\approx q(\boldsymbol{\mathcal{M}} \mid \mathbf{x}_t, \hat{\mathbf{z}}_t) \times q^{\boldsymbol{\phi}}_{t-1}(\boldsymbol{\mathcal{M}}) =: q^{\boldsymbol{\phi}}_t(\boldsymbol{\mathcal{M}}). \qquad (9)
\end{aligned}
$$

We begin by applying eq. (1) directly. The first approximation we make is in eq. (7), substituting the true previous joint filter $p(\boldsymbol{\mathcal{M}}, \mathbf{z}_{t-1} \mid H_{t-1})$ for $q^{\boldsymbol{\phi}}_{t-1}(\boldsymbol{\mathcal{M}}) q^{\boldsymbol{\phi}}_{t-1}(\mathbf{z}_{t-1})$. We do this for speed, sacrificing some modelling accuracy for the sake of directly reusing the previous marginal estimates. Appendix A discusses why using an approximation of the joint here is difficult. Next, in eq. (8) we use a single-sample MC approximation of the integral of $\mathbf{z}_t$. The nominal value $\hat{\mathbf{z}}_t$ we set to the mean of the current approximate state filter $q^{\boldsymbol{\phi}}_t(\mathbf{z}_t)$. Accepting some bias, we also do this for speed, as this is the best guess for $\mathbf{z}_t$ available at step t without extra computation. Our next approximation is the term $q(\boldsymbol{\mathcal{M}} \mid \mathbf{x}_t, \hat{\mathbf{z}}_t)$ in eq. (9), which represents a closed-form map update. Intuitively, it approximately inverts the emission $p(\mathbf{x}_t \mid \hat{\mathbf{z}}_t, \boldsymbol{\mathcal{M}})$ and populates the map with the content necessary for reconstructing the observation $\mathbf{x}_t$.

To understand how the map update comes to be, consider the following:

$$
\begin{aligned}
p(\mathbf{x}_t \mid \hat{\mathbf{z}}_t, \boldsymbol{\mathcal{M}}) &= p(\mathbf{x}_t, \boldsymbol{\mathcal{M}} \mid \hat{\mathbf{z}}_t) p(\boldsymbol{\mathcal{M}})^{-1} \\
&= p(\boldsymbol{\mathcal{M}} \mid \mathbf{x}_t, \hat{\mathbf{z}}_t) \underbrace{p(\mathbf{x}_t \mid \hat{\mathbf{z}}_t)}_{\text{const in } \boldsymbol{\mathcal{M}}} p(\boldsymbol{\mathcal{M}})^{-1} \\
&= p(\boldsymbol{\mathcal{M}} \mid \mathbf{x}_t, \hat{\mathbf{z}}_t) \times p(\boldsymbol{\mathcal{M}})^{-1} \times \text{const} \\
&\approx q(\boldsymbol{\mathcal{M}} \mid \mathbf{x}_t, \hat{\mathbf{z}}_t),
\end{aligned}
$$

where $q(\boldsymbol{\mathcal{M}} \mid \mathbf{x}_t, \hat{\mathbf{z}}_t)$ has to be designed to be proportional to $p(\boldsymbol{\mathcal{M}} \mid \mathbf{x}_t, \hat{\mathbf{z}}_t) \times p(\boldsymbol{\mathcal{M}})^{-1}$. With the help of Bayes' theorem we invert the emission and then arrive at an expression which is the target for the map update approximation. The same strategy has been previously used by Grisetti et al. [69]. We specify the exact form of the update in appendix D, where we engineer the update such that it results in meaningful rendering.

D Map Update Formulation

Consider the emission model $p(\mathbf{x}_t \mid \mathbf{z}_t, \boldsymbol{\mathcal{M}})$. As described in appendix B, it determines the hit of a single surface during raycasting. Moreover, the linear interpolation described in appendix B means an SDF-like representation will match the rendering assumptions. Noting that we use an uninformed map prior, the map update then needs to approximate $q(\boldsymbol{\mathcal{M}} \mid \mathbf{x}_t, \hat{\mathbf{z}}_t) \approx p(\boldsymbol{\mathcal{M}} \mid \mathbf{x}_t, \hat{\mathbf{z}}_t) \approx p(\boldsymbol{\mathcal{M}} \mid \mathbf{x}_t, \hat{\mathbf{z}}_t) \times p(\boldsymbol{\mathcal{M}})^{-1}$. Because of this, we define the map update as:

$$
\begin{aligned}
q(\boldsymbol{\mathcal{M}} \mid \mathbf{x}_t, \mathbf{z}_t) &= \prod_{ijk} q(\boldsymbol{\mathcal{M}}_{ijk} \mid \mathbf{x}_t, \mathbf{z}_t) \\
q(\boldsymbol{\mathcal{M}}_{ijk} \mid \mathbf{x}_t, \mathbf{z}_t) &= \mathcal{N}\Big(\boldsymbol{\mathcal{M}}_{ijk} \;\Big|\; f_{\text{update}}(\mathbf{z}_t, \mathbf{x}_t)_{ijk}, \operatorname{diag}(\boldsymbol{\sigma}^{\text{update}}_{ijk})^2\Big) \\
f_{\text{update}}(\mathbf{z}_t, \mathbf{x}_t)_{ijk} &= [-f_{\text{sdf}}(\mathbf{p}_{ijk}, \mathbf{z}_t, \mathbf{x}_t), f_{\text{rgb}}(\mathbf{p}_{ijk}, \mathbf{z}_t, \mathbf{x}_t)]^T.
\end{aligned}
$$

Here the indices ijk run over the voxels in a 3D grid. The function f_{sdf} computes the SDF update values for the particular voxel center $\mathbf{p}_{ijk}$ based on the observed depth image $\mathbf{x}^d_t$ and the camera

pose $\mathbf{z}_t^{\text{pose}}$. f_{rgb} computes the RGB update values analogously. Both functions follow a traditional implementation [41]. We use the negative SDF value in the update to match the assumptions of Mirchev et al. [15] (see appendix B). Since the update is engineered in advance to match the generative assumptions, we empirically validate whether it is appropriate in section 5.

Next we recap the parametric form of the approximate map filter:

$$q_t^{\boldsymbol{\phi}}(\boldsymbol{\mathcal{M}}) = \prod_{ijk} \mathcal{N}\big(\boldsymbol{\mathcal{M}}_{ijk} \mid \boldsymbol{\mu}_{ijk,t}^{\boldsymbol{\mathcal{M}}}, \text{diag}(\boldsymbol{\sigma}_{ijk,t}^{\boldsymbol{\mathcal{M}}})^2\big).$$

Applying the update is straightforward, as it comes down to solving the multiplication of Gaussians for each voxel in closed form. This is because both the update and the approximate map filter factorise over voxels. Therefore, the application of the updates can be easily parallelised on a GPU. In the following we present the update equations for the whole map at once for the sake of simplicity of notation:

$$\begin{aligned} q(\boldsymbol{\mathcal{M}} \mid \mathbf{x}_t, \mathbf{z}_t) q_t^{\boldsymbol{\phi}}(\boldsymbol{\mathcal{M}}) &= \mathcal{N}\big(\boldsymbol{\mathcal{M}} \mid f_{\text{update}}(\mathbf{z}_t, \mathbf{x}_t), \text{diag}(\boldsymbol{\sigma}^{\text{update}})^2\big) \\ &\quad \times \mathcal{N}\big(\boldsymbol{\mathcal{M}} \mid \boldsymbol{\mu}_t^{\boldsymbol{\mathcal{M}}}, \text{diag}(\boldsymbol{\sigma}_t^{\boldsymbol{\mathcal{M}}})^2\big) \\ &= \mathcal{N}\big(\boldsymbol{\mathcal{M}} \mid \boldsymbol{\mu}_{t+1}^{\boldsymbol{\mathcal{M}}}, \text{diag}(\boldsymbol{\sigma}_{t+1}^{\boldsymbol{\mathcal{M}}})^2\big) := q_{t+1}^{\boldsymbol{\phi}}(\boldsymbol{\mathcal{M}}). \end{aligned} \tag{10}$$

The update equations are better described in terms of the Gaussian precisions (inverse covariances), denoting them as:

$$\begin{aligned} \boldsymbol{\Lambda}_t^{\boldsymbol{\mathcal{M}}} &= \text{diag}(\boldsymbol{\sigma}_t^{\boldsymbol{\mathcal{M}}})^{-2} \\ \boldsymbol{\Lambda}_{t+1}^{\boldsymbol{\mathcal{M}}} &= \text{diag}(\boldsymbol{\sigma}_{t+1}^{\boldsymbol{\mathcal{M}}})^{-2} \\ \boldsymbol{\Lambda}^{\text{update}} &= \text{diag}(\boldsymbol{\sigma}^{\text{update}})^{-2}. \end{aligned}$$

Solving for the parameters $\boldsymbol{\mu}_{t+1}^{\boldsymbol{\mathcal{M}}}$ and $\boldsymbol{\Lambda}_{t+1}^{\boldsymbol{\mathcal{M}}}$, from eq. (10) we have:

$$\begin{aligned} \boldsymbol{\mu}_{t+1}^{\boldsymbol{\mathcal{M}}} &= (\boldsymbol{\Lambda}_{t+1}^{\boldsymbol{\mathcal{M}}})^{-1}\big(\boldsymbol{\Lambda}_t^{\boldsymbol{\mathcal{M}}} \boldsymbol{\mu}^{\boldsymbol{\mathcal{M}}} + \boldsymbol{\Lambda}^{\text{update}} f_{\text{update}}(\mathbf{z}_t, \mathbf{x}_t)\big) \\ \boldsymbol{\Lambda}_{t+1}^{\boldsymbol{\mathcal{M}}} &= \boldsymbol{\Lambda}_t^{\boldsymbol{\mathcal{M}}} + \boldsymbol{\Lambda}^{\text{update}}. \end{aligned}$$

These equations reveal the connection to the traditional SDF equations of Curless and Levoy [41]:

$$\begin{aligned} D_{t+1}(\mathbf{p}) &= \frac{W_t(\mathbf{p}) D_t(\mathbf{p}) + w_t(\mathbf{p}) d_t(\mathbf{p})}{W_t(\mathbf{p}) + w_t(\mathbf{p})} \\ W_{t+1}(\mathbf{p}) &= W_t(\mathbf{p}) + w_t(\mathbf{p}). \end{aligned}$$

Here $\mathbf{p} \in \mathbb{R}^3$ is a point in the world frame (e.g. a voxel center), D_t is the accumulated SDF, d_t is the SDF update, W_t the accumulated weights so far and w_t the update weight. The algebraic form is the same, equating the mean of the filtering estimate to D, the mean of the update to d, the precision of the filtering estimate to W and the precision of the update to w. A similar probabilistic connection has been explored before in [77, 78].

E The Marginal State Filter

Before discussing the state filter, first we need to introduce the following linear approximation of the transition.

E.1 Transition Linearisation

We represent each state $\mathbf{z}_t = (\mathbf{z}_t^{\text{pose}}, \mathbf{z}_t^{\text{vel}})$ as the combination of a pose $\mathbf{z}^{\text{pose}} \in \text{SE}(3)$ and a velocity (translational and angular) $\mathbf{z}^{\text{vel}} \in \mathbb{R}^6$. Poses are parameterised as a combination of a 3D location and a quaternion. Our controls are translational and angular acceleration in the world reference frame: $\mathbf{u}_t = (\mathbf{u}_t^{\text{lin. accel}}, \mathbf{u}_t^{\text{ang. accel}})$. The assumed generative transition model is then the Euler integration of acceleration first, and then of velocity:

$$\begin{aligned} p(\mathbf{z}_{t+1} \mid \mathbf{z}_t, \mathbf{u}_t) &= p\big(\mathbf{z}_{t+1}^{\text{vel}} \mid f_{\text{vel}}(\mathbf{z}_t^{\text{vel}}, \mathbf{u}_t)\big) p\big(\mathbf{z}_{t+1}^{\text{pose}} \mid f_{\text{pose}}(\mathbf{z}_t^{\text{pose}}, \mathbf{z}_{t+1}^{\text{vel}})\big) \\ &= \mathcal{N}\big(\mathbf{z}_{t+1}^{\text{vel}} \mid f_{\text{vel}}(\mathbf{z}_t^{\text{vel}}, \mathbf{u}_t), \text{diag}(\boldsymbol{\sigma}^{\text{vel}})^2\big) \\ &\quad \times \mathcal{N}\big(\mathbf{z}_{t+1}^{\text{pose}} \mid f_{\text{pose}}(\mathbf{z}_t^{\text{pose}}, \mathbf{z}_{t+1}^{\text{vel}}), \text{diag}(\boldsymbol{\sigma}^{\text{pose}})^2\big). \end{aligned}$$

The function f_{vel} defines acceleration integration in the world frame and is linear:

$$\mathbf{z}_{t+1}^{\text{vel}} = f_{\text{vel}}(\mathbf{z}_t^{\text{vel}}, \mathbf{u}_t) = \mathbf{z}_t^{\text{vel}} + \mathbf{u}_t \cdot (\Delta t)^2.$$

The function f_{pose} defines Euler integration of the agents velocity, to obtain its new pose. When we do inference (not for prediction), we choose to linearise this function:

$$\begin{aligned}\mathbf{z}_{t+1}^{\text{pose}} &= f_{\text{pose}}(\mathbf{z}_t^{\text{pose}}, \mathbf{z}_{t+1}^{\text{vel}}) \\ &\approx \underbrace{\mathbf{A}_t \mathbf{z}_t^{\text{pose}} + \mathbf{B}_t \mathbf{z}_{t+1}^{\text{vel}} + \mathbf{c}_t}_{\text{first order Taylor apprx.}} =: \hat{f}_{\text{pose}}(\mathbf{z}_t^{\text{pose}}, \mathbf{z}_{t+1}^{\text{vel}}).\end{aligned}$$

This assumption lets us propagate uncertainty and infer velocity in closed form. Thus, we define an approximate linearised transition:

$$\begin{aligned}q(\mathbf{z}_{t+1} \mid \mathbf{z}_t, \mathbf{u}_t) &\approx p(\mathbf{z}_{t+1} \mid \mathbf{z}_t, \mathbf{u}_t) \qquad (11)\\ q(\mathbf{z}_{t+1} \mid \mathbf{z}_t, \mathbf{u}_t) &= p\big(\mathbf{z}_{t+1}^{\text{vel}} \mid f_{\text{vel}}(\mathbf{z}_t^{\text{vel}}, \mathbf{u}_t)\big) q\Big(\mathbf{z}_{t+1}^{\text{pose}} \mid \hat{f}_{\text{pose}}(\mathbf{z}_t^{\text{pose}}, \mathbf{z}_{t+1}^{\text{vel}})\Big) \\ &= \mathcal{N}\big(\mathbf{z}_{t+1}^{\text{vel}} \;\big|\; \mathbf{z}_t^{\text{vel}} + \mathbf{u}_t \cdot (\Delta t)^2, \text{diag}((\boldsymbol{\sigma}^{\text{vel}})^2)\big) \\ &\quad \times \mathcal{N}\big(\mathbf{z}_{t+1}^{\text{pose}} \;\big|\; \mathbf{A}_t \mathbf{z}_t^{\text{pose}} + \mathbf{B}_t \mathbf{z}_{t+1}^{\text{vel}} + \mathbf{c}_t, \text{diag}((\boldsymbol{\sigma}^{\text{pose}})^2)\big).\end{aligned}$$

Here, we introduce a linearisation $\hat{f}_{\text{pose}}$ of conventional Euler pose integration. Since the velocity integration function f_{vel} is linear by definition, this leaves us with two linear Gaussian conditionals, which we will use for closed-form updates [80]. In particular, the following integral can be solved in closed form:

$$\int q(\mathbf{z}_t \mid \mathbf{z}_{t-1}, \mathbf{u}_{t-1}) q_{t-1}^{\boldsymbol{\phi}}(\mathbf{z}_{t-1}) \, \mathrm{d}\mathbf{z}_{t-1} =: q_t(\mathbf{z}_t \mid \mathbf{u}_{t-1}, H_{t-1}), \qquad (12)$$

assuming $q_{t-1}^{\boldsymbol{\phi}}(\mathbf{z}_{t-1})$ is a Gaussian belief over the previous state, our state filter approximation introduced in appendix E.2.

E.2 State Filter Derivation

We can now derive the approximation of the true marginal state filter:

$$\begin{aligned}p(\mathbf{z}_t \mid H_t) &= \int p(\boldsymbol{\mathcal{M}}, \mathbf{z}_t \mid H_t) \, \mathrm{d}\boldsymbol{\mathcal{M}} \\ &\propto \int p(\mathbf{x}_t \mid \mathbf{z}_t, \boldsymbol{\mathcal{M}}) \int p(\mathbf{z}_t \mid \mathbf{z}_{t-1}, \mathbf{u}_{t-1}) p(\boldsymbol{\mathcal{M}}, \mathbf{z}_{t-1} \mid H_{t-1}) \, \mathrm{d}\mathbf{z}_{t-1} \, \mathrm{d}\boldsymbol{\mathcal{M}} \\ &\approx \int p(\mathbf{x}_t \mid \mathbf{z}_t, \boldsymbol{\mathcal{M}}) \int q(\mathbf{z}_t \mid \mathbf{z}_{t-1}, \mathbf{u}_{t-1}) q_{t-1}^{\boldsymbol{\phi}}(\boldsymbol{\mathcal{M}}) q_{t-1}^{\boldsymbol{\phi}}(\mathbf{z}_{t-1}) \, \mathrm{d}\mathbf{z}_{t-1} \, \mathrm{d}\boldsymbol{\mathcal{M}} \qquad (13)\\ &\approx p\Big(\mathbf{x}_t \mid \mathbf{z}_t, \hat{\boldsymbol{\mathcal{M}}}\Big) \int q(\mathbf{z}_t \mid \mathbf{z}_{t-1}, \mathbf{u}_{t-1}) q_{t-1}^{\boldsymbol{\phi}}(\mathbf{z}_{t-1}) \, \mathrm{d}\mathbf{z}_{t-1} \qquad (14)\\ &= p\Big(\mathbf{x}_t \mid \mathbf{z}_t^{\text{pose}}, \hat{\boldsymbol{\mathcal{M}}}\Big) q_t(\mathbf{z}_t \mid \mathbf{u}_{t-1}, H_{t-1}) \qquad (15)\\ &= p\Big(\mathbf{x}_t \mid \mathbf{z}_t^{\text{pose}}, \hat{\boldsymbol{\mathcal{M}}}\Big) q_t(\mathbf{z}_t^{\text{pose}} \mid \mathbf{u}_{t-1}, H_{t-1}) q_t(\mathbf{z}_t^{\text{vel}} \mid \mathbf{z}_t^{\text{pose}}, \mathbf{u}_{t-1}, H_{t-1}) \qquad (16)\\ &\approx q_t^{\boldsymbol{\phi}}(\mathbf{z}_t^{\text{pose}}) q_t(\mathbf{z}_t^{\text{vel}} \mid \mathbf{z}_t^{\text{pose}}, \mathbf{u}_{t-1}, H_{t-1}) =: q_t^{\boldsymbol{\phi}}(\mathbf{z}_t). \qquad (17)\end{aligned}$$

Our first approximation is in eq. (13), substituting $p(\boldsymbol{\mathcal{M}}, \mathbf{z}_{t-1} \mid H_{t-1})$ for $q_{t-1}^{\boldsymbol{\phi}}(\boldsymbol{\mathcal{M}}) q_{t-1}^{\boldsymbol{\phi}}(\mathbf{z}_{t-1})$ with the same reasoning as for the map filter. We also replace the true transition model for the linearised version from eq. (11). Next, in eq. (14) we MC-estimate the integral of $\boldsymbol{\mathcal{M}}$ with a single sample, using the mean $\hat{\boldsymbol{\mathcal{M}}}$ of the previous map belief $q_t^{\boldsymbol{\phi}}(\boldsymbol{\mathcal{M}})$. Next, in eq. (15) we solve the integral over $\mathbf{z}_{t-1}$ analytically (c.f. eq. (12)). We can do this because the approximate linear transition $q(\mathbf{z}_t \mid \mathbf{z}_{t-1}, \mathbf{u}_{t-1})$ forms a linear Gaussian system with $q_{t-1}^{\boldsymbol{\phi}}(\mathbf{z}_{t-1})$. Thus we obtain $q_t(\mathbf{z}_t \mid \mathbf{u}_{t-1}, H_{t-1})$, an approximate Gaussian prior over the current state. Next, in eq. (16) we can split this Gaussian into $q_t(\mathbf{z}_t^{\text{pose}} \mid \mathbf{u}_{t-1}, H_{t-1})$, a Gaussian prior over the current agent pose, and $q_t(\mathbf{z}_t^{\text{vel}} \mid \mathbf{z}_t^{\text{pose}}, \mathbf{u}_{t-1}, H_{t-1})$, a linear Gaussian velocity conditional given a pose. We can obtain both of them in closed form following standard multivariate Gaussian equations for linear Gaussian systems [80].

E.3 Pose Optimisation Objective

As already discussed in the main text, we obtain a pose belief $q_t^{\phi}(\mathbf{z}_t^{\text{pose}})$ using a maximum-aposteriori (MAP) objective

$$\arg\max_{\mathbf{z}_t^{\text{pose}}} \log p\Big(\mathbf{x}_t \mid \hat{\boldsymbol{\mathcal{M}}}, \mathbf{z}_t^{\text{pose}}\Big) + \log q_t(\mathbf{z}_t^{\text{pose}} \mid \mathbf{u}_{t-1}, H_{t-1}).$$

It arises naturally from the first two terms in eq. (16), serving as a likelihood and a prior. The term $\log p\Big(\mathbf{x}_t \mid \hat{\boldsymbol{\mathcal{M}}}, \mathbf{z}_t^{\text{pose}}\Big)$ represents rendering with the emission model, and evaluating it in every gradient step is inefficient. Because of this, we replace that term for the prediction-to-observation objective used by Kayalibay et al. [54], Nießner et al. [45], Newcombe et al. [84]. We refer to [54] for a discussion of why this surrogate is meaningful. Our final objective for optimising the pose is:

$$\begin{aligned}\arg\min_{\mathbf{z}_t^{\text{pose}}} \; & \mathcal{L}_{\text{geo}}\big(\mathbf{z}_t^{\text{pose}}, \mathbf{x}_t, \mathbf{z}_{t-1}^{\text{pose}}, \hat{\mathbf{x}}_{t-1}\big) + \mathcal{L}_{\text{rgb}}\big(\mathbf{z}_t^{\text{pose}}, \mathbf{x}_t, \mathbf{z}_{t-1}^{\text{pose}}, \hat{\mathbf{x}}_{t-1}\big) \\ & - \log q_t(\mathbf{z}_t^{\text{pose}} \mid \mathbf{u}_{t-1}, H_{t-1}). \end{aligned} \tag{18}$$

where we have:

$$\begin{aligned}\mathcal{L}_{\text{geo}}\big(\mathbf{z}_t^{\text{pose}}, \mathbf{x}_t, \mathbf{z}_{t-1}^{\text{pose}}, \hat{\mathbf{x}}_{t-1}\big) &= \sum_k \Big\| \langle \hat{\mathbf{p}}_{t-1}^k - \mathbf{T}_{\mathbf{z}_t^{\text{pose}}}^{\mathbf{z}_{t-1}^{\text{pose}}} \mathbf{p}_t^k, \hat{\mathbf{n}}_{t-1}^k \rangle \Big\|_1 \\ \mathcal{L}_{\text{rgb}}\big(\mathbf{z}_t^{\text{pose}}, \mathbf{x}_t, \mathbf{z}_{t-1}^{\text{pose}}, \hat{\mathbf{x}}_{t-1}\big) &= \sum_k \Big\| \hat{\mathbf{x}}_{t-1}^{\text{rgb}}[\pi(\mathbf{T}_{\mathbf{z}_t^{\text{pose}}}^{\mathbf{z}_{t-1}^{\text{pose}}} \mathbf{p}_t^k)] - \mathbf{x}_t^{\text{rgb}}[\pi(\mathbf{p}_t^k)] \Big\|_1 . \end{aligned}$$

We follow the notation of Kayalibay et al. [54] for consistency, and refer to that paper for further details. $\mathbf{z}_t^{\text{pose}}$ is the current unknown pose of the agent. $\mathbf{z}_{t-1}^{\text{pose}}$ is the mean pose of the previous pose belief. $\hat{\mathbf{x}}_{t-1}$ is a rendered observation from the preceding step, the mean of $p\Big(\mathbf{x}_{t-1} \mid \hat{\boldsymbol{\mathcal{M}}}, \mathbf{z}_{t-1}^{\text{pose}}\Big)$. $\mathbf{T}_{\mathbf{z}_t^{\text{pose}}}^{\mathbf{z}_{t-1}^{\text{pose}}}$ is the relative pose between the current and previous data step. $\mathbf{p}_t^k$ and $\hat{\mathbf{p}}_{t-1}^k$ are corresponding 3D points in respectively the current and previous camera frames. Accordingly, $\hat{\mathbf{n}}_{t-1}^k$ is the corresponding normal of the rendered depth image.

The two terms $\mathcal{L}_{\text{geo}}$ and $\mathcal{L}_{\text{rgb}}$ align the current RGB-D observation $\mathbf{x}_t$ to the preceding prediction $\hat{\mathbf{x}}_{t-1}$ rendered from the map, using geometric and photometric alignment, also known as point-to-plane ICP [41] and direct color image alignment [52, 51]. Because the preceding $\hat{\mathbf{x}}_{t-1}$ is rendered, this effectively anchors the current observation to the map by optimising the new pose of the agent. In addition, $\log q_t(\mathbf{z}_t^{\text{pose}} \mid \mathbf{u}_{t-1}, H_{t-1})$ is maximized, satisfying the linearised dynamics prior over the agent pose. The L1 norm in $\mathcal{L}_{\text{geo}}$ and $\mathcal{L}_{\text{rgb}}$ corresponds to a Laplace distribution assumption and the sole reason for it is robustness to outliers. When we apply a Laplace approximation[7] to obtain pose uncertainty, we implicitly change the L1 assumption. This is because we approximate a Hessian for the objective with the square of the full-objective Jacobian (with the dynamics prior term as well), which implies an assumed curvature of a square function. In practice we did not observe any major negative consequences from this fact.

F Approximation Gap

In our derivations, we have made a few crude approximations to reduce computations and make implementation simpler. For example, in eq. (8), we choose to use the previous state filter's mean instead of taking the expectation w.r.t. the whole distribution when computing the map marginal filter. Similarly, in eq. (14), we choose to use the previous map filter's mean instead of taking the expectation w.r.t. the whole distribution when computing the state marginal filter. Note that the implication of this conditioning is that state uncertainty is not reflected in the map updates (i.e. map does not become more uncertain if we are uncertain where to place the update) and map uncertainty is not reflected in the pose optimisation (i.e. states do not become more uncertain even if the map for which they are

[7] Not to be confused with a Laplace distribution assumption.

Table 2: Assumed environment sizes, one size per data set.

EuRoC	Blackbird	TUM-RGBD
14 m × 14 m × 14 m	25 m × 25 m × 25 m	14 m × 14 m × 14 m

Table 3: Transition scale hyperparameters.

translation		rotation	
$\boldsymbol{\sigma}^{\text{vel}}$	$\boldsymbol{\sigma}^{\text{pose}}$	$\boldsymbol{\sigma}^{\text{vel}}$	$\boldsymbol{\sigma}^{\text{pose}}$
0.03	0.05	0.03	0.02

optimised has not settled yet). We look forward to improving this aspect in future work, as proper uncertainty propagation can stabilise long-term operation of the proposed inference and allow for better overall uncertainty calibration of the model. In eqs. (7) and (13) we also approximate the joint posterior with the product of both marginal approximations. We believe that positioning the method as an approximation to $p(\mathbf{z}_t \mid H_t)$ and $p(\boldsymbol{\mathcal{M}} \mid H_t)$, the optimal marginal posteriors, reveals the exact places where compromises have been made. We expect that explicitly highlighting the current approximation gap will be conducive for future research.

G Experiment Details

G.1 Execution Details

We have implemented PRISM with JAX [88], using Accelerated Linear Algebra (XLA) to compile computations into kernels that can be executed on a GPU device. This lets us execute everything in real-time, while preserving the auto-differentiability of the generative model [15]. Rendering, map updates and the Laplace approximation for pose uncertainty are executed on GPU, as they involve a lot of parallel computations. The gradient-based pose optimisation is executed on CPU, as every optimisation step is lightweight and optimisation steps need to happen in sequence. The linear-Gaussian updates are executed on CPU as well.

G.2 Hyperparameters and Inference Details

Map parameters Grid resolution is $200 \times 200 \times 200$ across all experiments. The map is parameterised with a mean grid and a grid with standard deviations. Each mean grid cell contains occupancy and RGB color, four values in total. Each standard deviation grid cell also contains four respective uncertainty values. Each map covers a hypercube of real-world space, we list the environment sizes per data set in table 2. This determines the effective voxel size, between 7 cm for EuRoC and TUM-RGBD and 12.5 cm for Blackbird.

Transition parameters The transition is homoscedastic, with predefined scales for acceleration and velocity integration. We use different scales for the location and orientation components of the states. Table 3 provides the hyperparameters, with $\boldsymbol{\sigma}^{\text{vel}}$ governing acceleration integration and $\boldsymbol{\sigma}^{\text{pose}}$ governing velocity integration. They are the same for all data sets.

Rendering The maximum camera depth is set to 7.0 m for EuRoC, 8.0 m for TUM-RGBD and 20.0 m for Blackbird. The ray step size ϵ is set to $0.4 \times \text{voxel\_size}$. The threshold for hit determination τ is set to 0, so that rendering is compatible with the map updates.

Map updates Map updates are placed only for map voxels that fall inside the camera frustum, and fall between the camera optical center and an added truncation distance after the observed depth surface. The truncation distance is $4 \times \text{voxel\_size}$ for Blackbird [18] and $2 \times \text{voxel\_size}$ for EuRoC

[17] and TUM-RGBD [19]. We assume a constant scale $\sigma_{ijk}^{\text{update}} = 1.0$ for the update of all relevant voxels. For the very first map belief, we initialise occupancy (negative SDF) to -0.001 and color to 0.0 for all map voxels.

Pose optimisation Poses are optimised using the MAP objective from section 4.3.2. We use Adam [89] as an optimiser, disabling its momentum. Step sizes are set individually for the translation and rotation components of the optimised poses. For the translational part of the pose, we use a step size of 0.001. For the rotational part of the pose, we use a step size of 0.00036. We use 1000 optimisation steps, sampling 200 random pixels uniformly at each step to evaluate the objective. Pixels with a geometric error higher than 0.45 are ignored during optimisation. Pixels with a photometric error higher than 0.15 are ignored during optimisation. The same optimisation hyperparameters are used in all experiments.

In terms of the objective itself, we assume a different Laplace scale for the photometric and geometric terms in eq. (18) for each data set, based on our confidence in the sensors. A lower Laplace scale means higher priority is given to the respective observations (color or depth). For EuRoC and TUM-RGBD, we use a color scale of 0.1 and a geometric scale of 0.02, as the depth readings in these data sets are accurate. For Blackbird, we use a color scale of 0.02 and a geometric scale of 0.2, as the depth we use in Blackbird is rather inaccurate, estimated with SGBM [85].

Laplace approximation We approximate the pose optimisation objective's Hessian with the square product of the objective's Jacobian. Since the Laplace estimates are noisy over time due to approximation errors, we apply an exponential moving average over time with a coefficient of 0.8.

G.3 Data Preprocessing

We subsample images to a resolution of 60×80 pixels for EuRoC, 192×256 for Blackbird and 120×160 pixels for TUM-RGBD. Since the ground-truth depth readings for EuRoC from [10] are sparse we downscale them to a resolution of 60×80 pixels to densify. We ignore pixels with invalid depth throughout our method, as well as pixels for which depth is highly discontinuous.

G.4 Localisation Evaluation Details

We compare our localisation results to the published results of existing SLAM methods, carrying them over from the respective publications. Respectively, the choice of our evaluation trajectories was determined by whether a comparison is possible. We run inference experiments with 5 random seeds for each trajectory and report the mean and standard deviation of the relevant metrics.

H Uncertainty Analysis

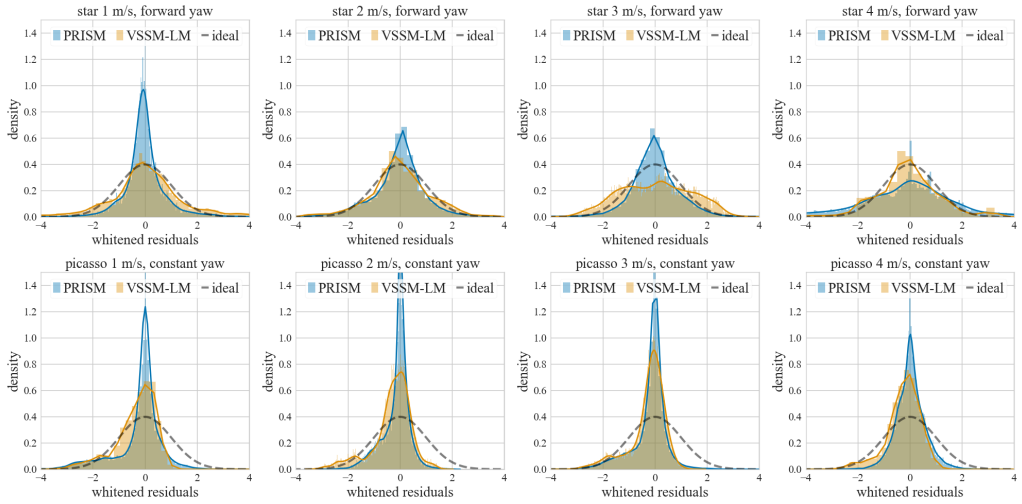


Figure 7: Whitened state residuals for PRISM and VSSM-LM [15] for eight different blackbird trajectories (denoted in the titles). The model is pessimistic when the distribution of whitened residuals is narrower than a standard Gaussian.

We analyse the uncertainty of PRISM and compare it to our reproduction of the off-line variational inference method of Mirchev et al. [15] in JAX, which we denote with VSSM-LM. We compare both state uncertainty and map uncertainty. Since we have ground-truth poses for the state, we can also evaluate the state uncertainty calibration quantitatively. The calibration tells us whether the uncertainty matches the estimation errors between the inferred state means and the ground truth.

State uncertainty To evaluate the calibration of state uncertainty, we first evaluate the residuals between the inferred mean poses and the ground-truth MOCAP poses, using the same eight trajectories of the Blackbird data set [18] from section 5.2. The emission and transition scales of both methods define the overall uncertainty magnitude of the estimates. To put both systems on equal ground and avoid tuning inefficiencies w.r.t. these scales, we estimate a single global scalar correction for each method and apply it to all state covariances. We then whiten[8] the computed residuals with their respective covariance estimates for all poses in all trajectories.

The whitened residuals of a perfectly calibrated model should form a standard normal distribution. This would indicate that the inferred covariances exactly match the distribution of errors the model makes. If the whitened residuals end up narrower than a standard Gaussian, then the model is too pessimistic as its uncertainty estimate was higher than the actual unnormalised residuals, and vice versa. Figure fig. 7 shows the distribution of the whitened residuals for both PRISM and VSSM-LM. While both models are not perfectly calibrated, their residual distributions are still reasonable, as they roughly match the support of the ideal Gaussian. PRISM is more pessimistic, indicated by the narrower distributions of whitened residuals. We find this is better than the alternative, as it would lead to more cautious control of the agent. Note that PRISM is also consistently more accurate in state estimation than VSSM-LM, as shown in section 5.2. The pessimism is more pronounced for trajectories of lower velocity (see titles of fig. 7), for which state estimation is easier.

[8] Normalise by the square root of the covariance matrix, i.e. inverse of a triangular matrix.

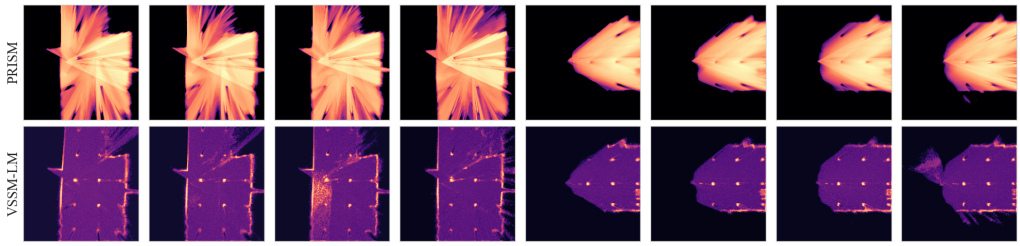


Figure 9: Horizontal slices of map uncertainty (orange means certain) for the same eight blackbird trajectories. PRISM on top, VSSM-LM below. From left to right: *star* {1, 2, 3, 4} m/s, forward yaw, then *picasso* {1, 2, 3, 4} m/s, constant yaw.

To quantify the calibration gap further, we perform a Chi-squared calibration analysis of the normalised squared sum of residuals (NSSR), following Jospin et al. [90], section VII. Sum here refers to summing over the pose dimension. Figure 8 shows the result. It compares the cumulative Chi-squared distribution of the normalised residuals (prediction distribution) to the cumulative distribution of observing that residual in the data (observation distribution). A model with an ideal calibration of its uncertainties relative to the errors it makes would match the identity diagonal. The x-axis corresponds to an ordering of the magnitude of the residuals (low to high). Model curves above the diagonal mean the model is pessimistic, and vice versa. The calibration curves of both methods indicate a reasonable correlation between the cumulative prediction distribution and the cumulative observation distribution of residuals, with PRISM exhibiting more bias towards pessimism. This is a confirmation of what we identified in the residual plots above. Note that the uncertainty of VSSM-LM is produced offline, ca. 15 times slower than PRISM in our implementation.

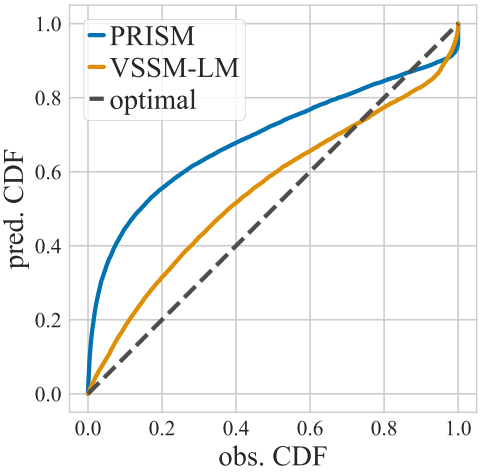


Figure 8: Chi-squared calibration curve.

Map uncertainty Next we qualitatively compare the map uncertainty of PRISM to the map uncertainty of VSSM-LM. Figure 9 shows horizontal slices of the uncertainty estimates at eye level for the same eight Blackbird trajectories from section 5.2. For all trajectories, the scene is a subway station with multiple columns.

VSSM-LM is certain primarily at surfaces, i.e. at walls and columns (the orange lines and dots in the images on the lower row). It assigns lower certainty to the empty space inside the scene (purple), which is technically observed just as much by the agent. We attribute this to the way differentiable rendering works in this method: since the renderer identifies a single hit along each ray when observations are reconstructed, gradients flow only to the map parameters that correspond to the hit. Therefore certainty is most pronounced there.

On the other hand, PRISM assigns certainty both to surfaces and to the observed empty space between (c.f. fig. 9, top row), due to the nature of the map updates. Note that whenever something is occluded in the view of the agent it remains less certain (e.g. the triangular patterns behind columns seen in the images). Also, whenever the agent spends more time observing a given region, the certainty of the map increases proportionally. This can be seen in all star trajectories, where the agent starts its flight

from the same spot after sitting still for a while (first four images in fig. 9, also see fig. 2b). Whether this characteristic difference between the methods matters is left to explore in future work.

Overall, both methods reliably leave regions out of view uncertain and increase certainty only in the observed space.

I Map Resolution Ablations

In this ablation we evaluate how localisation accuracy changes for increased map resolution. We set the new resolution to 400 $\times$ 400 $\times$ 400, doubling the resolution of each voxel grid side. We fix all other hyperparameters to their default values, as listed in appendix G. The localisation results are in table 4. We notice consistent improvements across the board, the largest of up to 3 cm for the Blackbird trajectories. This makes sense, as the Blackbird scenes are the biggest (25 m $\times$ 25 m $\times$ 25 m), where an increased resolution leads to a more substantial reduction in voxel size, from 12.5 cm to 6.25 cm per side.

We notice only one outlier, *star, 4 m/s*. For the increased resolution, five seeds for that trajectory resulted in RMSE scores of {0.300, 0.281, 0.420, 0.115, 0.269} m. None of the runs failed and they all showed low to moderate estimation bias around the corners of the trajectory.

We note that voxel grids are wasteful in terms of memory and limit the maximum feasible resolution. Schemes like voxel hashing [45] can lift this limitation, and fit all other presented assumptions.

Table 4: Localisation absolute error RMSE in meters on EuRoC [17], Blackbird [18] and TUM-RGBD [19], for a map resolution of 400 x 400 x 400 (twice as big as the default).

Trajectory	PRISM res. 200	PRISM res. 400
EuRoC/V101	0.041 ($\pm$ 0.002)	**0.038** ($\pm$ 0.003)
EuRoC/V102	0.035 ($\pm$ 0.002)	**0.030** ($\pm$ 0.001)
EuRoC/V103	0.042 ($\pm$ 0.002)	**0.038 ($\pm$ 0.002)**
EuRoC/V201	0.037 ($\pm$ 0.001)	**0.032 ($\pm$ 0.001)**
EuRoC/V202	**0.035 ($\pm$ 0.003)**	**0.035 ($\pm$ 0.003)**
EuRoC/V203	x	x

Trajectory	PRISM res. 200	PRISM res. 400
picasso, 1 m/s	0.064 ($\pm$ 0.003)	**0.045 ($\pm$ 0.003)**
picasso, 2 m/s	0.053 ($\pm$ 0.003)	**0.048 ($\pm$ 0.009)**
picasso, 3 m/s	0.061 ($\pm$ 0.003)	**0.042 ($\pm$ 0.002)**
picasso, 4 m/s	0.079 ($\pm$ 0.005)[9]	**0.061 ($\pm$ 0.002)**[9]
star, 1 m/s	0.089 ($\pm$ 0.007)[9]	**0.074 ($\pm$ 0.009)**[9]
star, 2 m/s	0.111 ($\pm$ 0.009)	**0.103 ($\pm$ 0.009)**
star, 3 m/s	0.115 ($\pm$ 0.012)	**0.082 ($\pm$ 0.008)**
star, 4 m/s	**0.153 ($\pm$ 0.015)**[9]	0.278 ($\pm$ 0.015) [9]

Trajectory	PRISM res. 200	PRISM res. 400
fr1/desk	0.053 ($\pm$ 0.003)	**0.052 ($\pm$ 0.001)**
fr2/xyz	0.029 ($\pm$ 0.001)	**0.021 ($\pm$ 0.000)**
fr3/office	0.083 ($\pm$ 0.001)	**0.081 ($\pm$ 0.002)**

J Motivation and Downstream Applicability

PRISM is an inference tailored to the state-space model introduced by Mirchev et al. [15], and this synergy has advantages. Markovian state-space models are a fundamental building block that enables

[9] Last 10 s are skipped, as the drone hits the ground during landing.

model-based control [2]. For this, both a predictive distribution and a state estimator are needed. The predictive distribution is already given by Mirchev et al. [15], PRISM fills the role of a real-time state estimator.

The advantages we foresee come from the probabilistic integration of a dense map, rendering and dynamics.

Prediction and Control PRISM's estimates harmonise with the predictive model:

$$\mathbb{E}_{q_t^{\boldsymbol{\phi}}(\boldsymbol{\mathcal{M}})q_t^{\boldsymbol{\phi}}(\mathbf{z}_t)}[p(\mathbf{z}_{t+1:t+k}, \mathbf{x}_{t:t+k} \mid \mathbf{u}_{t:t+k-1}, \mathbf{z}_t, \boldsymbol{\mathcal{M}})],$$

because we derive them as approximations to posterior distributions stemming from the same state-space model. A rollout can therefore start from the inference estimates $q_t^{\boldsymbol{\phi}}(\boldsymbol{\mathcal{M}})$ and $q_t^{\boldsymbol{\phi}}(\mathbf{z}_t)$ (in expectation above) and predict both rendered images and states with appropriate uncertainty for a candidate control sequence $\mathbf{u}_{t:t+k-1}$. We can then apply any technique from the literature on optimal control and reinforcement learning to control the agent [2, 91]. Note that because the control problem is of partial observability, i.e. a partially-observable Markov decision process (POMDP), an ideal solution may need to intertwine the estimator in the rollout to form beliefs, but we leave such considerations for the future to simplify discussion.

One advantage we see is that whole images can be predicted in the rollout. This is possible because of the dense map estimate, which supports rendering. It allows us to define rewards for the image observations, which can enable interesting control tasks that are not limited to point-to-goal navigation.

Another advantage is that the dense map directly provides obstacle information. We can combine this information with the map uncertainty estimates to be more robust when avoiding obstacles in the environment. Similarly, since the predictive rollout is fully-probabilistic, the uncertainty of $q_t^{\boldsymbol{\phi}}(\mathbf{z}_t)$ will propagate through the state transition and would be useful for robust collision avoidance as well.

On its own, the uncertainty of the map is useful for active learning (e.g. see [37]). In particular, we can go beyond frontier-based exploration and focus the agent on still uncertain map regions through an information-theoretic objective.

Adoption and Future Work We have made an effort to signpost all approximations we make in our filtering derivations in appendices C and E. We hope this will facilitate future research.

PRISM is also straightforward to implement in auto-differentiable frameworks (the current version is written entirely in JAX [88]).

A.3. Tracking and Planning with Spatial World Models

Paper Summary

TNP-SM shows how to solve navigation tasks under a dense state-space model (VSSM-LM, appendix A.1). It uses the dense maps obtained through differentiable rendering for collision avoidance, and combines high-level planning (search) with low level control to solve complex simulated environments (e. g. 3D mazes). The navigation works with only RGB-D images under noisy dynamics, using a state estimator to uncover the 6-DoF agent states needed for planning.

TNP-SM runs in real-time, and a key ingredient for that is that the state estimator derived from VSSM-LM uses a surrogate coloured-ICP objective to avoid gradients through the renderer. An analysis of the runtime is included, with a comparison to the alternative.

Author Contributions

The author's main contribution to the paper is the definition and implementation of a real-time state estimator (tracker) for the agent states, derived from the predictive assumptions of VSSM-LM, and the accompanying experiments that compare the state estimator's performance to estimation with gradients directly through a renderer (sections 3.2, 4.2 and "State estimation in real-time" in section 5). The planning and low-level control methods and associated experiments are contributions by Baris Kayalibay. Both authors co-wrote the paper.

problem definition	helped
literature review	helped
theoretical derivation	significant (state estimation)
implementation	significant (state estimation)
experiments	significant (state estimation)
writing	significant

Tracking and Planning with Spatial World Models

Barış Kayalıbay* BKAYALIBAY@ARGMAX.AI
Atanas Mirchev* ATANAS.MIRCHEV@ARGMAX.AI
Patrick van der Smagt
Justin Bayer BAYERJ@ARGMAX.AI
Machine Learning Research Lab, Volkswagen Group, Munich

Editors: R. Firoozi, N. Mehr, E. Yel, R. Antonova, J. Bohg, M. Schwager, M. Kochenderfer

Abstract

We introduce a method for real-time navigation and tracking with differentiably rendered world models. Learning models for control has led to impressive results in robotics and computer games, but this success has yet to be extended to vision-based navigation. To address this, we transfer advances in the emergent field of differentiable rendering to model-based control. We do this by planning in a learned 3D spatial world model, combined with a pose estimation algorithm previously used in the context of TSDF fusion, but now tailored to our setting and improved to incorporate agent dynamics. We evaluate over six simulated environments based on complex human-designed floor plans and provide quantitative results. We achieve up to 92% navigation success rate at a frequency of 15 Hz using only image and depth observations under stochastic, continuous dynamics.
Keywords: navigation, planning, model-based control, state estimation, differentiable rendering

1. Introduction

Autonomous systems need to understand their environment to make good decisions. Optimal control and reinforcement learning therefore hinge on learning or engineering accurate models of both dynamics and observations, which allow us to plan into the future and find optimal actions. The paradigm of learning models has been successfully applied to Atari games (e.g. Kaiser et al., 2020), walking with complex simulated robots (e.g. Hafner et al., 2019), controlling unmanned aerial vehicles (e.g. Becker-Ehmck et al., 2020) and meta-reinforcement learning for robotics (e.g. Zhao et al., 2020).

Prior work has attempted to learn such world models of spatial environments (Fraccaro et al., 2018; Mirchev et al., 2019). These early models were limited to simple scenes, which restricted their practical use in control to toy sce-

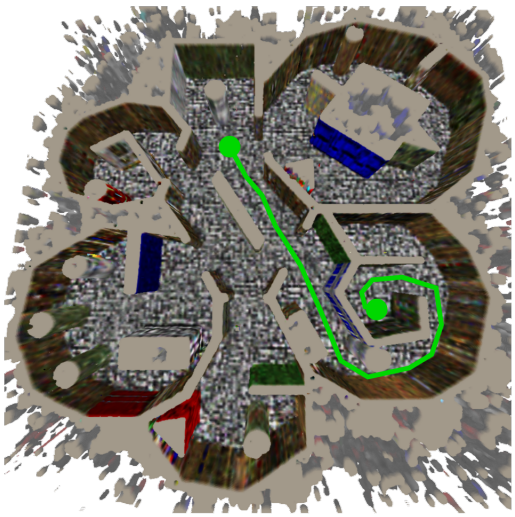


Figure 1: Planning in a world model. Isometric model view with planned trajectory.

* Equal contribution.

narios (Kayalibay et al., 2018). Their limitations were a result of attempting to model the complex interaction of a camera or depth sensor with its environment using a combination of neural networks and simple inductive biases. More recently, this balance has shifted in favour of the inductive bias, with an increased focus on engineering and domain knowledge (Mildenhall et al., 2020; Mirchev et al., 2021; Sitzmann et al., 2020; Müller et al., 2019). The result is the field of differentiable rendering, which models camera sensing by applying multiple-view geometry and adapting concepts from computer graphics, with neural networks being relegated to the relatively simple task of predicting properties like colour or occupancy over 3D space.

Differentiable rendering allows to model spatial environments with outstanding fidelity. It is then natural to use such models for control, an idea that was voiced as early as the 1990s (Oore et al., 1997). This has been done by Li et al. (2021) in the context of multi-link robots and Adamkiewicz et al. (2021) in the context of navigation, albeit without a quantitative evaluation of success.

Here we present a practical navigation algorithm for planar robots which uses a learned world model. Navigating based on visual observations alone requires inferring the agent's pose. We will therefore address the problem of state estimation as well. Our exact contributions are:

- We present a 15-Hz real-time algorithm for navigation under noisy agent dynamics and only local RGB-D observations in world models obtained with differentiable rendering.
- We test our method on simulated yet challenging indoor environments based on actual floor plans and provide a thorough quantitative evaluation.
- We improve an existing method for tracking camera poses by using the dynamics of the agent and apply it to our setting. This runs about five times faster than propagating gradients through the renderer, and is thus well-suited for vision-based control.

2. Related Work

Generative models and maps. Various generative models of space have been proposed (Fraccaro et al., 2018; Planche et al., 2019; Gregor et al., 2019; Mirchev et al., 2019). Among these, the work of Mirchev et al. (2019) was later adapted to a navigation task with the addition of A*-based planning (Kayalibay et al., 2018). The method was demonstrated on simple 2D mazes with 360° depth range observations. In contrast, others have proposed models aimed at tasks like navigation and exploration, which do not require generating new observations from the model. These approaches define read and write operations for distilling information from camera images into a 2D top-down map of the environment. Notable contributions here include the work of Parisotto and Salakhutdinov (2017), where a 2D neural map with abstract features is written to and read from to solve searching problems. Gupta et al. (2019) proposed a similar algorithm, extended by a planning module for navigation. The write operation was improved using projective geometry in (Chen et al., 2019) and (Chaplot et al., 2020), the latter achieving impressive navigation performance. Ramakrishnan et al. (2020) extended the setup by learning to predict occupancy beyond visible locations.

Differentiable rendering, pose estimation and control. Various methods for differentiably rendering images have been proposed in recent years. Of these, the method of Mildenhall et al. (2020) relies on modeling colour and spatial density with neural nets. Sitzmann et al. (2020) used an LSTM to find the point where a ray would hit the scene geometry. Lombardi et al. (2019) proposed using voxel grids and warping fields to encode and decode scenes. Mirchev et al. (2021) employ voxel

grids as well, focusing on localisation and mapping with unmanned aerial vehicles (UAV). Taking a different route, Niemeyer et al. (2020) and Yariv et al. (2020) rely on implicit differentiation to define the surface of the scene.

Several works deal with pose estimation in the context of differentiable rendering. Wang et al. (2021) jointly optimise pose, camera and scene parameters in a set of simple scenes. Assuming that the scene parameters have been learned in advance, Yen-Chen et al. (2021) propose finding camera poses using gradient descent and an efficient pixel subsampling scheme. Notably, Sucar et al. (2021) propose a real-time algorithm for joint tracking and mapping, though their evaluation is limited to small scenes. Mirchev et al. (2021) introduce a SLAM algorithm based on differentiable rendering, capable of tracking a UAV in photorealistically rendered simulations, though not in real time.

To the best of our knowledge, only two papers exist that apply differentiable rendering in a control context. Of these, (Li et al., 2021) focuses on manipulation tasks involving a robot arm. Adamkiewicz et al. (2021) investigate using neural radiance fields to solve navigation tasks. Their work is close to ours, though we differ in a couple of points. We focus on navigation with planar robots in indoor environments, and provide a quantitative evaluation on a set of complex scenes based on human-designed floor plans which were converted into levels for the Vizdoom simulator (Wydmuch et al., 2018). In contrast, they provide experiments with different kinds of robots, though they do not report any quantitative results indicating large-scale navigation success. Our work also approaches the problem of pose estimation differently. They approach pose estimation by combining the approach of Yen-Chen et al. (2021) with a transition loss and obtain a Bayesian filter. The benefit of our approach is its speed over computing gradients through the rendering pipeline. Finally, they propose a gradient-based algorithm for obstacle avoidance, which is initialised using A*-search. We similarly use A*-search, and track the planned trajectory using a simple low-level controller intended for a planar robot.

3. Background

3.1. Differentiable Rendering

The backbone of most differentiable rendering approaches is a parametric function that maps 3D space to geometric information (e.g. occupancy) and colour. Formally, we have two maps $f : \mathbb{R}^3 \to \mathbb{R}$ and $g : \mathbb{R}^3 \to \mathbb{R}^3$, where one learns the occupancy (alternatively, density or opacity) and the other corresponds to the RGB value. The colour model might also take a viewing angle as input, which allows modeling reflections. We can then generate an RGB-D image from any camera pose in the special Euclidean group $\mathrm{SE}(3)$ by defining a differentiable rendering procedure.

The first step of the rendering procedure involves finding the point where a ray intersects the scene geometry for the first time, the latter being implicitly defined by the occupancy model. We follow the approach of Mirchev et al. (2021), where the occupancy function is linearised around the point where it first exceeds a threshold, and the intersection point is then found using that linear approximation. This way of rendering is well-suited to our purposes, since it directly defines the occupancy of a point in space, which defines where the obstacles are when navigating. Formally, given a ray parameterised by the camera centre and an offset vector: $\mathbf{r} = \mathbf{c} + k\mathbf{v}$, we evaluate the occupancy function for $k \in \{\Delta, 2\Delta, \dots, n\Delta\}$, where Δ gives the distance between two points along the ray, and $n\Delta$ is the maximum range of the camera. Then, if $\mathbf{p}^+ = \mathbf{c} + k^+\mathbf{v}$ is the first point for which the occupancy function returns a value above some threshold τ, and $\mathbf{p}^- = \mathbf{c} + k^-\mathbf{v}$

is the point that precedes it, the intersection point $\mathbf{p}$ is defined as $\mathbf{p}^* = \mathbf{c} + k^*\mathbf{v}$ for:

$$k^* = \alpha k^+ + (1-\alpha)k^-, \quad \alpha = \frac{\tau - f(p^-)}{f(p^+) - f(p^-)}. \tag{1}$$

The setup of Mirchev et al. (2021) is also set apart from other differentiable rendering approaches as voxel grids are used instead of neural nets to capture the scene. Making use of trilinear interpolation, differentiability is maintained and hence the use of gradient-based techniques possible. We follow this decision as well, since indexing a voxel grid via interpolation is faster than evaluating a neural net. This reduces the time it takes to render a full image, a step required by our tracking method (we discuss rendering speed in section 5). We note that the voxel grid can be seen as an ensemble of small neural networks with partially fixed weights, where the networks are distributed over space in a grid-fashion, resembling the work of Reiser et al. (2021).

Formally the occupancy map is a 3D tensor $\mathcal{M}^{\text{occ}} \in \mathbb{R}^{h \times w \times d}$ and the colour map is a 4D tensor $\mathcal{M}^{\text{col}} \in \mathbb{R}^{h \times w \times d \times 3}$. The occupancy and colour functions f and g are then given by extracting the eight neighbor cells of the input point and trilinearly interpolating between them. The density of an RGB-D image $\mathbf{x}$ taken from the camera pose $\mathbf{z}$ under the differentiably rendered model is then:

$$p(\mathbf{x} \mid \mathbf{z}) = \prod_{i,j} p(\mathbf{x}_{ij} \mid \mathbf{z}, \mathcal{M}^{\text{occ}}, \mathcal{M}^{\text{col}}) = \prod_{\mathbf{x}_{ij} \in \mathbf{x}} \text{Laplace}\left(\begin{bmatrix} \mathbf{x}_{ij}^{\text{rgb}} \\ \mathbf{x}_{ij}^{\text{d}} \end{bmatrix} \middle| \begin{bmatrix} g(\mathbf{p}^*) \\ k^* \end{bmatrix}, \begin{bmatrix} \sigma_1 \\ \sigma_2 \end{bmatrix} \right). \tag{2}$$

Here, $\mathbf{x}_{ij}$ is a pixel from the image, and $\mathbf{x}_{ij}^{\text{rgb}}$ and $\mathbf{x}_{ij}^{\text{d}}$ are its RGB and depth readings respectively. $\mathbf{p}^*$ and k^* are the intersection point of the ray and its distance from the camera. The Laplace distribution's scales for colour and depth are parameterised by σ_1 and σ_2, which are treated as learnable parameters. The ray for a pixel is given by: $\mathbf{t}(\mathbf{z}) + d\mathbf{R}(\mathbf{z})\mathbf{K}^{-1} \begin{bmatrix} i & j & 1 \end{bmatrix}^T$, with $\mathbf{t}(\cdot)$ and $\mathbf{R}(\cdot)$ translation and rotation defined by the camera pose, $\mathbf{K}$ the camera intrinsic matrix, (i, j) the 2D coordinates of the pixel and d a depth reading.

This differentiable renderer was proposed by Mirchev et al. (2021) in the context of online simultaneous localisation and mapping. Here the map is learned offline on a set of RGB-D images with known camera poses. Given a dataset of images and camera poses $\mathcal{D} = \{(\mathbf{x}^{(i)}, \mathbf{z}^{(i)})\}_{i=1}^N$, the occupancy and colour parameters $\mathcal{M}^{\text{occ}}$ and $\mathcal{M}^{\text{col}}$ are obtained by gradient-based minimisation of:

$$\mathcal{L}(\mathcal{M}^{\text{occ}}, \mathcal{M}^{\text{col}}) = -\sum_{\mathbf{x},\mathbf{z} \in \mathcal{D}} \sum_{i,j} \log p(\mathbf{x}_{ij} \mid \mathbf{z}, \mathcal{M}^{\text{occ}}, \mathcal{M}^{\text{col}}). \tag{3}$$

We approximate the gradients by Monte-Carlo sampling images and pixels uniformly for speed.

3.2. Pose Estimation with Differentiable Renderers

Agents moving about freely normally cannot directly observe their state. Instead, they need to rely on estimating it from observations. Pose estimation in prior approaches with differentiable rendering has so far focused on backpropagating pixel-wise reconstruction errors through the renderer (Yen-Chen et al., 2021; Sucar et al., 2021; Adamkiewicz et al., 2021):

$$\arg\min_{\mathbf{z}} -\log p(\mathbf{x} \mid \mathbf{z}, \boldsymbol{\mathcal{M}}^{\text{occ}}, \boldsymbol{\mathcal{M}}^{\text{col}}).$$

We additionally consider a different route for state estimation, optimising both photometric and point-to-plane reprojection errors (Chen and Medioni, 1992). Given an RGB-D image $\hat{\mathbf{x}} = (\hat{\mathbf{x}}^{\mathrm{rgb}}, \hat{\mathbf{x}}^{\mathrm{d}})$ rendered by the model, a new observation $\mathbf{x} = (\mathbf{x}^{\mathrm{rgb}}, \mathbf{x}^{\mathrm{d}})$ and a relative pose $\mathbf{z}$ we minimise:

$$\arg\min_{\mathbf{z}} \sum_k \left\| \hat{\mathbf{x}}^{\mathrm{rgb}}[\pi(\mathbf{T}_{\mathbf{z}}\mathbf{p}^k)] - \mathbf{x}^{\mathrm{rgb}}[\pi(\mathbf{p}^k)] \right\|_1 + \sum_k \left\| \langle \hat{\mathbf{p}}^k - \mathbf{T}_{\mathbf{z}}\mathbf{p}^k, \hat{\mathbf{n}}^k \rangle \right\|_1 . \tag{4}$$

Here, $\mathbf{p}^k$ is a 3D point in the observed camera frame[1] and $\hat{\mathbf{p}}^k$ is a 3D point in the camera frame of the model prediction, related via projective association (Blais and Levine, 1995; Stotko, 2016). π denotes perspective projection from 3D into the image plane, using the known intrinsic matrix $\mathbf{K}$. The dot product $\langle \cdot, \hat{\mathbf{n}}^k \rangle$ with the corresponding normal vector $\hat{\mathbf{n}}^k$ defines the point-to-plane objective. First, normals are computed based on image gradients of the rendered prediction $\hat{\mathbf{x}}^{\mathrm{d}}$. Then the difference $\hat{\mathbf{p}}^k - \mathbf{T}_{\mathbf{z}}\mathbf{p}^k$ is projected onto the normal of the rendered surface, therefore reflecting the distance between the transformed point $\mathbf{T}\mathbf{p}^k$ and a hyperplane tangent to the surface. It drives points from the second camera frame to align to the surface implicitly defined by the rendered depth $\hat{\mathbf{x}}^{\mathrm{d}}$. The L1-norm (Laplace assumption) is used for increased robustness to outliers.

This method is commonly referred to as point-to-plane ICP (Chen and Medioni, 1992) combined with a photometric objective (Steinbrücker et al., 2011; Audras et al., 2011). Variants of this approach have been used to good effect for tracking new RGB-D observations w.r.t. a fused scene model (Newcombe et al., 2011; Nießner et al., 2013; Whelan et al., 2015). Traditionally, this is well-explored for maps obtained via TSDF fusion. We show the same concept is applicable to maps obtained by optimisation through differentiable rendering. In section 4.2 we will discuss its advantages compared to gradient descent directly through the renderer.

4. Method

4.1. Motion Planning

We focus on planar indoor robots trying to reach a goal position, the target. Given a starting location and target coordinates, we plan a trajectory using A*-search in the space of 2D coordinates (*planning* in fig. 2, top). Here, we discretise the environment using a uniform grid. Moving from one cell to another is possible only if a) there is no occupied point on the line connecting the two cells according to the occupancy model, and b) the cell we step into maintains a minimum safety distance to any obstacle. For the former, we sample a fixed (e.g. one hundred) number of points along the movement vector. For the latter, we extract obstacles from the occupancy model by evaluating it on each grid point, where the occupied points then give us a set of point obstacles. This requires many evaluations of the map, which in our case is computationally cheap due to the voxel grid parameterisation of the map, in contrast to a neural net. The cost of stepping from one cell to the other is the length of the step and we likewise use the Euclidean distance to the target as our A* heuristic.

Our A*-search yields a shortest path to the target without satisfying any constraints on the agent's movement in this phase. To follow this path, given as a sequence of waypoints, while obeying the limitations of the agent's dynamics, we use a low-level controller (fig. 2, top right). We assume that the agent can rotate in-place up to a maximum angular velocity, and always moves in the direction of its current heading subject to some maximum velocity. This allows using a simple

1. In practice, we obtain $\mathbf{p}^k$ from pixel coordinates, depth d^k and the inverse intrinsic matrix $\mathbf{K}^{-1}$ in the observed image. We use it directly in eq. (4) for brevity.

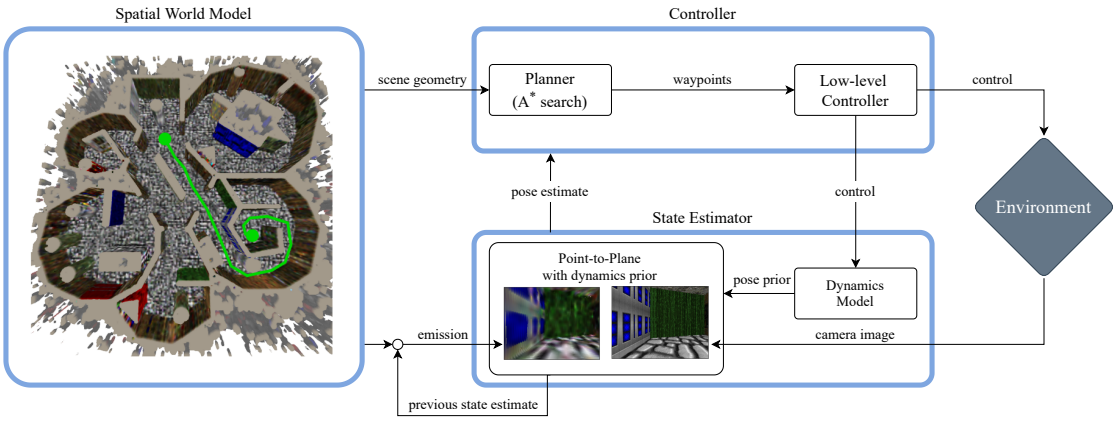


Figure 2: Overview of our method. The planner produces a set of waypoints based on the scene geometry learned by the world model. The low-level controller picks an action to move to the next waypoint. We estimate the agent's state by combining the agent's dynamics with the current camera image, which is aligned against an image predicted by the model.

low-level controller, which turns the agent at the nearest waypoint and then moves towards it. Our method can be applied to other systems, as long as a low-level controller capable of following the A* plan is available.

4.2. State Estimation

Our control rule requires us to continuously track the orientation and 2D location of the agent. We do so using a tracker that combines a transition model with the RGB-D images observed by the agent (*state estimator* in fig. 2). In the following, we will denote actions with $\mathbf{u}$, see section 5 for a description of the assumed noise model for the transition $p(\mathbf{z}_t \mid \mathbf{z}_{t-1}, \mathbf{u}_{t-1})$. Assuming a point-estimate of $\boldsymbol{\mathcal{M}}$ we can combine it with the transition into a state-space model:

$$p(\mathbf{z}_{2:T}, \mathbf{x}_{1:T} \mid \mathbf{u}_{1:T-1}, \mathbf{z}_1, \boldsymbol{\mathcal{M}}) = p(\mathbf{x}_1 \mid \mathbf{z}_1, \boldsymbol{\mathcal{M}}) \prod_{t=2}^{T} p(\mathbf{z}_t \mid \mathbf{z}_{t-1}, \mathbf{u}_{t-1}) p(\mathbf{x}_t \mid \mathbf{z}_t, \boldsymbol{\mathcal{M}}).$$

We are interested in the filtering posterior $p(\mathbf{z}_t \mid \mathbf{x}_t, \mathbf{x}_{1:t-1}, \mathbf{u}_{1:t-1}, \mathbf{z}_1, \boldsymbol{\mathcal{M}})$, shortened with $p^t_{\text{filter}}(\mathbf{z}_t)$:

$$\begin{aligned} p^t_{\text{filter}}(\mathbf{z}_t) &\propto p(\mathbf{x}_t \mid \mathbf{z}_t, \boldsymbol{\mathcal{M}}) p(\mathbf{z}_t \mid \mathbf{x}_{1:t-1}, \mathbf{u}_{1:t-1}, \mathbf{z}_1, \boldsymbol{\mathcal{M}}) \\ &= p(\mathbf{x}_t \mid \mathbf{z}_t, \boldsymbol{\mathcal{M}}) \mathbb{E}_{\mathbf{z}_{t-1} \sim p^{t-1}_{\text{filter}}(\cdot)} \left[p(\mathbf{z}_t \mid \mathbf{z}_{t-1}, \mathbf{u}_{t-1}) \right], \end{aligned} \tag{5}$$

for which we employ a maximum a-posteriori (MAP) approximation:

$$\begin{aligned} &\arg\max_{\mathbf{z}_t} \ \log p(\mathbf{x}_t \mid \mathbf{z}_t, \boldsymbol{\mathcal{M}}) + \log \left(\mathbb{E}_{\mathbf{z}_{t-1} \sim p^{t-1}_{\text{filter}}(\cdot)} \left[p(\mathbf{z}_t \mid \mathbf{z}_{t-1}, \mathbf{u}_{t-1}) \right] \right) \\ &\geq \log p(\mathbf{x}_t \mid \mathbf{z}_t, \boldsymbol{\mathcal{M}}) + \mathbb{E}_{\mathbf{z}_{t-1} \sim p^{t-1}_{\text{filter}}(\cdot)} \left[\log p(\mathbf{z}_t \mid \mathbf{z}_{t-1}, \mathbf{u}_{t-1}) \right] \\ &\approx \log p(\mathbf{x}_t \mid \mathbf{z}_t, \boldsymbol{\mathcal{M}}) + \log p(\mathbf{z}_t \mid \mathbf{z}^*_{t-1}, \mathbf{u}_{t-1}). \end{aligned} \tag{6}$$

The second line follows from Jensen's inequality. In the last line we approximate the expectation using the previous MAP estimate $\mathbf{z}^*_{t-1}$.

Integrating the agent's desired angular and forward velocities from the action $\mathbf{u}_{t-1}$, we can arrive at a prediction for the current state through $p(\mathbf{z}_t \mid \mathbf{z}^*_{t-1}, \mathbf{u}_{t-1})$, reflected in the second term of eq. (6). This prediction is imperfect due to noisy dynamics. We can refine it by comparing the agent's RGB-D observation against the map by optimising for $p(\mathbf{x}_t \mid \mathbf{z}_t, \boldsymbol{\mathcal{M}})$. In the literature of differentiable rendering, this is typically done by minimising the difference between the RGB-D observation $\mathbf{x}_t$ and an image $\hat{\mathbf{x}}_t$ rendered from the pose variable $\mathbf{z}_t$ that is subject to optimisation. This is equivalent to maximising the log-likelihood term $\log p(\mathbf{x}_t \mid \mathbf{z}_t, \boldsymbol{\mathcal{M}})$ above, and we will refer to it as emission-based tracking going forward.

While eq. (6) is well-aligned with the assumed generative model we follow (Mirchev et al., 2021), it is weighed down by the optimisation of $\log p(\mathbf{x}_t \mid \mathbf{z}_t, \boldsymbol{\mathcal{M}})$ due to the high computational cost of propagating gradients through the renderer. We provide an empirical analysis of this aspect in section 5. With this in mind, we opt for one more approximation, substituting the first term in eq. (6) for the prediction-to-observation objective introduced in section 3.2:

$$\begin{aligned}\arg\min_{\mathbf{z}_t} \sum_k \left\| \hat{\mathbf{x}}^{\mathrm{rgb}}_{t-1}[\pi(\mathbf{T}^{\mathbf{z}^*_{t-1}}_{\mathbf{z}_t}\mathbf{p}^k_t)] - \mathbf{x}^{\mathrm{rgb}}_t[\pi(\mathbf{p}^k_t)] \right\|_1 \\ + \sum_k \left\| \langle \hat{\mathbf{p}}^k_{t-1} - \mathbf{T}^{\mathbf{z}^*_{t-1}}_{\mathbf{z}_t}\mathbf{p}^k_t, \hat{\mathbf{n}}^k_{t-1} \rangle \right\|_1 - \log p(\mathbf{z}_t \mid \mathbf{z}^*_{t-1}, \mathbf{u}_{t-1}).\end{aligned} \tag{7}$$

Here, $\mathbf{T}^{\mathbf{z}^*_{t-1}}_{\mathbf{z}_t}$ denotes the relative pose between step t and $t-1$ (going backward in time), which is a function of the optimised $\mathbf{z}_t$ and the previous MAP estimate $\mathbf{z}^*_{t-1}$. With that pose, we reproject points between $\hat{\mathbf{x}}^{\mathrm{rgb}}_{t-1}$, a rendered mean prediction of the RGB image at the previous time step, and the current RGB observation $\mathbf{x}^{\mathrm{rgb}}_t$. For the point-to-plane objective, $\mathbf{p}^k_t$ is a 3D point in the camera frame of the current observation, computed based on the observed depth $\mathbf{x}^{\mathrm{d}}_t$, and $\hat{\mathbf{p}}^k_{t-1}$ and $\hat{\mathbf{n}}^k_{t-1}$ are respectively the corresponding 3D point and normal in the previous camera frame, found through projective data association (Blais and Levine, 1995). The optimised states $\mathbf{z}_t$ are parameterised in the $\mathfrak{se}(3)$ Lie-algebra for the special Euclidean group $\mathrm{SE}(3)$, and we assume an L1 norm (Laplace assumption) for robustness to outliers. We uniformly sample a constant number of pixels for which the objective is evaluated, approximating gradients in expectation. We perform 100 steps of gradient descent for each new time step using the Adam optimizer (Kingma and Ba, 2015).

By substituting the rendering log-likelihood from eq. (6) for the photometric and point-to-plane loss terms in eq. (7), we maintain the necessary geometric constraints in addition to satisfying the assumed transition model. Note that in the revised objective new observations are still anchored to the map, as the projective transformation happens between a *rendered* prediction $\hat{\mathbf{x}}_{t-1}$ and the new incoming observation $\mathbf{x}_t$. The chosen objective reflects a conditional independence assumption $\mathbf{x}_t \perp\!\!\!\perp \boldsymbol{\mathcal{M}} \mid \mathbf{x}_{t-1}, \mathbf{z}_{t-1}, \mathbf{z}_t$ between the map $\boldsymbol{\mathcal{M}}$ and current observation $\mathbf{x}_t$ given a previous $\mathbf{x}_{t-1}$ and relative pose offset, which is a reasonable approximation for consecutive time steps. We opt for this tracking approach in favour of optimising through the emission model as it is faster and just as accurate in our setting, as we will show in experiments. To the best of our knowledge combining photometric, point-to-plane and dynamics constraints in one objective for state estimation has not been explored before for differentiable rendering.

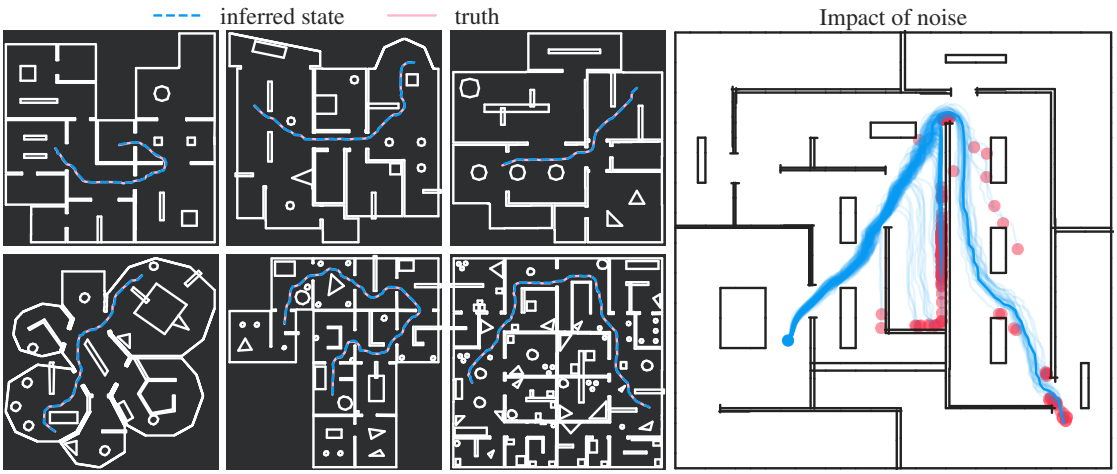


Figure 3: Evaluation levels with successful navigation runs (left) and the training maze (right). The solid blue line is a reference trajectory. The stochasticity of the dynamics is illustrated by showing multiple samples based on the same control inputs: since each trajectory ends up in a different place, state estimation and closed-loop control are crucial.

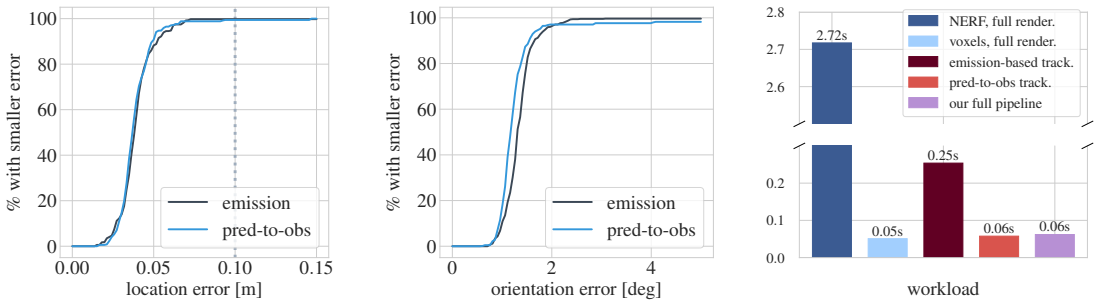
5. Experiments

We use the Vizdoom simulator for our quantitative experiments (Wydmuch et al., 2018). Despite its limited visual fidelity, Vizdoom allows us to focus on environments that are large and complex, i.e. with multiple rooms and realistic floor plans, as we can easily modify the simulation maps. To that end, we extracted six floor plans from the HouseExpo dataset Li et al. (2020), which we converted to Vizdoom levels with added obstacles. Figure 3 shows the evaluation environments, along with a training level taken from (Savinov et al., 2018). An agent body length of 0.2m, as suggested by Anderson et al. (2018), corresponds to an environment size of up to a $11.4 \times 11.4\text{m}^2$ sized square. A voxel map is learned for each environment from 5000 RGB-D images with pose labels. Our extended technical report describes the setup and models in detail (Kayalibay et al., 2022).

To analyze navigation performance under stochastic dynamics, we add noise to the agent motion in the simulator. Here, we make two considerations: noise should only be applied when the agent tries to move and it should not invert the direction of an action. The latter means that if the agent tries to rotate left, the noise should not make it rotate right. Likewise, if the agent tries to move forward, noise should not make it move backward. Guided by these principles, we add clipped Gaussian distributed noise to the angular velocity and speed, where the clipping ensures that the direction of turning (left/right) or movement (forward/backward) is not inverted.

Formally, the controls of the agent are $\mathbf{u} = (\dot{\alpha}, o, s)$, where $\dot{\alpha}$ is the angular velocity, o the angle of the movement direction and s the speed along that direction. The movement angle is distinct from the current heading angle α but must align with it up to a threshold of $5°$, and there are likewise maximum values defined for the angular velocity and speed at $11.5°$ and 0.8 times the agent's own body length. If we take l to be the 2D location of the agent, the noisy dynamics then amount to:

$$\alpha_{t+1} = \alpha_t + \text{sign}(\dot{\alpha}_t)\max(0, |\dot{\alpha}| + \epsilon_t^\alpha), \qquad \epsilon_t^\alpha \sim \mathcal{N}(0, \sigma_\alpha)$$



(a) (b) (c)

Figure 4: (a,b) CDF of the location & orientation RMSEs. The vertical line shows $0.5 \times \text{agent\_size}$. (c) Average runtimes of a single step for different computations: NeRF rendering, voxel map rendering, emission–based tracking, tracking with our scheme, tracking and control.

for the orientation and

$$l_{t+1} = l_t + \begin{bmatrix} \cos(\epsilon_t^\alpha + o_t) \\ \sin(\epsilon_t^\alpha + o_t) \end{bmatrix} \max(0, s_t + \epsilon_t^s), \quad \epsilon_t^s \sim \mathcal{N}(0, \sigma_s)$$

for the location, where the angular velocity noise is added to the direction of movement and t denotes time. In addition to the noise, there is another source of error in the agent's dynamics, which results from the simulator. Vizdoom only allows for setting an integer angle in $[0°, 360°]$ degrees and the 2D location can only be specified up to two decimal points of precision. Together, these factors make for enough perturbation to pose a non-trivial challenge against navigation. This effect is illustrated in the right plot of fig. 3.

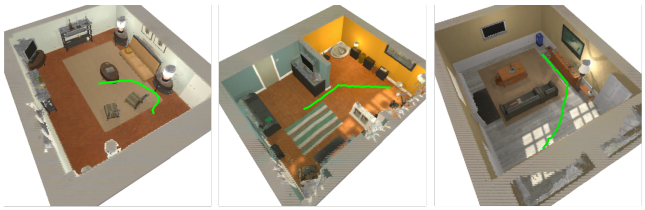
State estimation in real-time First, we compare the tracker described in section 4.2 to pose estimation via gradient backpropagation through the rendering emission. We use 3° and 5% of the agent body length for the rotational and translational Gaussian scales of the noisy simulator dynamics. Figures 4(a) and 4(b) show the accuracy of the two approaches, displaying the percentage of trajectories with an RMSE smaller than the number on the x-axis. Both estimators have adequate performance, with a location RMSE smaller than 0.1m more than 99% of the time. Similarly, the orientation RMSEs are roughly on par.

However, the main advantage of the state estimator from section 4.2 is its speed. As shown in fig. 4(c), a single step takes 0.06s (16.7Hz) on average, compared to 0.25s (4Hz) for the emission-based tracking. This is due to avoiding the computational cost of propagating gradients through the renderer (cf. section 4.2). We also note the major runtime difference when rendering from a NeRF map (Mildenhall et al., 2020) at 2.72s (0.4Hz) and from a voxel map (Mirchev et al., 2021) at 0.05s (20Hz) per image (fig. 4(c), left) respectively. The aim for on-line control made us employ the latter approach. The overall runtime of our pipeline is 0.062s (16.1Hz) per time step, with the bulk of it occupied by the state estimator. The proposed approach amounts to stable real-time tracking under noisy dynamics with only RGB-D observations. We believe this is a necessary component in the context of on-line control.

Navigation Next we focus on our overarching objective – successful point-to-point navigation. Since our tracker works by combining transition estimates with RGB-D observations that are compared against the map, we conduct an ablation study to check how much each part is contributing.

noise	ours	no map	dynamics
high	**0.46**	0.37	0.33
mid	**0.79**	0.51	0.52
low	**0.92**	0.61	0.61

(a)



(b)

Figure 5: (a) Average SPL over six environments. Higher is better. **No map** tracks by optimising eq. (7) using the previous camera image instead of the model prediction. **Dynamics** merely integrates the dynamics. (b) Isometric views of the learned maps and successful navigation trajectories from AI2-THOR. The **low** noise setting was used.

Here, we compare our pipeline to a) tracking by using the transition model only, i.e. path integration; b) tracking by optimising a pose offset between two consecutive RGB-D observations (instead of using an emitted image from the map). We repeat our evaluation under three different levels of noise: **high** ($\sigma_a = 9°$, $\sigma_s = 30\%$ of agent size), **medium** ($\sigma_a = 6°$, $\sigma_s = 15\%$) and **low** ($\sigma_a = 3°$, $\sigma_s = 10\%$).

In our evaluation, we sample random pairs of starting positions and targets from the free space of the level, with the constraint that the start and target must be farther apart than three times the agent's body length. We consider a navigation attempt as successful if the agent's final position is closer to the target than two times the agent's body length, and use 200 navigation tasks per environment. The same navigation tasks are used for each noise level and method.

Our central evaluation metric is *success weighted by path length* (SPL) (Anderson et al., 2018), which is defined as:

$$\text{SPL} = \sum_{s_i \in \mathcal{S}} s_i \frac{l_i}{\max(p_i, l_i)}, \tag{8}$$

where $\mathcal{S}$ is a set of navigation tasks, s_i is a binary variable indicating the success of a task, and p_i and l_i are the length of the path taken by the agent and the length of the optimal path. We have found that SPL and success rate are almost the same for all three approaches. This indicates that successful navigation runs align well with the optimal trajectory.

The results of our evaluation can be found in fig. 5(a). We find that our proposed navigation approach performs best under all noise levels, yet it also degrades at higher levels of noise. Finally, we qualitatively demonstrate that our work extends to AI2-THOR (Kolve et al., 2017) in fig. 5(b), which is visually much more realistic than Vizdoom.

6. Conclusion

We have introduced a method for real-time navigation based on a differentiably rendered world model. Our approach is able to solve navigation tasks in complex environments using only RGB-D observations and under different levels of noise. We demonstrated that the state estimation problem can be solved by applying a point-to-plane metric to the differentiable rendering setting and combining it with a dynamics model, which allows for faster tracking compared to differentiating through the renderer. In future work, we will extend our method to environments with a higher visual fidelity and more complex dynamics.

References

Michal Adamkiewicz, Timothy Chen, Adam Caccavale, Rachel Gardner, Preston Culbertson, Jeanette Bohg, and Mac Schwager. Vision-only robot navigation in a neural radiance world, 2021. URL https://mikh3x4.github.io/nerf-navigation/assets/NeRF_Navigation.pdf.

Peter Anderson, Angel Chang, Devendra Singh Chaplot, Alexey Dosovitskiy, Saurabh Gupta, Vladlen Koltun, Jana Kosecka, Jitendra Malik, Roozbeh Mottaghi, Manolis Savva, and Amir R. Zamir. On evaluation of embodied navigation agents, 2018.

Cedric Audras, A Comport, Maxime Meilland, and Patrick Rives. Real-time dense appearance-based slam for rgb-d sensors. In Australasian Conf. on Robotics and Automation, volume 2, pages 2–2, 2011.

Philip Becker-Ehmck, Maximilian Karl, Jan Peters, and Patrick van der Smagt. Learning to fly via deep model-based reinforcement learning. CoRR, abs/2003.08876, 2020. URL https://arxiv.org/abs/2003.08876.

Gérard Blais and Martin D. Levine. Registering multiview range data to create 3d computer objects. IEEE Transactions on Pattern Analysis and Machine Intelligence, 17(8):820–824, 1995.

Devendra Singh Chaplot, Dhiraj Gandhi, Saurabh Gupta, Abhinav Gupta, and Ruslan Salakhutdinov. Learning to explore using active neural slam, 2020.

Tao Chen, Saurabh Gupta, and Abhinav Gupta. Learning exploration policies for navigation, 2019.

Yang Chen and Gérard Medioni. Object modelling by registration of multiple range images. Image and vision computing, 10(3):145–155, 1992.

Marco Fraccaro, Danilo Jimenez Rezende, Yori Zwols, Alexander Pritzel, S. M. Ali Eslami, and Fabio Viola. Generative temporal models with spatial memory for partially observed environments, 2018.

Karol Gregor, Danilo Jimenez Rezende, Frederic Besse, Yan Wu, Hamza Merzic, and Aaron van den Oord. Shaping belief states with generative environment models for rl, 2019.

Saurabh Gupta, Varun Tolani, James Davidson, Sergey Levine, Rahul Sukthankar, and Jitendra Malik. Cognitive mapping and planning for visual navigation, 2019.

Danijar Hafner, Timothy P. Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA, pages 2555–2565, 2019. URL http://proceedings.mlr.press/v97/hafner19a.html.

Lukasz Kaiser, Mohammad Babaeizadeh, Piotr Milos, Blazej Osinski, Roy H Campbell, Konrad Czechowski, Dumitru Erhan, Chelsea Finn, Piotr Kozakowski, Sergey Levine, Afroz Mohiuddin, Ryan Sepassi, George Tucker, and Henryk Michalewski. Model-based reinforcement learning for atari, 2020.

Baris Kayalibay, Atanas Mirchev, Maximilian Soelch, Patrick van der Smagt, and Justin Bayer. Navigation and planning in latent maps, 2018. URL http://reinforcement-learning.ml/papers/pgmrl2018_kayalibay.pdf.

Baris Kayalibay, Atanas Mirchev, Patrick van der Smagt, and Justin Bayer. Tracking and planning with spatial world models. arXiv, 2022. URL https://arxiv.org/abs/2201.10335.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings, 2015. URL http://arxiv.org/abs/1412.6980.

Eric Kolve, Roozbeh Mottaghi, Winson Han, Eli VanderBilt, Luca Weihs, Alvaro Herrasti, Daniel Gordon, Yuke Zhu, Abhinav Gupta, and Ali Farhadi. AI2-THOR: An Interactive 3D Environment for Visual AI. arXiv, 2017.

Tingguang Li, Danny Ho, Chenming Li, Delong Zhu, Chaoqun Wang, and Max Q. H. Meng. House-expo: A large-scale 2d indoor layout dataset for learning-based algorithms on mobile robots, 2020.

Yunzhu Li, Shuang Li, Vincent Sitzmann, Pulkit Agrawal, and Antonio Torralba. 3d neural scene representations for visuomotor control, 2021.

Stephen Lombardi, Tomas Simon, Jason Saragih, Gabriel Schwartz, Andreas Lehrmann, and Yaser Sheikh. Neural volumes. ACM Transactions on Graphics, 38(4):114, Jul 2019. ISSN 1557-7368. doi: 10.1145/3306346.3323020. URL http://dx.doi.org/10.1145/3306346.3323020.

Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis, 2020.

Atanas Mirchev, Baris Kayalibay, Maximilian Soelch, Patrick van der Smagt, and Justin Bayer. Approximate bayesian inference in spatial environments, 2019.

Atanas Mirchev, Baris Kayalibay, Patrick van der Smagt, and Justin Bayer. Variational state-space models for localisation and dense 3d mapping in 6 dof. In International Conference on Learning Representations, 2021. URL https://openreview.net/forum?id=XAS3uKeFWj.

Thomas Müller, Brian McWilliams, Fabrice Rousselle, Markus Gross, and Jan Novk. Neural importance sampling, 2019.

Richard A Newcombe, Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew J Davison, Pushmeet Kohi, Jamie Shotton, Steve Hodges, and Andrew Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In 2011 10th IEEE international symposium on mixed and augmented reality, pages 127–136. IEEE, 2011.

Michael Niemeyer, Lars Mescheder, Michael Oechsle, and Andreas Geiger. Differentiable volumetric rendering: Learning implicit 3d representations without 3d supervision, 2020.

Matthias Nießner, Michael Zollhöfer, Shahram Izadi, and Marc Stamminger. Real-time 3d reconstruction at scale using voxel hashing. ACM Transactions on Graphics (ToG), 32(6):1–11, 2013.

Sageev Oore, Geoffrey E. Hinton, and Gregory Dudek. A mobile robot that learns its place. Neural Comput., 9(3):683–699, 1997. doi: 10.1162/neco.1997.9.3.683. URL https://doi.org/10.1162/neco.1997.9.3.683.

Emilio Parisotto and Ruslan Salakhutdinov. Neural map: Structured memory for deep reinforcement learning, 2017.

Benjamin Planche, Xuejian Rong, Ziyan Wu, Srikrishna Karanam, Harald Kosch, YingLi Tian, Jan Ernst, and Andreas Hutter. Incremental scene synthesis, 2019.

Santhosh K. Ramakrishnan, Ziad Al-Halah, and Kristen Grauman. Occupancy anticipation for efficient exploration and navigation, 2020.

Christian Reiser, Songyou Peng, Yiyi Liao, and Andreas Geiger. Kilonerf: Speeding up neural radiance fields with thousands of tiny mlps, 2021.

Nikolay Savinov, Alexey Dosovitskiy, and Vladlen Koltun. Semi-parametric topological memory for navigation, 2018.

Vincent Sitzmann, Michael Zollhfer, and Gordon Wetzstein. Scene representation networks: Continuous 3d-structure-aware neural scene representations, 2020.

Frank Steinbrücker, Jürgen Sturm, and Daniel Cremers. Real-time visual odometry from dense rgb-d images. In 2011 IEEE international conference on computer vision workshops (ICCV Workshops), pages 719–722. IEEE, 2011.

Patrick Stotko. State of the art in real-time registration of rgb-d images. In Central European Seminar on Computer Graphics for Students (CESCG 2016), 2016.

Edgar Sucar, Shikun Liu, Joseph Ortiz, and Andrew J. Davison. imap: Implicit mapping and positioning in real-time, 2021.

Zirui Wang, Shangzhe Wu, Weidi Xie, Min Chen, and Victor Adrian Prisacariu. Nerf–: Neural radiance fields without known camera parameters, 2021.

Thomas Whelan, Stefan Leutenegger, R Salas-Moreno, Ben Glocker, and Andrew Davison. Elasticfusion: Dense slam without a pose graph. Robotics: Science and Systems, 2015.

Marek Wydmuch, Michał Kempka, and Wojciech Jaśkowski. Vizdoom competitions: Playing doom from pixels. IEEE Transactions on Games, 2018.

Lior Yariv, Yoni Kasten, Dror Moran, Meirav Galun, Matan Atzmon, Ronen Basri, and Yaron Lipman. Multiview neural surface reconstruction by disentangling geometry and appearance, 2020.

Lin Yen-Chen, Pete Florence, Jonathan T. Barron, Alberto Rodriguez, Phillip Isola, and Tsung-Yi Lin. Inerf: Inverting neural radiance fields for pose estimation, 2021.

Tony Z. Zhao, Anusha Nagabandi, Kate Rakelly, Chelsea Finn, and Sergey Levine. MELD: meta-reinforcement learning from images via latent state models. CoRR, abs/2010.13957, 2020. URL https://arxiv.org/abs/2010.13957.

Prior Work (Not Thesis)

B

B.1. Approximate Bayesian Inference in Spatial Environments

Paper Summary

DVBF-LM (Mirchev et al., 2019) is a neural state-space model with a 2D latent map. The transition is an MLP working in an abstract latent space and the emission is an MLP that predicts either 2D range scans or colour images from the map. DVBF-LM simultaneously addresses SLAM, navigation and exploration in toy scenarios. The model, its inference and exploration were defined in the author's master's thesis. Navigation was added after a separate workshop publication (Kayalibay et al., 2018).

In the early days of the doctorate it was already apparent that DVBF-LM does not have sufficient inductive biases to reliably scale to 3D modelling. At the same time, the demonstrated results in DVBF-LM, albeit in toy environments, were motivation enough to further pursue the direction of spatial generative state-space models. This lead to the development of the three core publications, and thus the thesis as a whole.

Author Contributions

Please note that this publication does not count towards the dissertation, as sections from it overlap with the author's master's thesis. It is included for context, as it set the whole doctorate in motion.

problem definition	significant
literature review	significant
theoretical derivation	significant
implementation	significant
experiments	significant
writing	significant

Robotics: Science and Systems 2019
Freiburg im Breisgau, June 22-26, 2019

Approximate Bayesian inference in spatial environments

Atanas Mirchev[†], Baris Kayalibay[†], Maximilian Soelch[†], Patrick van der Smagt[‡] and Justin Bayer[†]

Machine Learning Research Lab, Volkswagen Group
80805, Munich, Germany
[†]{firstname.lastname}@argmax.ai
[‡]smagt@argmax.ai

***Abstract*—Model-based approaches bear great promise for decision making of agents interacting with the physical world. In the context of spatial environments, different types of problems such as localisation, mapping, navigation or autonomous exploration are typically adressed with specialised methods, often relying on detailed knowledge of the system at hand. We express these tasks as probabilistic inference and planning under the umbrella of deep sequential generative models. Using the frameworks of variational inference and neural networks, our method inherits favourable properties such as flexibility, scalability and the ability to learn from data. The method performs comparably to specialised state-of-the-art methodology in two distinct simulated environments.**

I. INTRODUCTION

Sequential decision making is a framework to represent the interaction of an agent with its environment: an observation of the world is presented to the agent, upon which the informed agent picks an action, which in turn alters the world's state. One instance of interest are spatial environments such as mobile robots on a factory floor, autonomous cars, robot arms or unmanned aerial vehicles. Various tasks are of interest in these scenarios. Localisation or pose estimation considers the relation of the agent itself to the environment. This is often combined with establishing a map of its surroundings and has been referred to as simultaneous localisation and mapping (SLAM) by the robotics community. If the aim is to obtain that map efficiently, autonomous exploration is about devising trajectories that uncover the map with as little effort as possible. Another goal is navigation, referring to the generation of plans that allow the agent to reach a pre-specified location.

We set out to address these problems in a unified framework. We augment a non-linear state space model, the deep variational Bayes filter [12] with a global latent variable representing a map (DVBF-LM). We propose the necessary learning algorithms that enable end-to-end learning and make it possible to express the aforementioned tasks as either inference or planning in that model. SLAM is performed by approximate, variational inference of the joint posterior over maps and pose trajectories. We rely on a standard formulation as probabilistic inference in a graphical model. For autonomous exploration, we optimise the expected information gain with respect to the control signals [25, 21]. This is possible due to the probabilistic treatment of the map, where unexploredness is related to the remaining uncertainty in the respective region. Navigation is implemented as planning in a discretisation of the learned model.

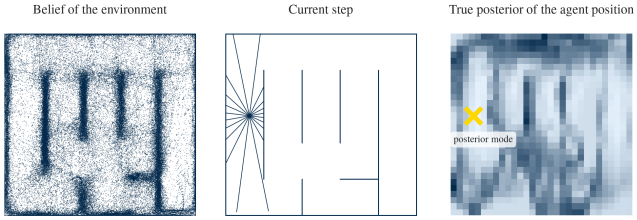


Fig. 1: Example for the agent pose posterior in a LiDAR environment (pybox2d): (a) Scatter plot of observations predicted from the learned DVBF-LM map. (b) The current agent position and LiDAR readings. (c) Plot of the pose posterior over locations given the current observation. It is highly non-Gaussian and has several maxima.

Our contributions are:

- a deep non-linear state-space model with an explicit map component that can be estimated from data;
- methods of performing SLAM, autonomous exploration and navigation in said model;
- a solution to train stochastic recurrent models on a single, long, consecutive time series;
- a variational posterior formulation that copes with the complex joint posterior prevalent in SLAM (cf. Figure 1).

We validate the claims in a series of extensive experiments where we perform comparably to baselines tailored specifically to the respective settings.

II. RELATED WORK

The problem of concurrent estimation of an agent's pose and its surrounding has seen considerable attention in the last decades. We refer the interested reader to the survey of Cadena et al. [3]. A contribution of Murphy [17] is most similar to our approach: the map is a matrix-valued global latent variable inferred through Bayesian methods.

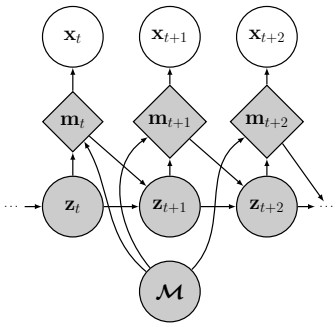


Fig. 2: Sequential graphical model with global map $\mathcal{M}$ and local charts $\mathbf{m}_t$.

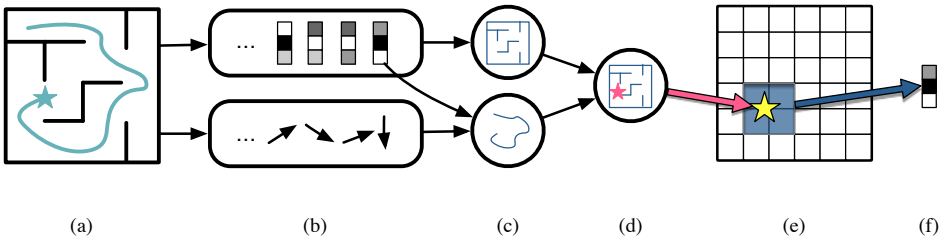


Fig. 3: *Illustration of pose inference.* (a) An agent (teal star) traverses a maze, (b) collecting sensor readings (top) and control signals (bottom). (c) A belief of the map is formed from observations (top), a belief of the trajectory is formed from observations and controls (bottom). (d) The two beliefs are fused. (e) The map is indexed with a pose-based attention mechanism. (f) The attended region of the map is used to reconstruct the observation.

Mapping and localisation has been adopted in the machine-learning community mostly to solve reinforcement-learning or visual-navigation problems [18, 19]. Fraccaro et al. [7] proposed a generative model for spatial environments. While their approach is similar to ours, their focus was primarily on simulator performance over long time spans. Further, an external memory is used which does not directly represent a random variable as part of a graphical model.

Early discussions of the importance of exploration of model parameters can be found in [20, 23], and information-theoretic methods for spatial exploration can be traced back to [6, 26]. Our work follows the framework of *curiosity-driven exploration* [20]. In the spatial environment context, it represents an instance of active SLAM [25, 3].

A large body of recent spatial exploration methods [21, 1, 24, 27] is driven by information theory, but assumes particular discretisations of the state space (occupancy grids) or skeletonisations of the possible action paths. Variational information maximizing exploration (VIME) [11] is closely related to our method, but imposes an intentional Gaussian constraint for tractability and mutual information is estimated for one step t at a time. Our method has no such assumptions.

III. DEEP VARIATIONAL BAYES FILTER WITH A LATENT MAP

Our aim is to provide a unified model that covers all of the aforementioned tasks. We defer the discussion of their exact implementations to Section IV, Section V and Section VI. Here we will focus on the underlying probabilistic generative model. We step on the solid foundation of state space models to represent sequential agent interactions:

$$p(\mathbf{x}_{1:T}, \mathbf{z}_{1:T} \mid \mathbf{u}_{1:T-1}) = p(\mathbf{z}_1) \prod_{t=2}^{T} p(\mathbf{z}_t \mid \mathbf{z}_{t-1}, \mathbf{u}_{t-1}) \prod_{t=1}^{T} p(\mathbf{x}_t \mid \mathbf{z}_t),$$

where

$\mathbf{x}_{1:T} \in \mathbb{R}^{T \times D_x}$	is a sequence of observations,
$\mathbf{z}_{1:T} \in \mathbb{R}^{T \times D_z}$	is a sequence of poses, and
$\mathbf{u}_{1:T-1} \in \mathbb{R}^{T-1 \times D_u}$	is a sequence of control inputs.

In terms of spatial environments, the *transition model* $p(\mathbf{z}_t \mid \mathbf{z}_{t-1}, \mathbf{u}_{t-1})$ represents the dynamics and the *emission model* $p(\mathbf{x}_t \mid \mathbf{z}_t)$ simulates the formation of observations.

The traditional state space model above is a powerful tool for the analysis of stochastic dynamical systems [12]. In order to unlock further inferences specific to spatial environments, more transparency is required. We thus introduce additional structure to provide the necessary entry points. First, we identify a part of the latent space with the environment itself. To that end we extend the traditional graphical model, incorporating a global map latent variable $\mathcal{M} \sim p(\mathcal{M})$. We introduce a latent middle layer of local charts $\mathbf{m}_t \sim p(\mathbf{m}_t \mid \mathbf{z}_t, \mathcal{M})$. Intuitively, the chart $\mathbf{m}_t$ represents the currently relevant attended region of the map. It shapes the transition of poses over time, $p(\mathbf{z}_{t+1} \mid \mathbf{z}_t, \mathbf{m}_t, \mathbf{u}_t)$. The observation emission model operates solely on these charts, $\mathbf{x}_t \sim p(\mathbf{x}_t \mid \mathbf{m}_t)$. In total, this yields the graphical model (cf. Figure 2)

$$p(\mathbf{x}_{1:T}, \mathbf{z}_{1:T}, \mathbf{m}_{1:T}, \mathcal{M} \mid \mathbf{u}_{1:T-1}) = p(\mathcal{M})\rho(\mathbf{z}_1) \prod_{t=1}^{T} p(\mathbf{x}_t \mid \mathbf{m}_t)\, p(\mathbf{m}_t \mid \mathbf{z}_t, \mathcal{M}) \prod_{t=1}^{T-1} p(\mathbf{z}_{t+1} \mid \mathbf{z}_t, \mathbf{m}_t, \mathbf{u}_t).$$

A common assumption is that the factors are governed by a set of parameters $\boldsymbol{\theta}$, i.e. $\rho_{\boldsymbol{\theta}_I}(\mathbf{z}_1)$, $p_{\boldsymbol{\theta}_T}(\mathbf{z}_{t+1} \mid \mathbf{z}_t, \mathbf{u}_t, \mathbf{m}_t)$, $p_{\boldsymbol{\theta}_E}(\mathbf{x}_t \mid \mathbf{m}_t)$. We will leave out the dependency for notational brevity in the remainder of this work. Further, in the conducted experiments we assume the transition parameters $\boldsymbol{\theta}_T$ and the initial state distribution are learned *a priori* or engineered.

A. Approximation via Variational Inference

Exact inference in such models is typically intractable. We obtain variational approximations $q(\mathcal{M})$ and $q(\mathbf{z}_{1:T} \mid \mathbf{x}_{1:T}, \mathbf{u}_{1:T-1}, \mathcal{M})$ of the corresponding posteriors, collecting their learnable variational parameters in $\boldsymbol{\phi}$, where we rely on Bayes by backprop [2] for the former and on SGVB [14] for the latter. The negative *evidence lower bound* (ELBO)

is given as

$$\mathcal{L}_{\text{elbo}} = \underbrace{\mathbb{E}_q\left[-\log p\left(\mathbf{x}_{1:T} \mid \mathbf{z}_{1:T}, \boldsymbol{\mathcal{M}}\right)\right]}_{=:\ell^r} + \underbrace{\mathrm{KL}(q(\boldsymbol{\mathcal{M}}) \,||\, p(\boldsymbol{\mathcal{M}}))}_{=:\ell^{\boldsymbol{\mathcal{M}}}} + \underbrace{\mathbb{E}_q\left[\mathrm{KL}(q\left(\mathbf{z}_{1:T} \mid \mathbf{x}_{1:T}, \boldsymbol{\mathcal{M}}\right) \,||\, p\left(\mathbf{z}_{1:T} \mid \boldsymbol{\mathcal{M}}\right))\right]}_{=:\ell^{\mathbf{z}}}. \quad (1)$$

The conditioning on $\mathbf{u}_{1:T-1}$ is dropped for brevity. We call ℓ^r the *reconstruction loss*, $\ell^{\mathbf{z}}$ the *pose KL penalty* and $\ell^{\boldsymbol{\mathcal{M}}}$ the *map KL penalty*. Inference of poses and the map then comes down to the minimisation of Equation (1) with respect to $\boldsymbol{\phi}$.

B. Implementation of the Generative Model

In general, the geometric properties of the environment that need to be represented will determine the particular form of the map and the associated attention model. For the purposes of this work we follow [17], defining the map $\boldsymbol{\mathcal{M}}$ to be a finite grid of width w and height h. Each grid cell $\boldsymbol{\mathcal{M}}_{ij}$ is a real-valued vector of dimensionality D_m, i.e. $\boldsymbol{\mathcal{M}} \in \mathbb{R}^{w \times h \times D_m}$. As prior for such a latent map cell we use a standard normal, $\boldsymbol{\mathcal{M}}_{ij} \sim \mathcal{N}(\mathbf{0}, \mathbf{1})$. Extracting *local charts* $\mathbf{m}_t$ from the map is done through a convex combination of the memory cells:

$$\mathbf{m}_t = f_m(\mathbf{z}_t, \boldsymbol{\mathcal{M}}) = \sum_{i,j} \alpha(\mathbf{z}_t)_{ij} \boldsymbol{\mathcal{M}}_{ij}.$$

The result is then a point mass:

$$p\left(\mathbf{m}_t \mid \mathbf{z}_t, \boldsymbol{\mathcal{M}}, \boldsymbol{\theta}_M\right) \propto \mathbb{I}\left[\mathbf{m}_t = f_m(\mathbf{z}_t, \boldsymbol{\mathcal{M}})\right].$$

In this implementation, we choose α to be a bilinear interpolation kernel, combining four cells at a time.

The *emission model* and *transition model* are conditional Gaussian distributions with fixed diagonal covariances. The respective means are given by neural networks parameterised by $\boldsymbol{\theta}_E$ and $\boldsymbol{\theta}_T$:

$$p\left(\mathbf{x}_t \mid \mathbf{m}_t\right) = \mathcal{N}\left(\boldsymbol{\mu}_E(\mathbf{m}_t), \mathrm{diag}(\boldsymbol{\sigma}_E^2)\right),$$
$$p\left(\mathbf{z}_{t+1} \mid \mathbf{z}_t, \mathbf{u}_t, \mathbf{m}_t\right) = \mathcal{N}\left(\boldsymbol{\mu}_T(\mathbf{z}_t, \mathbf{u}_t, \mathbf{m}_t), \sigma_T^2 \mathbf{1}\right).$$

C. Design of the Variational Posterior

Inference of poses is done through a variational approximation of the true posterior $q\left(\mathbf{z}_t \mid \mathbf{x}_{1:T}\right) \approx p\left(\mathbf{z}_t \mid \mathbf{x}_{1:T}\right)$, where we left out the control signals $\mathbf{u}_{1:T-1}$ for brevity and will do so for the remainder of this section. The global variable $\boldsymbol{\mathcal{M}}$ poses an atypical challenge for stochastic recurrent models trained with amortised variational inference, for which an intuitive explanation is as follows. Consider the true posterior, which has to account for all possible maps:

$$p\left(\mathbf{z}_t \mid \mathbf{x}_{1:T}\right) = \int p\left(\mathbf{z}_t \mid \boldsymbol{\mathcal{M}}, \mathbf{x}_{1:T}\right) p\left(\boldsymbol{\mathcal{M}} \mid \mathbf{x}_{1:T}\right) d\boldsymbol{\mathcal{M}}.$$

Any parameterised variational approximation $q\left(\mathbf{z}_t \mid \mathbf{x}_{1:T}\right)$ will have to implement its own belief of the map implicitly. During training, this will prove difficult as it has to track the current belief of the generative model to conform to it, as it essentially implements its inverse. The task of the inference model can be substantially eased by informing it of the current belief of the map $q(\boldsymbol{\mathcal{M}})$ explicitly. We choose to do so by implementing q as a bootstrap particle filter [8] with the particle forwarding distribution from Section III-D2 as a proposal distribution:

$$q\left(\mathbf{z}_{1:T} \mid \mathbf{x}_{1:T}, \boldsymbol{\mathcal{M}}\right) = \prod_{t=1}^{T} q\left(\mathbf{z}_t \mid \mathbf{x}_{1:t}, \boldsymbol{\mathcal{M}}\right),$$
$$q\left(\mathbf{z}_t \mid \mathbf{x}_{1:t}, \boldsymbol{\mathcal{M}}\right) \propto \mathbb{E}_{\mathbf{z}_t^{(k)} \sim \hat{q}(\mathbf{z}_t)}\left[\sum_{k=1}^{K} \hat{\omega}_i \mathbb{I}(\mathbf{z}_t = \mathbf{z}_t^{(k)})\right],$$
$$\hat{\omega}_k = \frac{\omega_k}{\sum_j \omega_j},$$
$$\omega_k = \frac{p\left(\mathbf{x}_t \mid \mathbf{z}_t^{(k)}, \boldsymbol{\mathcal{M}}\right) p\left(\mathbf{z}_t^{(k)}\right)}{\hat{q}(\mathbf{z}_t^{(k)})}.$$

This has two immediate consequences. First, the variational posterior used does not have any parameters and is hence not optimised directly. Second, the true posterior is recovered for $K \to \infty$. But most importantly, the importance weights explicitly reflect the map (sampled from an outer expectation over $q(\boldsymbol{\mathcal{M}})$ in Equation (1)) and the proposals in conflict with it will be sorted out in a natural manner as they have lower weights.

The variational approximation of the posterior map $q(\boldsymbol{\mathcal{M}})$ was chosen to follow a mean-field approach with a factorised Gaussian $q(\boldsymbol{\mathcal{M}}) = \prod_i \prod_j \mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{\mathcal{M}}_{ij}}, \boldsymbol{\sigma}^2_{\boldsymbol{\mathcal{M}}_{ij}})$, with variational parameters $\boldsymbol{\mu}_{\boldsymbol{\mathcal{M}}_{ij}}, \boldsymbol{\sigma}^2_{\boldsymbol{\mathcal{M}}_{ij}} \in \boldsymbol{\phi}$.

D. Faster Training with Mini Batches

In practice, inference is typically performed on very long, continuous streams of data recorded from a moving agent. Evaluating the ELBO for the whole trajectory at once proves prohibitive for learning or is downright impossible due to memory limitations. We therefore seek to relax the optimisation while still respecting the underlying model.

1) Decomposing the Loss into a Sum over Time Steps: Under the Markov assumptions, the evidence lower bound from Equation (1) can be written as a sum over time steps:

$$\mathcal{L}_{\text{elbo}} = \ell^r + \ell^{\mathbf{z}} + \ell^{\boldsymbol{\mathcal{M}}} = \mathbb{E}_q\left[\sum_{t=1}^{T} \ell_t^r + \ell_t^{\mathbf{z}} + \ell_t^{\boldsymbol{\mathcal{M}}}\right] \quad (2)$$

with, leaving out the control signals $\mathbf{u}_{1:T-1}$ for brevity:

$$\ell_t^r = -\log p\left(\mathbf{x}_t \mid \mathbf{z}_t, \boldsymbol{\mathcal{M}}\right), \qquad \ell^r = \mathbb{E}_q\left[\sum_{t=1}^{T} \ell_t^r\right],$$
$$\ell_t^{\mathbf{z}} = \log \frac{q(\mathbf{z}_t \mid \mathbf{x}_{1:t}, \boldsymbol{\mathcal{M}})}{p\left(\mathbf{z}_t \mid \mathbf{z}_{t-1}, \boldsymbol{\mathcal{M}}\right)}, \qquad \ell^{\mathbf{z}} = \mathbb{E}_q\left[\sum_{t=1}^{T} \ell_t^{\mathbf{z}}\right],$$
$$\ell_t^{\boldsymbol{\mathcal{M}}} = \frac{1}{T} \log \frac{q(\boldsymbol{\mathcal{M}})}{p(\boldsymbol{\mathcal{M}})}, \qquad \ell^{\boldsymbol{\mathcal{M}}} = \mathbb{E}_q\left[\sum_{t=1}^{T} \ell_t^{\boldsymbol{\mathcal{M}}}\right].$$

Following [2], we distribute the contribution of the map KL penalty term over different time steps, reflected in $\ell_t^{\boldsymbol{\mathcal{M}}}$. We denote the overall loss at time step t as $\mathcal{L}_t$.

If the loss function is a sum over independent terms, a gradient estimator using only a subset of those terms will be unbiased. Unfortunately the terms for each time step in Equation (2) are not independent, which requires ancestral sampling from the whole Markov chain.

2) Approximate Asynchronous Particle Representation: To overcome this issue we maintain sets of N particles $\boldsymbol{\xi}_t^{(n)}, n = 1, \ldots, N; t = 1, \ldots, T$ that cache samples for each step of the variational posterior over poses $q(\mathbf{z}_{1:T} \mid \mathbf{x}_{1:T}, \boldsymbol{\mathcal{M}}) = \prod_{t=1}^{T} q(\mathbf{z}_t \mid \mathbf{x}_{1:t}, \boldsymbol{\mathcal{M}})$ during training.

We define an estimator of the gradients from the complete ELBO, akin to stochastic gradient descent. The time steps we wish to use for gradient estimation are gathered in a minibatch $\mathcal{B}$. We then approximate the loss given in Equation (2) via

$$\tilde{\mathcal{L}} = \frac{T}{|\mathcal{B}|} \sum_{t \in \mathcal{B}} \mathbb{E}_{\boldsymbol{\mathcal{M}} \sim q} \left[\mathbb{E}_{\mathbf{z}_t \sim \tilde{q}} \left[\ell_t^r + \ell_t^{\mathbf{z}} + \ell_t^{\boldsymbol{\mathcal{M}}} \right] \right], \tag{3}$$

where $\tilde{q}(\mathbf{z}_{1:T}) = \prod_{t=1}^{T} \tilde{q}(\mathbf{z}_t)$ is an approximation of $q(\mathbf{z}_{1:T} \mid \mathbf{x}_{1:T}, \boldsymbol{\mathcal{M}})$ based on the cached particles that allows more efficient sampling of $\mathbf{z}_t$. In particular, every $\tilde{q}(\mathbf{z}_t)$ is importance-resampled from an underlying proposal distribution $\hat{q}(\mathbf{z}_t)$, which in turn is based on the particles $\boldsymbol{\xi}_t^{(n)}, n = 1, \ldots, N$. In this work, $\hat{q}(\mathbf{z}_t)$ is represented as a Normal random variable with moments matched from the set of N particles for time step t: $\hat{q}(\mathbf{z}_t) = \mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{\xi}_t}, \boldsymbol{\sigma}^2_{\boldsymbol{\xi}_t})$. The mean $\boldsymbol{\mu}_{\boldsymbol{\xi}_t} = \frac{1}{N} \sum_{n=1}^{N} \boldsymbol{\xi}_t^{(n)}$ and variance $\boldsymbol{\sigma}^2_{\boldsymbol{\xi}_t} = \frac{1}{N} \sum_{n=1}^{N} (\boldsymbol{\mu}_{\boldsymbol{\xi}_t} - \boldsymbol{\xi}_t^{(n)})^2$ are the empirical mean and variance of the particles respectively. The approximating particle sets are updated during gradient estimation: for any training iteration with $t \in \mathcal{B}$, we can update the particles at following time steps $t + k, k = 1, \ldots$:

$$\boldsymbol{\xi}_{t+k}^{(n)} \sim \tilde{q}(\mathbf{z}_t) \prod_{i=1}^{k} p\left(\mathbf{z}_{t+i} \mid \mathbf{z}_{t+i-1}, \boldsymbol{\mathcal{M}}\right),$$

effectively performing importance resampling and moving particles forward through the transition model, refreshing the approximation $\hat{q}(\mathbf{z}_{t+k})$. This leads to an asynchronous procedure: expectations in Equation (2) w.r.t. the approximate posterior over agent poses are implemented through particles stemming from previous training iterations, potentially biasing the gradients. This bias can be controlled with small parameter updates (i.e. $\boldsymbol{\phi}^{(i+1)} \approx \boldsymbol{\phi}^{(i)}$), since we can then expect the expectations to be close as well. In practice, we will choose chunks of consecutive time steps to be the elements of mini batches, requiring to only update the particles at the beginning of each such chunk.

IV. DVBF-LM AS A METHOD FOR SLAM

We first investigate the capabilities of DVBF-LM as a solution to SLAM. Our model is evaluated in two simulated, precisely controlled environments—a 2D environment with laser range finder observations and VizDoom [13]. A detailed description of each, along with additional information regarding the experimental setup can be found in the supplementary material [1].

We randomised seven distinct 2D maze patterns and replicated them in both environments. Each maze was traversed multiple times by two human operators to collect data. For both environments, the transition model $p\left(\mathbf{z}_{t+1} \mid \mathbf{z}_t, \mathbf{u}_t, \mathbf{m}_t\right)$ is pretrained on a first maze that is not considered during evaluation. Performing SLAM then consists of approximating the posterior of the poses and the map $p\left(\mathbf{z}_{1:t}, \boldsymbol{\mathcal{M}} \mid \mathbf{x}_{1:t}, \mathbf{u}_{1:t-1}\right)$ for a single traversal through the optimisation of Equation (1) with respect to the variational posteriors $q(\mathbf{z}_{1:T})$ and $q(\boldsymbol{\mathcal{M}})$.

We consider two cases, *offline* and *online SLAM*. In the first case we optimise for all time steps at once, whereas in the second case we sweep $t = 1, \ldots, T$ to obtain time-step-wise estimates $q(\mathbf{z}_{1:t})$.

All distances in the following experiments are unit-less, the width and height of the considered mazes were set to 1.0.

A. Pybox2d Environment

For this environment we implemented our own 2D simulator using pybox2d, in which the agent's sensors are laser range finders (LiDAR readings).

a) Improving Path Integration: The aim of this set of experiments is to test whether the proposed approximation of the graphical model improves upon direct path integration based on the pretrained transition $p\left(\mathbf{z}_{t+1} \mid \mathbf{z}_t, \mathbf{u}_t, \mathbf{m}_t\right)$ only. For both online and offline SLAM, using a map clearly outperforms the path integration baseline in terms of localisation error: 0.03 ± 0.02 and 0.04 ± 0.02 at time step 3000 for online and offline SLAM respectively. At this time step, the motion model has diverged for most of the sequences with an average error of 0.14 ± 0.1. Most notably, the use of a map practically eliminates drift: after 3000 steps, a relative error of less than $20{,}000^{-1}$, effectively zero, is obtained. This shows that our method stabilises the motion model and keeps the location estimate from diverging. We illustrate the findings in Figure 4.

b) Comparison to Cartographer: Next we compare DVBF-LM's online localisation performance to that of Google's Cartographer [10], which we consider a representative baseline model for 2D LiDAR SLAM. Cartographer is a realtime SLAM system which operates on laser range finder data and is capable of detecting loop closures. In addition to the LiDAR observations collected from pybox2d, we provided Cartographer with the angular velocity of the agent at every time step in the form of IMU readings. In order to improve upon the default Cartographer configuration and tune it to our setup, we performed a hyperparameter search over more than 40 of Cartographer's hyperparameters, with 6000 trials on a held out trajectory of 1000 steps. Both Cartographer and DVBF-LM manage to eliminate drift, with respective errors of 0.05 ± 0.04 and 0.03 ± 0.02 at time step 3000. The proposed graphical model approximation leads to localisation performance that is consistently on par in quality to that of Cartographer. The results from the comparison are depicted in Figure 5a.

[1] Available at: https://arxiv.org/abs/1805.07206.

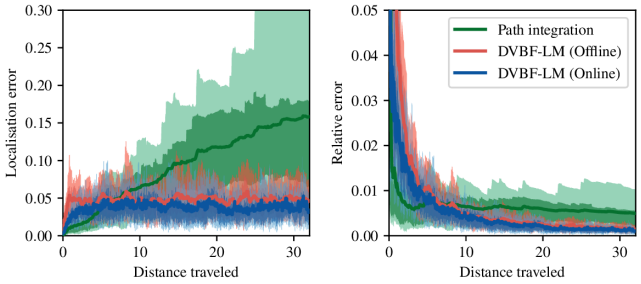
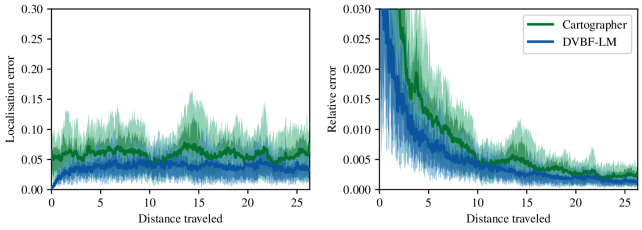
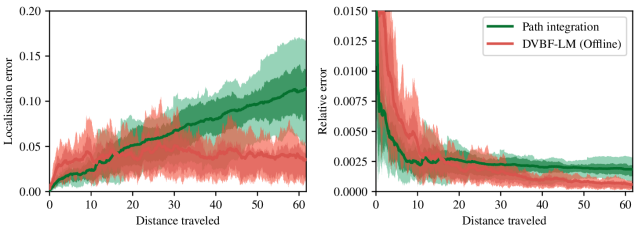


Fig. 4: We compare the online and offline DVBF-LM localisation error to path integration for all 24 test traversals accross 6 mazes in the pybox2d environment. The plots show aggregate results, shaded regions contain 50% and 80% of the traversals. The second plot shows the localisation error relative to the distance travelled.



(a) Pybox2d



(b) VizDoom

Fig. 5: (a) Online DVBF-LM localisation error compared to that of Cartographer on test mazes in the pybox2d environment. We omit the offline DVBF-LM results from Figure 4 for the sake of legibility. (b) Offline DVBF-LM localisation error compared to path integration on test maze traversals in the VizDoom environment. The plots show aggregate results, shaded regions contain 50% and 80% of the traversals. The plots on the right show the localisation error relative to the distance travelled.

B. *VizDoom Environment*

The VizDoom experiments take place in the same set of mazes as pybox2d. Observations are now two-dimensional images taken from the perspective of the agent.

For the VizDoom environment we only investigate offline SLAM performance. The experiment procedure was identical to the pybox2d counterpart. All model components apart from the emission model are kept the same. The latter is modified to better fit visual observations.

We summarise quantitative localisation results in Figure 5b. The method performs on a similar level as in the laser scan-based environment: localisation error is 0.04 ± 0.03 after 5000 time steps. We can see that DVBF-LM is capable of correcting the drift resulting from path integration (an error of 0.11 ± 0.05). The final relative error is $0.06\% \pm 0.06\%$ of the trajectory length. The results indicate that our method is able to perform accurate localisation when applied to different observation modalities in an offline fashion by adapting only the architecture of the emission model.

V. NAVIGATION IN LEARNED ENVIRONMENTS

One incentive for learning a generative model of a spatial environment is that such a model can be used to plan interactions with that environment. As an example, we use DVBF-LM as a black-box environment simulator and provide its predictions to a classical path planning algorithm to solve navigation tasks.

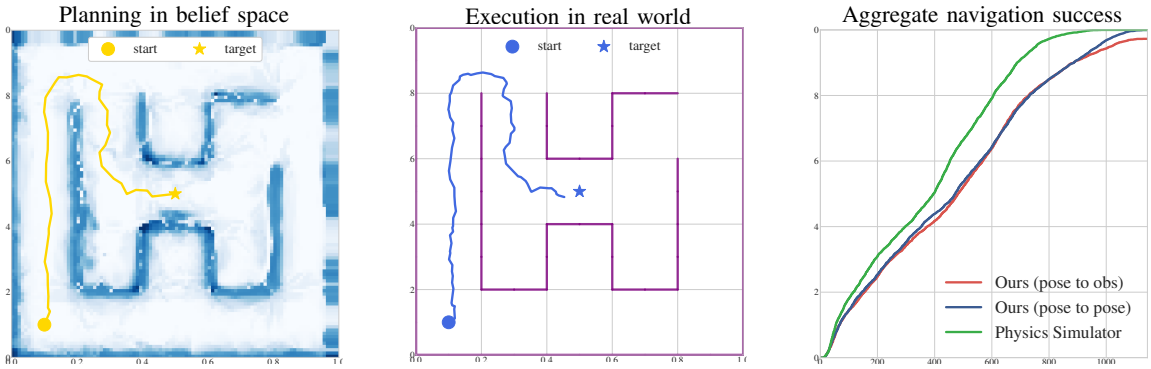
A. *Latent Hybrid-A**

The latent map $\boldsymbol{\mathcal{M}}$ of DVBF-LM conserves the Euclidean geometry of the true environment through the inductive bias of the pretrained transition. Hence, we are able to use the *hybrid-A** algorithm [5] for path planning. The goal of hybrid-A^* is to find a path from a *continuous* starting pose to a *continuous* target pose. This is done by discretising the search space into a grid of N cells $\{\mathbf{c}_n\}_{n \in [N]}$, which the conventional A^*-search [9] can operate on.

To obtain smooth navigation trajectories, every discrete state $\mathbf{c}_n$ is associated with a continuous state $\mathbf{z}_n$—the agent's state when that cell was explored for the first time. New cells $\mathbf{c}_{n+1}$ are explored by picking random sequences of controls $\mathbf{u}^n_{1:K}$ and predicting a following continuous state by applying the controls to the current $\mathbf{z}_n$. The successor states are found by applying the transition model of DVBF-LM and picking the mean of the resulting Gaussian distribution. When the target state is found, we backtrack to obtain a consistent sequence of controls $\mathbf{u}_{1:T}$, which can be executed by the agent to reach the target.

One issue with using an approximate transition model is that collisions with obstacles cannot always be modelled accurately. The problem is exacerbated by the fact that shortest paths in spatial environments tend to stay close to obstacles. In order to alleviate the negative effect of these two factors on navigation success, we introduce a safety term which penalises closeness to obstacles: $\mathcal{L}_{\text{safe}} = \sum_i s(l_i)$. Here, l_i is the reading of the i-th range sensor of the agent, as predicted by DVBF-LM based on the learned map $q(\boldsymbol{\mathcal{M}})$, and $s(\cdot)$ is a sigmoidal function mirrored along the y-axis.

We add $\mathcal{L}_{\text{safe}}$ to the travel distance when calculating edge weights. The penalty term assumes that we have access to depth readings, which is true in the case of laser scan-based settings but not when the agent only has access to visual observations.



(a) (b) (c)

Fig. 6: (a) Navigation plan constructed by using the belief space of the learned environment. (b) Executed trajectory in the actual simulator. (c) Ratio of successful navigation attempts over the number of simulation steps in A^*. In green the upper bound, A^* with access to the ground truth map and transition.

Generalising this path planning framework to visual observations will be part of future work.

B. Results

In our experiments, we verify that the model described in Section III can be applied to navigation.

a) Generative Model Learning: Before DVBF-LM can be used for planning, the emission model and the map $\mathcal{M}$ must first be learned. Here, we use the same models that were acquired as part of the SLAM experiments. For the transition we use an engineered model that allows the agent to move forward along its heading as long as the predicted LiDAR reading in that direction is greater than the desired step length. All navigation tasks are performed in the same set of mazes introduced earlier.

b) Pose-to-Pose Navigation with Hybrid-A^*: For each maze, we exhaustively pick pairs of starting and target pose from a 5×5 grid over the map. For each pair of poses, we apply the hybrid-A^* as described in Section V-A to plan a trajectory. The obtained controls are then executed in the true physics simulator. The navigation task is considered successful if the agent lands in a proximity of 0.05 or less from the target pose. As a baseline, we consider the navigation performance when the planning algorithm is executed directly in the physics simulator, with access to the true transition and emission. This allows us to assess the drop in performance resulting from approximating the true environment with DVBF-LM. Figure 6c shows the results of the evaluation. Planning based on our generative model comes very close in terms of navigation efficiency to planning in the simulator, affirming the usability of the learned environment maps for navigation tasks. Furthermore, all planned trajectories are successful in reaching the target.

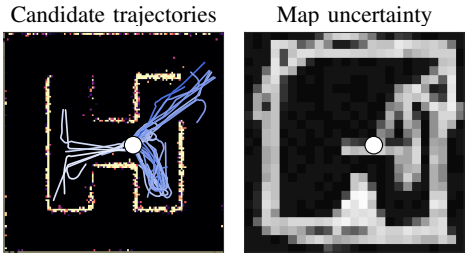
c) Pose-to-Observation Navigation with Hybrid-A^*: In this scenario, the observation targets are sensor readings from the environment simulator. The corresponding starting poses are the same as in the previous case. Before we can apply the algorithm from V-A the observation targets must first be mapped to pose targets. To that end, we train a separate variational auto-encoder (VAE, [14]) on the same data used for learning a map of the environment. We set the generative part of the VAE to the emission model of DVBF-LM, we condition on the learned map $q(\mathcal{M})$, and freeze the map and emission parameters. Thus, we obtain an approximation $q(\mathbf{z} \mid \mathbf{x})$ of the posterior over poses $p(\mathbf{z} \mid \mathbf{x})$ that conforms to the learned spatial map. This is done once for each of the six mazes. The obtained approximation can be reused for multiple navigation tasks in the given environment. The rest of the evaluation proceeds analogously to the *pose-to-pose* case, using the mode of the approximate posterior, $\mathbf{z}^* = \arg\max_{\mathbf{z}} q(\mathbf{z} \mid \mathbf{x})$, as a target. Figure 6c illustrates the results of the evaluation. Performance is very similar to the *pose-to-pose* case, with less than 2% of all trajectories failing to reach their actual target. The slight drop in performance can be attributed to *perceptual aliasing*—ambiguity in the pose given an observation—that is typical for spatial environments.

VI. EXPLORATION

Next, we tackle the problem of exploration, expressed in the efficient mapping of the environment. Fast inference of the map is a prerequisite for making informed decisions in spatial settings, as was demonstrated in the navigation task. Autonomous exploration amounts to the selection of control signals such that the data acquired makes inference progress fast. The control signals are then executed in the environment and the process repeats. We will now discuss how we use DVBF-LM to define an exploration policy.

A. Exploration via Active Learning

We choose to follow an information-theoretic approach—we define optimal exploration to be that which leads to the largest change in information in the map variable, a metric commonly referred to as infogain [15]. The change in information is quantified by the mutual information (MI) between future observations $\mathbf{x}_{1:T}$ predicted by DVBF-LM for a sequence of



(a) White means high MI score. (b) Black means high uncertainty.

Fig. 7: Generated candidates conform with the obstacles belief. Candidates leading into uncertain regions have high MI scores.

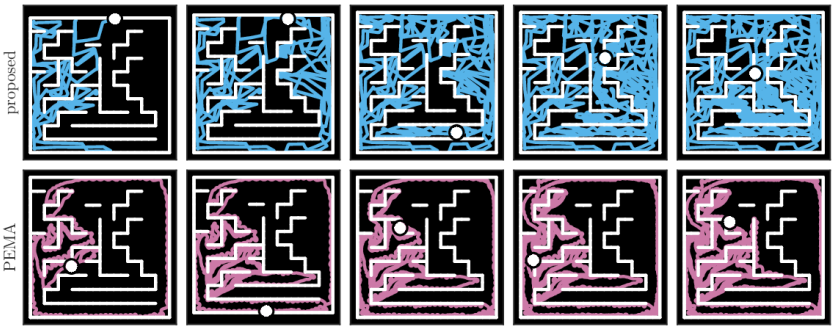


Fig. 8: Qualitative exploration comparison: proposed method vs. LSTM baseline. The plot shows the parallel exploration progress of both agents over time.

planned controls and the map $\boldsymbol{\mathcal{M}}$:

$$\begin{aligned} &\mathbb{E}_{\mathbf{x}_{1:T}\sim p(\cdot)}[\mathrm{KL}(p(\boldsymbol{\mathcal{M}} \mid \mathbf{x}_{1:T}) \mid\mid p(\boldsymbol{\mathcal{M}}))] \\ =\ &\mathrm{KL}(p(\mathbf{x}_{1:T}, \boldsymbol{\mathcal{M}}) \mid\mid p(\mathbf{x}_{1:T})p(\boldsymbol{\mathcal{M}})) \\ =\ &I(\mathbf{x}_{1:T}; \boldsymbol{\mathcal{M}}), \end{aligned} \tag{4}$$

omitting $\mathbf{u}_{1:T-1}$ for brevity. Intuitively, the goal is to select those control signals which maximise the information gained through them in expectation. Thus, we pose the following optimal-control problem:

$$\mathbf{u}^*_{1:T-1} = \arg\max_{\mathbf{u}_{1:T-1}} I(\mathbf{x}_{1:T}; \boldsymbol{\mathcal{M}} | \mathbf{u}_{1:T-1}). \tag{5}$$

In light of this goal, the generative nature of DBVF-LM and the explicit modelling of a global latent map appear essential to the formulation of a principled exploration solution. To solve the posed problem, two issues need to be addressed: the intractability of computing mutual information from Equation (4) and conducting the optimisation in Equation (5) (w.r.t. $\mathbf{u}_{1:T-1}$).

B. Approximating Mutual Information

Equation (4) depends on the highly nonlinear intractable joint $p(\mathbf{x}_{1:T}, \boldsymbol{\mathcal{M}} \mid \mathbf{u}_{1:T-1})$ and marginal $p(\mathbf{x}_{1:T} \mid \mathbf{u}_{1:T-1})$. For brevity, we will omit the conditioning on $\mathbf{u}_{1:T-1}$ until the end of this section. Approximation poses a challenge because of the double integration in the multi-dimensional spaces of $\mathbf{x}_{1:T}$ and $\boldsymbol{\mathcal{M}}$. We resort to combining MC sampling with applying a black-box entropy estimator, as done in [4]. This results in the following estimation:

$$\begin{aligned} I(\boldsymbol{\mathcal{M}}; \mathbf{x}_{1:T}) &= H[\mathbf{x}_{1:T}] - H[\mathbf{x}_{1:T} \mid \boldsymbol{\mathcal{M}}] \\ &\approx \hat{H}(\bar{\mathbf{X}}) - \frac{1}{M}\sum_{m=1}^{M} \hat{H}(\tilde{\mathbf{X}}^{(m)}). \end{aligned} \tag{6}$$

$\hat{H}$ represents a black-box entropy estimator that works on sample sets. $\bar{\mathbf{X}}$ is a set of samples from the marginal $p(\mathbf{x}_{1:T})$ and each $\tilde{\mathbf{X}}^{(m)}, m = 1, \ldots, M$ is a set of samples from a conditional $p\left(\mathbf{x}_{1:T} \;\middle|\; \boldsymbol{\mathcal{M}}^{(m)}\right)$ for $\boldsymbol{\mathcal{M}}^{(m)} \sim q(\boldsymbol{\mathcal{M}})$. All samples are obtained through ancestral sampling from DVBF-LM, which is only possible since DVBF-LM is a generative model.

In practice, exploration is performed in parallel with the inferences of $q(\mathbf{z}_{1:T})$ and $q(\boldsymbol{\mathcal{M}})$, gradually adding new data points and increasing the overall data set size. Note that we do not maximise MI once w.r.t. the prior $p(\boldsymbol{\mathcal{M}})$ for all future time steps, but we maximise it on-line multiple times for T steps ahead w.r.t. the current variational posterior $q(\boldsymbol{\mathcal{M}}) \approx p(\boldsymbol{\mathcal{M}} \mid \mathcal{D})$. This is well-grounded due to the validity of sequential Bayesian updates, i. e. $p(\boldsymbol{\mathcal{M}} \mid \mathcal{D}, \mathbf{x}_{1:T}) \propto p(\mathbf{x}_{1:T} \mid \boldsymbol{\mathcal{M}})p(\boldsymbol{\mathcal{M}} \mid \mathcal{D})$. In this work, a k-NN black-box entropy estimator is used for $\hat{H}$ [22].

C. Optimising Mutual Information

Optimisation of Equation (5) is performed in two stages. First, a set of proposal controls $\mathcal{U}$ is generated by using the current belief of the map $q(\boldsymbol{\mathcal{M}}) \approx p(\boldsymbol{\mathcal{M}} \mid \mathcal{D})$. Second, the best candidate control sequence $\mathbf{u}^*_{1:T-1} = \arg\max_{\mathbf{u}_{1:T-1}\in\mathcal{U}} I(\mathbf{x}_{1:T}; \boldsymbol{\mathcal{M}} \mid \mathbf{u}_{1:T-1}, \mathcal{D})$ is selected among the candidates and executed by the agent, following the scheme from the previous section.

Generating control sequences at random is very sample-inefficient, as the majority of sampled trajectories pass through obstacles in the environment. Instead we follow a heuristic approach, exploiting the laser range nature of our sensors. First we define an obstacle penalty $\mathcal{L}_{\text{obstacle}}(\mathbf{z})$ by building an occupancy map of the environment using $q(\boldsymbol{\mathcal{M}})$ and the DVBF-LM emission model. We then draw F random control sequences and form a set of candidates, minimising $\mathcal{L}_{\text{obstacle}}$ in expectation over the model:

$$\bar{\mathbf{u}}^{(f)}_{1:T-1} = \arg\min_{\mathbf{u}_{1:T-1}} \mathbb{E}_{\mathbf{z}_{1:T}\sim p(\cdot|\mathbf{u}_{1:T-1},\mathcal{D})}\left[\sum_{t=1}^{T} \mathcal{L}_{\text{obstacle}}(\mathbf{z}_t)\right], \tag{7}$$

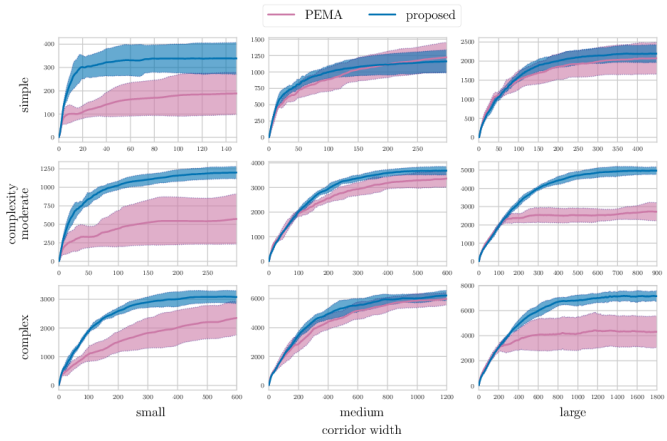
with $f = 1, \ldots, F$. We then approximate Equation (5) via

$$\mathbf{u}^*_{1:T-1} = \arg\max_{\mathbf{u}^{(f)}_{1:T-1}\in\mathcal{U}} I(\mathbf{x}_{1:T-1}; \boldsymbol{\mathcal{M}} \mid \mathbf{u}^{(f)}_{1:T-1}).$$

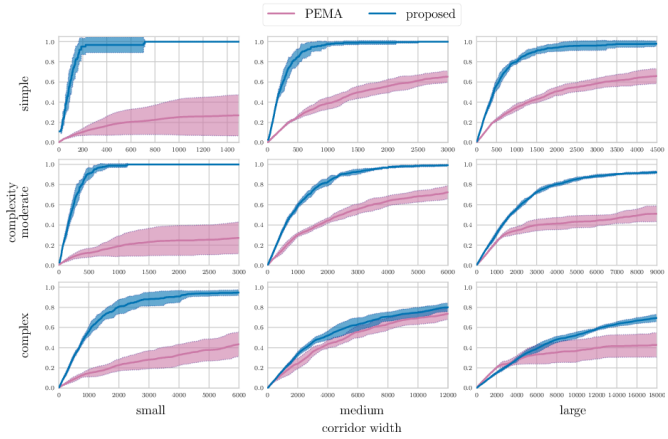
Figure 7a illustrates the described steps.

D. Results

The exploration experiments were carried out in our pybox2d simulator. All model components are learned during exploration, with the exception of the transition model, which is handled in



(a)



(b)

Fig. 9: Comparison of the proposed method (blue) against the LSTM baseline (violet): (a) infogain (b) exploration ratio. Higher values are better. The x axis shows the number of interaction steps divided by 10. Shaded area corresponds to one standard deviation over seven runs.

the same way as in the navigation experiments. We evaluate explorative performance based on two metrics—information gain over the course of exploration and the fraction of "tiles" visited by the agent, the exploration ratio. For the latter, we divide the mazes into a fixed number of equally sized tiles.

As a baseline we consider a method that directly optimises the second metric, the exploration ratio. The baseline agent is represented by a deep deterministic LSTM network. Given a sequence of observations as its input, it puts out a control signal at each time step. We refer to it as the pose-entropy maximising agent (PEMA). Since the exploration ratio objective is non-differentiable w.r.t. the LSTM parameters, we use Augmented Random Search [16] to optimise it.

Qualitatively, we see that the proposed method rapidly traverses the maze. DVBF-LM exploration exhibits nearly uniform coverage, being driven to places that are visited least and are thus most uncertain (see Figure 7). This holds even for the most complex mazes we considered, as we show in Figure 8.

The quantitative evaluation shows that our method consistently and significantly outperforms PEMA, even though PEMA is directly trained on the exploration ratio evaluation criterion. Figure 9 summarises the comparison for the different metrics over time, aggregated over multiple runs in mazes with different complexity and corridor width.

VII. Conclusion and Future Work

We have introduced a deep variational Bayes filter that integrates a global latent variable of spatial form. The novelty of our contribution lies in the flexibility that is inherited from neural networks and variational inference: contrary to most recent work in the area, our model still constitutes a generative model, which allows for a number of essential types of inference in spatial environments. We validated the proposed method by applying it to the problems of SLAM, autonomous exploration and navigation. Our model exhibits competitive localisation performance in comparison to an existing 2D LiDAR SLAM system, outperforms a strong baseline for exploration and can be used as a simulator for planning with virtually no loss in performance. The results bear promise for real world application, which we will address in upcoming studies, along with comparisons to state-of-the-art visual SLAM methods.

References

[1] Shi Bai, Jinkun Wang, Fanfei Chen, and Brendan Englot. Information-theoretic exploration with bayesian optimization. In *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*, pages 1816–1822. IEEE, 2016.

[2] Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural networks. *CoRR*, abs/1505.05424, 2015. URL http://arxiv.org/abs/1505.05424.

[3] Cesar Cadena, Luca Carlone, Henry Carrillo, Yasir Latif, Davide Scaramuzza, José Neira, Ian D. Reid, and John J. Leonard. Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. *IEEE Trans. Robotics*, 32(6):1309–1332, 2016. doi: 10.1109/TRO.2016.2624754. URL https://doi.org/10.1109/TRO.2016.2624754.

[4] Stefan Depeweg, Jose-Miguel Hernandez-Lobato, Finale Doshi-Velez, and Steffen Udluft. Decomposition of uncertainty in Bayesian deep learning for efficient and risk-sensitive learning. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1192–1201. PMLR, 10–15 Jul 2018. URL http://proceedings.mlr.press/v80/depeweg18a.html.

[5] Dmitri Dolgov, Sebastian Thrun, Michael Montemerlo, and James Diebel. Path planning for autonomous vehicles in unknown semi-structured environments. *The International Journal of Robotics Research*, 29(5):485–501, 2010. doi:

10.1177/0278364909359210. URL https://doi.org/10.1177/0278364909359210.

[6] Alberto Elfes. Robot navigation: Integrating perception, environmental constraints and task execution within a probabilistic framework. In *Reasoning with Uncertainty in Robotics*, pages 91–130. Springer, 1996.

[7] Marco Fraccaro, Danilo Jimenez Rezende, Yori Zwols, Alexander Pritzel, S. M. Ali Eslami, and Fabio Viola. Generative temporal models with spatial memory for partially observed environments. *CoRR*, abs/1804.09401, 2018. URL http://arxiv.org/abs/1804.09401.

[8] N. J. Gordon, D. J. Salmond, and A. F. M. Smith. Novel approach to nonlinear/non-gaussian bayesian state estimation. *IEE Proceedings F - Radar and Signal Processing*, 140(2):107–113, April 1993. ISSN 0956-375X. doi: 10.1049/ip-f-2.1993.0015.

[9] P. E. Hart, N. J. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107, July 1968. ISSN 0536-1567. doi: 10.1109/TSSC.1968.300136.

[10] Wolfgang Hess, Damon Kohler, Holger Rapp, and Daniel Andor. Real-time loop closure in 2d lidar slam. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1271–1278, 2016.

[11] Rein Houthooft, Xi Chen, Yan Duan, John Schulman, Filip De Turck, and Pieter Abbeel. Vime: Variational information maximizing exploration. In *Advances in Neural Information Processing Systems*, pages 1109–1117, 2016.

[12] Maximilian Karl, Maximilian Sölch, Justin Bayer, and Patrick van der Smagt. Deep variational bayes filters: Unsupervised learning of state space models from raw data. *CoRR*, abs/1605.06432, 2016. URL http://arxiv.org/abs/1605.06432.

[13] Michał Kempka, Marek Wydmuch, Grzegorz Runc, Jakub Toczek, and Wojciech Jaśkowski. ViZDoom: A Doom-based AI research platform for visual reinforcement learning. In *IEEE Conference on Computational Intelligence and Games*, pages 341–348, Santorini, Greece, Sep 2016. IEEE. URL http://arxiv.org/abs/1605.02097. The best paper award.

[14] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. In *Proceedings of the 2nd International Conference on Learning Representations (ICLR)*, 2014.

[15] David JC MacKay. Information-based objective functions for active data selection. *Neural computation*, 4(4):590–604, 1992.

[16] Horia Mania, Aurelia Guy, and Benjamin Recht. Simple random search provides a competitive approach to reinforcement learning. *CoRR*, abs/1803.07055, 2018. URL http://arxiv.org/abs/1803.07055.

[17] Kevin P. Murphy. Bayesian map learning in dynamic environments. In *Advances in Neural Information Processing Systems 12, [NIPS Conference, Denver, Colorado, USA, November 29 - December 4, 1999]*, pages 1015–1021, 1999. URL http://papers.nips.cc/paper/1716-bayesian-map-learning-in-dynamic-environments.

[18] Emilio Parisotto and Ruslan Salakhutdinov. Neural map: Structured memory for deep reinforcement learning. *CoRR*, abs/1702.08360, 2017. URL http://arxiv.org/abs/1702.08360.

[19] Nikolay Savinov, Alexey Dosovitskiy, and Vladlen Koltun. Semi-parametric topological memory for navigation. *CoRR*, abs/1803.00653, 2018. URL http://arxiv.org/abs/1803.00653.

[20] Jürgen Schmidhuber. Curious model-building control systems. In *Neural Networks. 1991 IEEE International Joint Conference on*, pages 1458–1463. IEEE, 1991.

[21] Cyrill Stachniss, Giorgio Grisetti, and Wolfram Burgard. Information gain-based exploration using rao-blackwellized particle filters. In *Robotics: Science and Systems I, June 8-11, 2005, Massachusetts Institute of Technology, Cambridge, Massachusetts, USA*, pages 65–72, 2005. URL http://www.roboticsproceedings.org/rss01/p09.html.

[22] Zoltán Szabó. Information theoretical estimators toolbox. *Journal of Machine Learning Research*, 15:283–287, 2014.

[23] S. Thrun. The role of exploration in learning control. In D.A. White and D.A. Sofge, editors, *Handbook for Intelligent Control: Neural, Fuzzy and Adaptive Approaches*. Van Nostrand Reinhold, Florence, Kentucky 41022, 1992.

[24] Chaoqun Wang, Lili Meng, Teng Li, Clarence W De Silva, and Max Q-H Meng. Towards autonomous exploration with information potential field in 3d environments. In *Advanced Robotics (ICAR), 2017 18th International Conference on*, pages 340–345. IEEE, 2017.

[25] Peter Whaite and Frank P. Ferrie. Autonomous exploration: driven by uncertainty. In *Conference on Computer Vision and Pattern Recognition, CVPR 1994, 21-23 June, 1994, Seattle, WA, USA*, pages 339–346, 1994. doi: 10.1109/CVPR.1994.323849. URL https://doi.org/10.1109/CVPR.1994.323849.

[26] Peter Whaite and Frank P Ferrie. Autonomous exploration: Driven by uncertainty. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(3):193–205, 1997.

[27] Kai Xu, Lintao Zheng, Zihao Yan, Guohang Yan, Eugene Zhang, Matthias Niessner, Oliver Deussen, Daniel Cohen-Or, and Hui Huang. Autonomous reconstruction of unknown indoor scenes guided by time-varying tensor fields. *ACM Transactions on Graphics (TOG)*, 36(6):202, 2017.

Part IV.

Appendix

Additional Background

C

C.1. Linear Gaussian Systems

Consider a Gaussian marginal

$$p(\mathbf{z}) = \mathcal{N}(\mathbf{z} \mid \boldsymbol{\mu}_{\mathbf{z}}, \boldsymbol{\Sigma}_{\mathbf{z}}), \tag{C.1}$$

and a linear Gaussian conditional

$$p(\mathbf{x} \mid \mathbf{z}) = \mathcal{N}(\mathbf{x} \mid \mathbf{A}\mathbf{z} + \mathbf{a}, \boldsymbol{\Sigma}_{\mathbf{x}}), \tag{C.2}$$

which form a factorisation of the joint distribution

$$p(\mathbf{x}, \mathbf{z}) = p(\mathbf{x} \mid \mathbf{z})p(\mathbf{z}). \tag{C.3}$$

This configuration is referred to as a *linear Gaussian system* (LGS), because of the linear dependence in the functional form $f(\boldsymbol{\theta}, \mathbf{z}) = \mathbf{A}\mathbf{z} + \mathbf{a}$ that defines the conditional mean. Given this directed model, one can obtain the terms of the inverse factorisation $p(\mathbf{x})p(\mathbf{z} \mid \mathbf{x})$ and the joint $p(\mathbf{x}, \mathbf{z})$ itself in closed-form. All three are Gaussian distributions. The formulas are

$$p(\mathbf{x}, \mathbf{z}) = \mathcal{N}(\mathbf{x}, \mathbf{z} \mid \boldsymbol{\mu}^*, \boldsymbol{\Sigma}^*) \tag{C.4}$$
$$p(\mathbf{z} \mid \mathbf{x}) = \mathcal{N}(\mathbf{z} \mid \boldsymbol{\mu}^*_{\mathbf{z}}, \boldsymbol{\Sigma}^*_{\mathbf{z}}) \tag{C.5}$$
$$p(\mathbf{x}) = \mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}^*_{\mathbf{x}}, \boldsymbol{\Sigma}^*_{\mathbf{x}}), \tag{C.6}$$

where

$$\boldsymbol{\mu}^*_{\mathbf{z}} = \mathbf{K}\mathbf{x} + (\mathbf{I} - \mathbf{K}\mathbf{A})\boldsymbol{\mu}_{\mathbf{z}} - \mathbf{K}\mathbf{a} \tag{C.7}$$
$$\boldsymbol{\Sigma}^*_{\mathbf{z}} = (\mathbf{I} - \mathbf{K}\mathbf{A})\boldsymbol{\Sigma}_{\mathbf{z}} \tag{C.8}$$
$$\mathbf{K} = \boldsymbol{\Sigma}_{\mathbf{z}}\mathbf{A}^{\mathsf{T}}\left(\mathbf{A}\boldsymbol{\Sigma}_{\mathbf{z}}\mathbf{A}^{\mathsf{T}} + \boldsymbol{\Sigma}_{\mathbf{x}}\right)^{-1}, \tag{C.9}$$

and

$$\boldsymbol{\mu}^*_{\mathbf{x}} = \mathbf{A}\boldsymbol{\mu}_{\mathbf{z}} + \mathbf{a} \tag{C.10}$$
$$\boldsymbol{\Sigma}^*_{\mathbf{x}} = \boldsymbol{\Sigma}_{\mathbf{x}} + \mathbf{A}\boldsymbol{\Sigma}_{\mathbf{z}}\mathbf{A}^{\mathsf{T}}, \tag{C.11}$$

and

$$\boldsymbol{\mu}^* = \begin{pmatrix} \boldsymbol{\mu}_\mathbf{z} \\ \mathbf{A}\boldsymbol{\mu}_\mathbf{z} + \mathbf{a} \end{pmatrix} \tag{C.12}$$

$$\boldsymbol{\Sigma}^* = \begin{bmatrix} \boldsymbol{\Sigma}_\mathbf{z} & \boldsymbol{\Sigma}_\mathbf{z}\mathbf{A}^\mathsf{T} \\ \mathbf{A}\boldsymbol{\Sigma}_\mathbf{z} & \boldsymbol{\Sigma}_\mathbf{x} + \mathbf{A}\boldsymbol{\Sigma}_\mathbf{z}\mathbf{A}^\mathsf{T} \end{bmatrix}. \tag{C.13}$$

The equations are derived by "completing the square" for each target Gaussian, after algebraically expanding the known product $p(\mathbf{z})p(\mathbf{x} \mid \mathbf{z})$ (see Bishop (2007) for a proof).

C.2. Connections to Gauss-Newton

If the model is small, i. e. its variables and computations easily fit in memory, then MAP optimisation lends itself to second-order optimisation methods. This is why flavors of Gauss-Newton (GN) are common in the field of sparse SLAM and sparse odometry estimation. We will now take a moment to relate Gauss-Newton to full-posterior estimation, seen as MAP plus a Laplace approximation. The purpose is to distinguish it from the inferences proposed in the core publications of the thesis.

In its basic form, a Gauss-Newton objective is a sum of squared errors (SSE)

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \sum_k r_k(\boldsymbol{\theta})^2, \tag{C.14}$$

where $r_k(\mathbf{z})$ are residuals between predictions and observations, and $\boldsymbol{\theta}$ are latent variables we seek to infer. For our discussion, we will upgrade this to the generalised Gauss-Newton formulation

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} f(g(\boldsymbol{\theta})), \tag{C.15}$$

where we replace the sum of square functions with $f(\cdot)$, a convex function, and we replace the residuals $r_k(\boldsymbol{\theta})$ with a non-linear function of $g(\boldsymbol{\theta})$ (e. g. see Diehl and Messerer (2019)). We will make f and g concrete at the end of this section.

Gauss-Newton can then be seen as an optimisation under a linear approximation of $g(\boldsymbol{\theta})$.[1] In other words, we choose to approximate $g(\boldsymbol{\theta})$ by its first-order Taylor expansion

$$\hat{g}(\boldsymbol{\theta}) = g(\boldsymbol{\theta}_0) + \mathbf{J}_g(\boldsymbol{\theta}_0)(\boldsymbol{\theta} - \boldsymbol{\theta}_0). \tag{C.16}$$

[1] We choose this perspective on Gauss-Newton to highlight the linear approximation.

Here $\mathbf{J}_g(\theta_0)$ is the Jacobian of $g(\cdot)$ evaluated at the expansion point θ_0, the current estimate in an optimisation context. Under this approximation, the Hessian of the objective becomes

$$\nabla^2 f(\hat{g}(\theta)) = \mathbf{J}_g(\theta_0)^T \cdot \underbrace{\nabla^2 f(\theta)}_{\text{Hessian of } f} \cdot \mathbf{J}_g(\theta_0) := \tilde{\mathbf{H}}(\theta). \tag{C.17}$$

The matrix $\hat{\mathbf{H}}$ is positive semi-definite. This follows from f being convex, its Hessian $\nabla^2 f(\theta)$ thus being positive semi-definite, and the product with the Jacobian from both sides preserving that. GN update steps are then

$$\theta_1 = \theta_0 + \alpha \tilde{\mathbf{H}}^{-1}(\theta_0) \nabla f(g(\theta_0)). \tag{C.18}$$

This is a generalised perspective on Gauss-Newton, where if we substitute for a square f (or sum of) and for residuals in place of g we recover the traditional variant. We defer to Nocedal and Wright (1999) for further information on GN and relaxations like Levenberg-Marquardt.

For our purposes, we are interested in how the above relates to posterior estimation. We can see MAP optimisation as a special case of eq. (C.15). Define $\{\mathbf{z}_1, \mathbf{z}_2, ...\mathbf{z}_M\}$ to be a set of latent variables, the equivalent of θ above, and let $\{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N\}$ be a set of observed variables. Assume we have a joint graphical model for $p(\mathbf{z}_{1:N}, \mathbf{x}_{1:M})$ with the factorisation

$$p(\mathbf{z}_{1:N}, \mathbf{x}_{1:M}) = \left(\prod_{n=1}^{N} p(\mathbf{z}_n \mid \mathcal{P}(\mathbf{z}_n)) \right) \left(\prod_{m=1}^{M} p(\mathbf{x}_m \mid \mathcal{P}(\mathbf{x}_m)) \right). \tag{C.19}$$

Here $\mathcal{P}(\cdot)$ denotes the parent nodes of a variable and the individual product terms reflect the factorisation of the joint. Then MAP optimisation boils down to

$$\mathbf{z}^*_{1:N} = \arg\max_{\mathbf{z}_{1:N}} \sum_{n=1}^{N} \log p(\mathbf{z}_n \mid \mathcal{P}(\mathbf{z}_n)) + \sum_{m=1}^{M} \log p(\mathbf{x}_m \mid \mathcal{P}(\mathbf{x}_m)). \tag{C.20}$$

If all terms $p(\mathbf{z}_n \mid \mathcal{P}(\mathbf{z}_n))$ and $p(\mathbf{x}_m \mid \mathcal{P}(\mathbf{x}_m))$ are homoscedastic Gaussians, we recover a weighted sum of square residuals that fits the generalised Gauss-Newton definition. Note that the conditionals can be non-linear or even non-Gaussian, e. g. they can be any parametric $p_\theta(\mathbf{z}_n \mid g(\mathcal{P}(\mathbf{z}_n)))$ in the sense of section 1.2.1. We only require that $\log p(\cdot)$ is convex and twice differentiable in the outputs of $g(\mathcal{P}(\mathbf{z}_n))$, which should have a valid Jacobian.

If we optimise eq. (C.20) with Gauss-Newton, we obtain a point-mass approximation of the posterior for the model in eq. (C.19). We can additionally apply a Laplace approximation using the Hessian estimate from eq. (C.17). We then obtain a posterior approximation

$$q(\mathbf{z}_{1:N}) \approx p(\mathbf{z}_{1:N} \mid \mathbf{x}_{1:M}) \tag{C.21}$$

$$q(\mathbf{z}_{1:N}) = \mathcal{N}\big(\mathbf{z}_{1:N} \;\big|\; \mathbf{z}^*_{1:N}, -\tilde{\mathbf{H}}(\mathbf{z}^*_{1:N})^{-1}\big). \tag{C.22}$$

Since the generalised Gauss-Newton Hessian estimate is positive semi-definite, it is straightforward to make it p.d. and use it in a Laplace approximation, regardless of whether $\mathbf{z}^*_{1:N}$ is a perfect posterior mode.[2]

The method is efficient for sparse data or sparse Jacobians (Dellaert and Kaess, 2017). This makes it suitable for sparse odometry and sparse SLAM inference under real-time constraints (Cadena et al., 2016). However, this work focuses on dense volumetric rendering and steers away from sparsity. The resulting posterior is a joint Gaussian over all latents and $g(\cdot)$ is linearised, which is restrictive. One goal of the thesis is to explore how to relax these assumptions.

C.3. Stochastic Gradients and the VI Reparameterisation Trick

To make the variational inference objective in eqs. (1.17) and (1.18) tractable, we estimate the expectation over $q_{\boldsymbol{\phi}}(\mathbf{z})$ with Monte Carlo. The expectation integral is replaced with an average over samples $\mathbf{z}_k \sim q_{\boldsymbol{\phi}}(\mathbf{z})$,

$$\mathcal{L}_{\text{elbo}}(\boldsymbol{\phi}, \boldsymbol{\theta}) \approx \frac{1}{K} \sum_{k=1}^{K} \log p_{\boldsymbol{\theta}}(\mathbf{x} \mid \mathbf{z}_k) - \mathrm{KL}\big(q_{\boldsymbol{\phi}}(\mathbf{z}) \;\big\|\; p_{\boldsymbol{\theta}}(\mathbf{z})\big). \tag{C.23}$$

This assumes that $\mathrm{KL}\big(q_{\boldsymbol{\phi}}(\mathbf{z}) \;\big\|\; p_{\boldsymbol{\theta}}(\mathbf{z})\big)$ is tractable (e. g. for Gaussians). Alternatively, one can approximate the KL via Monte Carlo as well,

$$\mathcal{L}_{\text{elbo}}(\boldsymbol{\phi}, \boldsymbol{\theta}) \approx \frac{1}{K} \sum_{k=1}^{K} \log p_{\boldsymbol{\theta}}(\mathbf{x} \mid \mathbf{z}_k) - \log q_{\boldsymbol{\phi}}(\mathbf{z}_k) + \log p_{\boldsymbol{\theta}}(\mathbf{z}_k). \tag{C.24}$$

Both variants make evaluation tractable, but for gradient-based optimisation of the posterior parameters $\boldsymbol{\phi}$ we additionally need tractable gradients through the MC sampling $\mathbf{z}_k \sim q_{\boldsymbol{\phi}}(\mathbf{z})$.

[2] Usually small positive values are already added to the diagonal of $\tilde{\mathbf{H}}(\mathbf{z}^*_{1:N})$ in a Levenberg-Marquardt optimisation context.

The *reparameterisation trick* makes this possible for a wide family of distributions (Kingma and Welling, 2014). The variational family $q_{\phi}(z)$ is chosen such that its samples z_k can be computed as

$$z_k = f(\phi, \epsilon), \quad \epsilon \sim p(\epsilon). \tag{C.25}$$

Here, ϵ is a random seed drawn from an out-of-band distribution, e. g. a standard Gaussian. Then, the sample z_k is the output of a deterministic function $f(\cdot)$, of the variational parameters ϕ and the random seed ϵ.[3] This construction allows gradients to flow through f into ϕ. For example, Gaussian, Laplace, Logistic, Gumbel and normalising flow distributions are variational families that can be easily framed in this way. When q is multivariate, the above applies to its individual factors, i. e. different families can be composed together for different latents.

Effectively, the reparameterisation shifts the expectation from $q_{\phi}(z)$ to $p(\epsilon)$, i. e.

$$\mathbb{E}_{q_{\phi}(z)}(g(z)) = \mathbb{E}_{p(\epsilon)}[g(f(\phi, \epsilon))] \approx \frac{1}{K} \sum_{k=1}^{K} g(f(\phi, \epsilon_k)), \tag{C.26}$$

for any differentiable function g, so that the random sampling of ϵ is decoupled from gradient flow w. r. t. ϕ and gradients can be MC-estimated w. r. t. $p(\epsilon)$.

C.4. Approximate Empirical Distributions

Sometimes we may want to trade the convenience of parametric distributions for expressiveness. Imagine we want to approximate a complex target distribution $p(x)$, but only have a function $g(x)$ proportional to it, i. e.

$$p(x) = \frac{g(x)}{\mathfrak{Z}}, \tag{C.27}$$

with $\mathfrak{Z} > 0$. In this case an empirical weighted set of particles can be a proxy for the unknown $p(x)$, enabling Monte-Carlo estimation. See fig. 1.1d for an example (particles are coloured by their weights).

[3] In practice, f is usually defined by hand in terms of ϵ and the distribution parameters ψ of q, e. g. a mean and a covariance for a Gaussian q, which in turn are a function of ϕ.

Importance sampling is one way to achieve this. We seek to evaluate expectations

$$\mathbb{E}_{p(\mathbf{x})}(f(\mathbf{x})) = \int p(\mathbf{x}) f(\mathbf{x}) \, d\mathbf{x}. \tag{C.28}$$

Because we cannot sample from $p(\mathbf{x})$, we use a proposal distribution $q(\mathbf{x})$ from which we can sample and which has a known PDF. For example, this could be a parametric distribution $q_\theta(\mathbf{x})$ as per the previous section. We also require that $\forall \mathbf{x}, p(\mathbf{x}) > 0 : q(\mathbf{x}) > 0$, i. e. the proposal should cover the support of the unknown $p(\mathbf{x})$. Then we can rewrite eq. (C.28) as

$$\begin{aligned} \mathbb{E}_{p(\mathbf{x})}[f(\mathbf{x})] &= \mathbb{E}_{q(\mathbf{x})}\left[\frac{p(\mathbf{x})}{q(\mathbf{x})} f(\mathbf{x})\right] = \mathbb{E}_{q(\mathbf{x})}\left[\frac{g(\mathbf{x})}{q(\mathbf{x})} f(\mathbf{x})\right] / \mathcal{Z} \\ &= \frac{\mathbb{E}_{q(\mathbf{x})}\left[\frac{g(\mathbf{x})}{q(\mathbf{x})} f(\mathbf{x})\right]}{\mathbb{E}_{q(\mathbf{x})}\left[\frac{g(\mathbf{x})}{q(\mathbf{x})}\right]}, \end{aligned} \tag{C.29}$$

where eq. (C.29) follows from

$$\frac{1}{\mathcal{Z}} = \int g(\mathbf{x}) \, d\mathbf{x} = \mathbb{E}_{q(\mathbf{x})}\left[\frac{g(\mathbf{x})}{q(\mathbf{x})}\right]. \tag{C.30}$$

Since we known $q(\mathbf{x})$ and $g(\mathbf{x})$, we can apply MC estimation to obtain:

$$\begin{aligned} \mathbb{E}_{p(\mathbf{x})}[f(\mathbf{x})] &= \frac{\mathbb{E}_{q(\mathbf{x})}\left[\frac{g(\mathbf{x})}{q(\mathbf{x})} f(\mathbf{x})\right]}{\mathbb{E}_{q(\mathbf{x})}\left[\frac{g(\mathbf{x})}{q(\mathbf{x})}\right]} \approx \frac{\sum_{k=1}^{K} \frac{g(\mathbf{x}^k)}{q(\mathbf{x}^k)} f(\mathbf{x}^k)}{\sum_{l=1}^{K} \frac{g(\mathbf{x}^l)}{q(\mathbf{x}^l)}} \\ &= \frac{\sum_{k=1}^{K} \tilde{w}^k f(\mathbf{x}^k)}{\sum_{l=1}^{K} \tilde{w}^l} = \sum_{k=1}^{K} w^k f(\mathbf{x}^k) \end{aligned} \tag{C.31}$$

Here we define $\tilde{w}^k = \frac{g(\mathbf{x})}{q(\mathbf{x})}$ and $w^k = \frac{\tilde{w}^k}{\sum_{l=1}^{K} \tilde{w}^l}$. The terms w^k are known as the normalised *importance weights* of each sampled *particle* $\mathbf{x}^k \sim q(\mathbf{x})$. The set of particles and their weights forms the weighted *empirical distribution*

$$\left\{ \left(w^k, \mathbf{x}^k \right) \right\}_{k=1,\dots K}. \tag{C.32}$$

Through eq. (C.31) it allows us to approximate any expectation w. r. t. the target $p(\mathbf{x})$.

Most commonly, this is applicable when we know $p(\mathbf{z})p(\mathbf{x} \mid \mathbf{z})$ and seek $p(\mathbf{z} \mid \mathbf{x})$. Through Bayes' law we see the above applies with

$$\begin{aligned} p(\mathbf{z} \mid \mathbf{x}) &= \frac{g(\mathbf{z})}{\mathcal{Z}} \\ \text{where} \quad g(\mathbf{z}) &= p(\mathbf{z}, \mathbf{x}) = p(\mathbf{z})p(\mathbf{x} \mid \mathbf{z}) \\ \mathcal{Z} &= p(\mathbf{x}). \end{aligned} \tag{C.33}$$

For the particles $\mathbf{z}^k \sim q(\mathbf{z}^k)$ the importance weights are then

$$\tilde{w}^k = \frac{p(\mathbf{z}^k, \mathbf{x})}{q(\mathbf{z}^k)}, \quad w^k = \frac{\tilde{w}^k}{\sum_{l=1}^K \tilde{w}_l}, \tag{C.34}$$

and the respective empirical is $\{w^k, \mathbf{z}^k\}_{k=1,\dots K}$. Note that $\mathbf{x}$ above is from the conditional $p(\mathbf{z} \mid \mathbf{x})$.

Empirical approximations are approximate due to the Monte-Carlo estimation in eq. (C.31). The number of particles needed to reduce the approximation gap grows exponentially with variable size. This makes empirical approximations too expensive for large problems, like the dense spatial modelling used throughout the thesis. On the other hand, empiricals are unbiased in the asymptotic limit, which cannot be easily guaranteed for parametric distributions.

C.5. KALMAN FILTERS

The most basic filter relevant to this work is the *Kalman filter* (Kalman, 1960). The prediction step of Kalman filters inspires a small part of the more complex spatial filter proposed in appendix A.2.

Kalman filters assume a state-space model with a linear Gaussian emission and a linear Gaussian transition,

$$p(\mathbf{z}_t \mid \mathbf{z}_{t-1}) = \mathcal{N}(\mathbf{z}_t \mid \mathbf{A}\mathbf{z}_{t-1} + \mathbf{a}, \mathbf{S}) \tag{C.35}$$

$$p(\mathbf{x}_t \mid \mathbf{z}_t) = \mathcal{N}(\mathbf{x}_t \mid \mathbf{B}\mathbf{z}_t + \mathbf{b}, \mathbf{L}). \tag{C.36}$$

Note that the constants $\mathbf{a}, \mathbf{b}$ are added for generality, extending the purely linear mappings. Also note that control inputs to the transition are omitted for brevity, but are there in practice. Let us assume the previous filter is Gaussian, such that

$$p(\mathbf{z}_{t-1} \mid \mathbf{x}_{1:t-1}) = \mathcal{N}(\mathbf{z}_{t-1} \mid \boldsymbol{\mu}_{t-1}, \boldsymbol{\Sigma}_{t-1}).$$

Under Markovian independence, $p(\mathbf{z}_{t-1} \mid \mathbf{x}_{1:t-1})$ and $p(\mathbf{z}_t \mid \mathbf{z}_{t-1})$ form a linear Gaussian system, in the sense of section 1.2.2. Equations (C.10) and (C.11) then give the one-step *prediction* distribution

$$p(\mathbf{z}_t \mid \mathbf{x}_{1:t-1}) = \int p(\mathbf{z}_t \mid \mathbf{z}_{t-1}) p(\mathbf{z}_{t-1} \mid \mathbf{x}_{1:t-1}) \, \mathrm{d}\mathbf{z}_{t-1} \tag{C.37}$$

$$= \mathcal{N}\left(\mathbf{z}_t \;\middle|\; \hat{\boldsymbol{\mu}}_t, \hat{\boldsymbol{\Sigma}}_t\right) \tag{C.38}$$

$$\hat{\boldsymbol{\mu}}_t = \mathbf{A}\boldsymbol{\mu}_{t-1} + \mathbf{a} \tag{C.39}$$

$$\hat{\boldsymbol{\Sigma}}_t = \mathbf{S} + \mathbf{A}\boldsymbol{\Sigma}_{t-1}\mathbf{A}^\mathsf{T}. \tag{C.40}$$

In the above, uncertainty from the previous filter at step $t-1$ is propagated through the transition into the current step t, marginalising $\mathbf{z}_{t-1}$ in the process.

In turn, $p(\mathbf{z}_t \mid \mathbf{x}_{1:t-1})$ and $p(\mathbf{x}_t \mid \mathbf{z}_t)$ form another LGS. Equations (C.7) and (C.8) then give the target filter

$$p(\mathbf{z}_t \mid \mathbf{x}_{1:t}) = \mathcal{N}(\mathbf{z}_t \mid \boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t) \tag{C.41}$$

$$\boldsymbol{\mu}_t = \mathbf{K}_t\mathbf{x}_t + (\mathbf{I} - \mathbf{K}_t\mathbf{B})\hat{\boldsymbol{\mu}}_t - \mathbf{K}_t\mathbf{b} \tag{C.42}$$

$$\boldsymbol{\Sigma}_t = (\mathbf{I} - \mathbf{K}_t\mathbf{B})\hat{\boldsymbol{\Sigma}}_t \tag{C.43}$$

$$\mathbf{K}_t = \hat{\boldsymbol{\Sigma}}_t\mathbf{B}^\mathsf{T}\left(\mathbf{B}\hat{\boldsymbol{\Sigma}}_t\mathbf{B}^\mathsf{T} + \mathbf{L}\right)^{-1}. \tag{C.44}$$

The result is again a Gaussian, and the same procedure can repeat for future steps by induction. The matrix $\mathbf{K}_t$ is known as a *Kalman gain* (Särkkä, 2013). It is often interpreted as an interpolation term, weighing the newest observation $\mathbf{x}_t$ against the prior over $\mathbf{z}_t$ (see algebraic structure of eq. (C.42)).

In practice, variants of the Kalman filter are applied to non-linear state-space models. *Extended Kalman filters* (EKFs) linearise both the emission and transition via a first-order Taylor expansion, while *unscented Kalman filters* (UKFs) rely on an unscented transformation of sigma points for uncertainty propagation (Särkkä, 2013).

The Gaussian and linearity assumptions of Kalman filters are restrictive. For that reason, EKF assumptions are used very sparingly in a small part of the derivations in appendix A.2, for speed. Additionally, Kalman filters do not directly scale to the dense spatial setting, with millions of latent parameters, without strong independence assumptions (diagonalisation).

C.6. Filtering via Optimisation

Filtering can be approached with variational inference as well. This is similar to *assumed density filters* (ADF) (e. g. Opper and Winther (1999) and Minka (2001)), where first a parametric approximate distribution family is assumed and then its divergence to the filtering posterior is minimised. The only difference is that VI optimises a $\mathrm{KL}(q \parallel p)$ while ADFs optimise a $\mathrm{KL}(p \parallel q)$ divergence (i. e. expectation propagation). Since VI was already covered in the background, we will now consider how it applies to the recursion of the Bayes filter in SSMs.

The main idea is to interpret the recursion from eq. (1.26) as the graphical model $p(\mathbf{x}_t \mid \mathbf{z}_t)p(\mathbf{z}_t \mid \mathbf{x}_{1:t-1})$ for the joint $p(\mathbf{x}_t, \mathbf{z}_t \mid \mathbf{x}_{1:t-1})$, which is correct under Markovian assumptions. Assuming both terms $p(\mathbf{x}_t \mid \mathbf{z}_t)$ and $p(\mathbf{z}_t \mid \mathbf{x}_{1:t-1})$ are known, one can use the ELBO to find the posterior $p(\mathbf{z}_t \mid \mathbf{x}_{1:t}) \propto p(\mathbf{x}_t \mid \mathbf{z}_t)p(\mathbf{z}_t \mid \mathbf{x}_{1:t-1})$. Following section 1.3.1, for an assumed approximation $q_{\boldsymbol{\phi}}(\mathbf{z}_t)$ we pose the optimisation objective

$$\begin{aligned}&\arg\max_{\boldsymbol{\phi}} \mathcal{L}_{\text{elbo}}(\boldsymbol{\phi}) \\ &\quad = \arg\max_{\boldsymbol{\phi}} \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}_t)}[\log p(\mathbf{x}_t \mid \mathbf{z}_t)] - \mathrm{KL}\big(q_{\boldsymbol{\phi}}(\mathbf{z}_t) \,\big|\big|\, p(\mathbf{z}_t \mid \mathbf{x}_{1:t-1})\big).\end{aligned}$$

Formulating filters in this way has the advantage that $q_{\boldsymbol{\phi}}(\mathbf{z}_t)$ is a flexible parametric distribution, up to the reparameterisation trick (appendix C.3). The emission and transition are also not limited, as long as their PDFs are known and they are differentiable. The main caveat is that the prior $p(\mathbf{z}_t \mid \mathbf{x}_{1:t-1}) = \mathbb{E}_{p(\mathbf{z}_{t-1} \mid \mathbf{x}_{1:t-1})}[p(\mathbf{z}_t \mid \mathbf{z}_{t-1})]$ is given by an expectation integral that needs to be approximated.

For dense spatial models, computing both a prior approximation and a following variational inference approximation in real-time is challenging, at least at the time of writing. Still, similar ideas underpin a pose tracking objective of the proposed spatial filters in appendix A.2 and appendix A.3, with the compromise that the optimisation is max-a-posteriori under real-time constraints. In this case q is a point mass, and can be extended with Gaussian uncertainty via Laplace approximation.

C.7. Particle Filters

Where Kalman filters are the archetype of closed-form but restrictive assumptions, *particle filters* (PFs) are their flexible but more unwieldy counterpart.

Formally, a PF only requires that the transition and emission have a tractable PDF. It is further assumed the previous filter $p(\mathbf{z}_{t-1} \mid \mathbf{x}_{1:t-1})$ is approximated by importance-weighted particles, as per appendix C.4:

$$\left\{\left(w_{t-1}^k, \mathbf{z}_{t-1}^k\right)\right\}_{k=1,\dots K}. \tag{C.45}$$

A particle filter then approximates $p(\mathbf{z}_t \mid \mathbf{x}_{1:t})$ with a new particle set, moving the recursion forward. First, using the particle set for $t-1$ one can MC-estimate the expectation in eq. (1.28) like so

$$p(\mathbf{z}_t \mid \mathbf{x}_{1:t}) \propto p(\mathbf{x}_t \mid \mathbf{z}_t)\mathbb{E}_{p(\mathbf{z}_{t-1}|\mathbf{x}_{1:t-1})}[p(\mathbf{z}_T \mid \mathbf{z}_{T-1})] \tag{C.46}$$

$$\approx p(\mathbf{x}_t \mid \mathbf{z}_t) \sum_{k=1}^{K} w_{t-1}^k p\left(\mathbf{z}_t \mid \mathbf{z}_{t-1}^k\right) \tag{C.47}$$

$$= \sum_{k=1}^{K} w_{t-1}^k p(\mathbf{x}_t \mid \mathbf{z}_t) p\left(\mathbf{z}_t \mid \mathbf{z}_{t-1}^k\right) := g(\mathbf{z}_t). \tag{C.48}$$

It approximately holds that $p(\mathbf{z}_t \mid \mathbf{x}_{1:t}) \propto g(\mathbf{z}_t)$. This matches the setup for importance sampling (see appendix C.4), and here it is used to form a new particle set. An assumed proposal distribution $q(\mathbf{z}_t)$[4] generates K new candidate particles $\mathbf{z}_t^k \sim q(\mathbf{z}_t)$ which are then assigned the weights

$$\tilde{w}_t^k = \frac{g(\mathbf{z}_t^k)}{q\left(\mathbf{z}_t^k\right)}, \quad w_t^k = \frac{\tilde{w}_t^k}{\sum_{l=1}^{K} \tilde{w}_t^l}. \tag{C.49}$$

The filter $p(\mathbf{z}_t \mid \mathbf{x}_{1:t})$ is then approximated by the particle set

$$\left\{\left(w_t^k, \mathbf{z}_t^k\right)\right\}_{k=1,\dots K}. \tag{C.50}$$

The above is an instance of sequential importance sampling (SIS), a sequential Monte Carlo (SMC) method (Doucet, Johansen, et al., 2009).[5] For the sake of brevity, the reader is referred to the excellent tutorial by Doucet, Johansen, et al. (2009) for more details on sequential importance resampling (SIR), the benefits of resampling in PFs and Rao-Blackwellised PFs.

PF posteriors appear in the non-core publication in appendix B.1, which preceded the core contributions of this thesis. They are discussed here

[4] Can be sampled from and must have a tractable PDF.

[5] For brevity, the discussion is narrowed to the assumptions of state-space models, instead of more general SMC and SIS formulations.

only for the purpose of comparison. Particle filters are asymptotically ideal, but the number of particles needed for a reasonable estimate suffers from the curse of dimensionality (Doucet, Johansen, et al., 2009). This is a general limitation of sampling methods, and the main reason they were not explored further for the proposed dense spatial models. In particular, in this thesis $p(\mathbf{x}_t \mid \mathbf{z}_t)$ is analogous with a renderer and evaluating it for a very large number of particles is expensive (cf. eq. (C.48)). Direct application of particle filtering is hence challenging for real-time inference and remains to be explored in future work.

C.8. Rotation Matrices

The elements of the algebraic group $\mathrm{SO}(3)$ map one-to-one to orthogonal rotation matrices $\mathbf{R} \in \mathbb{R}^{3\times3}$ with a positive determinant. For brevity, we will treat rotation matrices $\mathbf{R}$ as the direct elements of $\mathrm{SO}(3)$:

$$\mathbf{R} \in \mathrm{SO}(3): \quad \mathbf{R}^\mathsf{T}\mathbf{R} = \mathbf{I},\ \det(\mathbf{R}) = 1. \tag{C.51}$$

The abstract $\mathrm{SO}(3)$ operator $\circ$ is then equivalent to a matrix product:

$$\forall \mathbf{R}_1, \mathbf{R}_2 \in \mathrm{SO}(3): \quad \mathbf{R}_1\mathbf{R}_2 \in \mathrm{SO}(3). \tag{C.52}$$

When we multiply rotation matrices we concatenate consecutive rotations. The null-element of the rotation group maps to the identity matrix

$$\forall \mathbf{R} \in \mathrm{SO}(3): \quad \mathbf{R}^{-1}\mathbf{R} = \mathbf{R}^\mathsf{T}\mathbf{R} = \mathbf{I} \in \mathrm{SO}(3). \tag{C.53}$$

Respectively, $\mathbf{Rp}$ rotates a point $\mathbf{p} \in \mathbb{R}^3$ (i. e. the group action of $\mathrm{SO}(3)$).

Rotation matrices are real-valued, but they are not closed under addition.

C.9. Lie-Algebra Parameters

The rotation group $\mathrm{SO}(3)$ is a *Lie group*, as it is both a group and a differentiable manifold. The group manifold is characterised by a tangent space at the identity, known as the *Lie algebra* $\mathfrak{so}(3)$. This tangent space makes for an elegant parameterisation, because it is isomorphic to $\mathbb{R}^3$ and it is straightforward to map from it to the Lie group and back. The parameterisation is minimal, with three degrees of freedom.

To remain on topic, this section covers only practical aspects about the parameterisation. The reader is referred to the tutorial by Deray and Solà (2020) and Hall and Hall (2013) for more theory.

Consider a curve of rotations $\mathbf{R}(t)$ that lies on the rotation manifold $SO(3)$. An expression for its derivative $\dot{\mathbf{R}}(t)$ can be obtained by differentiating $\mathbf{R}(t)\mathbf{R}(t)^T$:

$$\mathbf{R}(t)\mathbf{R}(t)^T = \mathbf{I} \tag{C.54}$$

$$\dot{\mathbf{R}}(t)\mathbf{R}(t)^T + (\dot{\mathbf{R}}(t)\mathbf{R}(t)^T)^T = 0 \tag{C.55}$$

$$\dot{\mathbf{R}}(t)\mathbf{R}(t)^T = \underbrace{-(\dot{\mathbf{R}}(t)\mathbf{R}(t)^T)^T}_{:=[\theta]_\times(t)}. \tag{C.56}$$

Equation (C.56) tells us that $[\theta]_\times(t) = -(\dot{\mathbf{R}}(t)\mathbf{R}(t)^T)^T$ is a skew-symmetric matrix of the form

$$[\theta]_\times(t) = \begin{bmatrix} 0 & -\theta_z & \theta_y \\ \theta_z & 0 & -\theta_x \\ -\theta_y & \theta_x & 0 \end{bmatrix}. \tag{C.57}$$

With eqs. (C.56) and (C.57) the rotation derivative is

$$\dot{\mathbf{R}}(t) = [\theta]_\times(t)\mathbf{R}(t). \tag{C.58}$$

A first-order Taylor expansion around $\mathbf{R}(0) = \mathbf{I}$[6] then gives us:

$$\dot{\mathbf{R}}(0) = [\theta]_\times(0) \tag{C.59}$$

$$\mathbf{R}(t) \approx \mathbf{I} + t[\theta]_\times(0). \tag{C.60}$$

The Taylor expansion gives the tangent space, or Lie-algebra, which is the space of all skew-symmetric matrices

$$\mathfrak{so}(3) = \left\{ [\theta]_\times \in \mathbb{R}^{3\times3} \mid [\theta]_\times = -[\theta]_\times^T \right\}. \tag{C.61}$$

The notation $[\theta]_\times \in \mathbb{R}^{3\times3}$ is intentional here, because skew-symmetric matrices are isomorphic to the space of vectors $\theta \in \mathbb{R}^3$ in terms of cross products, in the sense that

$$\forall \mathbf{p} \in \mathbb{R}^3 : \quad [\theta]_\times \mathbf{p} = \theta \times \mathbf{p}. \tag{C.62}$$

The elements of $\theta = [\theta_x, \theta_y, \theta_z] \in \mathbb{R}^3$ are the three degrees of freedom that appear in the skew-symmetric matrix. In practice, a rotation is parameterised with the vector θ directly.

[6] The convention is to express Lie-algebras at the identity, but it should be noted that tangent spaces have similar structure at any element on the Lie group manifold.

With rotation parameters $\theta \in \mathbb{R}^3$ the mapping to rotations is defined as

$$f_{rot} : \mathbb{R}^3 \to SO(3) \tag{C.63}$$

$$f_{rot}(\theta) = \exp\Big(\big[\theta\big]_\times\Big) \tag{C.64}$$

$$= \mathbf{I} + \left[\frac{\theta}{\|\theta\|}\right]_\times \sin\|\theta\| + \left[\frac{\theta}{\|\theta\|}\right]_\times^2 (1 - \cos\|\theta\|). \tag{C.65}$$

This is known as the *exponential map* of $SO(3)$ and also as the *Euler-Rodrigues formula* (Dai, 2015). The exponential arises from solving for $\mathbf{R}(t)$ in eq. (C.58). The expression is differentiable, and thus suitable for gradient-based optimisation. Rotations can also be mapped back to vectors θ via a logarithmic map, for brevity we refer to Deray and Solà (2020) for more details.

The vectors θ associated with the Lie-algebra are also referred to as axis-angle or rotation vectors. Their magnitude $\|\theta\|$ gives the rotation angle around the normalised axis $\frac{\theta}{\|\theta\|}$. Respectively, this means that

$$f_{rot}(\theta) = f_{rot}(\frac{\|\theta\| + 2k\pi}{\|\theta\|}\theta), \quad k \in \mathbb{Z}, \tag{C.66}$$

The mapping is many-to-one, and parameters are equivalent in a sinusoidal pattern with a period of 2π. The discussion of further properties of Lie-algebras, such as the Lie-bracket, are purposefully left out as they are not necessary to understand the proposed methods.

Publication Permission Information

D

All of the publications included in the thesis appear only in *open-access* conference proceedings, they were not published in traditional journals. The authors did not give up their copyright in the process (i. e. it was not transferred). Following are the details for each published paper.

D.1. Included Core Publications

Each of the following subsections has the publication permissions of the included core papers in the dissertation.

D.1.1. Variational State-Space Models for Localisation and Dense 3D Mapping in 6 DoF (Mirchev et al., 2021)

The publication appeared at the 9th International Conference on Learning Representations (ICLR) 2021. It is available on OpenReview (an *open-access* conference proceedings website) under the following link: https://openreview.net/forum?id=XAS3uKeFWj. The paper copyright remains with the authors and was not transferred to ICLR, as explained on the ICLR website: https://iclr.cc/FAQ/Copyright. Here is the verbatim quote:

> **Copyright and Patents on ICLR Papers**
>
> According to U.S. Copyright Office's page What is a Copyright. When you create an original work you are the author and the owner and hold the copyright, unless you have an agreement to transfer the copyright to a third party such as the company or school you work for. Authors do not transfer the copyright of their paper to ICLR, instead they grant ICLR a non-exclusive, perpetual, royalty-free, fully-paid, fully-assignable license to copy, distribute and publicly display all or part of the paper.

Patents

You cannot file a patent once a paper is publicly disclosed. The US has a 1-year grace period.

D.1.2. PRISM: Probabilistic Real-Time Inference in Spatial World Models (Mirchev et al., 2022)

The publication appeared at the Conference on Robot Learning (CoRL) 2022. It was published in the online, *open-access* Proceedings of Machine Learning Research (PMLR) under a Creative Commons Attribution 4.0 International License (CC BY 4.0), which is permissive and allows redistribution of the published version. According to the CC BY 4.0 agreement, the authors "retain ownership of all rights under copyright in all versions of the article, and all rights not expressly granted in this agreement". The publication is available online under the following link: https://proceedings.mlr.press/v205/mirchev23a.html.

D.1.3. Tracking and Planning with Spatial World Models (Kayalibay et al., 2022)

The publication appeared at the Learning for Dynamics and Control Conference (L4DC) 2022. It was published in the online, *open-access* Proceedings of Machine Learning Research (PMLR) under a Creative Commons Attribution 4.0 International License (CC BY 4.0), which is permissive and allows redistribution of the published version. According to the CC BY 4.0 agreement, the authors "retain ownership of all rights under copyright in all versions of the article, and all rights not expressly granted in this agreement". The publication is available online under the following link: https://proceedings.mlr.press/v168/kayalibay22a.html.

D.2. Included Non-Core Prior Work

The non-core publication (Mirchev et al., 2019) was included in a thesis appendix for context, but it does not count towards the dissertation (it was published immediately before the doctorate). It appeared at the Robotics Science and Systems (RSS) 2019 conference and is available online in the online Robotics Science and Systems Proceedings under the following link: https://www.roboticsproceedings.org/rss15/p83.

html. The RSS proceedings are open-access, copyright was not transferred to RSS when the work was published.

Own publications

Core Publications

Mirchev, Atanas, Baris Kayalibay, Patrick van der Smagt, and Justin Bayer (2021). "Variational State-Space Models for Localisation and Dense 3D Mapping in 6 DoF." In: *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net. URL: https://openreview.net/forum?id=XAS3uKeFWj.

Mirchev, Atanas, Baris Kayalibay, Ahmed Agha, Patrick van der Smagt, Daniel Cremers, and Justin Bayer (2022). "PRISM: Probabilistic Real-Time Inference in Spatial World Models." In: *Conference on Robot Learning, CoRL 2022, 14-18 December 2022, Auckland, New Zealand*. Vol. 205. Proceedings of Machine Learning Research. PMLR, pp. 161–174. URL: https://proceedings.mlr.press/v205/mirchev23a.html.

Kayalibay, Baris, Atanas Mirchev, Patrick van der Smagt, and Justin Bayer (2022). "Tracking and Planning with Spatial World Models." In: *Learning for Dynamics and Control Conference, L4DC 2022, 23-24 June 2022, Stanford University, Stanford, CA, USA*. Vol. 168. Proceedings of Machine Learning Research. PMLR, pp. 124–137.

Non-Core Publications

Bayer, Justin, Maximilian Soelch, Atanas Mirchev, Baris Kayalibay, and Patrick van der Smagt (2021). "Mind the Gap when Conditioning Amortised Inference in Sequential Latent-Variable Models." In: *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net. URL: https://openreview.net/forum?id=a2gqxKDvYys.

Kayalibay, Baris, Atanas Mirchev, Patrick van der Smagt, and Justin Bayer (2021). "Less Suboptimal Learning and Control in Variational POMDPs." In: *Self-Supervision for Reinforcement Learning Workshop - ICLR 2021*. URL: https://openreview.net/forum?id=oe4q7ZiXwkL.

Kayalibay, Baris, Atanas Mirchev, Ahmed Agha, Patrick van der Smagt, and Justin Bayer (2023). "Filter-Aware Model-Predictive Control." In: *Learning for Dynamics and Control Conference, L4DC 2023, 15-16 June 2023, Philadelphia, PA, USA*. Vol. 211. Proceedings of Machine Learning Research. PMLR, pp. 1441–1454. URL: https://proceedings.mlr.press/v211/kayalibay23a.html.

Prior Work (Before Doctorate)

Kayalibay, Baris, Atanas Mirchev, Maximilian Soelch, Patrick van der Smagt, and Justin Bayer (2018). *Navigation and planning in latent maps*. URL: http://reinforcement-learning.ml/papers/pgmrl2018_kayalibay.pdf.

Mirchev, Atanas, Baris Kayalibay, Maximilian Soelch, Patrick van der Smagt, and Justin Bayer (2019). "Approximate Bayesian Inference in Spatial Environments." In: *Robotics: Science and Systems XV, University of Freiburg, Freiburg im Breisgau, Germany, June 22-26, 2019*. DOI: 10.15607/RSS.2019.XV.083. URL: https://doi.org/10.15607/RSS.2019.XV.083.

Mirchev, Atanas and Seyed-Ahmad Ahmadi (2018). "Classification of sparsely labeled spatio-temporal data through semi-supervised adversarial learning." In: *arXiv preprint arXiv:1801.08712*.

Golkov, Vladimir, Marcin J. Skwark, Atanas Mirchev, Georgi Dikov, Alexander R. Geanes, Jeffrey L. Mendenhall, Jens Meiler, and Daniel Cremers (2020). "3D Deep Learning for Biological Function Prediction from Physical Fields." In: *8th International Conference on 3D Vision, 3DV 2020, Virtual Event, Japan, November 25-28, 2020*. IEEE, pp. 928–937. DOI: 10.1109/3DV50981.2020.00103. URL: https://doi.org/10.1109/3DV50981.2020.00103.

Leis, Viktor, Andrey Gubichev, Atanas Mirchev, Peter A. Boncz, Alfons Kemper, and Thomas Neumann (2015). "How Good Are Query Optimizers, Really?" In: *Proc. VLDB Endow.* 9.3, pp. 204–215. DOI: 10.14778/2850583.2850594. URL: http://www.vldb.org/pvldb/vol9/p204-leis.pdf.

Bibliography

Antonini, Amado, Winter Guerra, Varun Murali, Thomas Sayre-McCord, and Sertac Karaman (2020). "The Blackbird UAV dataset." In: *Int. J. Robotics Res.* 39.10-11. DOI: 10.1177/0278364920908331. URL: https://doi.org/10.1177/0278364920908331.

Asgharivaskasi, Arash, Shumon Koga, and Nikolay Atanasov (2022). "Active Mapping via Gradient Ascent Optimization of Shannon Mutual Information over Continuous SE(3) Trajectories." In: *IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2022, Kyoto, Japan, October 23-27, 2022*. IEEE, pp. 12994–13001. DOI: 10.1109/IROS47612.2022.9981875. URL: https://doi.org/10.1109/IROS47612.2022.9981875.

Audras, Cedric, A Comport, Maxime Meilland, and Patrick Rives (2011). "Real-time dense appearance-based SLAM for RGB-D sensors." In: *Australasian Conf. on Robotics and Automation*. Vol. 2, pp. 2–2.

Ayache, Nicholas and Olivier D. Faugeras (1988). "Building, Registrating, and Fusing Noisy Visual Maps." In: *Int. J. Robotics Res.* 7.6, pp. 45–65. DOI: 10.1177/027836498800700605. URL: https://doi.org/10.1177/027836498800700605.

Bay, Herbert, Tinne Tuytelaars, and Luc Van Gool (2006). "SURF: Speeded Up Robust Features." In: *Computer Vision - ECCV 2006, 9th European Conference on Computer Vision, Graz, Austria, May 7-13, 2006, Proceedings, Part I*. Ed. by Ales Leonardis, Horst Bischof, and Axel Pinz. Vol. 3951. Lecture Notes in Computer Science. Springer, pp. 404–417. DOI: 10.1007/11744023_32. URL: https://doi.org/10.1007/11744023%5C_32.

Becker-Ehmck, Philip, Maximilian Karl, Jan Peters, and Patrick van der Smagt (2020). "Learning to Fly via Deep Model-Based Reinforcement Learning." In: *CoRR* abs/2003.08876. arXiv: 2003.08876.

Bertsekas, Dimitri P. (2005). *Dynamic programming and optimal control, 3rd Edition*. Athena Scientific. ISBN: 1886529264. URL: https://www.worldcat.org/oclc/314894080.

Bishop, Christopher M. (2007). *Pattern recognition and machine learning, 5th Edition*. Information science and statistics. Springer. ISBN: 9780387310732. URL: https://www.worldcat.org/oclc/71008143.

Bradbury, James, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang (2018). *JAX: composable transformations of Python+NumPy programs*. Version 0.3.13. URL: http://github.com/google/jax.

Brohan, Anthony, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alexander Herzog, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Tomas Jackson, Sally Jesmonth, Nikhil J. Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Isabel Leal, Kuang-Huei Lee, Sergey Levine, Yao Lu, Utsav Malla, Deeksha Manjunath, Igor Mordatch, Ofir Nachum, Carolina Parada, Jodilyn Peralta, Emily Perez, Karl Pertsch, Jornell Quiambao, Kanishka Rao, Michael S. Ryoo, Grecia Salazar, Pannag R. Sanketi, Kevin Sayed, Jaspiar Singh, Sumedh Sontakke, Austin Stone, Clayton Tan, Huong Tran, Vincent Vanhoucke, Steve Vega, Quan Vuong, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Tianhe Yu, and Brianna Zitkovich (2023). "RT-1: Robotics Transformer for Real-World Control at Scale." In: *Robotics: Science and Systems XIX, Daegu, Republic of Korea, July 10-14, 2023*. Ed. by Kostas E. Bekris, Kris Hauser, Sylvia L. Herbert, and Jingjin Yu. DOI: 10.15607/RSS.2023.XIX.025. URL: https://doi.org/10.15607/RSS.2023.XIX.025.

Brown, Duane (1976). "The bundle adjustment-progress and prospect." In: *XIII Congress of the ISPRS, Helsinki, 1976*.

Cadena, Cesar, Luca Carlone, Henry Carrillo, Yasir Latif, Davide Scaramuzza, José Neira, Ian D. Reid, and John J. Leonard (2016). "Past, Present, and Future of Simultaneous Localization and Mapping: Toward the Robust-Perception Age." In: *IEEE Trans. Robotics* 32.6, pp. 1309–1332. DOI: 10.1109/TRO.2016.2624754. URL: https://doi.org/10.1109/TRO.2016.2624754.

Campos, Carlos, Richard Elvira, Juan J. Gomez, Jose M. M. Montiel, and Juan D. Tardos (2021). "ORB-SLAM3: An Accurate Open-Source Library for Visual, Visual-Inertial and Multi-Map SLAM." In: *IEEE Transactions on Robotics* 37.6, pp. 1874–1890.

Canelhas, Daniel Ricao, Todor Stoyanov, and Achim J. Lilienthal (2018). "A Survey of Voxel Interpolation Methods and an Evaluation of Their Impact on Volumetric Map-Based Visual Odometry." In: *2018 IEEE International Conference on Robotics and Automation, ICRA 2018, Brisbane, Australia, May 21-25, 2018*. IEEE, pp. 3637–3643. DOI: 10.1109/ICRA.2018.8461227. URL: https://doi.org/10.1109/ICRA.2018.8461227.

Cen, Jiazhong, Zanwei Zhou, Jiemin Fang, Wei Shen, Lingxi Xie, Dongsheng Jiang, Xiaopeng Zhang, and Qi Tian (2023). "Segment Anything in 3D with NeRFs." In: *CoRR* abs/2304.12308. DOI: 10.48550/arXiv.2304.12308. arXiv: 2304.12308. URL: https://doi.org/10.48550/arXiv.2304.12308.

Chaplot, Devendra Singh, Dhiraj Gandhi, Saurabh Gupta, Abhinav Gupta, and Ruslan Salakhutdinov (2020). "Learning To Explore Using Active Neural SLAM." In: *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net. URL: https://openreview.net/forum?id=HklXn1BKDH.

Chaplot, Devendra Singh, Emilio Parisotto, and Ruslan Salakhutdinov (2018). "Active Neural Localization." In: *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net. URL: https://openreview.net/forum?id=ry6-G%5C_66b.

Charrow, Benjamin, Gregory Kahn, Sachin Patil, Sikang Liu, Ken Goldberg, Pieter Abbeel, Nathan Michael, and Vijay Kumar (2015). "Information-Theoretic Planning with Trajectory Optimization for Dense 3D Mapping." In: *Robotics: Science and Systems*. Vol. 11. Rome, pp. 3–12.

Chatila, Raja and Jean-Paul Laumond (1985). "Position referencing and consistent world modeling for mobile robots." In: *Proceedings of the 1985 IEEE International Conference on Robotics and Automation, St. Louis, Missouri, USA, March 25-28, 1985*. IEEE, pp. 138–145. DOI: 10.1109/ROBOT.1985.1087373. URL: https://doi.org/10.1109/ROBOT.1985.1087373.

Chen, Yang and Gérard G. Medioni (1992). "Object modelling by registration of multiple range images." In: *Image Vis. Comput.* 10.3, pp. 145–155. DOI: 10.1016/0262-8856(92)90066-C. URL: https://doi.org/10.1016/0262-8856(92)90066-C.

Civera, Javier and Seong Hun Lee (2020). "RGB-D Odometry and SLAM." In: *CoRR* abs/2001.06875. arXiv: 2001.06875. URL: https://arxiv.org/abs/2001.06875.

Corneil, Dane, Wulfram Gerstner, and Johanni Brea (Oct. 2018). "Efficient Model-Based Deep Reinforcement Learning with Variational State Tabulation." In: ed. by Jennifer Dy and Andreas Krause. Vol. 80. Proceedings of Machine Learning Research. Stockholmsmässan, Stockholm Sweden: PMLR, pp. 1049–1058.

Crowley, James L. (1989). "World modeling and position estimation for a mobile robot using ultrasonic ranging." In: *Proceedings of the 1989 IEEE International Conference on Robotics and Automation, Scottsdale, Arizona, USA, May 14-19, 1989*. IEEE Computer Society, pp. 674–680. DOI: 10.

1109/ROBOT.1989.100062. URL: https://doi.org/10.1109/ROBOT.1989.100062.

Curless, Brian and Marc Levoy (1996). "A Volumetric Method for Building Complex Models from Range Images." In: *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 1996, New Orleans, LA, USA, August 4-9, 1996*. Ed. by John Fujii. ACM, pp. 303–312. DOI: 10.1145/237170.237269. URL: https://doi.org/10.1145/237170.237269.

Dai, Jian S (2015). "Euler–Rodrigues formula variations, quaternion conjugation and intrinsic connections." In: *Mechanism and Machine Theory* 92, pp. 144–152.

Davison, Andrew J., Ian D. Reid, Nicholas Molton, and Olivier Stasse (2007). "MonoSLAM: Real-Time Single Camera SLAM." In: *IEEE Trans. Pattern Anal. Mach. Intell.* 29.6, pp. 1052–1067. DOI: 10.1109/TPAMI.2007.1049. URL: https://doi.org/10.1109/TPAMI.2007.1049.

Deitke, Matt, Eli VanderBilt, Alvaro Herrasti, Luca Weihs, Jordi Salvador, Kiana Ehsani, Winson Han, Eric Kolve, Ali Farhadi, Aniruddha Kembhavi, and Roozbeh Mottaghi (2022). "ProcTHOR: Large-Scale Embodied AI Using Procedural Generation." In: *NeurIPS*. Outstanding Paper Award.

Dellaert, Frank and Michael Kaess (2017). "Factor Graphs for Robot Perception." In: *Found. Trends Robotics* 6.1-2, pp. 1–139. DOI: 10.1561/2300000043. URL: https://doi.org/10.1561/2300000043.

Deray, Jérémie and Joan Solà (2020). "Manif: A micro Lie theory library for state estimation in robotics applications." In: *J. Open Source Softw.* 5.46, p. 1371. DOI: 10.21105/joss.01371. URL: https://doi.org/10.21105/joss.01371.

Diehl, Moritz and Florian Messerer (2019). "Local Convergence of Generalized Gauss-Newton and Sequential Convex Programming." In: *58th IEEE Conference on Decision and Control, CDC 2019, Nice, France, December 11-13, 2019*. IEEE, pp. 3942–3947. DOI: 10.1109/CDC40024.2019.9029288. URL: https://doi.org/10.1109/CDC40024.2019.9029288.

Doucet, Arnaud, Adam M Johansen, et al. (2009). "A tutorial on particle filtering and smoothing: Fifteen years later." In: *Handbook of nonlinear filtering* 12.656-704, p. 3.

Durrant-Whyte, Hugh and Tim Bailey (2006). "Simultaneous localization and mapping: part I." In: *IEEE robotics & automation magazine* 13.2, pp. 99–110.

Durrant-Whyte, Hugh F. (1988). "Uncertain geometry in robotics." In: *IEEE J. Robotics Autom.* 4.1, pp. 23–31. DOI: 10.1109/56.768. URL: https://doi.org/10.1109/56.768.

Engel, Jakob (2017). "Large-Scale Direct SLAM and 3D Reconstruction in Real-Time." PhD thesis. Technical University Munich, Germany. URL: https://nbn-resolving.org/urn:nbn:de:bvb:91-diss-20171016-1326959-1-5.

Engel, Jakob, Vladlen Koltun, and Daniel Cremers (2018). "Direct Sparse Odometry." In: *IEEE Trans. Pattern Anal. Mach. Intell.* 40.3, pp. 611–625. DOI: 10.1109/TPAMI.2017.2658577. URL: https://doi.org/10.1109/TPAMI.2017.2658577.

Engel, Jakob, Thomas Schöps, and Daniel Cremers (2014). "LSD-SLAM: Large-Scale Direct Monocular SLAM." In: *Computer Vision - ECCV 2014 - 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part II*. Ed. by David J. Fleet, Tomás Pajdla, Bernt Schiele, and Tinne Tuytelaars. Vol. 8690. Lecture Notes in Computer Science. Springer, pp. 834–849. DOI: 10.1007/978-3-319-10605-2_54. URL: https://doi.org/10.1007/978-3-319-10605-2%5C_54.

Forster, Christian, Matia Pizzoli, and Davide Scaramuzza (2014). "SVO: Fast semi-direct monocular visual odometry." In: *2014 IEEE International Conference on Robotics and Automation, ICRA 2014, Hong Kong, China, May 31 - June 7, 2014*. IEEE, pp. 15–22. DOI: 10.1109/ICRA.2014.6906584. URL: https://doi.org/10.1109/ICRA.2014.6906584.

Fraccaro, Marco, Danilo Jimenez Rezende, Yori Zwols, Alexander Pritzel, S. M. Ali Eslami, and Fabio Viola (2018). "Generative Temporal Models with Spatial Memory for Partially Observed Environments." In: *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*. Ed. by Jennifer G. Dy and Andreas Krause. Vol. 80. Proceedings of Machine Learning Research. PMLR, pp. 1544–1553. URL: http://proceedings.mlr.press/v80/fraccaro18a.html.

Gallier, Jean (2011). *Geometric methods and applications: for computer science and engineering*. Vol. 38. Springer Science & Business Media.

Gregor, Karol, Danilo Jimenez Rezende, Frederic Besse, Yan Wu, Hamza Merzic, and Aäron van den Oord (2019). "Shaping Belief States with Generative Environment Models for RL." In: *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*. Ed. by Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, pp. 13475–13487. URL: https://proceedings.neurips.cc/paper/2019/hash/2c048d74b3410237704eb7f93a10c9d7-Abstract.html.

Grisetti, Giorgio, Cyrill Stachniss, and Wolfram Burgard (2007). "Improved Techniques for Grid Mapping With Rao-Blackwellized Particle Filters." In: *IEEE Trans. Robotics* 23.1, pp. 34–46. DOI: 10.1109/TRO.2006.889486. URL: https://doi.org/10.1109/TRO.2006.889486.

Ha, David and Jürgen Schmidhuber (2018). "Recurrent World Models Facilitate Policy Evolution." In: *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*. Ed. by Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett, pp. 2455–2467. URL: https://proceedings.neurips.cc/paper/2018/hash/2de5d16682c3c35007e4e92982f1a2ba-Abstract.html.

Hafner, Danijar, Timothy P. Lillicrap, Jimmy Ba, and Mohammad Norouzi (2020). "Dream to Control: Learning Behaviors by Latent Imagination." In: *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net. URL: https://openreview.net/forum?id=S1lOTC4tDS.

Hafner, Danijar, Timothy P. Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson (2019). "Learning Latent Dynamics for Planning from Pixels." In: *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*. Ed. by Kamalika Chaudhuri and Ruslan Salakhutdinov. Vol. 97. Proceedings of Machine Learning Research. PMLR, pp. 2555–2565. URL: http://proceedings.mlr.press/v97/hafner19a.html.

Hafner, Danijar, Jurgis Pasukonis, Jimmy Ba, and Timothy P. Lillicrap (2023). "Mastering Diverse Domains through World Models." In: *CoRR* abs/2301.04104. DOI: 10.48550/arXiv.2301.04104. arXiv: 2301.04104. URL: https://doi.org/10.48550/arXiv.2301.04104.

Hall, Brian C and Brian C Hall (2013). *Lie groups, Lie algebras, and representations*. Springer.

Hartley, Andrew and Andrew Zisserman (2006). *Multiple view geometry in computer vision (2. ed.)* Cambridge University Press. ISBN: 978-0-521-54051-3.

Huang, Qiangqiang and John J. Leonard (2023). "GAPSLAM: Blending Gaussian Approximation and Particle Filters for Real-Time Non-Gaussian SLAM." In: *CoRR* abs/2303.14283. DOI: 10.48550/arXiv.2303.14283. arXiv: 2303.14283. URL: https://doi.org/10.48550/arXiv.2303.14283.

Huang, Shoudong and Gamini Dissanayake (2016). "A critique of current developments in simultaneous localization and mapping." In: *International Journal of Advanced Robotic Systems* 13.5, p. 1729881416669482.

Kaiser, Lukasz, Mohammad Babaeizadeh, Piotr Milos, Blazej Osinski, Roy H. Campbell, Konrad Czechowski, Dumitru Erhan, Chelsea Finn, Piotr Kozakowski, Sergey Levine, Afroz Mohiuddin, Ryan Sepassi, George Tucker, and Henryk Michalewski (2020). "Model Based Reinforcement Learning for Atari." In: *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net. URL: https://openreview.net/forum?id=S1xCPJHtDB.

Kalman, R. E. (Mar. 1960). "A New Approach to Linear Filtering and Prediction Problems." In: *Journal of Basic Engineering* 82.1, pp. 35–45. ISSN: 0021-9223. DOI: 10.1115/1.3662552. URL: https://doi.org/10.1115/1.3662552.

Kayalibay, Baris (to be submitted in 2023/2024). "Control with Spatial World Models." PhD thesis. Technical University Munich, Germany.

Kayalibay, Baris, Atanas Mirchev, Ahmed Agha, Patrick van der Smagt, and Justin Bayer (2023). "Filter-Aware Model-Predictive Control." In: *Learning for Dynamics and Control Conference, L4DC 2023, 15-16 June 2023, Philadelphia, PA, USA*. Vol. 211. Proceedings of Machine Learning Research. PMLR, pp. 1441–1454. URL: https://proceedings.mlr.press/v211/kayalibay23a.html.

Kayalibay, Baris, Atanas Mirchev, Patrick van der Smagt, and Justin Bayer (2022). "Tracking and Planning with Spatial World Models." In: *Learning for Dynamics and Control Conference, L4DC 2022, 23-24 June 2022, Stanford University, Stanford, CA, USA*. Vol. 168. Proceedings of Machine Learning Research. PMLR, pp. 124–137.

Kayalibay, Baris, Atanas Mirchev, Maximilian Soelch, Patrick van der Smagt, and Justin Bayer (2018). *Navigation and planning in latent maps*. URL: http://reinforcement-learning.ml/papers/pgmrl2018_kayalibay.pdf.

Kingma, Diederik P. and Jimmy Ba (2015). "Adam: A Method for Stochastic Optimization." In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. Ed. by Yoshua Bengio and Yann LeCun. URL: http://arxiv.org/abs/1412.6980.

Kingma, Diederik P. and Max Welling (2014). "Auto-Encoding Variational Bayes." In: *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*. Ed. by Yoshua Bengio and Yann LeCun. URL: http://arxiv.org/abs/1312.6114.

Klein, Georg and David William Murray (2007). "Parallel Tracking and Mapping for Small AR Workspaces." In: *Sixth IEEE/ACM International Symposium on Mixed and Augmented Reality, ISMAR 2007, 13-16 November 2007, Nara, Japan*. IEEE Computer Society, pp. 225–234. DOI: 10.1109/ISMAR.2007.4538852. URL: https://doi.org/10.1109/ISMAR.2007.4538852.

Klingensmith, Matthew, Siddartha S. Sirinivasa, and Michael Kaess (2016). "Articulated Robot Motion for Simultaneous Localization and Mapping (ARM-SLAM)." In: *IEEE Robotics Autom. Lett.* 1.2, pp. 1156–1163. DOI: 10.1109/LRA.2016.2518242. URL: https://doi.org/10.1109/LRA.2016.2518242.

Koller, Daphne and Nir Friedman (2009). *Probabilistic Graphical Models - Principles and Techniques*. MIT Press. ISBN: 978-0-262-01319-2. URL: http://mitpress.mit.edu/catalog/item/default.asp?ttype=2%5C&tid=11886.

Kosiorek, Adam R., Heiko Strathmann, Daniel Zoran, Pol Moreno, Rosalia Schneider, Sona Mokrá, and Danilo Jimenez Rezende (2021). "NeRF-VAE: A Geometry Aware 3D Scene Generative Model." In: *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*. Ed. by Marina Meila and Tong Zhang. Vol. 139. Proceedings of Machine Learning Research. PMLR, pp. 5742–5752. URL: http://proceedings.mlr.press/v139/kosiorek21a.html.

Laplace, Pierre Simon (1986). "Memoir on the Probability of the Causes of Events." In: *Statistical Science* 1.3, pp. 364–378. ISSN: 08834237. URL: http://www.jstor.org/stable/2245476 (visited on 05/13/2022).

Lee, Alex X., Anusha Nagabandi, Pieter Abbeel, and Sergey Levine (2020). "Stochastic Latent Actor-Critic: Deep Reinforcement Learning with a Latent Variable Model." In: *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*. Ed. by Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin. URL: https://proceedings.neurips.cc/paper/2020/hash/08058bf500242562c0d031ff830ad094-Abstract.html.

Leutenegger, Stefan, Paul Timothy Furgale, Vincent Rabaud, Margarita Chli, Kurt Konolige, and Roland Siegwart (2013). "Keyframe-Based Visual-Inertial SLAM using Nonlinear Optimization." In: *Robotics: Science and Systems IX, Technische Universität Berlin, Berlin, Germany, June 24 - June 28, 2013*. Ed. by Paul Newman, Dieter Fox, and David Hsu. DOI: 10.15607/RSS.2013.IX.037. URL: http://www.roboticsproceedings.org/rss09/p37.html.

Li, Mingyang and Anastasios I Mourikis (2013). "High-precision, consistent EKF-based visual-inertial odometry." In: *The International Journal of Robotics Research* 32.6, pp. 690–711.

Li, Zhengqi, Simon Niklaus, Noah Snavely, and Oliver Wang (2021). "Neural Scene Flow Fields for Space-Time View Synthesis of Dynamic Scenes." In: *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021*. Computer Vision Foundation / IEEE, pp. 6498–6508. DOI: 10.1109/CVPR46437.2021.00643. URL: https://openaccess.thecvf.com/content/CVPR2021/html/Li%5C_Neural%5C_Scene%5C_Flow%5C_Fields%5C_for%5C_Space-Time%5C_View%5C_Synthesis%5C_of%5C_Dynamic%5C_CVPR%5C_2021%5C_paper.html.

Liu, Wenxin, David Caruso, Eddy Ilg, Jing Dong, Anastasios I. Mourikis, Kostas Daniilidis, Vijay Kumar, and Jakob Engel (2020). "TLIO: Tight Learned Inertial Odometry." In: *IEEE Robotics Autom. Lett.* 5.4, pp. 5653–5660. DOI: 10.1109/LRA.2020.3007421. URL: https://doi.org/10.1109/LRA.2020.3007421.

Lombardi, Stephen, Tomas Simon, Jason M. Saragih, Gabriel Schwartz, Andreas M. Lehrmann, and Yaser Sheikh (2019). "Neural volumes: learning dynamic renderable volumes from images." In: *ACM Trans. Graph.* 38.4, 65:1–65:14. DOI: 10.1145/3306346.3323020. URL: https://doi.org/10.1145/3306346.3323020.

Lombardi, Stephen, Tomas Simon, Gabriel Schwartz, Michael Zollhöfer, Yaser Sheikh, and Jason M. Saragih (2021). "Mixture of volumetric primitives for efficient neural rendering." In: *ACM Trans. Graph.* 40.4, 59:1–59:13. DOI: 10.1145/3450626.3459863. URL: https://doi.org/10.1145/3450626.3459863.

Lowe, David G. (1999). "Object Recognition from Local Scale-Invariant Features." In: *Proceedings of the International Conference on Computer Vision, Kerkyra, Corfu, Greece, September 20-25, 1999*. IEEE Computer Society, pp. 1150–1157. DOI: 10.1109/ICCV.1999.790410. URL: https://doi.org/10.1109/ICCV.1999.790410.

Lucas, Bruce D. and Takeo Kanade (1981). "An Iterative Image Registration Technique with an Application to Stereo Vision." In: *Proceedings of the 7th International Joint Conference on Artificial Intelligence, IJCAI '81, Vancouver, BC, Canada, August 24-28, 1981*. Ed. by Patrick J. Hayes. William Kaufmann, pp. 674–679.

Luiten, Jonathon, Georgios Kopanas, Bastian Leibe, and Deva Ramanan (2023). "Dynamic 3D Gaussians: Tracking by Persistent Dynamic View Synthesis." In: *preprint*.

Meagher, Donald (1982). "Geometric modeling using octree encoding." In: *Computer graphics and image processing* 19.2, pp. 129–147.

Mildenhall, Ben, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng (2020). "NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis." In: *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part I*. Ed. by Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm. Vol. 12346. Lecture Notes in Computer Science. Springer, pp. 405–421. DOI: 10.1007/978-3-030-58452-8_24. URL: https://doi.org/10.1007/978-3-030-58452-8%5C_24.

Minka, Thomas P. (2001). "A family of algorithms for approximate Bayesian inference." PhD thesis. Massachusetts Institute of Technology, Cambridge, MA, USA. URL: https://hdl.handle.net/1721.1/86583.

Mirchev, Atanas, Baris Kayalibay, Ahmed Agha, Patrick van der Smagt, Daniel Cremers, and Justin Bayer (2022). "PRISM: Probabilistic Real-Time Inference in Spatial World Models." In: *Conference on Robot Learning, CoRL 2022, 14-18 December 2022, Auckland, New Zealand*. Vol. 205. Proceedings of Machine Learning Research. PMLR, pp. 161–174. URL: https://proceedings.mlr.press/v205/mirchev23a.html.

Mirchev, Atanas, Baris Kayalibay, Patrick van der Smagt, and Justin Bayer (2021). "Variational State-Space Models for Localisation and Dense 3D Mapping in 6 DoF." In: *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net. URL: https://openreview.net/forum?id=XAS3uKeFWj.

Mirchev, Atanas, Baris Kayalibay, Maximilian Soelch, Patrick van der Smagt, and Justin Bayer (2019). "Approximate Bayesian Inference in Spatial Environments." In: *Robotics: Science and Systems XV, University of Freiburg, Freiburg im Breisgau, Germany, June 22-26, 2019*. DOI: 10.15607/RSS.2019.XV.083. URL: https://doi.org/10.15607/RSS.2019.XV.083.

Montemerlo, Michael, Sebastian Thrun, Daphne Koller, and Ben Wegbreit (2002). "FastSLAM: A Factored Solution to the Simultaneous Localization and Mapping Problem." In: *Proceedings of the Eighteenth National Conference on Artificial Intelligence and Fourteenth Conference on Innovative Applications of Artificial Intelligence, July 28 - August 1, 2002, Edmonton, Alberta, Canada*. Ed. by Rina Dechter, Michael J. Kearns, and Richard S. Sutton. AAAI Press / The MIT Press, pp. 593–598. URL: http://www.aaai.org/Library/AAAI/2002/aaai02-089.php.

Moravec, Hans P. (1988). "Sensor Fusion in Certainty Grids for Mobile Robots." In: *AI Mag.* 9.2, pp. 61–74. URL: https://ojs.aaai.org/index.php/aimagazine/article/view/676.

Mourikis, Anastasios I. and Stergios I. Roumeliotis (2007). "A Multi-State Constraint Kalman Filter for Vision-aided Inertial Navigation." In: *2007 IEEE International Conference on Robotics and Automation, ICRA 2007, 10-14 April 2007, Roma, Italy*. IEEE, pp. 3565–3572. DOI: 10.1109/ROBOT.2007.364024. URL: https://doi.org/10.1109/ROBOT.2007.364024.

Müller, Thomas, Alex Evans, Christoph Schied, and Alexander Keller (2022). "Instant neural graphics primitives with a multiresolution hash encoding." In: *ACM Trans. Graph.* 41.4, 102:1–102:15. DOI: 10.1145/3528223.3530127. URL: https://doi.org/10.1145/3528223.3530127.

Mur-Artal, Raul, J. M. M. Montiel, and Juan D. Tardós (2015). "ORB-SLAM: A Versatile and Accurate Monocular SLAM System." In: *IEEE Trans. Robotics* 31.5, pp. 1147–1163. DOI: 10.1109/TRO.2015.2463671. URL: https://doi.org/10.1109/TRO.2015.2463671.

Murphy, Kevin P. (1999). "Bayesian Map Learning in Dynamic Environments." In: *Advances in Neural Information Processing Systems 12, NIPS Conference, Denver, Colorado, USA, November 29 - December 4, 1999*, pp. 1015–1021. URL: http://papers.nips.cc/paper/1716-bayesian-map-learning-in-dynamic-environments.

Newcombe, Richard A., Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew J. Davison, Pushmeet Kohli, Jamie Shotton, Steve Hodges, and Andrew W. Fitzgibbon (2011a). "KinectFusion: Real-time dense surface mapping and tracking." In: *10th IEEE International Symposium on Mixed and Augmented Reality, ISMAR 2011, Basel, Switzerland, October 26-29, 2011*. IEEE Computer Society, pp. 127–136. DOI: 10.1109/ISMAR.2011.6092378. URL: https://doi.org/10.1109/ISMAR.2011.6092378.

Newcombe, Richard A., Steven Lovegrove, and Andrew J. Davison (2011b). "DTAM: Dense tracking and mapping in real-time." In: *IEEE International Conference on Computer Vision, ICCV 2011, Barcelona, Spain, November 6-13, 2011*. Ed. by Dimitris N. Metaxas, Long Quan, Alberto Sanfeliu, and Luc Van Gool. IEEE Computer Society, pp. 2320–2327. DOI: 10.1109/ICCV.2011.6126513. URL: https://doi.org/10.1109/ICCV.2011.6126513.

Nießner, Matthias, Michael Zollhöfer, Shahram Izadi, and Marc Stamminger (2013). "Real-time 3D reconstruction at scale using voxel hashing." In: *ACM Trans. Graph.* 32.6, 169:1–169:11. DOI: 10.1145/2508363.2508374. URL: https://doi.org/10.1145/2508363.2508374.

Nisar, Barza, Philipp Foehn, Davide Falanga, and Davide Scaramuzza (2019). "VIMO: Simultaneous Visual Inertial Model-Based Odometry and Force Estimation." In: *Robotics: Science and Systems XV, University of Freiburg, Freiburg im Breisgau, Germany, June 22-26, 2019*. Ed. by Antonio

Bicchi, Hadas Kress-Gazit, and Seth Hutchinson. DOI: 10.15607/RSS.2019.XV.082. URL: https://doi.org/10.15607/RSS.2019.XV.082.

Nocedal, Jorge and Stephen J. Wright (1999). *Numerical Optimization*. Springer. ISBN: 978-0-387-98793-4. DOI: 10.1007/b98874. URL: https://doi.org/10.1007/b98874.

Opper, Manfred and Ole Winther (1999). "A Bayesian approach to on-line learning." In:

Parker, Steven G., Peter Shirley, Yarden Livnat, Charles D. Hansen, and Peter-Pike J. Sloan (1998). "Interactive ray tracing for isosurface rendering." In: *9th IEEE Visualization Conference, IEEE Vis 1998, Research Triangle Park, North Carolina, USA, October 18-23, 1998, Proceedings*. Ed. by David S. Ebert, Holly E. Rushmeier, and Hans Hagen. IEEE Computer Society and ACM, pp. 233–238. DOI: 10.1109/VISUAL.1998.745713. URL: https://doi.org/10.1109/VISUAL.1998.745713.

Pietzsch, Tobias (2008). "Planar Features for Visual SLAM." In: *KI 2008: Advances in Artificial Intelligence, 31st Annual German Conference on AI, KI 2008, Kaiserslautern, Germany, September 23-26, 2008. Proceedings*. Ed. by Andreas Dengel, Karsten Berns, Thomas M. Breuel, Frank Bomarius, and Thomas Roth-Berghofer. Vol. 5243. Lecture Notes in Computer Science. Springer, pp. 119–126. DOI: 10.1007/978-3-540-85845-4_15. URL: https://doi.org/10.1007/978-3-540-85845-4%5C_15.

Placed, Julio A., Jared Strader, Henry Carrillo, Nikolay Atanasov, Vadim Indelman, Luca Carlone, and José A. Castellanos (2023). "A Survey on Active Simultaneous Localization and Mapping: State of the Art and New Frontiers." In: *IEEE Trans. Robotics* 39.3, pp. 1686–1705. DOI: 10.1109/TRO.2023.3248510. URL: https://doi.org/10.1109/TRO.2023.3248510.

Qin, Tong, Peiliang Li, and Shaojie Shen (2018). "VINS-Mono: A Robust and Versatile Monocular Visual-Inertial State Estimator." In: *IEEE Trans. Robotics* 34.4, pp. 1004–1020. DOI: 10.1109/TRO.2018.2853729. URL: https://doi.org/10.1109/TRO.2018.2853729.

Qin, Tong, Jie Pan, Shaozu Cao, and Shaojie Shen (2019). "A General Optimization-based Framework for Local Odometry Estimation with Multiple Sensors." In: *CoRR* abs/1901.03638. arXiv: 1901.03638. URL: http://arxiv.org/abs/1901.03638.

Reed, Scott E., Konrad Zolna, Emilio Parisotto, Sergio Gómez Colmenarejo, Alexander Novikov, Gabriel Barth-Maron, Mai Gimenez, Yury Sulsky, Jackie Kay, Jost Tobias Springenberg, Tom Eccles, Jake Bruce, Ali Razavi, Ashley Edwards, Nicolas Heess, Yutian Chen, Raia Hadsell, Oriol Vinyals, Mahyar Bordbar, and Nando de Freitas (2022). "A Generalist Agent."

In: *Trans. Mach. Learn. Res.* 2022. URL: https://openreview.net/forum?id=1ikK0kHjvj.

Reiser, Christian, Songyou Peng, Yiyi Liao, and Andreas Geiger (2021). "KiloNeRF: Speeding up Neural Radiance Fields with Thousands of Tiny MLPs." In: *2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021, Montreal, QC, Canada, October 10-17, 2021*. IEEE, pp. 14315–14325.

Rezende, Danilo Jimenez and Shakir Mohamed (2015). "Variational Inference with Normalizing Flows." In: *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*. Ed. by Francis R. Bach and David M. Blei. Vol. 37. JMLR Workshop and Conference Proceedings. JMLR.org, pp. 1530–1538. URL: http://proceedings.mlr.press/v37/rezende15.html.

Rosinol, Antoni, Marcus Abate, Yun Chang, and Luca Carlone (2020). "Kimera: an Open-Source Library for Real-Time Metric-Semantic Localization and Mapping." In: *2020 IEEE International Conference on Robotics and Automation, ICRA 2020, Paris, France, May 31 - August 31, 2020*. IEEE, pp. 1689–1696. DOI: 10.1109/ICRA40945.2020.9196885. URL: https://doi.org/10.1109/ICRA40945.2020.9196885.

Rosinol, Antoni, John J. Leonard, and Luca Carlone (2022). "NeRF-SLAM: Real-Time Dense Monocular SLAM with Neural Radiance Fields." In: *CoRR* abs/2210.13641. DOI: 10.48550/arXiv.2210.13641. arXiv: 2210.13641. URL: https://doi.org/10.48550/arXiv.2210.13641.

Rosten, Edward, Reid B. Porter, and Tom Drummond (2010). "Faster and Better: A Machine Learning Approach to Corner Detection." In: *IEEE Trans. Pattern Anal. Mach. Intell.* 32.1, pp. 105–119. DOI: 10.1109/TPAMI.2008.275. URL: https://doi.org/10.1109/TPAMI.2008.275.

Rublee, Ethan, Vincent Rabaud, Kurt Konolige, and Gary R. Bradski (2011). "ORB: An efficient alternative to SIFT or SURF." In: *IEEE International Conference on Computer Vision, ICCV 2011, Barcelona, Spain, November 6-13, 2011*. Ed. by Dimitris N. Metaxas, Long Quan, Alberto Sanfeliu, and Luc Van Gool. IEEE Computer Society, pp. 2564–2571. DOI: 10.1109/ICCV.2011.6126544. URL: https://doi.org/10.1109/ICCV.2011.6126544.

Särkkä, Simo (2013). *Bayesian Filtering and Smoothing*. Vol. 3. Institute of Mathematical Statistics textbooks. Cambridge University Press. ISBN: 978-1-10-761928-9. URL: http://www.cambridge.org/de/academic/subjects/statistics-probability/applied-probability-and-stochastic-networks/bayesian-filtering-and-smoothing?format=PB.

Schöps, Thomas, Torsten Sattler, and Marc Pollefeys (2019). "BAD SLAM: Bundle Adjusted Direct RGB-D SLAM." In: *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*. Computer Vision Foundation / IEEE, pp. 134–144. DOI: 10.1109/CVPR.2019.00022. URL: http://openaccess.thecvf.com/content%5C_CVPR%5C_2019/html/Schops%5C_BAD%5C_SLAM%5C_Bundle%5C_Adjusted%5C_Direct%5C_RGB-D%5C_SLAM%5C_CVPR%5C_2019%5C_paper.html.

Scona, Raluca, Simona Nobili, Yvan R. Petillot, and Maurice F. Fallon (2017). "Direct visual SLAM fusing proprioception for a humanoid robot." In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2017, Vancouver, BC, Canada, September 24-28, 2017*. IEEE, pp. 1419–1426. DOI: 10.1109/IROS.2017.8205943. URL: https://doi.org/10.1109/IROS.2017.8205943.

Smith, Randall, Matthew Self, and Peter C. Cheeseman (1990). "Estimating Uncertain Spatial Relationships in Robotics." In: *Autonomous Robot Vehicles*. Ed. by Ingemar J. Cox and Gordon T. Wilfong. Springer, pp. 167–193. DOI: 10.1007/978-1-4613-8997-2_14. URL: https://doi.org/10.1007/978-1-4613-8997-2%5C_14.

Smith, Randall C and Peter Cheeseman (1986). "On the representation and estimation of spatial uncertainty." In: *The international journal of Robotics Research* 5.4, pp. 56–68.

Stachniss, Cyrill (2009). *Robotic Mapping and Exploration*. Vol. 55. Springer Tracts in Advanced Robotics. Springer. ISBN: 978-3-642-01096-5. DOI: 10.1007/978-3-642-01097-2. URL: https://doi.org/10.1007/978-3-642-01097-2.

Stachniss, Cyrill and Wolfram Burgard (2003). "Exploring Unknown Environments with Mobile Robots using Coverage Maps." In: *IJCAI-03, Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence, Acapulco, Mexico, August 9-15, 2003*. Ed. by Georg Gottlob and Toby Walsh. Morgan Kaufmann, pp. 1127–1134. URL: http://ijcai.org/Proceedings/03/Papers/162.pdf.

Steinbrücker, Frank, Christian Kerl, and Daniel Cremers (2013). "Large-Scale Multi-resolution Surface Reconstruction from RGB-D Sequences." In: *IEEE International Conference on Computer Vision, ICCV 2013, Sydney, Australia, December 1-8, 2013*. IEEE Computer Society, pp. 3264–3271. DOI: 10.1109/ICCV.2013.405. URL: https://doi.org/10.1109/ICCV.2013.405.

Steinbrücker, Frank, Jürgen Sturm, and Daniel Cremers (2011). "Real-time visual odometry from dense RGB-D images." In: *IEEE International*

Conference on Computer Vision Workshops, ICCV 2011 Workshops, Barcelona, Spain, November 6-13, 2011. IEEE Computer Society, pp. 719–722. DOI: 10.1109/ICCVW.2011.6130321. URL: https://doi.org/10.1109/ICCVW.2011.6130321.

Strasdat, Hauke, J. M. M. Montiel, and Andrew J. Davison (2012). "Visual SLAM: Why filter?" In: *Image Vis. Comput.* 30.2, pp. 65–77. DOI: 10.1016/j.imavis.2012.02.009. URL: https://doi.org/10.1016/j.imavis.2012.02.009.

Sturm, Jürgen, Nikolas Engelhard, Felix Endres, Wolfram Burgard, and Daniel Cremers (2012). "A benchmark for the evaluation of RGB-D SLAM systems." In: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2012, Vilamoura, Algarve, Portugal, October 7-12, 2012*. IEEE, pp. 573–580. DOI: 10.1109/IROS.2012.6385773. URL: https://doi.org/10.1109/IROS.2012.6385773.

Sucar, Edgar, Shikun Liu, Joseph Ortiz, and Andrew J. Davison (2021). "iMAP: Implicit Mapping and Positioning in Real-Time." In: *2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021, Montreal, QC, Canada, October 10-17, 2021*. IEEE, pp. 6209–6218. DOI: 10.1109/ICCV48922.2021.00617. URL: https://doi.org/10.1109/ICCV48922.2021.00617.

Sutton, Richard S and Andrew G Barto (2018). *Reinforcement learning: An introduction*. MIT press.

Thrun, Sebastian, Wolfram Burgard, and Dieter Fox (2000). "A Real-Time Algorithm for Mobile Robot Mapping With Applications to Multi-Robot and 3D Mapping." In: *Proceedings of the 2000 IEEE International Conference on Robotics and Automation, ICRA 2000, April 24-28, 2000, San Francisco, CA, USA*. IEEE, pp. 321–328. DOI: 10.1109/ROBOT.2000.844077. URL: https://doi.org/10.1109/ROBOT.2000.844077.

– (2005). *Probabilistic robotics*. Intelligent robotics and autonomous agents. MIT Press. ISBN: 978-0-262-20162-9.

Thrun, Sebastian, Daphne Koller, Zoubin Ghahramani, Hugh F. Durrant-Whyte, and Andrew Y. Ng (2002). "Simultaneous Mapping and Localization with Sparse Extended Information Filters: Theory and Initial Results." In: *Algorithmic Foundations of Robotics V, Selected Contributions of the Fifth International Workshop on the Algorithmic Foundations of Robotics, WAFR 2002, Nice, France, December 15-17, 2002*. Ed. by Jean-Daniel Boissonnat, Joel W. Burdick, Ken Goldberg, and Seth Hutchinson. Vol. 7. Springer Tracts in Advanced Robotics. Springer, pp. 363–380. DOI: 10.1007/978-3-540-45058-0_22. URL: https://doi.org/10.1007/978-3-540-45058-0%5C_22.

Tomasi, Carlo and Takeo Kanade (1991). "Detection and Tracking of Point Features." In: *Carnegie Mellon University Technical Report CMU-CS-91-132*.

Tosi, Fabio, Youmin Zhang, Ziren Gong, Erik Sandström, Stefano Mattoccia, Martin R. Oswald, and Matteo Poggi (2024). "How NeRFs and 3D Gaussian Splatting are Reshaping SLAM: a Survey." In: *CoRR* abs/2402.13255. DOI: 10.48550/ARXIV.2402.13255. arXiv: 2402.13255. URL: https://doi.org/10.48550/arXiv.2402.13255.

Tran, Dustin, Matthew D. Hoffman, Dave Moore, Christopher Suter, Srinivas Vasudevan, Alexey Radul, Matthew Johnson, and Rif A. Saurous (2018). "Simple, Distributed, and Accelerated Probabilistic Programming." In: *Neural Information Processing Systems*.

Wagner, René, Udo Frese, and Berthold Bäuml (2014). "Graph SLAM with signed distance function maps on a humanoid robot." In: *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2014, Chicago, IL, USA, September 14-18, 2014*. IEEE, pp. 2691–2698. DOI: 10.1109/IROS.2014.6942930. URL: https://doi.org/10.1109/IROS.2014.6942930.

Whelan, Thomas, Stefan Leutenegger, Renato F. Salas-Moreno, Ben Glocker, and Andrew J. Davison (2015). "ElasticFusion: Dense SLAM Without A Pose Graph." In: *Robotics: Science and Systems XI, Sapienza University of Rome, Rome, Italy, July 13-17, 2015*. Ed. by Lydia E. Kavraki, David Hsu, and Jonas Buchli. DOI: 10.15607/RSS.2015.XI.001. URL: http://www.roboticsproceedings.org/rss11/p01.html.

Wu, Philipp, Alejandro Escontrela, Danijar Hafner, Pieter Abbeel, and Ken Goldberg (2022). "DayDreamer: World Models for Physical Robot Learning." In: *Conference on Robot Learning, CoRL 2022, 14-18 December 2022, Auckland, New Zealand*. Ed. by Karen Liu, Dana Kulic, and Jeffrey Ichnowski. Vol. 205. Proceedings of Machine Learning Research. PMLR, pp. 2226–2240. URL: https://proceedings.mlr.press/v205/wu23c.html.

Yang, Yulin, Chuchu Chen, Woosik Lee, and Guoquan P. Huang (2022). "Decoupled Right Invariant Error States for Consistent Visual-Inertial Navigation." In: *IEEE Robotics Autom. Lett.* 7.2, pp. 1627–1634. DOI: 10.1109/LRA.2021.3140054. URL: https://doi.org/10.1109/LRA.2021.3140054.

Zhang, Zhengdong, Theia Henderson, Sertac Karaman, and Vivienne Sze (2020). "FSMI: Fast computation of Shannon mutual information for information-theoretic mapping." In: *Int. J. Robotics Res.* 39.9. DOI: 10.1177/0278364920921941. URL: https://doi.org/10.1177/0278364920921941.

Zhao, Ziqing (2023). "Information Theory for Active Exploration in Spatial World Models." MA thesis. Technical Univesity Munich, Germany.

Zhu, Zihan, Songyou Peng, Viktor Larsson, Zhaopeng Cui, Martin R. Oswald, Andreas Geiger, and Marc Pollefeys (2023). "NICER-SLAM: Neural Implicit Scene Encoding for RGB SLAM." In: *CoRR* abs/2302.03594. DOI: 10.48550/arXiv.2302.03594. arXiv: 2302.03594. URL: https://doi.org/10.48550/arXiv.2302.03594.

Zhu, Zihan, Songyou Peng, Viktor Larsson, Weiwei Xu, Hujun Bao, Zhaopeng Cui, Martin R. Oswald, and Marc Pollefeys (2022). "NICE-SLAM: Neural Implicit Scalable Encoding for SLAM." In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022*. IEEE, pp. 12776–12786. DOI: 10.1109/CVPR52688.2022.01245. URL: https://doi.org/10.1109/CVPR52688.2022.01245.

Zollhöfer, Michael, Patrick Stotko, Andreas Görlitz, Christian Theobalt, Matthias Nießner, Reinhard Klein, and Andreas Kolb (2018). "State of the Art on 3D Reconstruction with RGB-D Cameras." In: *Comput. Graph. Forum* 37.2, pp. 625–652. DOI: 10.1111/CGF.13386. URL: https://doi.org/10.1111/cgf.13386.

Addendum

D.1. Clarification of Experiment Details

Exploration The plots in fig. 5.14 were created by the author using an experiment setup and data courtesy of Ziqing Zhao (see attributions in section 5.6.2, she did exploration experiments while working on her thesis, Zhao (2023), Technical University Munich). Based on that data, the entropies in fig. 5.14c were reported up to an added constant. Inevitably, the exact values depend on the details of the simulator, map resolution and map updates, and are thus meant only as a qualitative result.

Also, around eq. (5.27) it is explained that a part of the exploration objective in eq. (5.25) is MC-estimated by rendering images $\mathbf{x}_{>t}$. This is the general theoretical perspective, however the accompanying experiments from fig. 5.14 use a single $\mathbf{x}_{>t}$ for the estimation, namely the mean of the rendering emission conditioned on the mean of the current map belief $q_t^{\Phi}(\mathfrak{M})$. This is biased towards optimism when the map mean is initialised empty (SDF-wise), but works well in the chosen environment. This could be relevant for reproducibility, otherwise conclusions remain the same.

QCar Navigation Around eq. (5.20) it is explained that aggregated value iteration is used to approximate the navigation cost-to-go. In the experiments, this was done using the map mean to evaluate the cost values $c(\mathbf{z}_\tau, \mathfrak{M})$. The assumed transition model at this point was also deterministic (bicycle kinematics).

D.2. Typos And Notation

1. In eq. (5.18), $\mathbf{z}_t$ should be $\boldsymbol{\mu}_{\mathbf{z}_t}$, i. e. the transition mean (notation).
2. Around fig. 2.1, 1D refers to one DoF (the rotations are in 2D space).