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

Anonymous authors
Paper under double-blind review

ABSTRACT

We solve the problem of 6-DoF localisation and 3D dense reconstruction in spatial environments as approximate Bayesian inference in a deep generative approach which combines learned with engineered models. This principled treatment of uncertainty and probabilistic inference overcomes the shortcoming of current state-of-the-art solutions to rely on heavily engineered, heterogeneous pipelines. Variational inference enables us to use neural networks for system identification, while a differentiable raycaster is used for the emission model. This 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 a state-of-the-art visual inertial odometry system. The applicability of the model to downstream tasks such as generative prediction and planning is investigated.

1 INTRODUCTION

In recent years, researchers have made considerable progress in solving the simultaneous localisation and mapping (SLAM) task (Durrant-Whyte et al., 1996). Modern SLAM and visual odometry systems (Mur-Artal & Tardós, 2017; Engel et al., 2018; Qin et al., 2018) have been shown to be an effective, real-time solution under moderate environment conditions. However, they are often built with assumptions optimised for different objectives, combining point selection front ends, feature extractors (e.g. (Rublee et al., 2011)), bundle adjustment (Triggs et al., 1999), pose-graph optimisation and BoW (Bag of Words) loop closure (Gálvez-López & Tardós, 2012). This makes the tight integration of learning challenging, which in turn limits how much a model can generalise and adapt. For the sake of efficiency, parameters are typically either hand-tuned or point-estimated and do not capture uncertainty. Current visual SLAM systems also normally focus on inference and do not additionally offer the predictive capabilities needed for planning and control.

In this work, we address SLAM in a probabilistic generative model (Koller & Friedman, 2009) that facilitates the direct integration of learned components, offers principled inference, generative predictions for control and uncertainty treatment. Building upon Mirchev et al. (2019), we propose a spatial solution that captures free 6-DoF agent movement and simulates the surrounding 3D world. We define our model with the following design goals in mind:

1. End-to-end learning with respect to a single objective.
2. Principled treatment of uncertainty in pose and map.
3. Expressive predictive distribution suitable for downstream tasks.

Our method implements agent localisation and dense environment reconstruction with Stochastic Gradient Variational Bayes (SGVB) (Kingma & Welling, 2014) and Bayes by Backprop (Blundell et al., 2015). All variables of interest are inferred as variational posteriors. Furthermore, integrating learned components in the form of neural networks is straightforward, as they are differentiable and comply with the Stochastic Gradient Descent (SGD) optimisation. To illustrate this, we propose a state-space model of agent dynamics learned from real flight data which we seamlessly include as a prior in the graphical model. We apply the overall model to the Blackbird dataset (Antonini et al., 2018): real drone flight trajectories with simulated realistic stereo images. The method currently runs offline and its localisation performance nears that of a state-of-the-art visual inertial odometry (VIO) system.

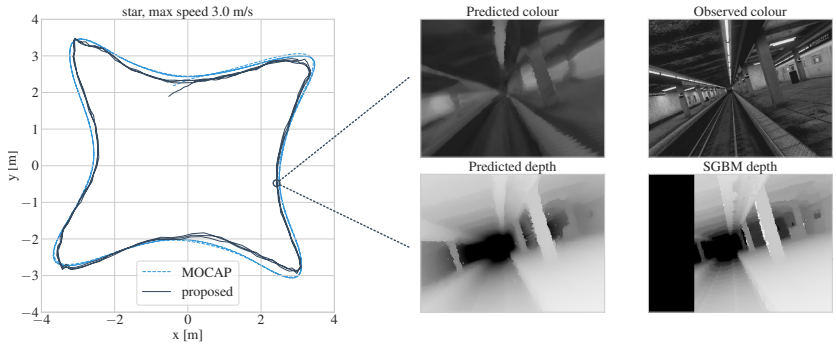


Figure 1: Illustration of the proposed quadcopter localisation and dense mapping. Left: top-down view of the localisaton estimate. Right: generative depth and colour reconstructions for one time step.

2 Related work

Traditional SLAM To keep exposition brief, we refer to (Cadena et al., 2016) for an overview and focus only on approaches that share properties with our model. A part of the map variable used in our solution can be interpreted as an occupancy grid map (Thrun et al., 2005), using a Gaussian distribution. We note that the map and raycasting models in this work are simple, leaving the discussion of more efficient implementations for future work. In KinectFusion (Newcombe et al., 2011a) and DTAM (Newcombe et al., 2011b) a virtual image is formed from a dense map for a given pose, resembling the emission in our generative model. 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.

Integration of learning and SLAM The idea of integrating learning with traditional SLAM methods has become more prevalent in recent research. D3VO (Yang et al., 2020) predicts depth and odometry with neural networks and successfully incorporates them in DSO (Engel et al., 2018), a well-established visual odometry system. In DeepTAM (Zhou et al., 2018) learning is used for odometry and depth estimation. NeuralSLAM (Zhang et al., 2017) uses learning for writing and reading from a spatial map. CodeSLAM and SceneCode (Bloesch et al., 2018; Zhi et al., 2019) focus on encoding geometric depth readings and semantic information in a low-dimensional latent variable. ∇SLAM (Jatavallabhula et al., 2019) explores ways to apply automatic differentiation to traditional engineered SLAM pipelines. Our proposed solution benefits from end-to-end differentiation as well, while putting the main emphasis on deep generative Bayesian modelling. DVBF-LM (Mirchev et al., 2019) is a deep generative model for 2D environments that addresses the tasks of SLAM, navigation and exploration. Our model shares the same probabilistic factorisation as DVBF-LM, but in contrast to our method the design choices in DVBF-LM do not allow real-world 3D modelling.

Variational Bayesian inference for SLAM 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. In (Fatemi et al., 2015) SLAM for radar data is addressed through traditional variational expectation maximisation. 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

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:

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

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 $\mathbf{z}_{1:T}$.

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:

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

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:

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

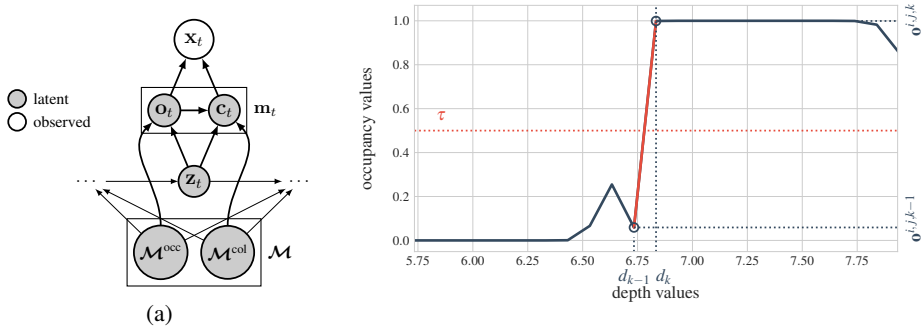
Here $\mathbf{T}(\mathbf{z}_t) \in \mathbb{SE}(3)$ denotes the rigid camera transformation defined by the current agent state $\mathbf{z}_t$ and i, j, k index the points lying inside the attended camera frustum.

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:

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



(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 $\mathbf{o}^{i,j,k}$.

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 $\mathbf{z}_t = (\boldsymbol{\lambda}_t, \boldsymbol{o}_t, \mathbf{z}_t^{\text{rest}}) \in \mathbb{R}^{d_{\mathbf{z}}}$. $\boldsymbol{\lambda}_t \in \mathbb{R}^3$ is the agent location in space. $\boldsymbol{o}_t \in \mathbb{H}^4$ 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

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

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.

[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.

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.

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 $\dot{\boldsymbol{o}}_t^{\text{imu}}$ (angular velocity) over time, which we choose to treat as conditional inputs $\mathbf{u}_t = (\ddot{\boldsymbol{\lambda}}_t^{\text{imu}}, \dot{\boldsymbol{o}}_t^{\text{imu}})$.

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 $\mathbf{z}_t = (\boldsymbol{\lambda}_t, \boldsymbol{o}_t, \dot{\boldsymbol{\lambda}}_t)$ then contains the location, orientation and linear velocity of the agent at every time step. The transition is defined as:

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

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

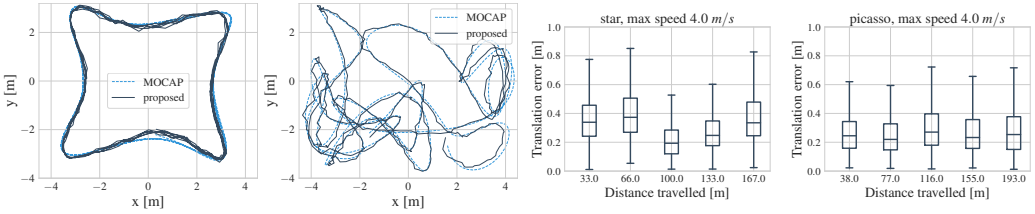
Learning a dynamics model Engineered models of the agent movement are often imperfect or not available. We therefore provide a method for learning a fully-probabilistic transition model from streams of prerecorded controls and agent pose observations, which we can then seamlessly include as a prior in 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 epistemic 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 $\hat{\boldsymbol{o}}_{1:T}$ 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 $\mathbf{u}_{1:T-1} = (\ddot{\boldsymbol{\lambda}}_{1:T-1}^{\text{imu}}, \dot{\boldsymbol{o}}_{1:T-1}^{\text{imu}}, \mathbf{u}_{1:T-1}^{\text{rpm}}, \mathbf{u}_{1:T-1}^{\text{pwm}})$. We define the generative state-space model:

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

The objective is to learn generative transition parameters $\boldsymbol{\theta}_T$, such that the marginal likelihood of observed agent poses $p_{\boldsymbol{\theta}_T}(\hat{\boldsymbol{\lambda}}_{1:T}, \hat{\boldsymbol{o}}_{1:T} \mid \mathbf{u}_{1:T-1})$ is maximised. The latent state is $\mathbf{z}_t = (\boldsymbol{\lambda}_t, \boldsymbol{o}_t, \dot{\boldsymbol{\lambda}}_t, \mathbf{z}_t^{\text{rest}})$, 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:

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

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.



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

Figure 3: (a) Top-down view of localisation for *star, forward yaw, max speed 4.0 m/s*. (b) Top-down view of localisation for *picasso, constant yaw, max speed 4.0 m/s*. Only a section of the trajectory is displayed for legibility. (c,d) Aggregated localisation error statistics for all steps up until a given travelled distance (x-axis) along the two trajectories, evaluated as per (Zhang & Scaramuzza, 2018).

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

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

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:

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

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

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

5 Experiments

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

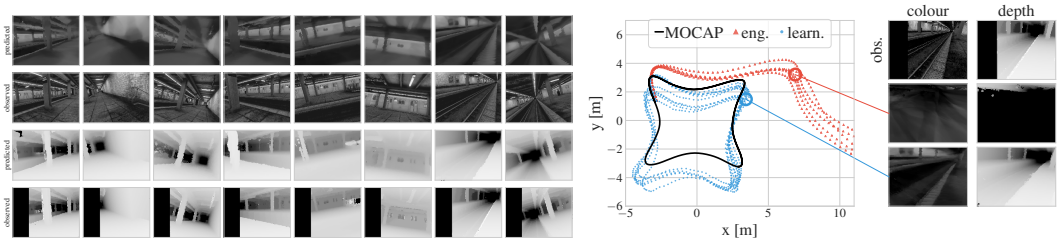
For evaluation, we use the Blackbird data set (Antonini et al., 2018). It consists of over ten hours of real flight data collected from a quadcopter equipped with an IMU sensor. The ground truth poses are recorded by a motion capture (MOCAP) system. For each trajectory, Blackbird contains realistic simulated 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). 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 B and C.

5.1 Agent localisation

We evaluate localisation performance of the complete spatial model following Zhang & Scaramuzza (2018). We report location and orientation RMSE averaged over the whole trajectory (absolute trajectory error, ATE), as well as relative translational RMSE for increasingly growing segments. Figure 3 shows two example localisation estimates for the fastest trajectories in the test set, where the quadcopter flies with up to 4.0 m/s. Over more than $150m$ the average translational error never exceeds $0.4m$ (cf. figs. 3c and 3d). The global map variable $\boldsymbol{\mathcal{M}}$ stabilises localisation and prevents drift from compounding over time. Empirically, we further found that modelling colour significantly helps with perceptual aliasing in the environment. In table 1 we compare the average localisation error of the proposed model over all steps to that of VIMO (Nisar et al., 2019) and VINS-MONO (Qin et al., 2018), two state-of-the-art VIO systems. We carry over the results from (Nisar et al., 2019). While the localisation estimates of our framework are not as accurate as those of the other two systems, we found that our solution had no difficulty with the fastest *star* test set trajectory, for which both VIO systems have been reported to fail without specifically being tuned for it. It should be noted, however, that both baselines run in real-time, whereas in its current form our method does not.

Table 1: Average localisation RMSE across all test trajectory time steps.

		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	0.110	0.092	0.088	0.102	0.045	0.044	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	-	-



(a) (b)

Figure 4: (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.

As for 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, except for two outliers—*picasso, max speed 4.0 m/s* and *star, max speed 2.0 m/s* (cf. table 1). 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 controls (cf. appendix A), for which the learned model does not generalise. Localisation beforehand is stable. Predictive performance is where the learned transition really pays off (section 5.4).

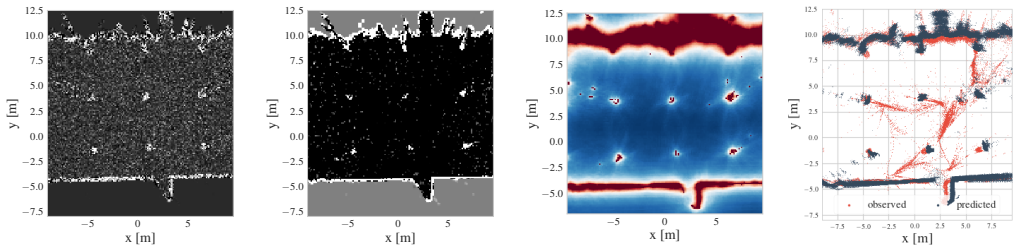
Overall, localisation is successful for all test trajectories and its accuracy is practical and approaching that of state-of-the-art VIO systems. At the same time, the proposed model is fully-probabilistic and flexible regarding the integration of learned components.

5.2 Dense geometric mapping

Next we focus on the reconstructed map, obtained as an approximate variational posterior $q_\phi(\boldsymbol{\mathcal{M}})$. We take the *NYC subway station* Blackbird environment as an example, in which the test set trajectories take place. Figure 4a shows model reconstructions generated along an example trajectory segment. The model can successfully generate both colour and depth based on the approximate map posterior $q_\phi(\boldsymbol{\mathcal{M}})$. Note that the true observations are not needed for this, as all of the information is recorded in $\boldsymbol{\mathcal{M}}^{\text{occ}}$ and $\boldsymbol{\mathcal{M}}^{\text{col}}$ 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 generated by the emission model. The inferred map correctly filters out wrong observations, as can be seen in the top-down point-cloud comparison in fig. 5d, noting the subway station columns.

5.3 Using maps for downstream tasks

Besides serving as an anchor to prevent localisation drift, a map can be used to define downstream navigation and exploration tasks. Since the map is modelled as a random variable, the approximate



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

Figure 5: 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).

posterior $q_\phi(\boldsymbol{\mathcal{M}})$ also gives us an uncertainty estimate, in this work only applying to $\boldsymbol{\mathcal{M}}^{occ}$. Figure 5a 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 dense map can also be used to construct navigation plans by computing a 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 5c shows $J(\mathbf{z})$ evaluated for all states along the considered 2D slice of the map. The collision cost is high near the walls and columns, and gradually drops off in the free space.

5.4 Generating future predictions

The proposed generative model lets us 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 planning and decision-making, and typically not readily available from modern visual SLAM systems. Figure 4b shows a representative example for predictive performance 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. Evaluated on 200 test set trajectories of length 100, the learned transition model significantly outperforms the engineered IMU integration, 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 integrating learned components in the proposed generative framework.

6 Conclusion

This work is the first to show that dense 3D mapping and 6-DoF localisation can be accomplished in a deep generative probabilistic framework using variational inference. The proposed spatial model is fully-differentiable and can be optimised with SGD. We further propose a probabilistic method for learning agent dynamics from prerecorded data. We demonstrate that deep priors can be integrated with ease in the overall spatial solution, using the learned transition as an example, which significantly boosts predictive performance. The proposed framework was used to model quadcopter flight data, exhibiting performance approaching that of a state-of-the-art VIO system and bearing promise for real-world applications. In the future, we will address the current model's speed limitations and move towards reasoning about spatial aspects that go beyond SLAM in the same framework.

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 dataset: A large-scale dataset for uav perception in aggressive flight. In *2018 International Symposium on Experimental Robotics (ISER)*, 2018.

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

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.

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.

Hugh Durrant-Whyte, David Rye, and Eduardo Nebot. Localization of autonomous guided vehicles. In *Robotics Research*, pp. 613–625, London, 1996. Springer London.

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

Maryam Fatemi, Lennart Svensson, Lars Hammarstrand, and Malin Lundgren. Variational bayesian em for slam. In *2015 IEEE 6th International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP)*, pp. 501–504. IEEE, 2015.

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.

Dorian Gálvez-López and J. D. Tardós. Bags of binary words for fast place recognition in image sequences. *IEEE Transactions on Robotics*, 28(5):1188–1197, October 2012.

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.

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.

Daphne Koller and Nir Friedman. *Probabilistic Graphical Models - Principles and Techniques*. MIT Press, 2009.

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, Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew J. Davison, Pushmeet Kohli, Jamie Shotton, Steve Hodges, and Andrew W. Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. *2011 10th IEEE International Symposium on Mixed and Augmented Reality*, pp. 127–136, 2011a.

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, 2011b.

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.

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.

E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. Orb: An efficient alternative to sift or surf. In *2011 International Conference on Computer Vision*, pp. 2564–2571, Nov 2011.

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.

Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press, 2005. ISBN 0262201623.

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.

Bill Triggs, Philip F McLauchlan, Richard I Hartley, and Andrew W Fitzgibbon. Bundle adjustment—a modern synthesis. In *International workshop on vision algorithms*, pp. 298–372. Springer, 1999.

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 CASE STUDY: DRONE LANDING

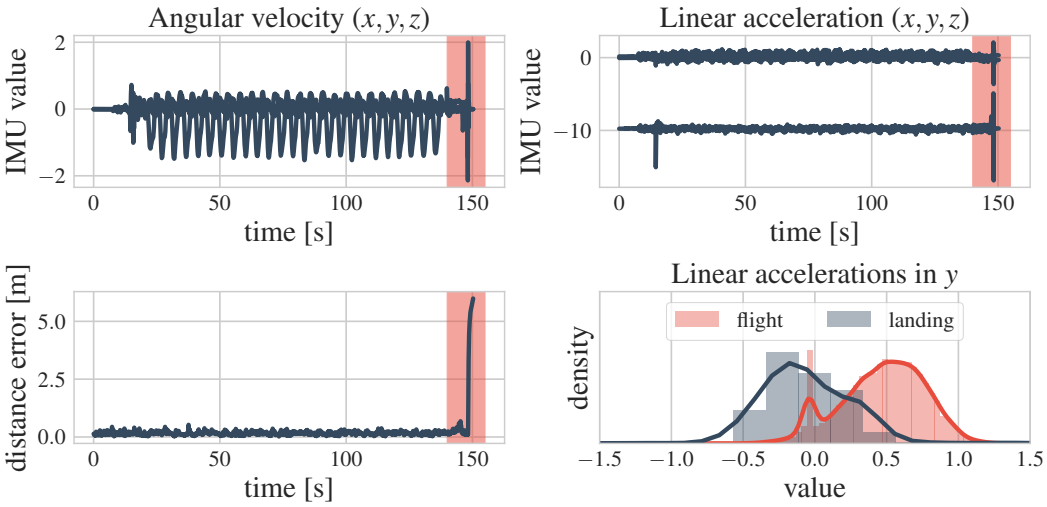


Figure 6: Illustration of the discussed failure landing of the quadcopter at the end of the two test set trajectories, taking *star, max speed 2.0 m/s* as an example. The problematic segment of the trajectory is marked with a red vertical band. Note the outlier controls at the end. Top left: angular 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.

B DATA DETAILS AND OVERALL SETUP

For all presented experiments we used the Blackbird data set (Antonini et al., 2018), which can be found here: `https://github.com/mit-fast/Blackbird-Dataset`. In the following we describe the exact pre-processing of the data.

B.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 in (Nisar et al., 2019) (considered as a baseline). Appendix F 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 C) were selected with random search. You can find details for the search ranges in appendix D.

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 in (Nisar et al., 2019). These trajectories take place in the *NYC subway station* environment.

B.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. 5d). The pixel values for the color images were normalized to be in the range $[0, 1]$.

C MODEL DETAILS

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

C.1 FULL SPATIAL MODEL

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

C.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 in (Mirchev et al., 2019), using chunks of 5 time steps for every gradient update, using 50 approximating state particles and applying the respective particle updates after every gradient step.

Adam (Kingma & Ba, 2014) is used to optimise the parameters $\boldsymbol{\phi}$ for the approximate posterior distribution $q_{\boldsymbol{\phi}}(\mathbf{z}_{1:T}, \boldsymbol{\mathcal{M}})$. Table 2a 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.

C.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 2b.

Table 2: Model details.

(a) 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

(b) Hyperparameters of the individual spatial model components.

Component	Parameter	Value
$\mathbf{z}_t$	init. value for $\boldsymbol{\mu}_t$	$\boldsymbol{\mu}_{t-1}$
	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

(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
$\hat{q}(\mathbf{z}_t \mid \hat{\boldsymbol{\lambda}}_{1:T}, \hat{\boldsymbol{o}}_{1:T}, \mathbf{u}_{1:T})$	RNN type	LSTM
	# RNN units	64

C.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.

C.1.4 Handling orientations

The portions $\boldsymbol{o}_t$ 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.

C.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 2b.

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 $[0, 20]$).

Table 3: HPS details.

(a) HPS ranges for the full spatial model.

Parameter	Range
reconstr. pixel count	$[100, 200, 500]$
$q(\mathcal{M}^{occ})$ init. value stddev	$[0.01, 0.1, 1.0]$
$p(\mathcal{M}^{occ})$ stddev	$[0.1, 1.0, 10.0]$
$\mathcal{M}^{occ}$ grid side	$[50, 100, 200]$
$\mathcal{M}^{occ}$ learning rate	$[0.01, 0.05, 0.001]$
$\mathcal{M}^{col}$ 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	$[0.0001, 0.001, 0.01]$
$\boldsymbol{o}$ stddev	$[0.0001, 0.001, 0.01]$
$\dot{\boldsymbol{\lambda}}$ stddev	$[0.0001, 0.001, 0.01]$

C.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 $\hat{\boldsymbol{o}}_t$, i.e. the mean of the emission $p(\hat{\boldsymbol{o}}_t \mid \mathbf{z}_t)$ is a rotation matrix constructed from the quaternion $\boldsymbol{o}_t$ 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.

C.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

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

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. 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. Additionally, $\boldsymbol{\sigma}_T$ (the diagonal for the assumed Gaussian) was selected baed on a hyperparameter search using 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.

D Hyperparameter search and execution details

All models are implemented in the TensorFlow framework (Abadi et al., 2016), making use of automatic differentiation to optimise model parameters. 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.

D.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.

D.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.

D.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.

E FURTHER INFERENCE EXAMPLES

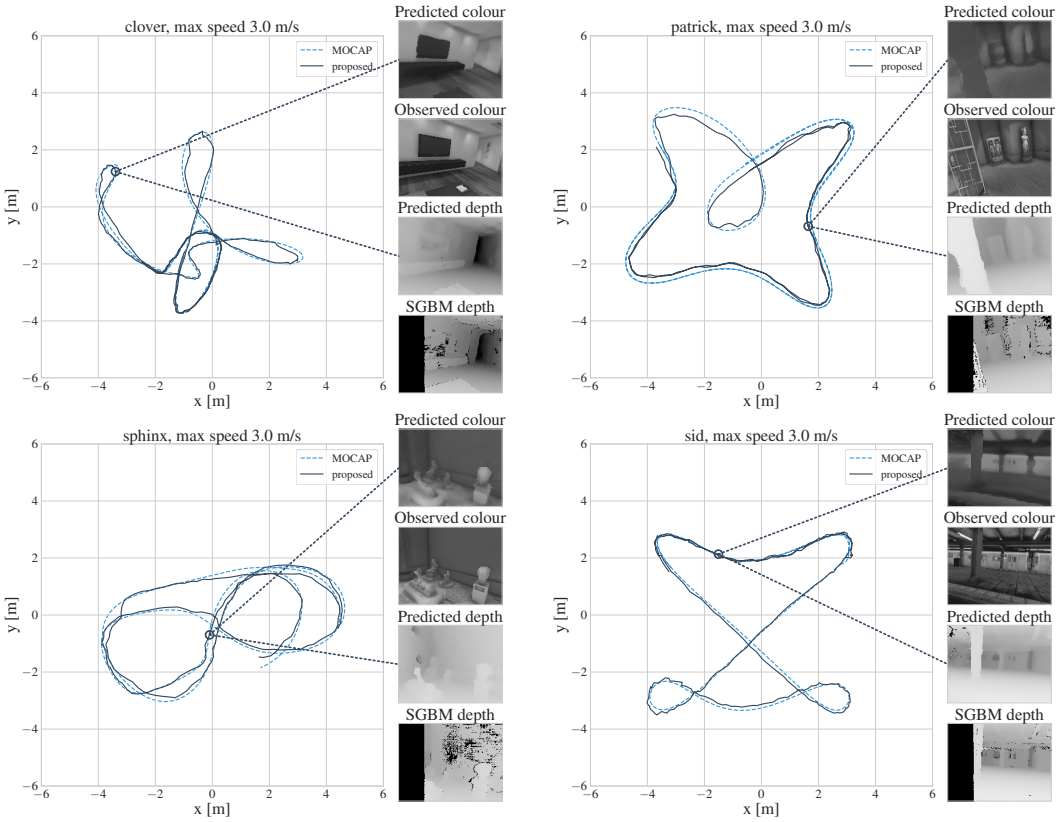


Figure 7: Examples of inference in the full spatial model, expressed in localisation and mapping for segments of the following trajectories: *clover, forward yaw, max speed 3.0 m/s*, *patrick, forward yaw, max speed 3.0 m/s*, *sphinx, forward yaw, max speed 3.0 m/s* and *sid, forward yaw, max speed 3.0 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.

F 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	833	83.3
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