Original Article

The International Journal of Robotics Research
2023, Vol. 0(0) 1–33
© The Author(s) 2023
Article reuse guidelines:
sagepub.com/journals-permissions
DOI: 10.1177/02783649231218720
journals.sagepub.com/home/ijr
Sage

Uncertainty-aware visually-attentive navigation using deep neural networks

Huan Nguyen[1], Rasmus Andersen[2], Evangelos Boukas[2] and Kostas Alexis[1]

Abstract
Autonomous navigation and information gathering in challenging environments are demanding since the robot's sensors may be susceptible to non-negligible noise, its localization and mapping may be subject to significant uncertainty and drift, and performing collision-checking or evaluating utility functions using a map often requires high computational costs. We propose a learning-based method to efficiently tackle this problem without relying on a map of the environment or the robot's position. Our method utilizes a Collision Prediction Network (CPN) for predicting the collision scores of a set of action sequences, and an Information gain Prediction Network (IPN) for estimating their associated information gain. Both networks assume access to a) the depth image (CPN) or the depth image and the detection mask from any visual method (IPN), b) the robot's partial state (including its linear velocities, z-axis angular velocity, and roll/pitch angles), and c) a library of action sequences. Specifically, the CPN accounts for the estimation uncertainty of the robot's partial state and the neural network's epistemic uncertainty by using the Unscented Transform and an ensemble of neural networks. The outputs of the networks are combined with a goal vector to identify the next-best-action sequence. Simulation studies demonstrate the method's robustness against noisy robot velocity estimates and depth images, alongside its advantages compared to state-of-the-art methods and baselines in (visually-attentive) navigation tasks. Lastly, multiple real-world experiments are presented, including safe flights at 2.5 m/s in a cluttered corridor, and missions inside a dense forest alongside visually-attentive navigation in industrial and university buildings.

Keywords
Autonomous navigation, deep neural networks, uncertainty-aware navigation, visually-attentive navigation, aerial robots

1. Introduction

Recent breakthroughs in the field of aerial robotics have enabled their widespread adoption in various applications including in subterranean exploration, construction, agriculture and forestry Tranzatto et al. (2022); Loquercio et al. (2021); Petracek et al. (2021); Zhou and Gheisari (2018); Kulbacki et al. (2018). Extremely agile navigation of quadrotors has been demonstrated recently in the context of drone-racing competitions Foehn et al. (2022); Wagter et al. (2021) or in broader field tests Loquercio et al. (2021); Kaufmann et al. (2020). However, the task of autonomous 3D navigation and efficient information gathering in challenging, geometrically complex, perceptually-degraded environments remains demanding since a) the robot's sensors may be susceptible to non-negligible noise, b) the onboard localization and mapping may be subject to significant uncertainty and drift Ebadi et al. (2022); Cadena et al. (2016), and c) performing collision-checking or evaluating utility functions for high-quality information sampling using a map often results in high computational cost Schmid et al. (2020).

While map-based methods require building a consistent map of the environment, for example via octrees Hornung et al. (2013), TSDFs Oleynikova et al. (2017); Han et al. (2019), or VDB structures Museth (2013), map-less methods follow another approach by only relying on a single observation or a (spatio)-temporal window of recent observations possibly combined with high-level commands from the operator or path planners. Traditional map-less approaches utilize various data structures such as kd-trees Florence et al. (2018); Gao et al. (2019), 3D circular buffers Usenko et al. (2017), rectangular pyramids Bucki et al. (2020) or directly use disparity images Matthies et al. (2014) for fast collision

[1]Autonomous Robots Lab, Department of Engineering Cybernetics, Norwegian University of Science and Technology, Trondheim, Norway
[2]Department of Electrical and Photonics Engineering, Technical University of Denmark, Kongens Lyngby, Denmark

Corresponding author:
Huan Nguyen, Autonomous Robots Lab., Department of Engineering Cybernetics, Norwegian University of Science and Technology, Høgskoleringen 1, Trondheim 7034, Norway.
Email: dinh.h.nguyen@ntnu.no

checking. Recent work on data-driven learning offers another promising pathway towards low-latency navigation by exploiting both the parallel computing capabilities of GPUs Kew et al. (2021); Kahn et al. (2021a) and the universal approximation power of deep neural networks Tabuada and Gharesifard (2022) to directly map raw sensor observations to control actions, thus bypassing the need for separate perception, mapping, and planning modules Loquercio et al. (2021); Kaufmann et al. (2020).

Our work falls into this latter category of approaches as we aim to develop an efficient collision-free and information-gathering navigation method that does not rely on the global map or position information of the robot. Learning-based methods can offer low computation costs, however, only a few works discuss the effects of different uncertainties on the robot's navigation capabilities. In turn, modern deep neural networks are notoriously famous for giving unjustifiably overconfident predictions Guo et al. (2017); Abdar et al. (2021). Hence, it is essential to handle uncertainty in neural network prediction properly in safety-critical applications.

Simultaneously, we further aim to address the challenge of combining such map-less safe navigation with efficient sampling of information about interesting areas in the environment. Relevant works in the literature of informative path planning have been various Hollinger and Sukhatme (2014); Forssen et al. (2008); Dang et al. (2018); Popovic et al. (2018), but usually require building maps of the environments and tend to be computationally expensive, which hinders their deployability or the quality of the achieved solution given the limited computing resources onboard most aerial robots.

Responding to the combined problem of map-less collision-free and visually-attentive navigation, we propose a duo of new methods, called "Attentive ORACLE" (A-ORACLE) and "ORACLE." Attentive ORACLE trains two deep neural networks: a Collision Prediction Network for predicting uncertainty-aware collision costs and an Information gain Prediction Network for estimating information gain values of a set of action sequences in a Motion Primitives Library. While the Collision Prediction Network utilizes only depth data—alongside a partial robot state that does not involve its position—and is built and expanded upon our earlier work Nguyen et al. (2022), the new Information gain Prediction Network utilizes both depth images and visual detection results and is trained with the information gain labels provided by an offline expert that relies on a volumetric mapping representation of the environments. Given the predictions from the two networks, in addition to a unit goal vector given by any high-level planner, the method derives the safe (collision-free) motion primitive having the highest information gain and leading towards a desired direction. This is then commanded and executed in a receding horizon fashion. It is noted that when the Information gain Prediction Network is not engaged, the method reduces to 3D ORACLE which ensures safe uncertainty-aware map-less navigation.

Compared to our previous work on ORACLE Nguyen et al. (2022), this manuscript represents a major extension and claims a set of contributions as outlined below.

First, it is about introducing visual attention-aware navigation into the framework through the Information gain Prediction Network which in turn allows to combine safe navigation with implicit information sampling (*contribution 1*). Second, we present a significant upgrade of Nguyen et al. (2022) as the new method a) extends the previous one from 2D to 3D navigation enabling safe flight in complex and cluttered scenes without the need for a map or position estimates (*contribution 2*) and further b) utilizes a deep ensembles method, called Deep Ensembles Lakshminarayanan et al. (2017), instead of Monte Carlo dropout Gal and Ghahramani (2016) for the neural network's epistemic uncertainty estimation thus offering performance robustness against sources of noise (*contribution 3*).

To realize these goals, ORACLE and A-ORACLE employ a novel supervised learning paradigm for collision prediction and assessment of the informativeness of candidate motion primitives where both the epistemic and aleatoric uncertainty are accounted for collision prediction through the Deep Ensembles and the Unscented Transform over the robot's partial state covariance (*contribution 4*). Finally, a new set of simulations and real-world experiments are conducted to verify the proposed uncertainty-aware and visually-attentive framework. The method is thus extensively evaluated including successful sim-to-real transfer, an ablation study, and comparative analysis against other methods of the state-of-the-art highlighting its advantages (*performance claim*).

Specifically, more thorough simulation studies are conducted to demonstrate the performance of our method against noisy inputs including the robot's velocity estimate and the depth image (linked to contributions 2–4). An ablation study regarding the role of the Deep Ensembles is also conducted in simulation (linked to contribution 3), alongside a comparative analysis of ORACLE with the work in Loquercio et al. (2021) (linked to contributions 2–4). Moreover, simulation results with different sources of visual attention are performed to illustrate the advantages of our visually-attentive navigation method compared to other baselines and an appropriately modified version of the informative planning work in Schmid et al. (2020) (linked to contribution 1). Finally, real-world experiments, a subset of which is depicted in Figure 1, including safe flights with a reference forward speed of 2.5 m/s in a cluttered environment (linked to contributions 2–4), autonomous missions in a highly cluttered forest (linked to contributions 2–4), and visual attention-aware navigation in industrial and campus buildings (linked to contribution 1) are also presented. As demonstrated and analyzed, the method not only utilizes partial state information and transfers well to the real system but also presents robustness to state uncertainty and exteroceptive sensor noise that is unseen during training (contributions 2–4). For the remainder of this manuscript, the 3D ORACLE method which ensures safe uncertainty-aware map-less navigation is simply called ORACLE.

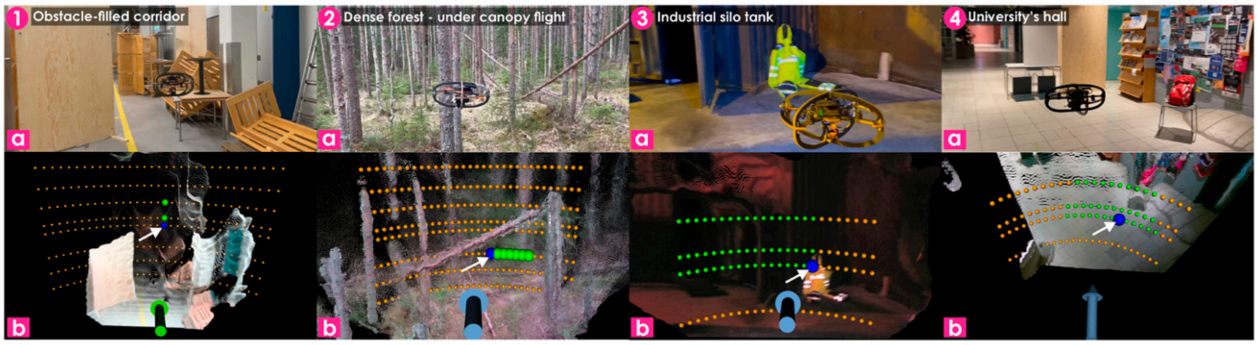
1 Obstacle-filled corridor
2 Dense forest - under canopy flight
3 Industrial silo tank
4 University's hall
a
b

Figure 1. Instances of real-world experiments demonstrating the proposed methods, including safe flights with a reference forward speed of 2.5 m/s in a cluttered corridor (1a), under canopy flights inside a dense forest (2a) and visual attention-aware navigation in an industrial silo tank (3a) and a university's hall (4a). The bottom row (b) illustrates prediction results from the method where the spherical markers correspond to the estimated trajectory endpoints of a set of action sequences, while among them green markers illustrate the subset of safe action sequences (with orange being unsafe), and the blue marker with an arrow corresponds to the selected action sequence.

The remainder of this paper is organized as follows: Section 2 presents related work, followed by the problem statement in Section 3. The proposed method is presented in Section 4 while evaluation studies are detailed in Section 5, followed by conclusions in Section 6.

2. Related work

A set of contributions in a) learning-based navigation, b) uncertainty-aware navigation and modeling uncertainty in deep neural networks, and c) visually-attentive navigation relate to this work.

2.1. Learning-based navigation

In recent years, a large amount of work has been devoted to harnessing the power of deep learning in various ways to solve the problem of autonomous navigation. A group of work focuses on solving the global path planning problem efficiently in which a top-down image or point cloud of the whole environment is provided a priori Ichter and Pavone (2019); Srinivas et al. (2018); Qureshi et al. (2021). However, in this work, we focus on the setting where the global map of the environment is not available and the robot needs to navigate in a collision-free manner given only local onboard observations. Several works utilize neural networks to solve the local navigation problem. The authors in Loquercio et al. (2021) and Tolani et al. (2021) use imitation learning to generate collision-free smooth trajectories which are then tracked by model-based controllers. Nevertheless, position information may not be reliable in many perceptually-degraded environments. On the other hand, other low-level commands (velocity/steering angle, acceleration, or angular velocity/thrust commands) can be inferred by deep navigation policies which can be trained by various schemes including reinforcement learning Francis et al. (2020), supervised learning where ground-truth commands are readily available in a driving dataset Loquercio et al. (2018), provided by human operators Shah and Levine (2022) or demonstrated by an expert Kaufmann et al. (2020), and self-supervised learning Gandhi et al. (2017); Kahn et al. (2021a); Kahn et al. (2021b). In this work, we choose to use velocity/steering angle commands to allow the robot to not rely on reliable position estimation.

A body of work utilizes deep learning to derive interpretable maps, which are then used by classical planners to plan collision-free paths Wang et al. (2021); Frey et al. (2022); Castro et al. (2023); Zeng et al. (2019). Instead of learning classical map representations from raw observation data, many works present methods to encode raw sensor data into an implicit latent vector Hoeller et al. (2021); Dugas et al. (2021); Ichter and Pavone (2019); Srinivas et al. (2018); Qureshi et al. (2021). Control actions can then be inferred through these latent representations thus offering the benefit of low-latency navigation Loquercio et al. (2021), utilizing the computing capability of modern GPU for efficient deep neural network's inference. The latent vectors in our work are learned to implicitly encode information about the environments as well as the robot's partial state to predict collision events and information gains at future time steps.

Like other works that apply deep learning to score each motion primitive in a discrete set (Veer and Majumdar, 2020); Kahn et al., 2021a); Kahn et al., 2021b), our work also falls into this category. However, we explicitly consider the effects of uncertainties when scoring each motion primitive.

2.2. Modeling uncertainty in deep neural networks and uncertainty-aware learning-based navigation

When using deep neural networks for making predictions, there are two kinds of uncertainty that need to be considered: a) aleatoric uncertainty which captures inherent and irreducible data noise and b) epistemic uncertainty which accounts for model uncertainty and cannot be negligible for out-of-distribution inputs Kendall and Gal (2017). Two main methods for estimating epistemic uncertainty that can be applied to large neural networks and large datasets are a) approximate Bayesian inference and b) ensembling (Gustafsson et al., 2020); Abdar et al., 2021). Monte Carlo (MC) dropout (Gal and Ghahramani, 2016) is an approximate Bayesian inference method that is widely used in deep learning due to its simplicity and efficiency. On the other hand, ensembling methods use an ensemble of neural networks to derive the output uncertainty. Empirically, studies in Gustafsson et al. (2020); Ovadia et al. (2019) conclude that Deep Ensembles (Lakshminarayanan et al., 2017), an ensemble method that assembles different neural networks trained with different initialization weights and shuffling of the same dataset, can provide more reliable and useful uncertainty estimates than MC dropout.

Additionally, methods for propagating aleatoric uncertainty from the input to the output of the neural network can be classified into two main groups: layer-wise and entire-network uncertainty propagation (Abdelaziz et al., 2015). Though layer-wise uncertainty propagation methods (Ghosh et al., 2016); Hernández-Lobato and Adams, 2015); Gast and Roth, 2018); Wang et al., 2016); Astudillo and Neto, 2011) can offer the distributions of hidden layers, they often require modification to the original network during the training or inference phases. Moreover, Abdelaziz et al. (2015); Chua et al. (2018) demonstrate that entire-network uncertainty propagation through particle-based propagation methods such as the Unscented Transform Julier and Uhlmann (1997) can be competitive in terms of accuracy and computation.

As demonstrated in traditional belief space planning methods (Bry and Roy, 2011); Agha-mohammadi et al., 2018); Sun et al., 2021), modeling uncertainty is vital to achieving safe navigation in challenging environments where the state of the robot or the map of the environment can be highly uncertain. Most existing works applying deep neural networks for autonomous navigation account for epistemic uncertainty only, for instance, by using autoencoders Richter and Roy (2017), dropout and bootstrap Kahn et al. (2017); Georgakis et al. (2022); Lütjens et al. (2019), 2D spatial dropout Amini et al. (2017), evidential fusion Liu et al. (2021). One of the exceptions is Loquercio et al. (2020) which accounts for both uncertainties in the image data using Assumed Density Filtering Ghosh et al. (2016) and epistemic uncertainty using MC dropout. Chua et al. (2018) propose to use particle propagation to estimate the aleatoric uncertainty and Deep Ensembles to derive the epistemic uncertainty. However, this work focuses on the different problem of control of robot dynamics as opposed to the task of safe and attentive flight exploiting exteroceptive sensor data, employs reinforcement learning instead of supervised learning, is not verified onboard a robot for autonomous navigation and thus does not address the sim-to-real challenge, especially with high-dimensional data. Moreover, in Chua et al. (2018), to predict future plausible state trajectories, all the state particles are initially created from the same current state since the aleatoric uncertainty considered is the inherent stochasticities of the dynamics model (e.g., process noise). Our work considers the aleatoric uncertainty of the system as the prediction uncertainty due to the noisy robot's partial state estimates. Thus, our particles are chosen as the sigma points around the current robot's partial state estimate, given by the Unscented Transform.

2.3. Visually-attentive navigation

Our problem is also closely related to the informative path planning (IPP) problem where the robots need to find trajectories to maximize information gathered along the trajectory, given a constrained budget of time, fuel, or energy (Hollinger and Sukhatme, 2014). Traditionally, the IPP problem can be tackled by performing coverage path planning and viewpoint selection on the pre-built map of the environment (Hollinger and Sukhatme, 2014; Forssen et al., 2008) or adapting the paths online based on the latest map to focus on the areas of interest (Dang et al., 2018; Popovic et al., 2018; Schmid et al., 2020). Learning-based methods have been applied to solve the IPP problem efficiently. While Choudhury et al. (2017) present an imitation learning approach where an agent imitates an "information-gathering" planner with full information about the world map, other works in Niroui et al. (2019); Chen et al. (2020); Zhu et al. (2018) train reinforcement learning agents to output the next frontiers to visit for autonomous exploration. Furthermore, the works in Tao et al. (2023); Georgakis et al. (2022) use neural networks to predict the occupancy maps and calculate the informative trajectories to reduce the uncertainties of the map.

The step of evaluating the information gains for all the trajectories, however, can be time-consuming Schmid et al. (2020). Accordingly, several works have proposed methods to reduce the computational time of the information gain calculation step, either by subsampling ray casting (Selin et al., 2019; Oleynikova et al., 2018; Zhou et al., 2021), avoiding redundant voxel checks (Zhou et al., 2021; Millane et al., 2018; Schmid et al., 2020), or calculating an analytical formula for a specific metric (Zhang et al., 2020). Rckin et al. (2022) combined tree search with offline-learned neural network predicting informative sensing actions. The method, however, requires the robot's position and a cost feature map input to the network which relies on the assumption that the robot's underlying localization and mapping are accurate.

Our work proposes to efficiently approximate an information gain formula tailored to obtaining high-quality observations of interesting areas with a neural network. The prediction is then combined with an uncertainty-aware Collision Prediction Network, exploiting the Unscented Transform and Deep Ensembles, alongside input from a high-level planner to achieve efficient uncertainty-aware visually-attentive navigation without relying on a map of the environment or the robot's position information.

3. Problem formulation and notations

The problem considered in this work is that of autonomous uncertainty-aware and visually-attentive aerial robot navigation. The method explicitly assumes no access to the map of the environment (neither offline nor online) and no information for the robot position but only a partial state estimate of the robot combined with the real-time depth data and a 2D detection mask representing the interestingness of every region within an angle- and range-constrained sensor frustum. We assume that there is a global planner providing the 3D unit goal vector $\mathbf{n}_t^g$ to the robot (e.g., for exploration or inspection), possibly by having access to a topological map of the environment. Given the above, the focus is on designing a local safe navigation planner to head towards the goal vector and not only avoid obstacles but simultaneously pay attention to interesting areas.

In the following sections, we will denote ${}^{\mathcal{F}}\mathbf{b}$ as vector $\mathbf{b}$ expressed in frame $\mathcal{F}$ and $[\mathbf{b}_x, \mathbf{b}_y, \mathbf{b}_z]$ as the projected components of vector $\mathbf{b}$ in x, y, z axes of the frame that $\mathbf{b}$ is expressed in. We also use $\xi(\tau)$ to represent the value of vector or scalar variable ξ at continuous time τ, and $\xi_t = \xi(t\Delta_t)$ to indicate the value of ξ at discrete time step t, where Δ_t is the time step duration. Let $\mathcal{B}, \mathcal{V}$ be the body frame and vehicle (or yaw-rotated inertial) frame of the robot, respectively, $\mathbf{o}_t$ the current depth image, $\boldsymbol{\mu}_t$ the current detection mask, coming from any visual detection methods, in which each pixel encodes the interestingness of the corresponding pixel in $\mathbf{o}_t$ and has the value between 0 (uninteresting pixel) and 1 (the most interesting pixel), and $\mathbf{s}_t = [\mathbf{v}_t^T, \omega_t, \phi_t, \theta_t]^T$ the estimated partial state of the robot consisting of a) the 3D velocity in $\mathcal{V}$ ($\mathbf{v}_t = [\mathbf{v}_{t,x}, \mathbf{v}_{t,y}, \mathbf{v}_{t,z}]^T \in \mathbb{R}^{3\times1}$), b) the angular velocity around the z-axis of $\mathcal{B}$ (ω_t), as well as c) the roll (ϕ_t) and pitch angles (θ_t). Let $\boldsymbol{\Sigma}_t$ denote the covariance matrix of the estimated robot's partial state, $\mathbf{n}_t^g$ the 3D unit goal vector—expressed in $\mathcal{V}$—given by the global planner, ψ_t the current yaw angle of the robot, and $\mathbf{a}_{t:t+H} = [\mathbf{a}_t, \mathbf{a}_{t+1}, \ldots, \mathbf{a}_{t+H-1}]$ an action sequence having length H where the action at time step $t+i$ ($i = 0, \ldots, H-1$) includes a) the reference speed expressed in the vehicle frame $\mathbf{v}_{t+i}^r$ and b) the steering angle (δ_{t+i}^r) from the current yaw angle of the robot (ψ_t), such that $\mathbf{a}_{t+i} = [(\mathbf{v}_{t+i}^r)^T, \delta_{t+i}^r]^T$. The exact problem considered is then formulated as that of finding an optimized collision-free sequence of actions $\mathbf{a}_{t:t+H}$ enabling the robot to safely navigate along the goal vector $\mathbf{n}_t^g$ and simultaneously "gather" additional information gain about interesting areas in the environment given ($\mathbf{o}_t$, $\mathbf{s}_t$, $\boldsymbol{\mu}_t$, $\boldsymbol{\Sigma}_t$).

4. Proposed approach

To satisfy the two objectives of collision-free navigation and information sampling, we design two deep neural networks to efficiently estimate the ground-truth collision score c^{col} and the information gain g for each action sequence, namely, the "Collision Prediction Network (CPN)" and "Information gain Prediction Network (IPN)," respectively. Both networks assume access to a) either the depth image (CPN) or the stacked matrix of the current depth image and the detection mask (IPN), alongside b) the estimates of the robot's linear velocities, z-axis angular velocity, and roll/pitch angles, as well as c) candidate action sequences from a Motion Primitives Library (MPL). The choice of using the MPL instead of regressing the action (Francis et al., 2020) or trajectory (Tolani et al., 2021) directly from the input is based on the observations that MPL is a multi-modal output by construction, which is vital for the collision-avoidance task (Loquercio et al., 2021). Attentive ORACLE identifies the next-best-sequence of actions, specifically 3D velocity-steering commands over certain time periods, that ensure that the system is navigating towards where the unit goal vector is pointing, while not only avoiding the obstacles but also gathering information about interesting areas in the environment. The first action of this sequence is executed by the robot, while the process continues iteratively in a receding horizon manner. Importantly, the "global" goal vector may be provided by any global planner thus allowing Attentive ORACLE to be combined with any high-level planning framework Dang et al. (2020); Galceran and Carreras (2013); Kim and Ostrowski (2003); Achtelik et al. (2014). Figure 2 provides an overview of the architecture of the method. It is noted that the CPN accounts both for a) the estimation uncertainty of the robot's partial state and b) the neural network's epistemic uncertainty, and thus considers sigma points given the partial state estimate and its covariance, while simultaneously using an ensemble of neural networks to evaluate the collision scores. IPN is not concerned with the uncertainty of the partial state estimate and the epistemic uncertainty for computational reasons.

4.1. Velocity-steering angle motion primitives library

For each candidate action sequence in the MPL, the commands at each time step have the same velocity in the corresponding $\mathcal{V}$ with zero velocity in the y-axis and the same steering angle from the yaw angle of the robot at the beginning of the action sequence, ψ_t. The steering angle is sampled within the field-of-view (FOV) of the depth sensor. Specifically, we have $\mathbf{a}_{t:t+H} = [(\mathbf{v}^r)^T, \delta^r, (\mathbf{v}^r)^T, \delta^r, \ldots, (\mathbf{v}^r)^T, \delta^r]^T \in \mathbb{R}^{4\times H}$. We assume the xOz-plane of $\mathcal{B}$ and the yOz-plane of the depth camera frame, $\mathcal{C}$, are identical. The x, y, z-axes of $\mathcal{B}$ point to the front, left of the robot, and upward, respectively. The x, y, z-axes of $\mathcal{C}$ point to the left of the depth camera, downward, and to the front of the depth camera, respectively. We denote $[F_h, F_v]$ as the FOV, $d_{\max}$ the maximum range of the depth camera, and θ_c the rotation angle of $\mathcal{C}$ around the y-axis of $\mathcal{B}$. For each candidate action sequence, we have:

$$\mathbf{v}_{t+i}^r = \mathbf{v}^r, \delta_{t+i}^r = \delta^r \; (i = 0, \ldots, H-1) \tag{1}$$

$$\delta^r \in \left[-\frac{F_h}{2}, \frac{F_h}{2}\right] \tag{2}$$

$$\mathbf{v}_x^r H \Delta_t \le d_{max} \tag{3}$$

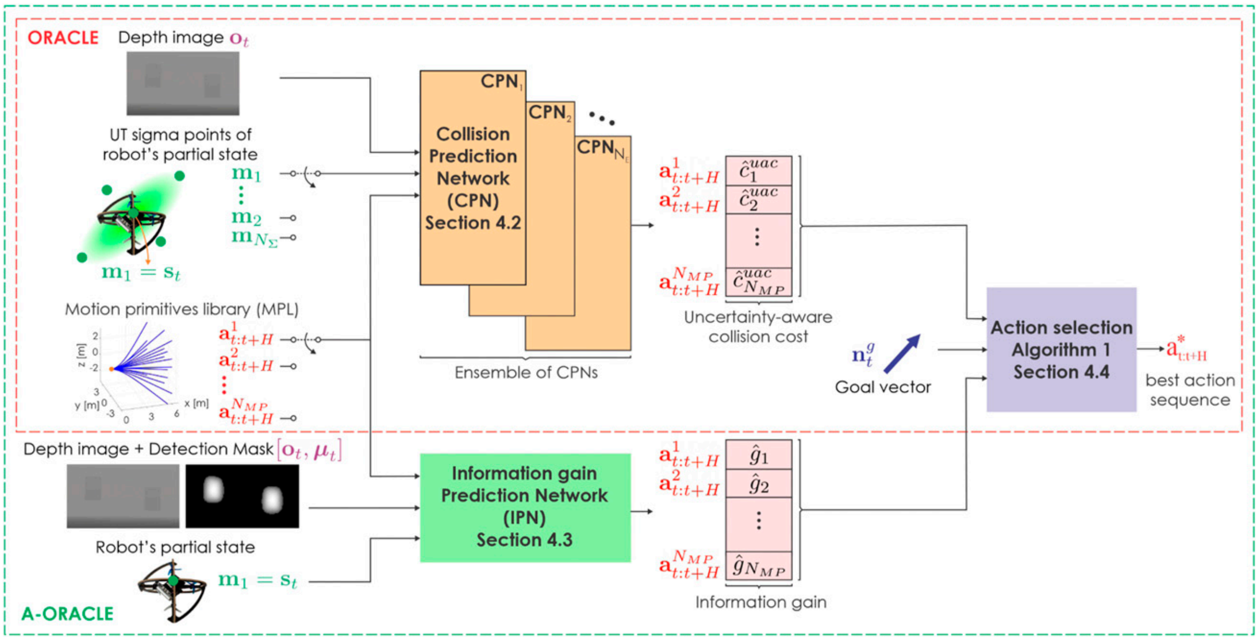
ORACLE
Depth image $\mathbf{o}_t$
UT sigma points of robot's partial state
$\mathbf{m}_1$
$\mathbf{m}_2$
$\mathbf{m}_{N_\Sigma}$
$\mathbf{m}_1 = \mathbf{s}_t$
Motion primitives library (MPL)
$\mathbf{a}^1_{t:t+H}$
$\mathbf{a}^2_{t:t+H}$
$\mathbf{a}^{N_{MP}}_{t:t+H}$
CPN$_1$
Collision Prediction Network (CPN) Section 4.2
CPN$_2$
CPN$_{N_E}$
Ensemble of CPNs
$\hat{c}^{uac}_1$
$\hat{c}^{uac}_2$
$\hat{c}^{uac}_{N_{MP}}$
Uncertainty-aware collision cost
$\mathbf{n}^g_t$
Goal vector
Action selection Algorithm 1 Section 4.4
$\mathbf{a}^*_{t:t+H}$
best action sequence
Depth image + Detection Mask $[\mathbf{o}_t, \boldsymbol{\mu}_t]$
Robot's partial state
$\mathbf{m}_1 = \mathbf{s}_t$
Information gain Prediction Network (IPN) Section 4.3
$\hat{g}_1$
$\hat{g}_2$
$\hat{g}_{N_{MP}}$
Information gain
A-ORACLE

Figure 2. Overview of the algorithmic architecture of Attentive ORACLE (A-ORACLE). We design two deep neural networks to efficiently estimate the uncertainty-aware collision score and the information gains for multiple action sequences, namely the "Collision Prediction Network (CPN)" and "Information gain Prediction Network (IPN)", respectively. Both networks assume access to (a) either the depth image (CPN) or the stacked matrix of the current depth image and the detection mask (IPN), alongside, (b) the estimates of the robot's linear velocities, z-axis angular velocity, and roll/pitch angles, and (c) candidate action sequences in a Motion Primitives Library (MPL). Notably, CPN utilizes $\mathbf{m}_1$ representing the current mean value of $\mathbf{s}_t$ and $\mathbf{m}_2 \ldots \mathbf{m}_{N_\Sigma}$ representing the remaining sigma points of the Unscented Transform to account for the uncertainty in the robot's partial state estimate, while an ensemble of CPNs is used to account for the epistemic uncertainty of the neural network model. The predicted uncertainty-aware collision cost $\hat{c}^{uac}$, information gain $\hat{g}$, and a unit goal vector $\mathbf{n}^g_t$ given by a high-level global planner are used to choose the optimal action sequence to be executed in a receding horizon fashion. When the IPN is not engaged, the method reduces to ORACLE method which ensures safe uncertainty-aware map-less navigation.

$$\mathbf{v}^r_y = 0 \tag{4}$$

$$\mathbf{v}^r_z = \mathbf{v}^r_x \tan(\beta), \beta \in \left[-\frac{F_v}{2} - \theta_t - \theta_c, \frac{F_v}{2} - \theta_t - \theta_c \right] \tag{5}$$

We denote $\mathbf{a}^k_{t:t+H}$ as the k^{th} action sequence in the MPL. As opposed to other MPL-based methods that sample the position space Veer and Majumdar (2020); Bucki et al. (2020), our planned sequences do not include the robot's position space but remain in velocity/steering angle space, similar to those proposed in Lopez and How (2017); Goel et al. (2021), as the underlying assumption is that ORACLE does not have—or does not need to have—access to a position estimate. The open-loop trajectories of the robot can be estimated by integrating the low-order approximation of the robot's low-level closed-loop dynamics model:

$$\begin{cases} \dot{\mathbf{v}}_j(\tau) = \dfrac{1}{\mathcal{T}_{v,j}} \left(\mathcal{K}_{v,j} \mathbf{v}^r_j(\tau) - \mathbf{v}_j(\tau) \right), j = x, y, z \\ \dot{\delta}^r(\tau) = \mathcal{K}_{p,\psi}(\delta^r(\tau) - \delta(\tau)) \\ \ddot{\delta}(\tau) = \dfrac{1}{\mathcal{T}_{\dot{\psi}}} (\mathcal{K}_{\dot{\psi}} \dot{\delta}^r(\tau) - \omega(\tau)) \end{cases} \tag{6}$$

where $\mathcal{T}_{v,j}, \mathcal{K}_{v,j}$ $(j = x, y, z)$ are the time constant and gain of the velocity controller for the velocity component in j-axis of $\mathcal{V}$, respectively, $\delta(\tau) = \psi(\tau) - \psi_t$ is the robot's current relative yaw angle with respect to the $\mathcal{V}$-frame at time step t when the first action in the action sequence is applied, $\mathcal{K}_{p,\psi}$ is the gain of the Proportional controller for the yaw angle of the robot which sends the yaw-rate command, $\dot{\psi}^r_t$ or $\dot{\delta}^r_t$, to the low-level yaw-rate controller having time constant $\mathcal{T}_{\dot{\psi}}$ and gain $\mathcal{K}_{\dot{\psi}}$ as in Brescianini et al. (2013). Figure 3(a) illustrates the estimated trajectories from an indicative MPL having 16 action sequences, while Figure 3(b) demonstrates the changes in the estimated trajectories when applying the MPL with noisy initial velocities of the robot.

4.2. Uncertainty-aware collision-free navigation

At the core of the collision-free navigation task is the CPN which processes a) the input depth image $\mathbf{o}_t$, b) the robot's partial state $\mathbf{s}_t$, and c) motion primitives-based sequences of future references $\mathbf{a}_{t:t+H}$ from the MPL discussed in Section 4.1, and is trained to predict the collision scores of the anticipated robot motion at each time step from $t+1$ to $t+H$ in the future:

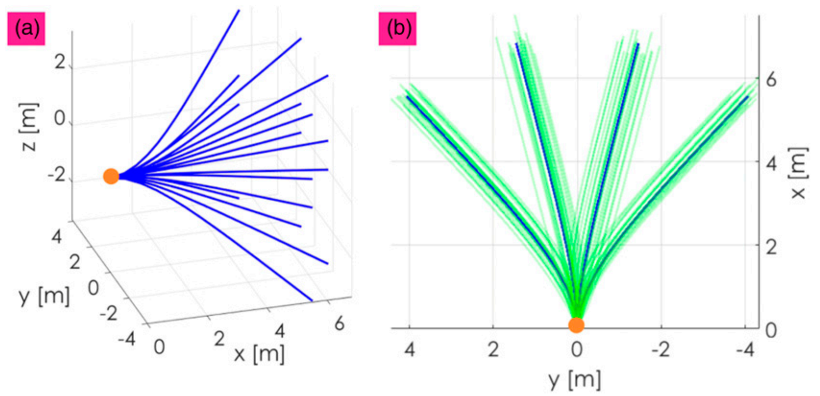

Figure 3. (a) Estimated trajectories from an indicative MPL having 16 action sequences and $\mathbf{s}_t = [2.5,0,0,0,0,0]^T$, (b) the changes in the estimated trajectories, visualized in green color, when applying the MPL with 20 different initial velocities $[\mathbf{v}_{t,x}, \mathbf{v}_{t,y}, \mathbf{v}_{t,z}]^T$ of the robot which are drawn randomly from Gaussian distributions $[\mathcal{N}(2.5, 0.5), \mathcal{N}(0, 0.5), \mathcal{N}(0, 0.5)]^T$ m/s.

$$\widehat{\mathbf{c}}^{col}_{t+1:t+H+1} = \left[\widehat{c}^{col}_{t+1}, \widehat{c}^{col}_{t+2}, ..., \widehat{c}^{col}_{t+H}\right] \quad (7)$$

By entirely using collision data in simulation. Thus, ORACLE avoids the need for hand-engineered collision checking algorithms such as in Bucki et al. (2020); Gao et al. (2019) or access to a reconstructed map of the environment Funk et al. (2021); Tabib et al. (2022). The collision costs for every action sequence in the MPL of velocity-steering commands can then be evaluated in parallel as per Kew et al. (2021), exploiting modern GPU architectures and thus enable high update rate compute. Notably, when evaluating the collision costs, ORACLE does not only consider the mean estimate of the robot's partial state but also its estimated uncertainty (exploiting the Unscented Transform) as calculated by any onboard localization system, as well as the epistemic uncertainty in the neural network model, as detailed in Section 4.2.3.

4.2.1. Neural network architecture. To predict a sequence of collision labels ($\widehat{\mathbf{c}}^{col}_{t+1:t+H+1}$) from a sequence of input actions ($\mathbf{a}_{t:t+H}$), given the current partial state of the robot ($\mathbf{s}_t$) and the depth image ($\mathbf{o}_t$), we use a Long Short-Term Memory (LSTM), a type of recurrent neural network, at the core of the CPN. In further detail, the input to the LSTM cells is generated by the velocity-steering angle action sequence provided by the MPL, while the initial state of the LSTM is a compressed latent vector encoding information about $\mathbf{s}_t$ and $\mathbf{o}_t$. This encoded latent vector is a concatenation of the output of a Convolutional Neural Network (CNN), which processes $\mathbf{o}_t$, and a Fully-Connected Network (FCN), which processes $\mathbf{s}_t$. It is noted that this encoded latent vector is learned simultaneously with the rest of the network, while CPN is trained in an end-to-end manner. Specifically, the outputs of the LSTM cells are passed through an FCN to predict a) the collision labels, as well as b) the positions, and c) relative yaw angles of the robot at each future time step with respect to the current $\mathcal{V}$-frame at time step t:

$$\begin{aligned} &\widehat{\mathbf{c}}^{col}_{t+1:t+H+1} \text{(a)} \\ &{}^{\mathcal{V}}\widehat{\mathbf{p}}_{t+1:t+H+1} = \left[{}^{\mathcal{V}}\widehat{\mathbf{p}}_{t+1}, {}^{\mathcal{V}}\widehat{\mathbf{p}}_{t+2}, \ldots, {}^{\mathcal{V}}\widehat{\mathbf{p}}_{t+H}\right] \text{(b)} \\ &\widehat{\boldsymbol{\delta}}_{t+1:t+H+1} = \left[\widehat{\delta}_{t+1}, \widehat{\delta}_{t+2}, ..., \widehat{\delta}_{t+H}\right] \text{(c)} \end{aligned} \quad (8)$$

Instead of regressing the robot's low-level commands directly from the network inputs, our CPN learns to perform collision checking implicitly for each action sequence. This is shown to generalize well to different simulated and real-world environments, as demonstrated in section 5. Intuitively, the method opts to rely on a priori set of motion primitives as candidate action sequences and then solves the simpler problem of collision checking on them instead of regressing directly the control action which would represent a more complex and thus potentially harder to generalize formulation. It is noted that the position and relative yaw angle prediction output heads are only executed in the training phase to provide additional back-propagated gradients to train the CPN and are not evaluated in the inference mode. The prediction network architecture, as shown in Figure 4, is inspired by the network in Kahn et al. (2021b). However, we replace the MobileNetV2 part with the ResNet-8 network as in Loquercio et al. (2018) for faster onboard inference speed.

4.2.2. Data collection and augmentation. The RotorS simulator (Furrer et al., 2016) is used to collect data for training the CPN. To ensure successful sim-to-real transfer, the dynamics of the simulated model should be matched with the intended real system, in this case, the custom quadrotor described in Section 4.5.2. Relevant methods for dynamic system identification of MAVs are presented in Sa et al. (2017). To collect data for predicting collision scores at the future time steps, an action sequence with random $\mathbf{v}^r$ and δ^r as described in (1)–(5) is drawn and is fully executed. This process is repeated until the robot collides with the obstacles or a timeout event occurs. One training data point $\mathbf{d}$ is recorded every time the robot moves more than Δ_{th}

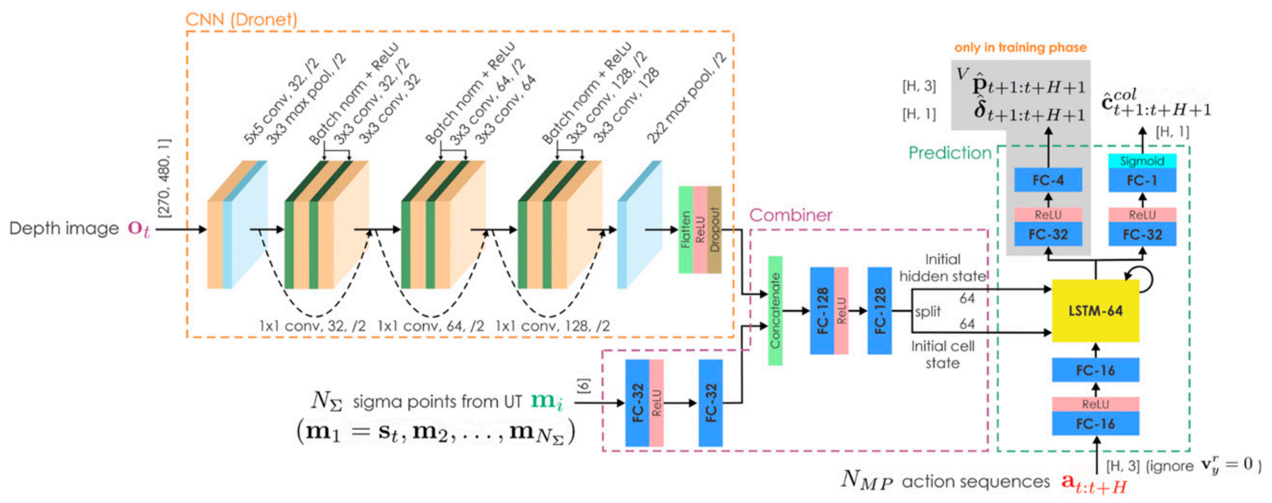

Figure 4. Architecture of the Collision Prediction Network (CPN). The convolutional hyperparameters are represented in the format ($\mathfrak{a} \times \mathfrak{b}$ conv , $\mathfrak{c}$, $/\mathfrak{d}$), where $\mathfrak{a} \times \mathfrak{b}$ refers to the kernel size, $\mathfrak{c}$ refers to the number of channels, and $\mathfrak{d}$ refers to the stride length. The dense layers only have the layer size mentioned alongside. The dimensions of the inputs and outputs are displayed next to their corresponding arrows where H denotes the action sequence's length.

meters or collides with the environment. Each such data point has the format:

$$\mathbf{d}_{CPN} = \left(\mathbf{o}_t, \mathbf{s}_t, \mathbf{a}_{t:t+H}, \mathbf{c}^{col}_{t+1:t+H+1}, {}^{\mathcal{V}}\mathbf{p}_{t+1:t+H+1}, \boldsymbol{\delta}_{t+1:t+H+1}\right)$$

where $\mathbf{c}^{col}_{t+1:t+H+1} = [c^{col}_{t+1}, c^{col}_{t+2}, ..., c^{col}_{t+H}]$, c^{col}_{t+i} denotes the ground-truth collision label between time steps $t+i-1$ and $t+i$, $i = 1, \ldots, H$ (equal to 1 for collision and 0 for non-collision status) and ${}^{\mathcal{V}}\boldsymbol{p}_{t+1:t+H+1} = [{}^{\mathcal{V}}\mathbf{p}_{t+1}, {}^{\mathcal{V}}\mathbf{p}_{t+2}, \ldots, {}^{\mathcal{V}}\mathbf{p}_{t+H}]$, $\boldsymbol{\delta}_{t+1:t+H+1} = [\delta_{t+1}, \delta_{t+2}, ..., \delta_{t+H}]$; ${}^{\mathcal{V}}\mathbf{p}_{t+i}$, $\delta_{t+i} = \psi_{t+i} - \psi_t$ denote the ground-truth position and relative yaw angle of the robot at the future time step $t+i$, $i = 1, \ldots, H$ expressed in the current $\mathcal{V}$-frame at time step t, respectively. When the collision happens midway an action sequence, for instance after the execution of $\mathbf{a}_{t+k}$ ($k < H$), then the collision labels corresponding to the remaining actions in the sequence $\mathbf{c}^{col}_{t+k+1:t+H}$ are set to 1, and augmented data points are also added to the dataset by replacing the actions after $\mathbf{a}_{t+k}$ with randomly sampled actions as in Kahn et al., (2021a). The number of data points created by augmenting the remaining actions is such that the number of data points with no collision and the number of data points with at least one collision label are almost equal; hence, the dataset is almost balanced. Moreover, we also perform the horizontal flip data augmentation following the below lemma:

Lemma IV.1 Consider the following assumptions:

1) The depth camera follows the pinhole camera model.
2) The xOz-plane of $\mathcal{B}$ and the yOz-plane of the depth camera frame, $\mathcal{C}$, are identical. The x, y, z-axes of $\mathcal{B}$ point to the front, left of the robot, and upward, respectively. The x, y, z-axes of $\mathcal{C}$ point to the left of the depth camera, downward, and to the front of the depth camera, respectively.
3) The low-level closed-loop dynamics of the robot can be approximated by the system of equation (6).

If the above assumptions are satisfied, the augmented data point $\mathbf{d}^{flip}_{CPN}$ can be added to the dataset where $\mathbf{d}^{flip}_{CPN} = (\mathbf{o}^{flip}_t, \mathbf{s}^{flip}_t, \mathbf{a}^{flip}_{t:t+H}, \mathbf{c}^{col}_{t+1:t+H+1}, {}^{\mathcal{V}}\mathbf{p}^{flip}_{t+1:t+H+1}, \boldsymbol{\delta}^{flip}_{t+1:t+H+1})$ where $\mathbf{o}^{flip}_t$ is the horizontally flipped image of $\mathbf{o}_t$ and $\mathbf{s}^{flip}_t, \mathbf{a}^{flip}_{t:t+H}, {}^{\mathcal{V}}\mathbf{p}^{flip}_{t+1:t+H+1}, \boldsymbol{\delta}^{flip}_{t+1:t+H+1}$ are created by changing the signs of $\mathbf{v}_{t,y}$, ω_t, ϕ_t; $\mathbf{v}^r_{t+i,y}, \delta^r_{t+i}$ ($i = 0, ..., H-1$); ${}^{\mathcal{V}}\mathbf{p}_{t+i,y}$; δ_{t+i} ($i = 1, \ldots, H$) in $\mathbf{s}_t$, $\mathbf{a}_{t:t+H}$, $\mathbf{p}_{t+1:t+H+1}$ and $\boldsymbol{\delta}_{t+1:t+H+1}$, respectively.

Proof. See Appendix B

In order to collect a comprehensive dataset for training the collision predictor, we randomized the initial position and orientation of the robot, as well as the obstacles' poses, categories, dimensions, and densities in order to collect around 1.5 million data points, including augmented ones, in total. The entire data collection process in the Gazebo simulator requires approximately 6 days on a laptop with AMD Ryzen 9 4900HS CPU with 32 GB of RAM. Figure 5 illustrates one indicative training environment which has a size of 40 × 40 × 10 m and includes obstacles having primitive shapes such as spheres, pyramids, cylinders, T-shape and U-shape blocks, as well as common real-world obstacles such as trees, tables, chairs, walls, or fences. The derived dataset was then split into a training and validation subset with an 80%: 20% ratio. The network is trained end-to-end with the Adam optimizer Kingma and Ba (2015) and the loss function as the weighted sum of the binary cross-entropy (BCE) loss for collision prediction (binary classification task) and the mean-squared error (MSE) loss for position and relative yaw angle predictions (regression tasks):

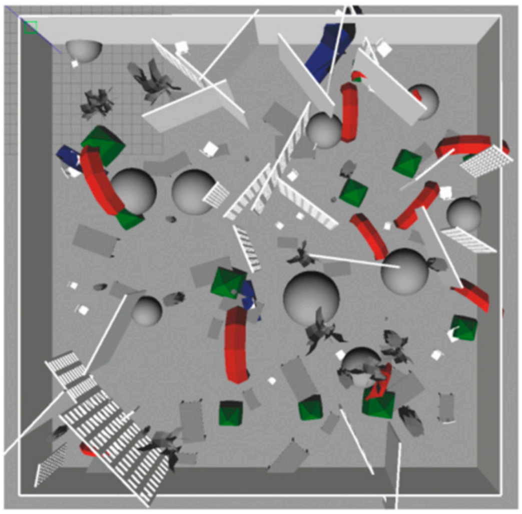

Figure 5. An indicative simulation environment for collecting training data.

$$\begin{aligned} L_{CPN} &= \alpha_1^{CPN} L_{BCE}(\mathbf{c}^{col}_{t+1:t+H+1}, \widehat{\mathbf{c}}^{col}_{t+1:t+H+1}) \\ &\quad + \alpha_2^{CPN} L_{MSE}({}^{\mathcal{V}}\mathbf{p}_{t+1:t+H+1}, {}^{\mathcal{V}}\widehat{\mathbf{p}}_{t+1:t+H+1}) \\ &\quad + \alpha_3^{CPN} L_{MSE}(\boldsymbol{\delta}_{t+1:t+H+1}, \widehat{\boldsymbol{\delta}}_{t+1:t+H+1}) \end{aligned} \tag{9}$$

where L_{MSE} loss is only calculated for time steps where the collision labels are zeros.

4.2.3. Uncertainty-aware prediction. As mentioned, the method further considers the uncertainty of the robot's partial state and the epistemic uncertainty of the collision prediction network. First, we calculate the combined collision cost for each action sequence in the MPL as the weighted sum of the collision scores at future time steps. Specifically, the sooner the collision event is predicted to happen, the higher its contribution to the final collision cost:

$$\widehat{c}^{col} = \sum_{i=1}^{H} \widehat{c}^{col}_{t+i} e^{-\lambda(i-1)}, \lambda > 0 \tag{10}$$

where λ is the time-step weighting factor. It is noted that this formula is similar to the geometric discount widely used in reinforcement learning where the discount rate is $e^{-\lambda} < 1$ with $\lambda > 0$. To account for the uncertainty of $\mathbf{s}_t$, which may not be negligible—especially in fast flight or within perceptually degraded environments—we utilize the Unscented Transform (UT) Julier and Uhlmann (1997) to approximately propagate the uncertainty in $\mathbf{s}_t$ to the predicted collision cost $\widehat{c}^{col}$ of an action sequence $\mathbf{a}_{t:t+H}$. In the UT, for a ζ-dimensional robot's partial state, $N_\Sigma = 2\zeta+1$ sigma points $\mathbf{m}_i$, and their associated weights W_i $(i = 1, \ldots, N_\Sigma)$, are computed based on the mean value $\mathbf{s}_t$ and the covariance matrix $\boldsymbol{\Sigma}_t$ using the following formulas:

$$\mathbf{m}_1 = \mathbf{s}_t \tag{11}$$

$$W_1 = \frac{\kappa}{\zeta + \kappa} \tag{12}$$

$$\mathbf{m}_i = \mathbf{s}_t + \left(\sqrt{(\zeta+\kappa)\boldsymbol{\Sigma}_t}\right)_i \tag{13}$$

$$W_i = \frac{1}{2(\zeta+\kappa)} \tag{14}$$

$$\mathbf{m}_{i+\zeta} = \mathbf{s}_t - \left(\sqrt{(\zeta+\kappa)\boldsymbol{\Sigma}_t}\right)_i \tag{15}$$

$$W_{i+\zeta} = \frac{1}{2(\zeta+\kappa)} \tag{16}$$

where $i = 2, \ldots, \zeta+1, \kappa \in \mathbb{R}$, $(\sqrt{(\zeta+\kappa)\boldsymbol{\Sigma}_t})_i$ is the $(i-1)$th row or column of the matrix square root of $\sqrt{(\zeta+\kappa)\boldsymbol{\Sigma}_t}$. These sigma points are then propagated through the CPN, and the mean and variance of the output distribution of $\widehat{c}^{col}$ are calculated based on the output predictions of the sigma points m_i and the weights W_i.

Additionally, the epistemic uncertainty—which can be significant for novel input data—can be captured by the variance between the outputs of different models in an ensemble of neural networks as in Lakshminarayanan et al. (2017). Specifically, we train the CPN with different initial weights and shuffling of the dataset to obtain multiple final weights for it. This is shown empirically to explore more diverse modes in function space compared to MC dropout Fort et al. (2019); Pop and Fulop (2018). For efficient neural network forward pass and uncertainty estimation, we split the neural network shown in Figure 4 into 3 parts, namely, the CNN, Combiner, and Prediction networks. Let a) N_E, N_{MP} be the number of neural networks in the ensemble and action sequences in the MPL, respectively, and b) $\sigma_n^{col}, \rho_n^{col}$ $(n = 1, \ldots, N_E)$ the variance and mean of the predicted collision cost of $\mathbf{a}_{t:t+H}$, estimated by the UT with different neural networks in the ensemble. As per Kendall and Gal (2017), the total variance can then be expressed as:

$$\sigma_{tot}^{col} = \frac{1}{N_E} \sum_{n=1}^{N_E} \left[\sigma_n^{col} + \left(\rho_n^{col} - \overline{\rho}^{col}\right)^2\right] \tag{17}$$

where $\overline{\rho}^{col} = 1/N_E \sum_{n=1}^{N_E} \rho_n^{col}$. The final uncertainty-aware collision cost for an action sequence follows the upper confidence bound policy as per Georgakis et al. (2022) and is given as:

$$\widehat{c}^{uac} = \overline{\rho}^{col} + \alpha\sqrt{\sigma_{tot}^{col}}, \alpha > 0 \tag{18}$$

Specifically, we denote $\widehat{c}_k^{uac}$ as the uncertainty-aware collision cost of the kth action sequence, $\mathbf{a}^k_{t:t+H}$, in the MPL $(k = 1, \ldots, N_{MP})$. It is noted that by splitting CPN into 3 parts, we can perform inference on the CNN, Combiner, and Prediction networks with different input batch sizes of 1, N_Σ, and $N_\Sigma \times N_{MP}$, respectively, avoiding the need to use the large input batch size of $N_\Sigma \times N_{MP}$ for each CPN in the ensemble. Figure 6 outlines the steps to derive the uncertainty-aware collision costs for every action sequence in the MPL.

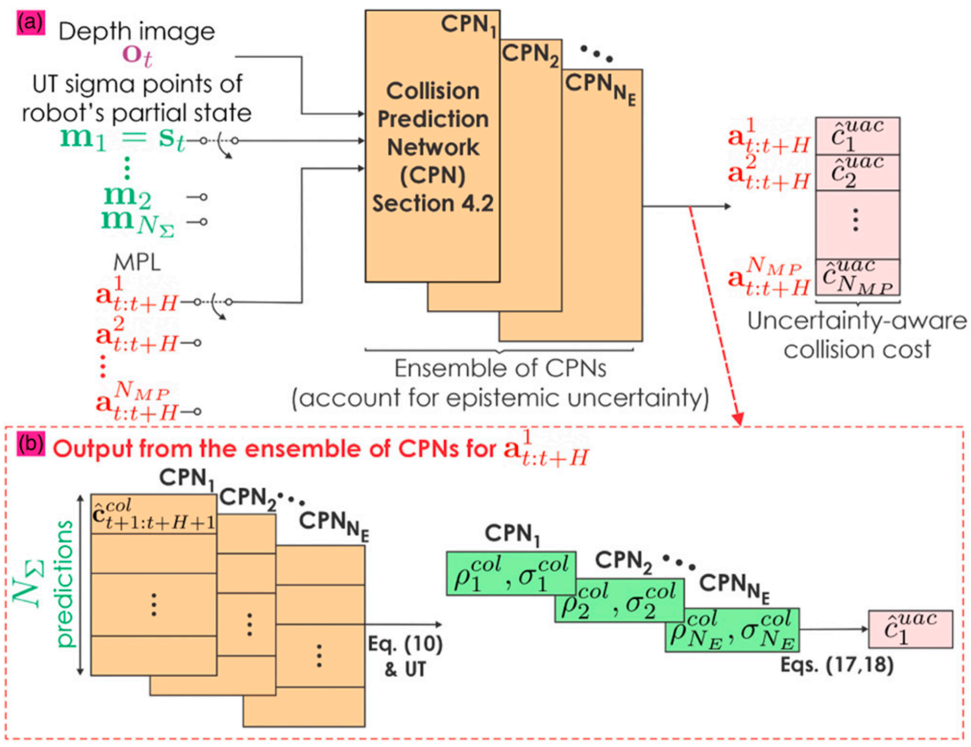
(a)
Depth image $\mathbf{o}_t$
UT sigma points of robot's partial state
$\mathbf{m}_1 = \mathbf{s}_t$
$\mathbf{m}_2$
$\mathbf{m}_{N_\Sigma}$
MPL
$\mathbf{a}^1_{t:t+H}$
$\mathbf{a}^2_{t:t+H}$
$\mathbf{a}^{N_{MP}}_{t:t+H}$
CPN1
CPN2
CPN$_{N_E}$
Collision Prediction Network (CPN) Section 4.2
Ensemble of CPNs (account for epistemic uncertainty)
$\hat{c}^{uac}_1$
$\hat{c}^{uac}_2$
$\hat{c}^{uac}_{N_{MP}}$
Uncertainty-aware collision cost
(b) Output from the ensemble of CPNs for $\mathbf{a}^1_{t:t+H}$
N_Σ predictions
$\hat{\mathbf{c}}^{col}_{t+1:t+H+1}$
Eq. (10) & UT
$\rho^{col}_1, \sigma^{col}_1$
$\rho^{col}_2, \sigma^{col}_2$
$\rho^{col}_{N_E}, \sigma^{col}_{N_E}$
Eqs. (17,18)

Figure 6. (a) The ensemble of CPNs takes as inputs the current depth image, the set of sigma points calculated from the Unscented Transform based on the mean value $\mathbf{s}_t$ and covariance matrix Σ_t, and the MPL to derive the uncertainty-aware collision costs for every action sequence in the MPL in parallel. (b) Steps to derive the uncertainty-aware collision cost for an action sequence in the MPL from the output of the ensemble of CPNs.

4.3. *Visually-attentive navigation*

Solutions to the information-gathering problem usually involve the evaluation of utility functions Fox et al. (1998); Popovic et al. (2018), which is one of the main computational bottlenecks in informative path planning Schmid et al. (2020); Rckin et al. (2022). In this work, we aim to allow efficient information gathering based on the latest sensor observations by designing an IPN to approximately estimate the information gains of multiple action sequences. Specifically, the IPN considered in this work is a neural network that takes as input a) the depth image and a 2D detection/interestingness mask stacked together $[\mathbf{o}_t, \boldsymbol{\mu}_t]$, b) the mean value of the robot's partial state estimate $\mathbf{s}_t$ (involving its linear velocities, z-axis angular velocities, and roll/pitch angles), and c) action sequences in the same library (MPL) as the CPN. The detection mask is such that each pixel of the depth image is associated with a value from 0 (lowest) to 1 (highest) based on its interestingness. As interestingness value, the output of relevant methods focusing on extrinsic top-down or intrinsic bottom-up motivations such as in object detection Redmon and Farhadi (2018) (top-down) or visual saliency image maps Tsotsos (2011); Frintrop et al. (2015); Kümmerer et al. (2018) (bottom-up) are considered. A-ORACLE is not bound to any particular type of interestingness concept and only assumes that the results of such methods are captured by an image mask aligned with the depth image thus annotating each depth pixel with an interestingness weight. Specifically, the task of the IPN is to predict the information gain obtained by the robot at each time step from $t+1$ to $t+H$ in the future:

$$\hat{\mathbf{g}}_{t+1:t+H+1} = [\hat{g}_{t+1}, \hat{g}_{t+2}, ..., \hat{g}_{t+H}] \tag{19}$$

by approximating the expert detailed in Section 4.3.2, utilizing modern GPU computing capabilities to speed up the computation.

4.3.1. Neural network architecture. Figure 7 describes the architecture of the IPN. To predict a sequence of information gain labels at future time steps ($\hat{\mathbf{g}}_{t+1:t+H+1}$), we need information about the anticipated positions and orientations of the robot at those time steps, as well as the latest understanding of the environment encoded in the stacked matrix of the current depth image and the associated detection mask ($[\mathbf{o}_t, \boldsymbol{\mu}_t]$). We use a 1D Long Short-Term Memory (LSTM) recurrent neural network whose output vector at each time step encodes information to predict the robot's position and relative yaw angle at that time step with respect to $\mathcal{V}$-frame at time step t. It is noted that the position (${}^{\mathcal{V}}\hat{\mathbf{p}}_{t+1:t+H+1} = [{}^{\mathcal{V}}\hat{\mathbf{p}}_{t+1}, {}^{\mathcal{V}}\hat{\mathbf{p}}_{t+2}, ..., {}^{\mathcal{V}}\hat{\mathbf{p}}_{t+H}]$) and relative yaw angle ($\hat{\boldsymbol{\delta}}_{t+1:t+H+1} = [\hat{\delta}_{t+1}, \hat{\delta}_{t+2}, ..., \hat{\delta}_{t+H}]$) prediction output heads are only executed in the training phase to provide additional back-propagated gradients to train the IPN and are not evaluated in the inference mode. Specifically, the input to the LSTM cells is generated by the velocity-steering angle action sequence $\mathbf{a}_{t:t+H}$ provided by the same MPL as the CPN, while the initial state of the LSTM is a latent vector encoding information about $\mathbf{s}_t$. The ResNet1 in Figure 7 outputs a multi-channel feature map that compresses the information in $[\mathbf{o}_t, \boldsymbol{\mu}_t]$. Then, the output from the LSTM network is fed to several FC layers whose output is expanded and added to the expanded output of ResNet1 to provide the input to ResNet2. Another FCN processes the output from ResNet2 to provide future information gain predictions.

We also tested the same architecture as the CPN, described in Figure 4, for the information prediction task. However, the output feature map from the CNN part of the CPN does not have enough spatial resolution to enable the information gain prediction task. Moreover, we tried to replace the ResNet2 (Figure 7) with a 2D Convolutional LSTM Shi et al. (2015) while using the output feature map from ResNet1 as the 2D LSTM's initial state, and the output sequence from the 1D LSTM as the 2D LSTM's input sequence. However, the resulting network was slow to train and perform inference. The choice of using the network architecture presented in Figure 7 (using a 1D LSTM for position and relative yaw angle predictions and using the ResNet2 with shared weights for information gain prediction at every future time step) balances the accuracy of the prediction and the speed of training and inference.

For efficient neural network forward pass, we split the neural network shown in Figure 7 into 2 parts, namely, the CNN and Prediction networks. We can then perform inference on the CNN and Prediction networks with different input batch sizes of 1 and N_{MP}, respectively, avoiding the need to use the same input batch size of N_{MP} for the whole IPN. Intuitively, given that the IPN can closely approximate the information gain then the ability to run it efficiently allows high planning rates which benefits online performance.

4.3.2. Ground-truth information gain label. The ground-truth information gain label for training the IPN is calculated using Voxblox's volumetric map (Oleynikova et al., 2017) augmented with an additional interestingness field for each voxel. Specifically, we denote I^k as the interestingness of voxel k in the occupancy map built only from the current depth image $\mathbf{o}_t$. The interestingness value of each voxel is calculated as the average interestingness of every pixel in the detection mask $\boldsymbol{\mu}_t$ whose 3D projected point lies in voxel k:

$$I^k = \frac{1}{\left|\mathbf{proj}^k_{\mu_t}\right|} \sum_{[u,v] \in \mathbf{proj}^k_{\mu_t}} \boldsymbol{\mu}_t(u,v) \tag{20}$$

where $\mathbf{proj}^k_{\mu_t}$ denotes the set of interesting pixels whose 3D projections lie on voxel k. Moreover, to encourage observing the unknown areas that are next to the observed interesting areas, we decay the interestingness of the observed interesting voxels to unknown neighbor voxels by the decay function:

$$I^{k^{unk}} = I^{k^{unk}_{\text{nearest}}} \lambda_1^{\gamma\left(k^{unk}, k^{unk}_{\text{nearest}}\right)}, 0 < \lambda_1 < 1 \tag{21}$$

where k^{unk}_{nearest} is the nearest observed interesting voxel of the unknown voxel k^{unk} and $\gamma(k^{unk}, k^{unk}_{\text{nearest}})$ is the diagonal distance between k^{unk}_{nearest} and k^{unk}. It is noted that equation (21) is only applied to unknown neighbor voxels satisfying $\gamma(k^{unk}, k^{unk}_{\text{nearest}}) \le \gamma_{th}$. Finally, to account for the resolution of

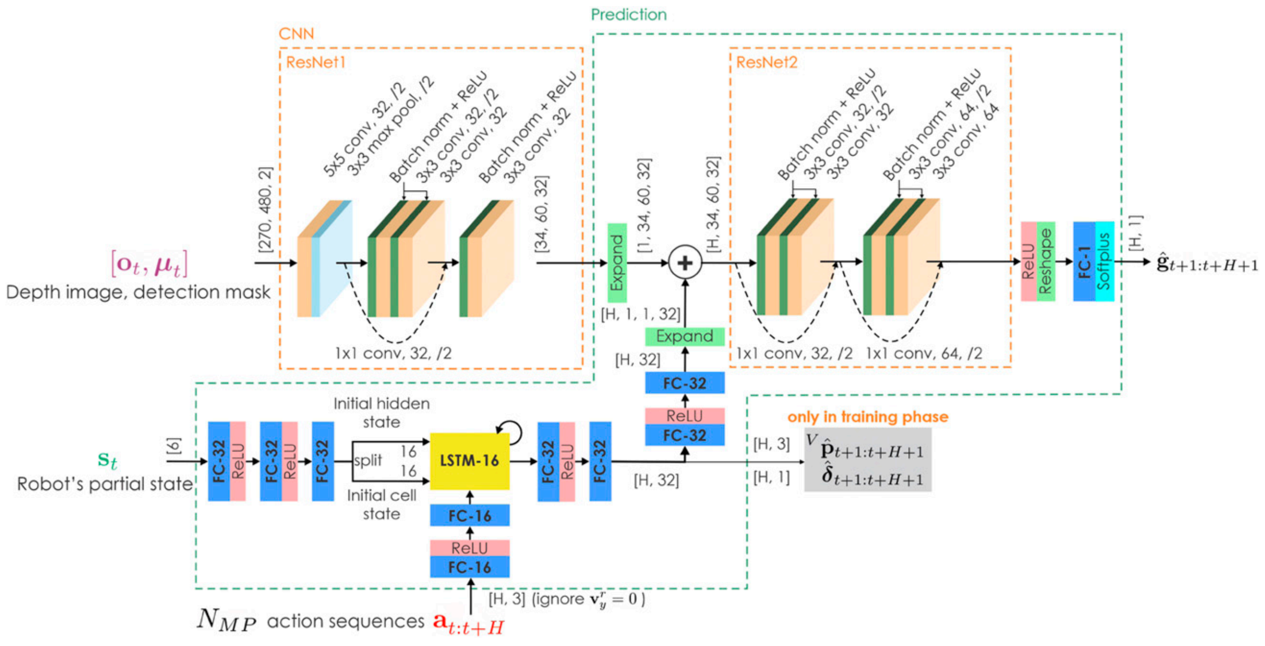

Figure 7. Architecture of the Information gain Prediction Network (IPN). The convolutional hyperparameters are represented in the format ($\mathfrak{a} \times \mathfrak{b}$ conv , $\mathfrak{c}$, /$\mathfrak{d}$), where $\mathfrak{a} \times \mathfrak{b}$ refers to the kernel size, $\mathfrak{c}$ refers to the number of channels, and $\mathfrak{d}$ refers to the stride length. The dense layers only have the layer size mentioned alongside. The dimensions of the inputs, outputs, and some internal signals inside the IPN are displayed next to their corresponding arrows where H denotes the action sequence's length.

the observation, we also weigh the contribution of each voxel by its corresponding Area per Pixel, denoted as AP^k for voxel k, as in Dang et al. (2018). The information gain for a viewpoint at time step $t+j$ is then calculated as:

$$\begin{aligned} g_{t+j} &= \sum_{k \in \mathbf{F}_{t+j}} I^k e^{-\lambda_2 \mathrm{AP}_k}, \lambda_2 > 0 \\ \mathrm{AP}_k &= \frac{z_k^2}{f_x^c f_y^c} \end{aligned} \tag{22}$$

where $\mathbf{F}_{t+j}$ is the frustum of the detection camera on the robot at time step $t+j$, z_k is the distance from voxel k to the detection camera, and f_x^c, f_y^c denote the focal length of the detection sensor based on the pinhole camera model. In practice, we perform ray casting in the frustum $\mathbf{F}_{t+j}$ and calculate the contribution of each voxel k lying on the rays, it is noted that each voxel is only counted once in equation (22). Moreover, the cast ray stops when it meets an occupied voxel. Figure 8 illustrates how the information gain label is calculated.

4.3.3. Data generation and augmentation. To create a diverse dataset for training the IPN, we utilize the same dataset collected for training the CPN and create synthetic detection masks, $\boldsymbol{\mu}_t$, based on the depth data. Specifically, multiple ellipses are created with random positions and dimensions to represent the interesting pixels (having interestingness equal 1) in the detection mask. To randomize between the cases where the observed information gain is high or low, we guarantee that there's an ellipse in the actual moving direction of the robot with probability p = 0.5. A Gaussian filter with random kernel size is further applied to the detection mask to create the final synthetic mask, reflecting a prior assumption that pixels that are next to the most interesting pixels can have small interestingness (between 0 and 1). Lastly, the synthetic mask pixels corresponding to the depth pixels that have invalid depth values (outside the depth range of the depth sensor) are removed and only pixels corresponding to the objects in the depth image are kept. Figure 9 illustrates the steps to generate the synthetic detection masks.

A data point d_{IPN} can then be created with the format:

$$\mathbf{d}_{IPN} = \left(\mathbf{o}_t, \boldsymbol{\mu}_t, \mathbf{s}_t, \mathbf{a}_{t:t+H}, \mathbf{g}_{t+1:t+H+1}, {}^{\mathcal{V}}\mathbf{p}_{t+1:t+H+1}, \boldsymbol{\delta}_{t+1:t+H+1}\right)$$

where $\mathbf{g}_{t+1:t+H+1} = [g_{t+1}, g_{t+2}, \ldots, g_{t+H}]$ denotes the information gain label at future time steps and ${}^{\mathcal{V}}\mathbf{p}_{t+1:t+H+1}, \boldsymbol{\delta}_{t+1:t+H+1}$ are defined as in Section 4.2.2. Assuming that all the assumptions in lemma IV.1 hold and the detection camera also satisfies the first two assumptions in lemma IV.1, we also perform the horizontal-flip data augmentation to obtain $\mathbf{d}_{IPN}^{flip} = (\mathbf{o}_t^{flip}, \boldsymbol{\mu}_t^{flip}, \mathbf{s}_t^{flip}, \mathbf{a}_{t:t+H}^{flip}, \mathbf{g}_{t+1:t+H+1}, {}^{\mathcal{V}}\mathbf{p}_{t+1:t+H+1}^{flip}, \boldsymbol{\delta}_{t+1:t+H+1}^{flip})$, where $\boldsymbol{\mu}_t^{flip}$ is the horizontally flipped image of $\boldsymbol{\mu}_t$.

4.3.4. Network training and inference. A weighted-MSE loss is calculated for the regression tasks of the three output prediction heads depicted in Figure 7 and the Adam optimizer Kingma and Ba (2015) is utilized to train the IPN. The loss function has the form:

$$\begin{aligned} L_{IPN} &= \alpha_1^{IPN} L_{MSE}(\mathbf{g}_{t+1:t+H+1}, \widehat{\mathbf{g}}_{t+1:t+H+1}) \\ &+ \alpha_2^{IPN} L_{MSE}({}^{\mathcal{V}}\mathbf{p}_{t+1:t+H+1}, {}^{\mathcal{V}}\widehat{\mathbf{p}}_{t+1:t+H+1}) \\ &+ \alpha_3^{IPN} L_{MSE}(\boldsymbol{\delta}_{t+1:t+H+1}, \widehat{\boldsymbol{\delta}}_{t+1:t+H+1}) \end{aligned} \tag{23}$$

where L_{MSE} is only calculated for time steps where the collision labels are zeros. The total information gain prediction of an action sequence can then be calculated as:

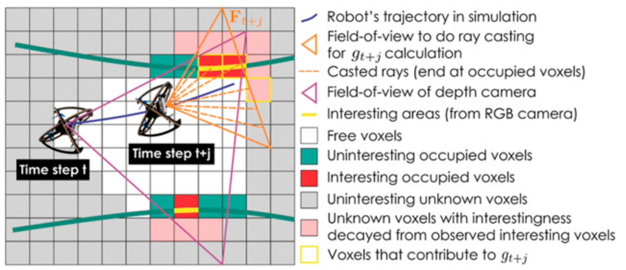
$\mathbf{F}_{t+j}$
Time step t
Time step t+j
Robot's trajectory in simulation
Field-of-view to do ray casting for g_{t+j} calculation
Casted rays (end at occupied voxels)
Field-of-view of depth camera
Interesting areas (from RGB camera)
Free voxels
Uninteresting occupied voxels
Interesting occupied voxels
Uninteresting unknown voxels
Unknown voxels with interestingness decayed from observed interesting voxels
Voxels that contribute to g_{t+j}

Figure 8. Outline of how the information gain label g_{t+j} at time step $t+j$ is calculated. At time step t, the robot builds an annotated volumetric map based only on the current depth image and the detection mask. The unknown voxels with decayed interestingness from observed interesting voxels are visualized in pink color. In this case, the decay equation (21) is only applied for unknown neighbor voxels having $\gamma(k^{unk}, k^{unk}_{\text{nearest}}) \leq 1$. The information gain label g_{t+j} is then calculated by performing ray casting within the detection camera's frustum $\mathbf{F}_{t+j}$ to calculate equation (22). The voxels that contribute to g_{t+j} are marked with yellow boundaries.

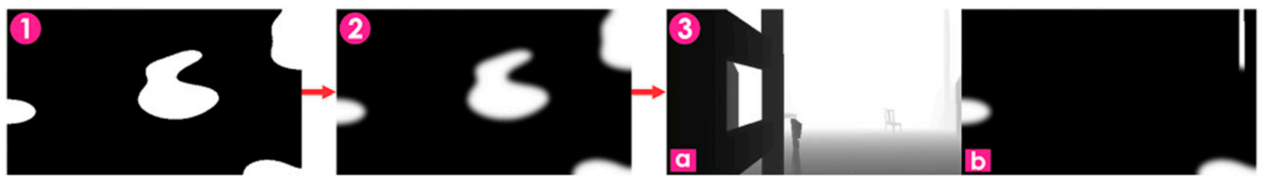

Figure 9. Steps to create synthetic detection masks to train the IPN. From left to right: (1) Multiple ellipses are created with random positions and dimensions for training purposes. (2) A Gaussian filter with random kernel size is further applied. (3a) The depth image is loaded. (3b) The final detection mask is generated by combining all valid pixels (having values within the range limits) of the depth image with the mask created by filtering randomly generated ellipses.

$$\widehat{g} = \sum_{i=1}^{H} \widehat{g}_{t+i} \tag{24}$$

Additionally, to reduce the computational time of the IPN in inference mode, we can estimate the information-gain for one in every K future time steps by reducing the size of the input to the Resnet2 in Figure 7 from $[H, 34, 60, 32]$ to $[H_{IPN}, 34, 60, 32]$, resulting in the number of prediction steps for the IPN$H_{IPN} < H$.

4.4. *Uncertainty-aware visually-attentive collision-free navigation*

Algorithm 1 Attentive-ORACLE Navigation Method

```
1: $\mathbf{s}_t, \mathbf{\Sigma}_t \leftarrow$ Get the robot's partial state and covariance
2: $\mathbf{m}_i\ (i = 1, \ldots, N_\Sigma) \leftarrow$ sigma points given by UT
3: $\mathbf{o}_t \leftarrow$ Get the depth image
4: $\boldsymbol{\mu}_t \leftarrow$ Get the detection mask
5: $\mathbf{n}_t^g \leftarrow$ Get the goal vector
6: for $n = 1$ to $N_E$ do
7:     CPN$_n \leftarrow$ Neural network $n$ in the ensemble
8:     for $\mathbf{a}_{t:t+H}$ in $MPL$ do
9:         for $i = 1$ to $N_\Sigma$ do
10:            // Parallel evaluation in GPU
11:            $\hat{\mathbf{c}}^{col}_{t+1:t+H+1} \leftarrow$ CPN$_n(\mathbf{o}_t, \mathbf{m}_i, \mathbf{a}_{t:t+H})$
12:            Calculate $\hat{c}^{col}$ by equation (10)
13:        end for
14:        Calculate $\sigma_n^{col}, \mu_n^{col}$ by UT on $\{\hat{c}^{col}\}_{N_\Sigma \times 1}$
15:    end for
16: end for
17: for $\mathbf{a}_{t:t+H}$ in $MPL$ do
18:     Calculate $\hat{c}^{uac}$ by equations (17), (18)
19: end for
20: $\hat{c}^{uac}_{min} \leftarrow \min_{k=1,\ldots,N_{MP}}\{\hat{c}^{uac}_k\}$
21: if $\hat{c}^{uac}_{min} > c_{de}$ then // Reach a dead end
22:     Send yaw-in-spot command
23:     Go to line 1
24: end if
25: for $\mathbf{a}_{t:t+H}$ in MPL having $\hat{c}^{uac} < \hat{c}^{uac}_{min} + c_{th}$ do
26:     if $\neg$ ($\boldsymbol{\mu}_t$ is empty) and $\neg$ timeout then
27:         $\hat{\mathbf{g}}_{t+1:t+H+1} \leftarrow$ IPN$(\mathbf{o}_t, \boldsymbol{\mu}_t, \mathbf{s}_t, \mathbf{a}_{t:t+H})$
28:         Calculate $\hat{g}$ by equation (24)
29:         Choose $\mathbf{a}^*_{t:t+H}$ to maximize $\hat{g}$
30:     else
31:         Calculate $c^g_\delta, c^g_\theta$ by equations (25),(26)
32:         Choose $\mathbf{a}^*_{t:t+H}$ to minimize $c^g_\delta$, then minimize $c^g_\theta$
33:     end if
34: end for
35: Execute the first action $\mathbf{a}^*_t$, then go to step 1
```

Algorithm 1 outlines Attentive-ORACLE's key steps. After calculating the uncertainty-aware predicted collision cost for each action sequence in the MPL, as described in section 4.2.3 (line 6–19), the minimum collision cost $\widehat{c}^{uac}_{\min}$ of all action sequences is calculated (line 20). If $\widehat{c}^{uac}_{\min} > c_{de}$, where c_{de} is a set positive threshold, the robot faces a dead end and it will rotate in the current position ("yaw-in-spot") until it finds a collision-free direction to follow (line 21–24). Then all the action sequences having collision cost greater than $\widehat{c}^{uac}_{\min} + c_{th}$, where c_{th} is a set positive threshold, are discarded (line 25). If the detection mask $\boldsymbol{\mu}_t$ is not empty and timeout does not occur, the IPN is queried to determine the most informative action sequence to follow (line 26–29), otherwise, the remaining safe action sequences are checked for deviation from the goal vector $\mathbf{n}_t^g$ (line 30–33):

$$c^g_\delta = |\mathrm{wrap}(\delta^r - \delta^g_t)| \tag{25}$$

$$c^g_\theta = \left|\mathrm{wrap}\left(\mathrm{atan}\,2\left(\mathbf{v}^r_z, \mathbf{v}^r_x\right) - \theta^g_t\right)\right| \tag{26}$$

where

$$\delta^g_t = \mathrm{atan}\,2\left(\mathbf{n}^g_{t,y}, \mathbf{n}^g_{t,x}\right) \tag{27}$$

$$\theta^g_t = \mathrm{atan}\,2\left(\mathbf{n}^g_{t,z}, \sqrt{\left(\mathbf{n}^g_{t,x}\right)^2 + \left(\mathbf{n}^g_{t,y}\right)^2}\right) \tag{28}$$

and wrap (.) is the function that wraps an angle in radians to $[-\pi, \pi]$. It is noted that lines 26–29 are not considered in the ORACLE (non-attentive) method. The best action sequence is then chosen and its first step is executed, while the whole procedure is repeated in a receding horizon fashion (line 35).

4.5. *Implementation details*

4.5.1. Hyper-parameters for training CPN and IPN. Both CPN and IPN are trained with the learning rate set to $5e-5$ for around 200 epochs and we choose $\alpha_1^{CPN} = 1.0, \alpha_2^{CPN} = \alpha_3^{CPN} = 0.01, \alpha_1^{IPN} = 1.0, \alpha_2^{IPN} = \alpha_3^{IPN} = 0.01$ to balance different loss terms in (9) and (23). It is noted that future work may involve learning the weights of distinct terms in the loss functions as per the work in Cipolla et al. (2018).

4.5.2. System overview. We design a quadrotor, dubbed Learning-based Micro Flyer (LMF), which inherited the collision-tolerant design of the Resilient Micro Flyer De Petris et al. (2020), yet with an increased diameter of 0.43 m and a mass of 1.2 kg. It integrates a Realsense D455 to obtain depth and RGB data at a 480 × 270 resolution with FOV of $[F_h, F_v] = [87, 58]°$ and a frequency of 15 FPS, a PixRacer Ardupilot-based autopilot delivering velocity and yaw-rate control, and a Realsense T265 fused with the IMU of the autopilot allowing it to estimate the velocity, orientation and angular rates of the robot. Notably, the position estimates of the T265 are not required by ORACLE or A-ORACLE except for calculating the unit goal vector $\mathbf{n}_t^g$ and checking if the robot had reached the waypoints. A Proportional controller for converting the reference steering angle to the yaw-rate command is also developed. The CPN, IPN, and the detection method for obtaining the detection mask (YOLO Redmon and Farhadi (2018) in this case) are implemented on a Jetson Xavier NX onboard LMF. Figure 10 illustrates the main hardware components of the LMF.

4.5.3. Image pre-processing step. Since real-life depth images are often subject to several shortcomings compared to simulated data, including a) missing information, b) loss of detail, and c) depth noise Hoeller et al. (2021), we perform an additional pre-processing step using the IP-Basic algorithm Ku et al. (2018) to refine the depth frame and thus reduce the mismatch between the real and simulated depth images. Specifically, this pre-processing step applies a series of morphological transformations and blurring operation to fill in empty pixels in the depth images. Figure 11 illustrates the effect of the depth image pre-processing step.

5. Evaluation studies

A set of evaluation studies were then conducted to verify the proposed learning-based attentive navigation method.

5.1. Simulation studies

5.1.1. Uncertainty-aware navigation. To evaluate ORACLE's ability to navigate cluttered environments in combination with degraded state estimates and noisy depth image inputs, we conducted simulation studies and compared ORACLE with 2 baselines. Specifically, the proposed approach was compared with a) the *"Naive"* method which utilizes the CPN directly to calculate the collision cost without considering the uncertainty of the partial state estimate $\mathbf{s}_t$ or that of the neural network model and b) the *"Ensemble"* method which uses Deep Ensembles only and not UT. Accordingly, the *Naive* method is not using neither the UT samples nor the ensemble of CPNs (cf. Figure 2). The type of simulation environments used is illustrated in Figure 12 and has dimensions of width × length × height = 60 × 100 × 9 m. Each such environment consists of two

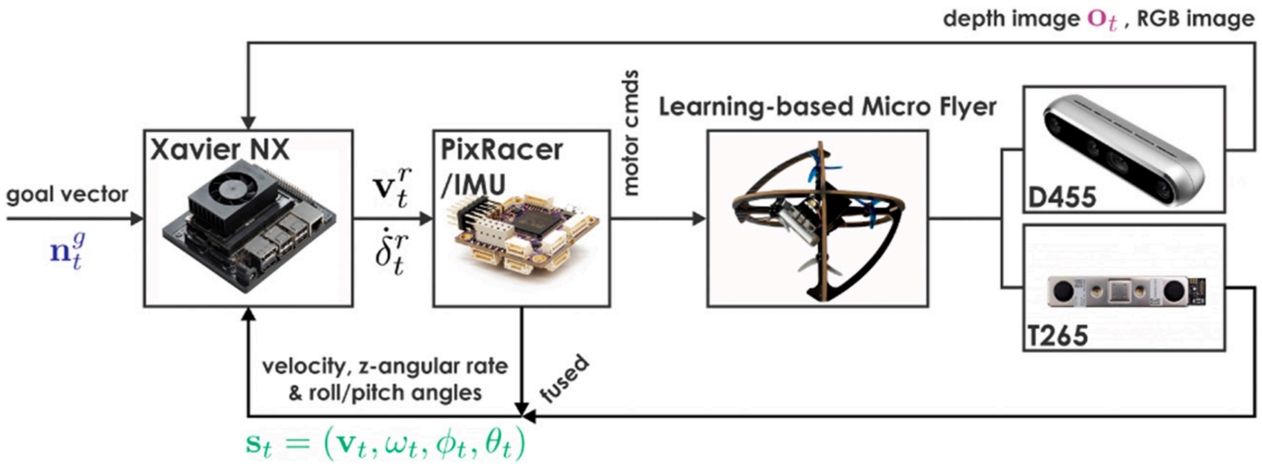
depth image $\mathbf{o}_t$, RGB image
Xavier NX
PixRacer /IMU
motor cmds
Learning-based Micro Flyer
D455
T265
goal vector
$\mathbf{n}_t^g$
$\mathbf{v}_t^r$
$\dot{\delta}_t^r$
velocity, z-angular rate & roll/pitch angles
fused
$\mathbf{s}_t = (\mathbf{v}_t, \omega_t, \phi_t, \theta_t)$

Figure 10. Main hardware components onboard LMF.

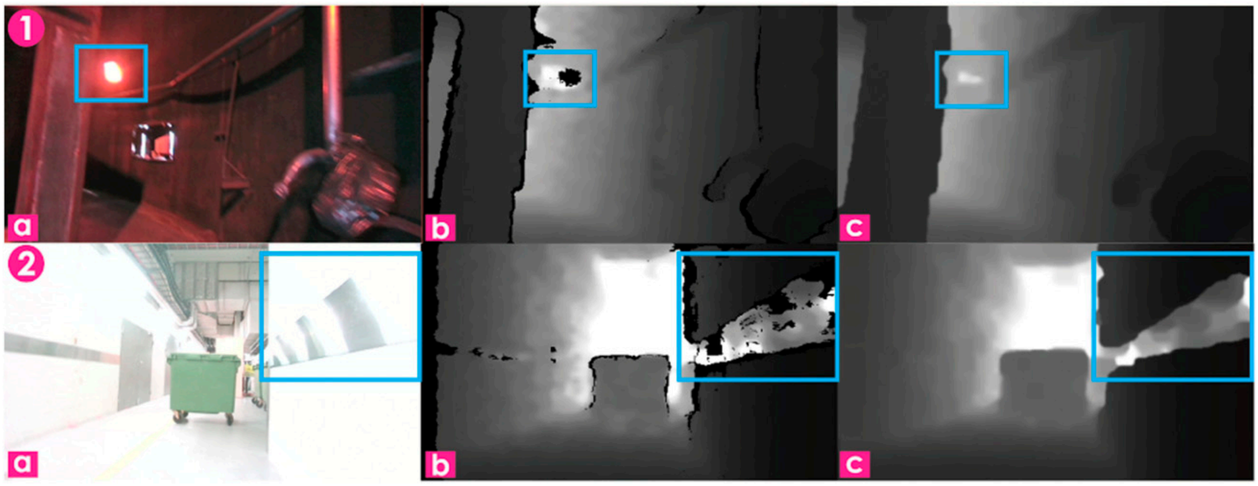

Figure 11. Depth-image preprocessing results. (a) RGB images from the Realsense D455 camera. (b) Raw depth images returned by the Realsense D455. Areas marked in blue boxes contain empty depth pixels due to textureless features caused by a light (1a) or reflective surface (2a). "Stereo shadow" regions can also be seen around the left object in 1b and the left part of (2b). (c) Depth images after the empty pixels are filled in.

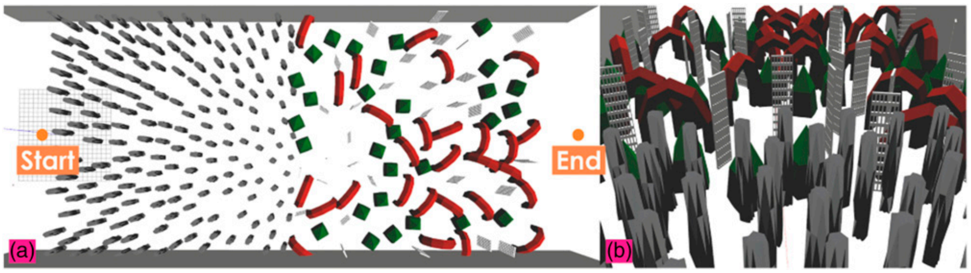

Figure 12. An indicative simulation environment for simulation studies evaluating ORACLE (a) and enlarged view of a specific section in the environment (b). The ceiling is removed for visualization purpose.

parts: the first part contains multiple cylinder-like obstacles with a diameter of about 1 m, distributed according to a Poisson disc sampling with a density of $\delta_1 = 3.5$ m, and the second part includes obstacles with different shapes, as illustrated in Figure 12(b), also following a Poisson disc sampling with a density of $\delta_2 = 5$ m.

The robot is modeled as a sphere of radius 0.22 m. We randomly generated 10 different environments, and both ORACLE and the 2 baselines are deployed in each environment 10 times with the same start point and end goal, which is 110 m ahead of the start point in the x-axis, but with different noise inputs at each run. Specifically, we deteriorated the partial state estimate with additive Gaussian noise on the x, y, and z-velocity components simultaneously, leading to $\mathbf{\Sigma}_t = \text{diag}(\sigma_v{}^2, \sigma_v{}^2, \sigma_v{}^2, 0, 0, 0)$, reflecting that the robot's z-axis angular velocity, roll/pitch angles can be estimated reasonably well (Weiss, 2012). The standard deviations (std) of the velocity noise in every axis σ_v is varied from 0.2 m/s to 0.6 m/s. For the image noise, we followed the empirical study in Ahn et al. (2019) to model the std of the depth noise as a quadratic function of the depth. Accordingly, if a pixel has the ground-truth depth of z, the simulated noisy depth value of that pixel z_{noisy} is given as:

$$z_{noisy} = z + \mathcal{N}(0, \sigma_z(z)) \tag{29}$$

$$\sigma_z(z) = d_z z^2 \tag{30}$$

It is noted that the negligible first- and zeroth-order terms are ignored in this formula. The Intel Realsense RGB-D camera D435 is found to have $d_z \approx 0.004$ in Ahn et al. (2019) while we use the later version of this sensor, Intel Realsense D455, in this work. We chose to simulate the depth image noise up to $d_z = 0.005$. For all simulations, the robot reaches the goal when it is within a radius of 5 m from the goal or if it crosses the line $x = 110$ m, additionally a timeout period of 100 s is also applied. The depth camera is simulated with a maximum range of $d_{max} = 10$ m, FOV of $[F_h, F_v] = [87, 58]°$. The MPL consists of $N_{MP} = 256$ action sequences and for each action sequence, $N_\Sigma = 7$ sigma points are evaluated since we only consider the noise in the velocity components of $\mathbf{s}_t$. The length of the action sequences in the MPL utilized by the CPN is $H = 14$ and the time-step duration is $\Delta_t = 0.2$ s, leading to $\mathbf{v}_x^r \leq 3.57 \text{m/s}$ from (3). The reference forward speed $\mathbf{v}_x^r = 2.5$ m/s is chosen for all simulations in Section 5.1.1 where the velocity and depth image's noise are also applied. The same collision threshold $c_{th} = 0.1$ and the time-step weighting factor $\lambda = 0.04$ are used for all methods, resulting in the weight of around 0.6 for the largest time step in (10). Additionally, all the methods replan at the rate of 15 Hz. For the *Ensemble* and ORACLE methods, we use an ensemble of $N_E = 5$ CPNs for collision-score prediction.

It is noted that the reference forward speed of 2.5 m/s is higher than the reference speed of 1.5 m/s used in Bartolomei et al. (2023), where no velocity or image noise is simulated and the density of the simulation environments is $\delta_1 = 2.23$ m. While our reference speed is lower than the flying speed of 10 m/s in Loquercio et al. (2021), where the maximum density of the simulated environments is $\delta_1 = 5$ m and the diameter of the obstacles is about 0.6 m, our maximum simulated velocity noise ($\sigma_v = 0.6$ m/s) is around 3 times the standard deviation of the velocity noise in the y-axis, the main reactive axis of the robot, simulated in Loquercio et al. (2021). Additionally, an empirical image noise model is applied in our simulation evaluation study. It is aimed to systematically evaluate the performance of our method when exposed to novel noisy depth images that are unseen during the training process. The average and 1-σ boundaries of the success (non-collision) rate of each simulation study are reported below.

Figure 13 demonstrates the success rate when the velocity estimation deteriorates. As shown, the *Naive* method exhibits more significant drops in the performance when the velocity noise in all axes is increased (drops by 28% at 0.2 m/s- but 50% at 0.4 m/s- and 60% at 0.6 m/s-noise level). On the other hand, the *Ensemble* method shows smaller drops in the performance at all levels of noise (drops by 12% at 0.2 m/s-, 28% at 0.4 m/s- and 32% at 0.6 m/s-noise level). Lastly, ORACLE is the least sensitive to velocity noise. Its success rate drops marginally at 0.2 m/s, while it drops by only 6% at 0.4 m/s- and 14% at 0.6 m/s-noise level. The predictions from 2 CPNs in the Ensemble are illustrated in Figure 14(a) and (b). The *Ensemble* baseline utilizes prediction from multiple CPNs, resulting in a more conservative set of safe action sequences, as presented in Figure 14(c). Lastly, when ORACLE is deployed and σ_v utilized in the UT is increased from 0.2 to 0.6 m/s (Figure 14(d)–(f)), the safe set of action sequences is further reduced, leading to a safer action sequence chosen finally when the velocity estimate is subjected to noise. However, a more conservative set of safe action sequences can lead to a larger deviation from the goal vector.

We also verified the performance of ORACLE when the velocity noise is wrongly estimated. Figure 15 shows the success rates of ORACLE when a fixed standard deviation $\sigma_v = 0.2$ m/s is used while the actual standard deviation of the velocity noise in all axes varies between 0 and 0.6 m/s. It can be concluded that the higher the true noise level is compared to the estimated noise level, the lower the success rate. However, the performance drops gracefully when the actual noise level is close to the estimated one (drops by only 2% at 0.4 m/s-noise level compared to using ORACLE with the actual σ_v) and the performance is still higher than the *Ensemble* baseline which does not use the UT.

The performance of the *Naive* baseline and *Ensemble* baseline, which is similar to ORACLE in this case, with different levels of depth image noise is given in Figure 16. As shown, the performance of the *Naive* method is greatly affected by the image noise (drops to 0%-success rate at the highest level of depth image noise). On the contrary, the use of the ensemble of CPNs renders the *Ensemble* method much less sensitive to depth image noise, only exhibiting a drop of around 20% at the highest noise level.

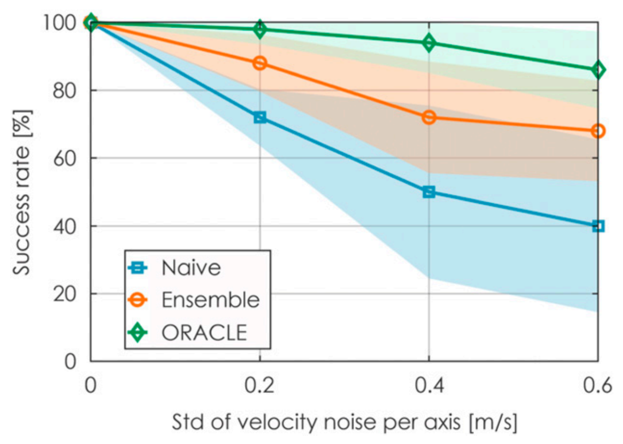

Figure 13. Sensitivity analysis of noise in velocity estimation (100 runs were performed). The x-axis shows the standard deviation of velocity noise applied simultaneously in all axes. The presented results relate to contributions 2–4.

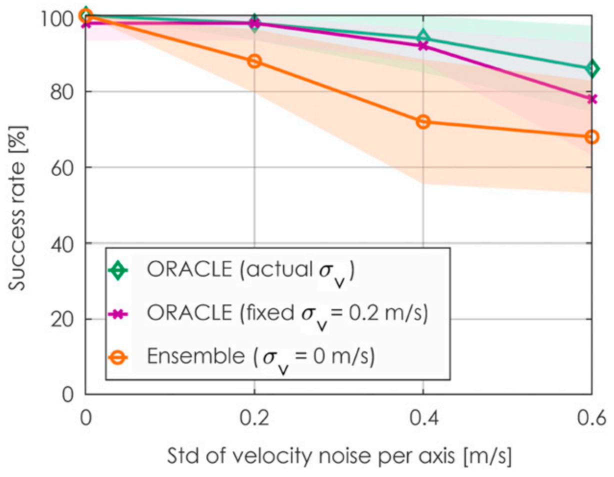

Figure 15. Sensitivity analysis of noise in velocity estimation when wrong σ_v is utilized in the UT (100 runs were performed). The x-axis shows the standard deviation of velocity noise applied simultaneously in all axes. The presented results relate to contributions 2-4.

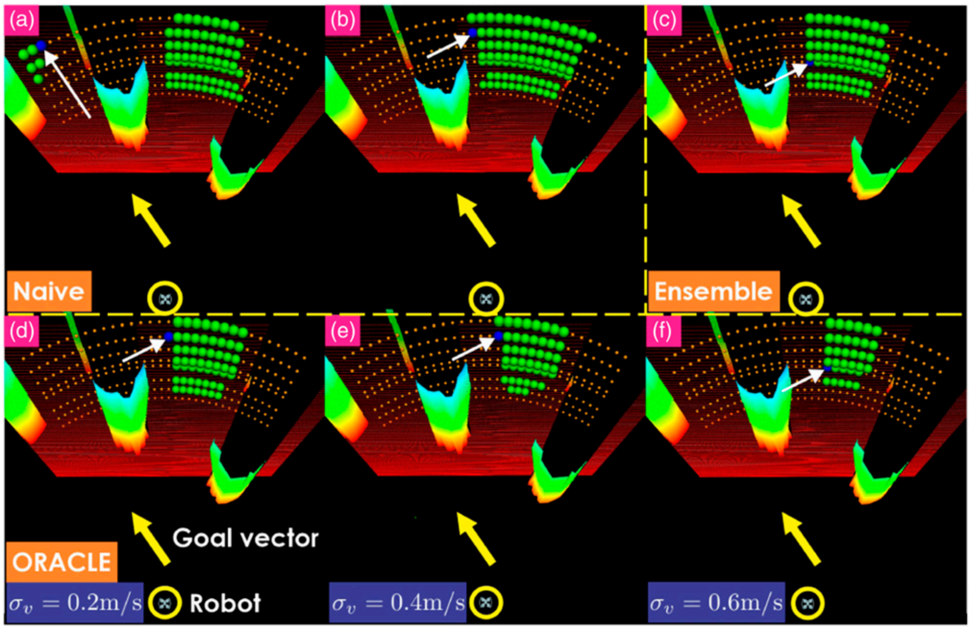

Figure 14. Collision-score predictions from *Naive* baseline with 2 different CPN's weights (a), (b), *Ensemble* baseline (c) and ORACLE with different σ_v utilized in UT (d, e, f). Green markers: estimated trajectory endpoints of safe action sequences, blue marker with an arrow: estimated trajectory endpoint of chosen action sequence. The presented results relate to contributions 2-4.

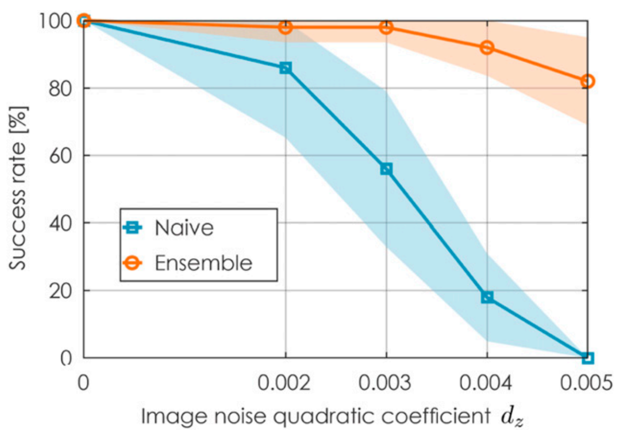

Figure 16. Sensitivity analysis of noise in depth image (100 runs were performed). The presented results relate to contributions 2-4.

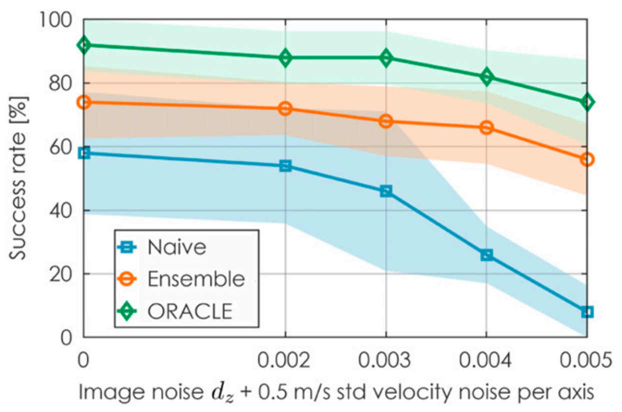

Figure 17. Sensitivity analysis of noise in both velocity estimation and depth image (100 runs were performed). The presented results relate to contributions 2-4.

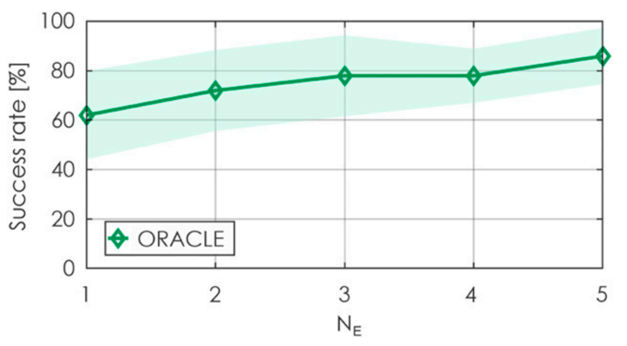

Figure 18. Planning performance when different N_E is utilized in ORACLE and velocity noise with $\sigma_v = 0.6$ m/s is applied on all *x*, *y*, *z* axes simultaneously (100 runs were performed). The presented results relate especially to contribution 3.

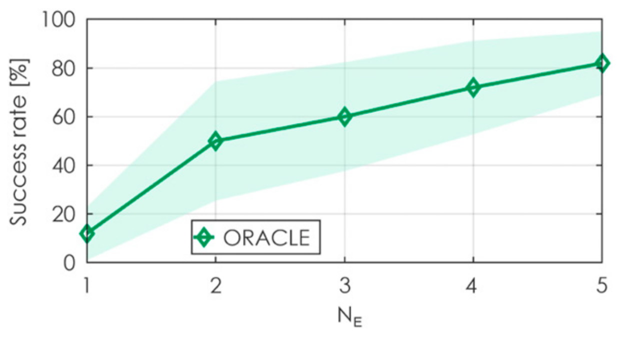

Figure 19. Planning performance when different N_E is utilized in ORACLE and image noise with $d_z = 0.005$ is applied (100 runs were performed). The presented results relate especially to contribution 3.

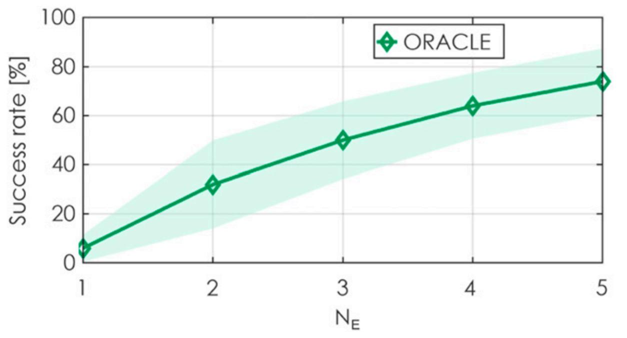

Figure 20. Planning performance when different N_E is utilized in ORACLE and both image noise with $d_z = 0.005$ and velocity noise ($\sigma_v = 0.5$ m/s on all *x*, *y*, *z* axes simultaneously) are applied (100 runs were performed). The presented results relate especially to contribution 3.

Lastly, we also compared ORACLE with the two baselines when both the velocity noise of 0.5 m/s std in all axes and the depth image noise are applied. As depicted in Figure 17, the performance of the *Naive* method drops drastically when the noise level is increased, reaching less than 10% at the highest noise level. On the other hand, the *Ensemble* method shows smaller degradation in the performance but still drops to around 56%-success rate at the highest noise level. On the other hand, ORACLE is the least sensitive to both velocity and depth image noise. Its success rate is around 74% when the noise level is the most significant.

An ablation study is also conducted to study the effect that the number of neural networks in the Deep Ensembles, N_E, has on the planning performance of ORACLE. Specifically, the highest noise levels in Figures 13, 16, and 17 are injected into the robot, and the success rates of ORACLE with different N_E parametrizations are reported in Figures 18–20 ($N_E = 1, \ldots, 5$). As shown, the planning performance generally increases when the ensemble utilizes a larger number of neural networks, albeit at the expense of increased running time. Onboard running time with different numbers of neural networks in the ensemble is analyzed in section 5.3. This analysis reveals the important role of the Deep Ensembles.

While most state-of-the-art learning-based navigation methods do not explicitly account for the uncertainty in the robot's partial state estimate and noisy exteroceptive data that is unseen during the training process (Loquercio et al., 2021; Kaufmann et al., 2020), we demonstrate that using the Unscented Transform and Deep Ensembles (Lakshminarayanan et al., 2017) make our method more resilient against a) noise in the robot's partial state estimate and b) novel noisy depth image inputs, while not relying on consistent position estimate.

Moreover, to verify the performance of our methods in context with the literature, we modified our code to work with the Flightmare simulator Song et al. (2020) and compared our method (ORACLE) alongside its two simplifications (*Naive, Ensemble*) against a state-of-the-art learning-based navigation method for drones, namely, the work in Loquercio et al. (2021) called *"Agile."*

Specifically, forest environments provided by Flightmare where the trees follow a Poisson disc sampling with a density of $\delta = 4.5$ m are chosen to benchmark the methods. The final waypoint is 50 m in front of the robot and the commanded velocity is 2.5 m/s for all the methods. Additionally, a timeout period of 100 s is applied. Notably, ORACLE and its simplifications have not been trained or fine-tuned explicitly for this type of environment, while it is clarified that we have used the pre-trained weights for *Agile* as provided by its authors in order to facilitate fairness. Whereas *Agile* employs a default camera model with a resolution of 640 × 480, FOV of [91, 75]°, and max range of d_{max} = 20 m, our methods (*Naive, Ensemble*, ORACLE) utilize depth data at a 480 × 270 resolution with an FOV of [87, 58]° and d_{max} = 10 m. Both camera models produce data at a frequency of 15 FPS. For all simulations, *Agile* utilizes the Hummingbird quadrotor model in RotorS Furrer et al. (2016) while we use the LMF model described in Section 4.5.2. For collision checking, both robots are modeled—for the purposes of this simulation—as a sphere of radius 0.18 m as this is the default value used by *Agile*. All tests in Flightmare were performed on a desktop with RTX3090 GPU and AMD Ryzen Threadripper 3970X 32-Core CPU with 64 GB of RAM. It is noted that while *Agile* requires access to position information for tracking the trajectory command, our methods (both the planners and the low-level controller) don't assume access to position information (except when calculating the goal vector and checking if the robot has reached the waypoint).

Different noise levels, including a) *"No noise"* where the ground-truth partial state estimates and depth images are given to the robot, b) *"Velocity noise"* where the velocity estimates are deteriorated by additive Gaussian noise with standard deviation σ_v = 0.5 m/s on all x, y, z axes simultaneously, c) *"Image noise"* where the noise model presented in equations (29) and (30) with d_z = 0.004 is employed for the depth images, as well as d) *"Both noise"* where both the noise in cases (b) and (c) are injected to the robot. We randomly created 10 different forest environments as per the previously mentioned parameters. For each such forest environment, 10 runs were performed for each method. The same collision threshold c_{th} = 0.075 is used for *Naive, Ensemble,* and ORACLE, while ORACLE also utilizes a fixed standard deviation for the velocity σ_v = 0.5 m/s which is considered "by default" even in the case that no actual state uncertainty was applied in simulation. The rest of the parameters are kept the same as mentioned earlier. The success (non-collision) rate, alongside the total traveled distance, the average acceleration, and jerk values for all the methods are provided in Table 1 and the robots' trajectories in indicative environments when ORACLE and *Agile* are employed are given in Figure 21.

The above examples demonstrate the performance and robustness characteristics of ORACLE against a state-of-the-art method. As can be seen, ORACLE presents high performance with good generalization in collision-free navigation across simulated forest environments, this is driven by a) it being robust against depth image noise-induced uncertainty through the Deep Ensembles, alongside b) accounting for partial state uncertainty in all cases. As shown in Table 1, the *Naive* method has generally similar performance with Agile but when the Deep Ensembles and the consideration of state uncertainty are factored in, ORACLE significantly outperforms Agile across simulated forest environments and noise conditions. Consideration of depth image noise through the Deep Ensembles supports safe navigation through added conservativeness. The introduced conservativeness promotes the selection of safer paths—even if potentially slightly longer than necessary—which is essential especially in cluttered environments and when operating subject to noisy depth images. Likewise, accounting for partial state uncertainty has similar positive effects. Interestingly, the two remain beneficial even in the "No noise" case as they still offer enhanced conservativeness (e.g., a fixed state uncertainty is considered by ORACLE in any case even when such noise is not presented in the simulated data), as explained in Figure 14. Despite its superior performance when it comes to success ratio, when only successful paths are considered for both methods, ORACLE on average employs longer paths compared to Agile, as illustrated in Figure 21.

5.1.2. Visually-attentive navigation. We conducted a set of simulation studies to evaluate the proposed visually-attentive navigation method (A-ORACLE) with two different sources of visual interestingness detection: visual saliency detection Frintrop et al. (2015) and object detection using YOLO Redmon and Farhadi (2018).

For the case of using saliency as a method to derive detection masks to guide the robot's attention, we use art gallery environments with varying densities (sparse, average, and dense) of salient objects (paintings and furniture) as in Dang et al. (2018) to evaluate A-ORACLE and the baselines. The detection mask $\boldsymbol{\mu}_t$ is derived by thresholding and then normalizing the saliency map output from the visual saliency detection method as described in Frintrop et al. (2015). The simulation environments are depicted in Figure 22.1 and several planning instances are shown in Figure 22.2-3. Specifically, four baselines are compared with A-ORACLE, namely, 1) ORACLE which is described in section 4.2 and utilizes only the CPN for collision-free

Table 1. Comparative Evaluation Metrics for Uncertainty-aware Navigation Simulations With Forest Environments in the Flightmare Simulator Using our Method (ORACLE), its Simplifications (Naive, Ensemble) and the State-of-The-Art Agile Open-source Method. The Mean and Standard Deviation (the Number Enclosed in the Parentheses) of the Metrics are Calculated From 100 Runs (10 Runs per Environment). These Results Relate to Contributions 2-4.

Noise levels	Metrics	Naive	Ensemble	ORACLE	Agile
No noise	Success rate (%)	65 (31)	80 (35)	**90** (10)	55 (29)
	Traveled distance of every run (m)	42.3 (14.8)	47.2 (10.5)	**50.2** (5.4)	38.4 (16.3)
	Traveled distance of successful runs (m)	51.2 (1.5)	51.4 (1.7)	51.4 (2.4)	**48.4** (0.4)
	Average acceleration [m/s^2]	0.7 (0.1)	**0.6** (0.1)	**0.6** (0.1)	1.5 (0.4)
	Average jerk [m/s^3]	5.5 (1.1)	4.6 (1.0)	**4.4** (0.8)	8.6 (2.4)
Velocity noise	Success rate (%)	55 (19)	60 (18)	**82** (15)	35 (25)
	Traveled distance of every run (m)	40.5 (15.3)	39.9 (16.5)	**49.0** (13.1)	33.4 (16.6)
	Traveled distance of successful runs (m)	51.6 (1.5)	51.7 (1.6)	53.9 (5.8)	**50.3** (0.6)
	Average acceleration [m/s^2]	1.0 (0.2)	**0.8** (0.2)	0.9 (0.2)	1.4 (0.2)
	Average jerk [m/s^3]	13.1 (2.2)	8.9 (1.6)	9.7 (1.3)	**8.0** (1.7)
Image noise	Success rate (%)	50 (25)	66 (32)	**73** (34)	32 (28)
	Traveled distance of every run (m)	46.3 (14.6)	45.6 (15.1)	**57.0** (18.1)	31.8 (15.0)
	Traveled distance of successful runs (m)	53.4 (4.2)	55.3 (4.0)	60.6 (16.6)	**48.4** (0.3)
	Average acceleration [m/s^2]	1.0 (0.1)	**0.8** (0.1)	**0.8** (0.1)	1.4 (0.3)
	Average jerk [m/s^3]	8.3 (1.0)	6.3 (1.0)	**5.4** (0.8)	8.5 (1.6)
Both noise	Success rate (%)	32 (16)	53 (16)	**64** (27)	24 (25)
	Traveled distance of every run (m)	42.9 (18.1)	44.1 (15.5)	**52.8** (22.1)	30.2 (15.5)
	Traveled distance of successful runs (m)	54.0 (5.4)	58.8 (9.7)	62.1 (10.8)	**50.2** (0.6)
	Average acceleration [m/s^2]	1.2 (0.1)	**0.9** (0.1)	1.0 (0.1)	1.5 (0.3)
	Average jerk [m/s^3]	15.3 (1.8)	9.9 (1.1)	10.4 (1.3)	**8.5** (1.8)

navigation (no attentive component), 2) the Visual Saliency-aware receding horizon Exploration Planner (VSEP) Dang et al. (2018) which generates exploration paths through a sampling-based planning step first, then another (nested) sampling-based planning step samples and evaluates the intermediate viewpoints to reach the first viewpoint of the exploration path in the most informative manner (in terms of looking towards visually salient areas), 3) an *Expert-baseline* (*"Expert"*) which employs Voxblox Oleynikova et al. (2017) to build the map of the environment incrementally and using the current full occupancy map of the environment to evaluate the information gain for the same action sequence library as our methods using equations ((4), (20), (22), and (24)) the Online Informative Path Planning approach (*"Online IPP"*) described in Schmid et al. (2020) which continuously expands, maintains, and improves a single RRT*-inspired tree of paths while simultaneously executing it. Specifically, the *Online IPP* is modified to utilize equations (20) and (22) to calculate the gain at each node in the tree (thus aligning it with the information gain in A-ORACLE), while the cost of a node is the expected execution time to reach it as per the default choice of the authors. It is noted that for the *Expert* and *Online IPP*, the interestingness level of a voxel is calculated from the current and all past observations, and the decay equation (21) is not used. To obtain the information gain label for training the IPN of A-ORACLE, we perform ray casting with the maximum range of 5 m and the angular resolution of [5, 5]° within the detection camera's frustum and choose $\lambda_1 = 0.9$, $\lambda_2 = 1000$, and a voxel size of 0.2 m. For both VSEP, the *Expert* and *Online IPP*, a voxel resolution of 0.2 m is also used. The depth and detection cameras are simulated with a maximum range of $d_{max} = 10$ m and FOV of $[F_h, F_v] = [87, 58]$°. The MPL consists of $N_{MP} = 256$ action sequences and for each action sequence $N_\Sigma = 7$ sigma points are evaluated. The length of the action sequences in the MPL is $H = 15$ while the time-step duration is $\Delta_t = 0.33$ s, leading to $\mathbf{v}_x^r \leq 2$ m/s from (3). For all simulations of all methods in Section 5.1.2, the reference forward velocity is chosen as $\mathbf{v}_x^r = 0.75$ m/s. Similar to Section 5.1.1, the same collision threshold $c_{th} = 0.1$, time-step weighting factor $\lambda = 0.04$, number of CPNs in the ensemble $N_E = 5$ are used for ORACLE, A-ORACLE, and the *Expert*. To reduce the computation time of the information gain prediction task, for A-ORACLE and *Expert*, we only estimate the information gain at one in every four future time steps, leading to $H_{IPN} = 4$. For VSEP, the maximum number of sampling points in the second planning phase is set to $N_{MP} \times H_{IPN} = 1024$. On the other hand, for the *Online IPP*, maximum $N_{MP} \times H_{IPN} = 1024$ new viewpoints are sampled for each update step of the trajectory tree. Since VSEP constraints a maximum travel time for the visual saliency-aware path, we also tune the timeout time in line 26 of Algorithm 1 so that the total traveled distances of A-ORACLE and VSEP are roughly similar in order to have a fair evaluation. Additionally, because our methods are not built for exploration purposes, we provide the four methods: ORACLE, A-ORACLE, *Expert,* and *Online IPP* with the waypoints defined by the exploration paths from the first planning step of VSEP and allow the methods to deviate from the exploration paths to capture higher quality observations. This is in order to allow all methods to be compared in the task of

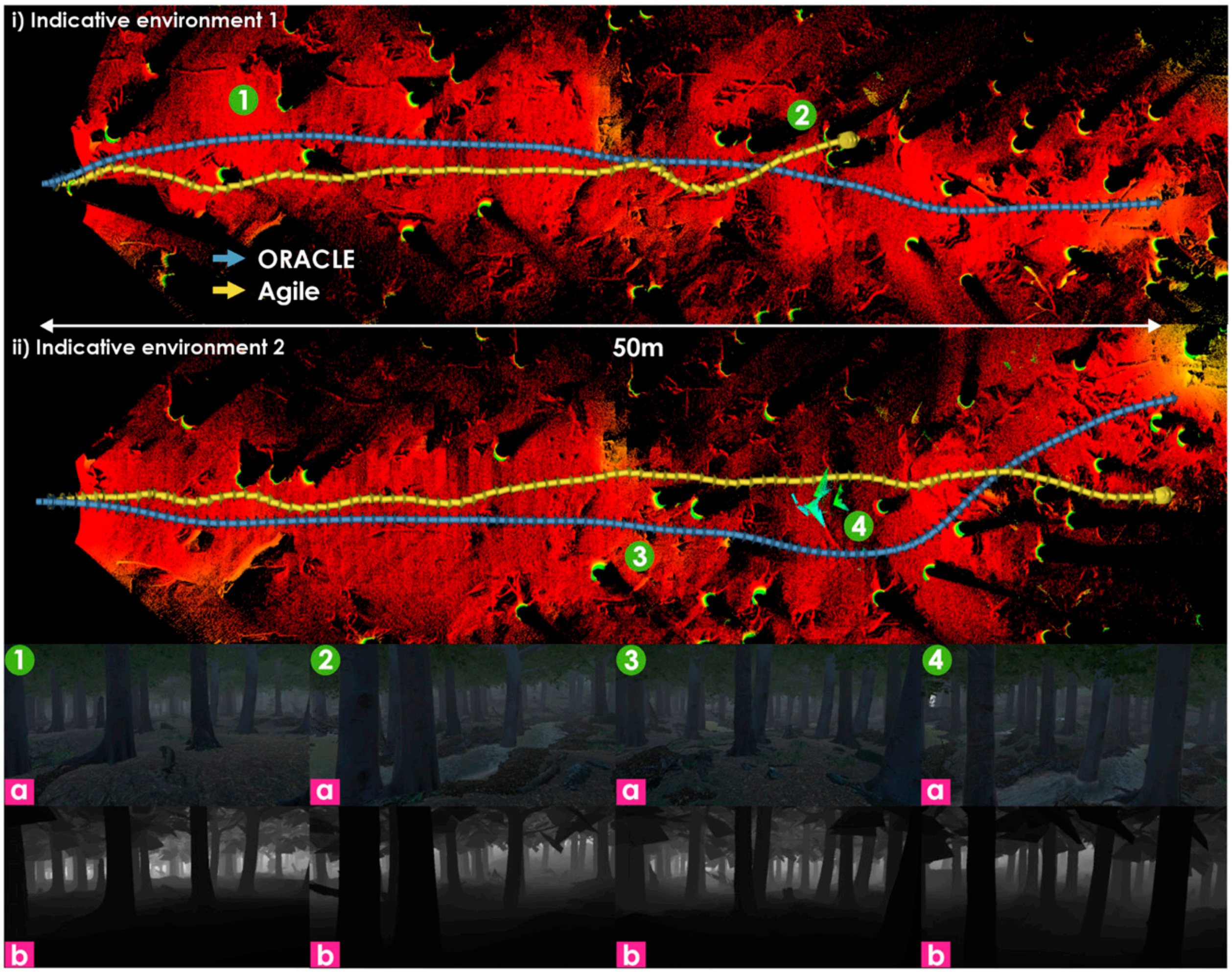

Figure 21. The robots' trajectories in indicative environments using Flightmare where (i) ORACLE succeeds and *Agile* fails and (ii) both ORACLE and *Agile* succeed. The onboard RGB-D images from ORACLE's sensor model are visualized in 1-4a,b (the ground-truth depth images from Flightmare are illustrated here, whereas they are saturated to d_{max} = 10 m before feeding to the CPN). The presented results relate to contributions 2-4.

navigation with implicit information sampling. Notably, the *Online IPP* is modified such that the next-best node in the trajectory tree to be reached at each planning step is the node containing the highest value and is within the neighborhood of the next target waypoint. A timeout value similar to A-ORACLE is also applied in the *Online IPP*. When a timeout event happens, the robot will follow the straight-line connection between the robot's current position and the chosen next-best node if this connection lies in the known collision-free space, otherwise, the remaining of the present planned path will be fully executed. It is noted that while VSEP and the *Online IPP* methods derive informative paths and execute them until the end before replanning and the *Expert* can only replan at the rate of 1 Hz, ORACLE and A-ORACLE replan at 5 Hz-rate (or possibly higher) in this simulation study due to their small processing time, as can be seen in the last row of Table 2.

To compare the five methods, we run the Voxblox mapping framework and annotate each voxel based on the saliency mask using equation (20). A valid interesting voxel k is defined as a voxel being observed in at least N_{th} camera frames and having interestingness $I^k > I_{th}$. For each valid interesting voxel, we also logged its minimum viewing distance from all observed camera frames. Figure 23 shows the percentage of valid interesting voxels, calculated based on the total number of valid interesting voxels seen by the *Expert*, plotted against their minimum viewing distances for each method. It can be seen that Attentive ORACLE views more valid interesting voxels from closer distances than ORACLE and VSEP, leading to a higher quality of observations of the objects. Table 2 presents the average metrics of 10 runs/environment for each method. As depicted, the number of valid interesting voxels observed by A-ORACLE is $1.08 - 1.48$ times that of those observed by ORACLE, $0.98 - 1.23$ times that seen by VSEP, $0.82 - 0.93$ times that observed by the *Expert*, and $0.82 - 0.89$ times that of those viewed by the *Online IPP*, while having average travel distances that are very similar to those of VSEP and the *Expert*, and only $0.5 - 0.6$ times that of those

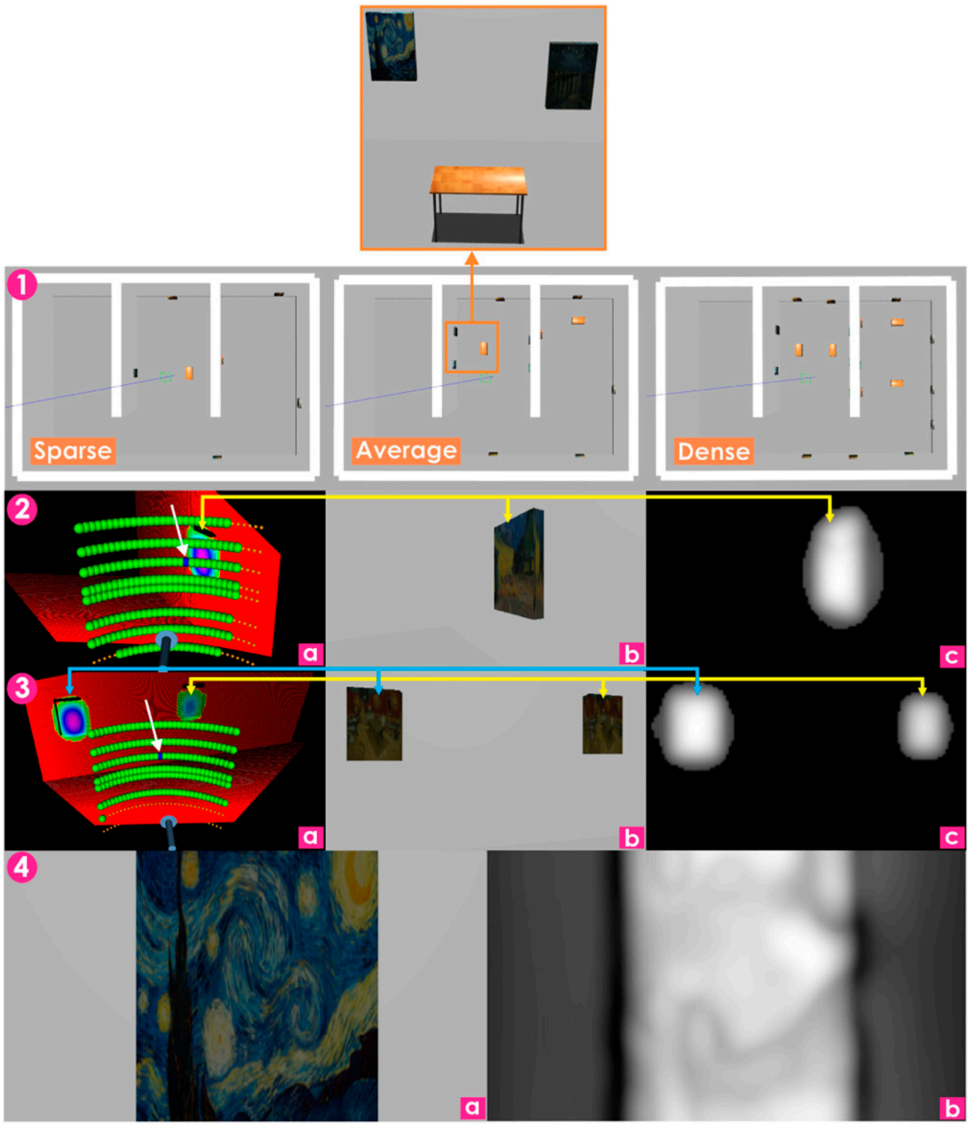

Figure 22. (1) Art gallery environments for visually-attentive navigation simulation with the detection masks derived from the saliency maps, the salient objects are the paintings and furniture (visualized in the orange boxes). (2) and (3) show specific planning instances with the point clouds annotated with saliency values from the detector and the network predictions (a), images from the onboard RGB camera (b), and saliency detection mask (c). Green markers: estimated trajectory endpoints of safe action sequences, blue marker with an arrow: estimated trajectory endpoint of chosen action sequence. (4) Illustration of a saliency mask (b) obtained from an onboard RGB image (a) when the robot is spawned in front of a painting. The presented results relate to contribution 1.

covered by the *Online IPP*. It is stressed that the *Online IPP* can plan the viewpoints outside the current FOV of the robot's depth camera, as opposed to the MPL utilized in ORACLE, A-ORACLE, and the *Expert*. Notably, the average inference time of A-ORACLE is just 6.3% of that of the *Expert*, 10.2% of that of VSEP, and 2.1% of that of *Online IPP*, while managing to achieve comparable or better performance than VSEP and comparable performance with the *Expert*. Figure 22.2-3 illustrates specific instances with the point clouds annotated with saliency values and the network predictions (left column), the onboard RGB image (middle column), and the detection mask (right column) where the brighter the color, the higher the saliency value.

To demonstrate that our method can work with multiple visual detection input sources, we also verified A-ORACLE using the output of the YOLO object detector Redmon and Farhadi (2018), as trained for the DARPA Subterranean Challenge by Team CERBERUS Tranzatto et al. (2022), as a cue for interestingness. Specifically, the detection mask $\boldsymbol{\mu}_t$ is created by assigning the interestingness values of 1 to all the pixels inside the detected objects' bounding boxes. We tested three methods: A-ORACLE, ORACLE, and the *Expert* in a realistic 3D subway station environment where the waypoints are given in a lawn mower pattern, as depicted in Figure 24(a)–(b). VSEP is not evaluated in this case since it requires a very high number of sampling points to find the feasible path in this large environment. The poses

Table 2. Evaluation metrics for visually-attentive navigation simulations with saliency detection inputs in art gallery environments. The metrics displayed in the table include 1) the number of valid interesting voxels (valid interesting voxels), 2) the volume of valid interesting voxels (volume), 3) the total traveled distance, and 4) the processing time. The average and standard deviation (the number enclosed in the parentheses) of processing time is calculated from 500 planning iterations on a laptop with AMD Ryzen 9 4900HS CPU and RTX 2060 GPU. These results relate to contribution 1.

Environments	Metrics	ORACLE	VSEP	Expert	A-ORACLE	Online IPP
Sparse	Valid interesting voxels	1965	2377	3131	2915	3276
	Volume (m^3)	15.72	19.02	25.05	23.32	26.21
	Traveled distance (m)	113.9	155.0	177.6	152.5	288.4
Average	Valid interesting voxels	3510	3884	4641	3799	4397
	Volume (m^3)	28.08	31.07	37.13	30.39	35.18
	Traveled distance (m)	105.9	161.2	180.9	151.5	302.7
Dense	Valid interesting voxels	5524	7036	8818	7510	9213
	Volume (m^3)	44.19	56.29	70.54	60.08	73.70
	Traveled distance (m)	110.0	158.6	173.5	154.5	260.8
	Processing time (ms)	21.3 (4.6)	423.8 (151.1)	688.3 (330.1)	43.4 (7.1)	2110 (1804)

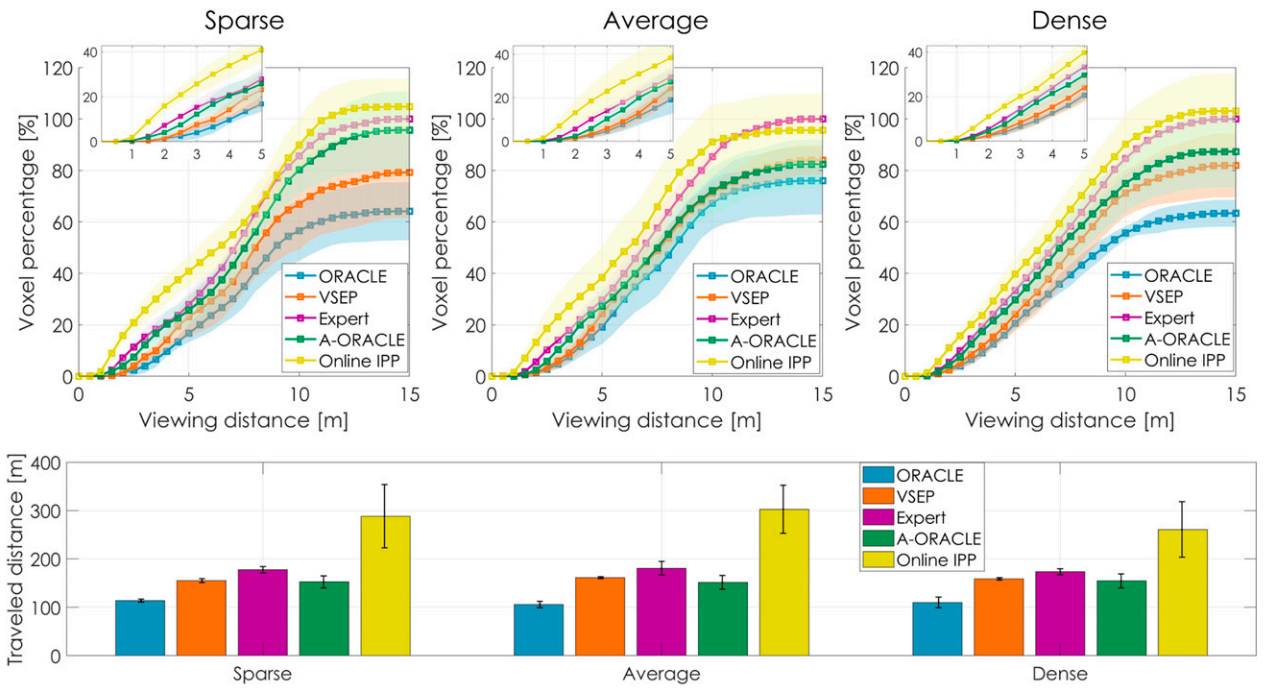

Figure 23. Simulation results for visually-attentive navigation with saliency detection inputs in art gallery environments. Top row: The x-axis shows the minimum viewing distances for the valid interesting voxels and the y-axis shows the percentage of seen valid interesting voxels (average value and $1-\sigma$ boundaries of 10 runs/environment), with respect to the *Expert*, having minimum viewing distances less than x. Bottom row: the mean and $1-\sigma$ error bar of the total traveled distance of each method. The presented results relate to contribution 1.

of the objects are randomized to create 10 different environments and the simulation parameters are the same as the experiments with saliency detection input. Similar to the simulations with saliency input, Voxblox is also run for evaluating the three methods.

Figure 25 shows the percentage of valid interesting voxels, calculated based on the total number of valid interesting voxels seen by the *Expert*, plotted against their minimum viewing distances for each method. It can be seen that A-ORACLE views more valid interesting voxels from closer distances than ORACLE, leading to a higher quality of observations of the objects. Table 3 presents the average metrics of 10 environments for each method. As depicted, the number of valid interesting voxels observed by A-ORACLE is 15% more than that of those observed by ORACLE, and 9% less than that observed by the Expert-baseline, while having an average travel distance that is only 9.5% longer than that of ORACLE (and 7.1% less than that of the *Expert*). Figure 24(b) shows the robot's trajectory when A-ORACLE is deployed in a specific environment where the red backpacks visualized are the objects of interest.

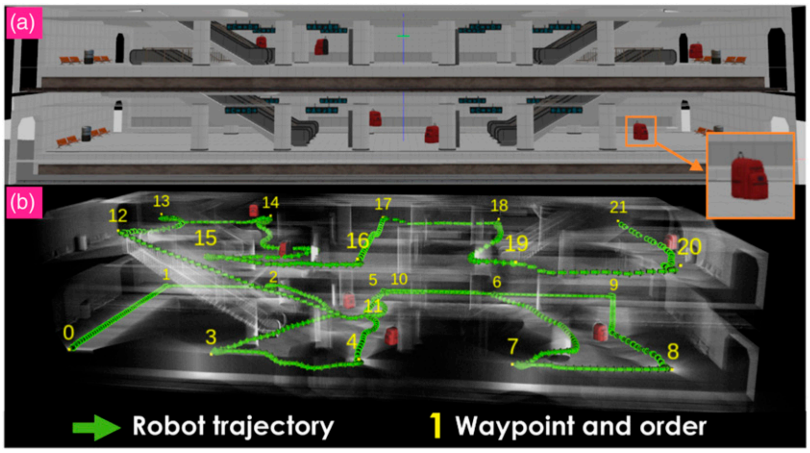

Figure 24. (a) Subway station environments for visually-attentive navigation simulation with YOLO detections as attention input. The red backpacks visualized in the image are the objects of interest in this case. (b) The robot's trajectory when A-ORACLE is deployed and the waypoints marked with the numbers representing the visiting order. The presented results relate to contribution 1.

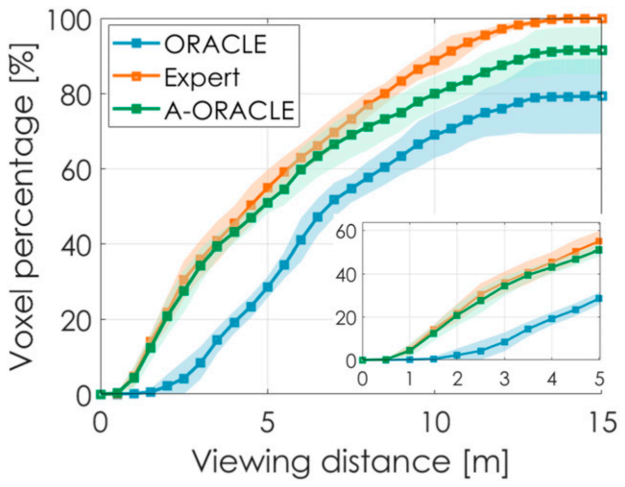

Figure 25. Simulation results for visually-attentive navigation based on YOLO detection inputs in subway station environments. The x-axis shows the minimum viewing distances for the valid interesting voxels and the y-axis shows the percentage of seen valid interesting voxels (average value and $1-\sigma$ boundaries of 10 runs), with respect to the *Expert*, having minimum viewing distances less than x. The presented results relate to contribution 1.

Table 3. Evaluation Metrics for Visually-Attentive Navigation Simulations Based on YOLO Detection Inputs in subway Station Environments. All Distances are in Meters, and all Volumes are in Cubic Meters. These Results Relate to Contribution 1.

Metrics	ORACLE	Expert	A-ORACLE
Number of valid interesting voxels	1233	1566	1423
Volume of valid interesting voxels	9.86	12.53	11.38
Traveled distance	296.8	350.1	325.1

5.2. Experimental studies

Moreover, to verify our methods in the real system, we performed a set of experiments in both pure navigation and visually-attentive navigation tasks with the robot platform described in Section 4.5.2. In all experiments, the position information was not required by ORACLE or A-ORACLE except for calculating the unit goal vector $\mathbf{n}_t^g$ and checking if the robot had reached the waypoints. The parameters used for all experiments are listed in Table 4.

Since the Intel Realsense T265 estimation output does not contain covariance information, we used a fixed standard deviation of $\sigma_v = 0.2$ m/s for the velocity noise. It is also noted that compared to the simulated parameters in Section 5.1, we used an ensemble of $N_E = 3$ neural networks in all real-world experiments and used $N_{MP} = 96$ with A-ORACLE to reduce the inference time of the CPN and the IPN onboard the robot. The running times of different components of ORACLE and A-ORACLE are provided in Section 5.3. Additionally, we used a lower threshold c_{th} for the first and second experiments compared to the other experiments to allow safer navigation when the robot was tasked to fly faster in more cluttered environments. We also chose $\lambda = 0.08$ in the second experiment in which the environment is the most cluttered, as shown in Figure 27, to prioritize the predictions at smaller time steps.

In the first experiment, illustrated in Figure 26 and Extension 1, the robot was tasked to reach a waypoint that is in front of it with the reference forward speed of $\mathbf{v}_x^r = 2.5$ m/s while navigating safely in a cluttered corridor filled with various types of obstacles. Figure 26.1-3 presents predictions of the CPN at some specific scenarios, where the end of trajectories are generated only for visualization purposes based on $\mathbf{s}_t$ and the MPL using the estimated dynamics models of the robot. The green dots correspond to the action sequences that pass the collision cost threshold check in line 25 of Algorithm 1, while the blue dot corresponds to the best action sequence chosen in line 32 of Algorithm 1. As shown, the visualized trajectories correlate well with the collision cost predicted by the CPN, showing

Table 4. Experiment parameters.

Parameter	ORACLE (1st experiment)	ORACLE (2nd experiment)	A-ORACLE (3rd, 4th experiments)
$\mathbf{v}_x^r$	2.5 m/s	1.5 m/s	0.6 m/s
N_{MP}	256	256	96
N_E	3	3	3
N_Σ	7	7	7
σ_v	0.2 m/s	0.2 m/s	0.2 m/s
H	14	14	15
H_{IPN}	None	None	4
Δ_t	0.2 s	0.2 s	0.33 s
λ	0.04	0.08	0.04
c_{th}	0.01	0.05	0.1

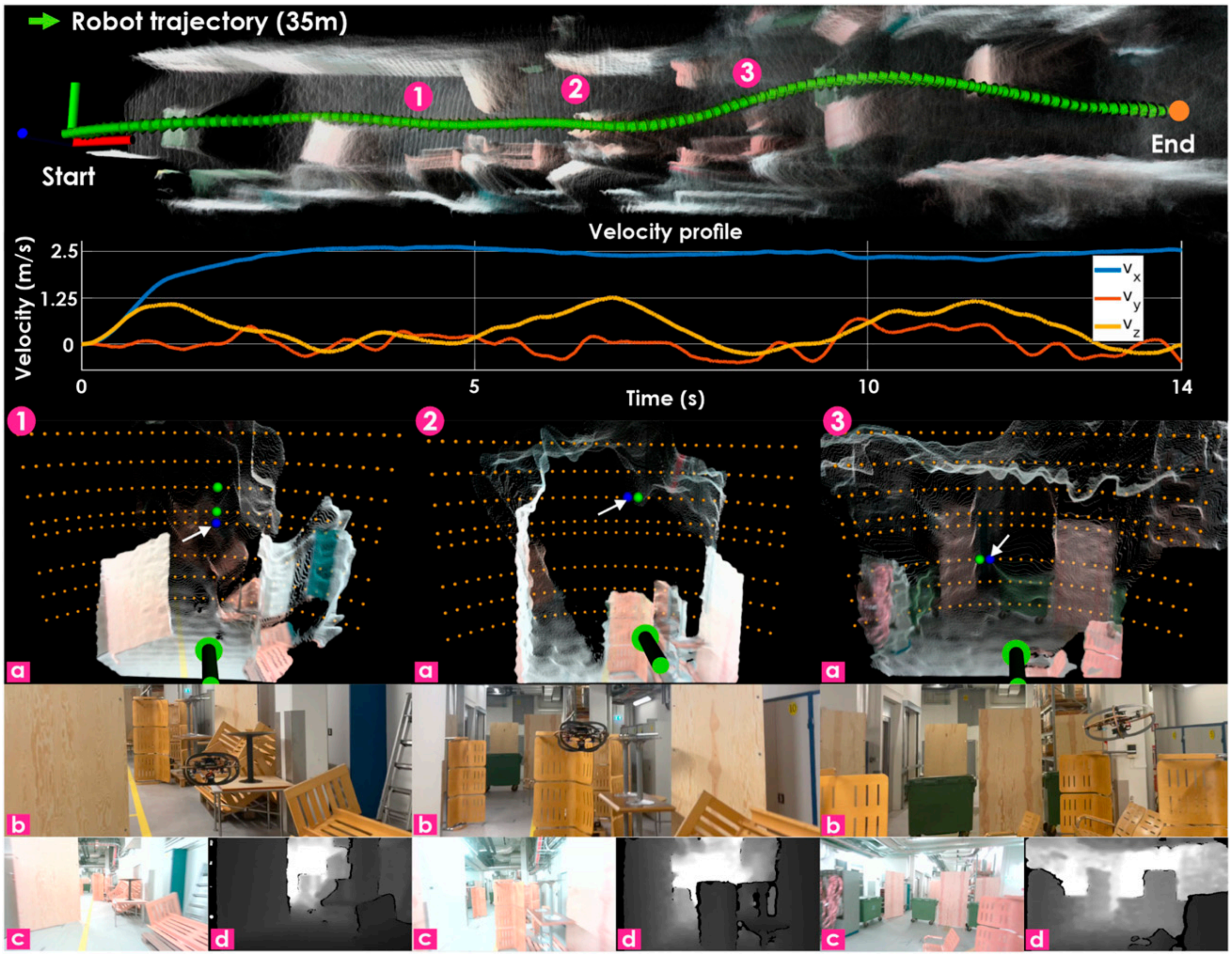

Figure 26. Experiment 1: experiment with ORACLE in a corridor filled with obstacles. The map of the environment, reconstructed from the Realsense T265's odometry and the Realsense D455's pointclouds, is given in the top row while some instances of the experiment are shown in 1-3 where the predictions from the CPN are illustrated in 1-3a (green markers: estimated trajectory endpoints of safe action sequences, blue marker with an arrow: estimated trajectory endpoint of chosen action sequence), the third-person views are displayed in 1-3b, and the onboard RGB-D images are visualized in 1-3c,d, respectively. The robot was commanded to fly toward a waypoint in front of it with the reference forward velocity of 2.5 m/s, as shown in the velocity profile plot. The presented results relate to contributions 2-4.

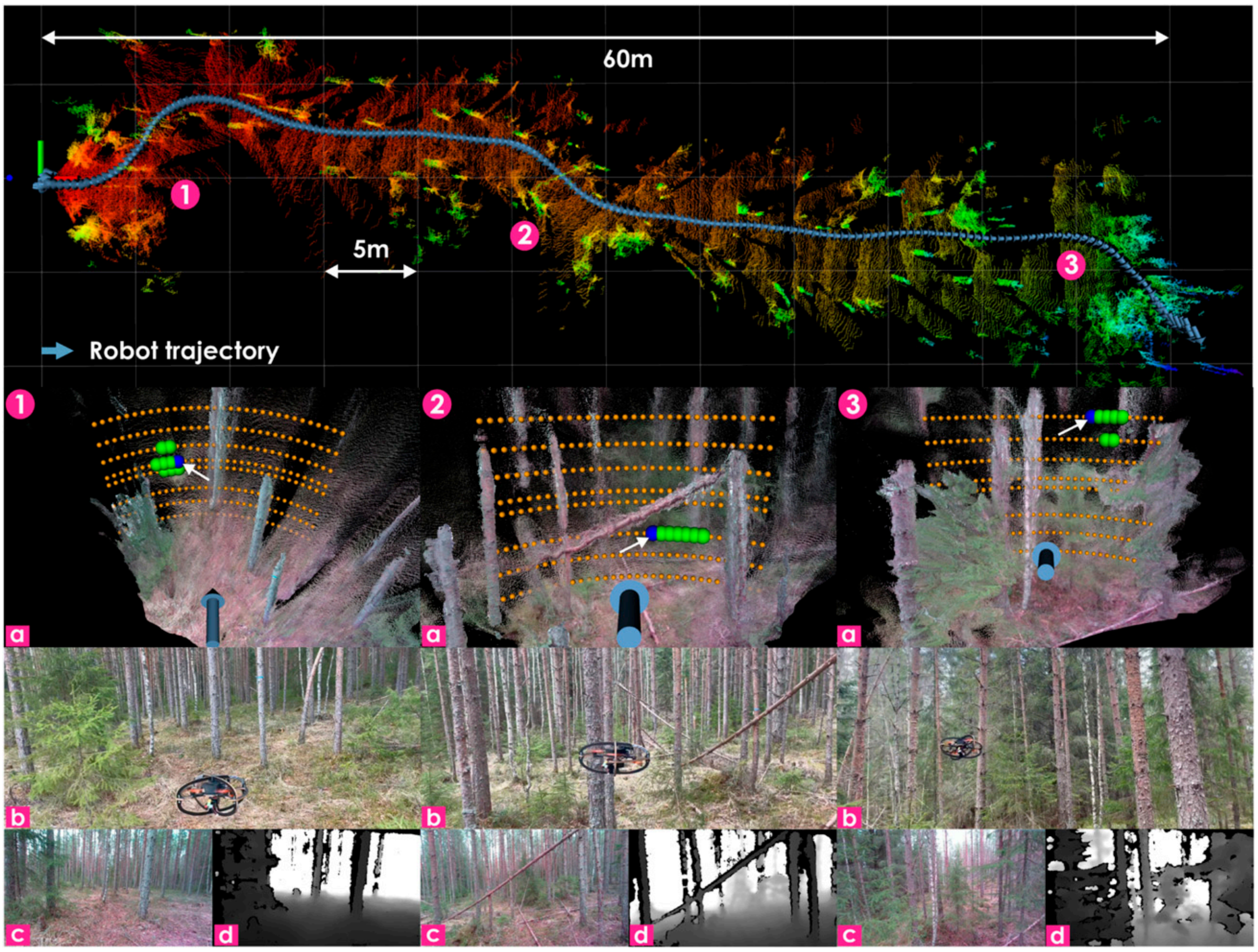

Figure 27. Experiment 2: experiment with ORACLE in a dense forest during under canopy flight. The maps of the environment and the odometry estimates of the robot, derived by RTAB Labbé and Michaud (2019), are given in the top row while some instances of the experiment are shown in 1-3 where the predictions from the CPN are illustrated in 1-3a (green markers: estimated trajectory endpoints of safe action sequences, blue marker with an arrow: estimated trajectory endpoint of chosen action sequence), the third-person views are displayed in 1-3b, and the onboard RGB-D images are visualized in 1-3c,d, respectively. The robot was commanded to fly towards a waypoint in front of it with the reference forward velocity of 1.5 m/s. Each square in the figure has dimensions equal to 5 m × 5 m. Note that the position estimates are not provided to the robot during the mission and the results of RTAB-based map are only derived in post-processing given the relative drift experienced by the onboard T265 odometry. The presented results relate to contributions 2-4.

the reliable performance of the CPN in real-world situations. The velocity profile is also given in Figure 26 where the z-component of the velocity is utilized to avoid obstacles in some instances, showing the benefit of navigating in full 3D compared to our prior (2D) work presented in Nguyen et al. (2022).

The second experiment related to a forest during under canopy flight and took place near Evo, Finland, and was presented in Extension 2. The robot was commanded to navigate safely towards a waypoint that is in front of it with reference forward speed of $\mathbf{v}_x^r = 1.5$ m/s. Figure 27.1-3 presents predictions of the CPN in some particular instances. It is noted that a large part of this environment has a density of around 0.2 trees/m^2, which corresponds to the densest forests simulated in Bartolomei et al. (2023) and 5 times more than the densest forests simulated in Loquercio et al. (2021) $\left(\text{with a density of } \frac{1}{25} \text{ trees/m}^2\right)$. Additionally, this environment presents challenging conditions for the navigation methods since thin tree branches are abundant, as can be seen in Figure 27.1-3(b),(c),(d). As shown, ORACLE can negotiate this environment successfully although it has never collected data in cluttered forests in simulation, demonstrating the generalization capability of our method.

In the third experiment, we performed flight tests with A-ORACLE and ORACLE in an industrial silo tank at the RelyOn training facility in Trondheim, Norway, as presented in Figure 28 and Extension 3. The robot was tasked to navigate safely in the environment following a pre-defined set of waypoints, while it was allowed to deviate from the intended waypoints to gather higher quality observations of objects of interest (a backpack and a protective suit simulating a human in this case). YOLO Redmon and Farhadi (2018), as trained for the DARPA Subterranean Challenge by Team CERBERUS Tranzatto et al. (2022), was utilized as the object detection algorithm, and its output detection masks are depicted in Figure 28.1-2(d). While

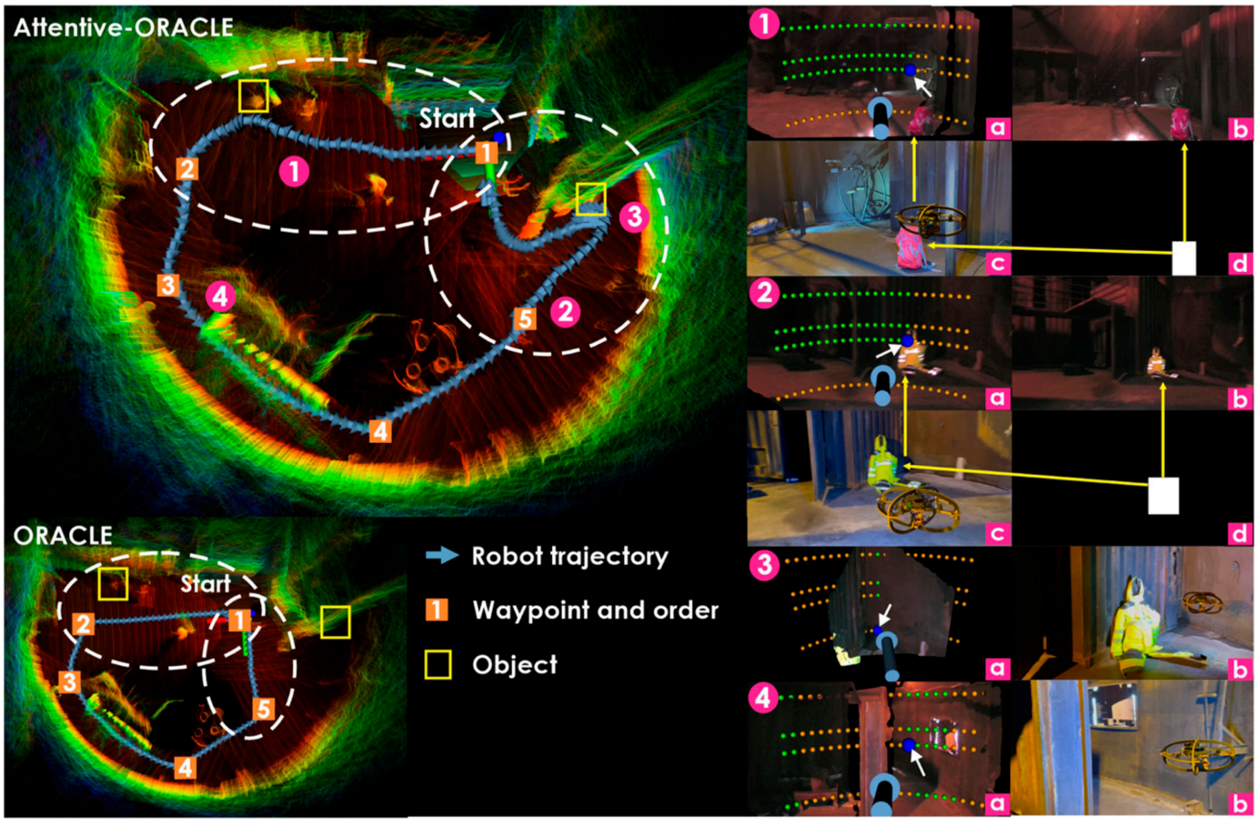

Figure 28. Experiment 3: experiments with both A-ORACLE and ORACLE in an industrial silo tank. The map of the environment (reconstructed from the Realsense T265's odometry and the Realsense D455's pointclouds), the given waypoints, and the trajectories taken by the robot when ORACLE and A-ORACLE are deployed are shown on the left. As shown in the white ellipses, the robot traversed closer to the interesting objects, which are marked with yellow boxes and visualized in 1-2c, when A-ORACLE was engaged compared to ORACLE. Some instances of the experiment with A-ORACLE are shown in 1-4. The predictions from the CPN are illustrated in 1-4a where the green markers correspond to the estimated trajectory endpoints of safe action sequences, and the blue marker with an arrow corresponds to the estimated trajectory endpoint of chosen action sequence (determined using both the prediction results from the CPN and the IPN). The third-person views are displayed in 1-2c, and 3-4b, while the onboard RGB images and detection masks from YOLO are visualized in 1-2b, and d, respectively. The presented results relate to contributions 1-4.

moving from waypoint 1 to 2 and 5 to 1 (marked with the white ellipses in Figure 28), the robot deviated from the straight-line connection between the waypoints in order to look at the objects of interest from closer distances, as illustrated in Figure 28.1-2(b). For comparison, ORACLE was also deployed with the same set of waypoints and the trajectory of the robot is visualized in the bottom left of Figure 28. As can be seen, since ORACLE does not consider the quality of observations of interesting objects, straight-line connections between the waypoints were usually chosen (except when moving from waypoint 3 to 4). It is noted that the straight-line connection between waypoints 3 and 4 is not collision-free. Notably, when the robot traversed closer to the survivor when A-ORACLE was engaged, it detected a dead end, depicted in Figure 28.3, and performed a yaw-in-spot action until it found a free direction, as presented in line 21 of Algorithm 1. Figure 28.4 shows the CPN's prediction around waypoint 3, demonstrating the capability of our methods to provide a multi-modal navigation solution where the robot can choose to turn left or right to avoid the front obstacle.

The fourth experiment, as seen in Extension 4, was conducted in a hall inside a building on the campus of NTNU. The robot was given a waypoint that is in front of it and the straight-line connection between the start and end points is not collision-free. Three backpacks were placed along the hall to represent the objects of interest and YOLO Redmon and Farhadi (2018) was again utilized to detect the objects. Similar to the second experiment, when A-ORACLE was deployed, the robot traversed closer to the objects of interest to view them from smaller distances. Notably, in this environment, the position estimates from the Realsense T265 drifted significantly, possibly due to the darkness in some parts of the environment, as can be seen from the onboard RGB image in Figure 29.c. The ground-truth reconstructed maps and odometry estimates of the robot, visualized in the top row (A-ORACLE) and the left column in the second row of Figure 29 (ORACLE), are estimated offline using the method presented in Labbé and Michaud (2019). The drifted map with wrong dimensions (25 m versus 40 m) and odometry estimates from Realsense T265 are visualized in the right column in the second row of

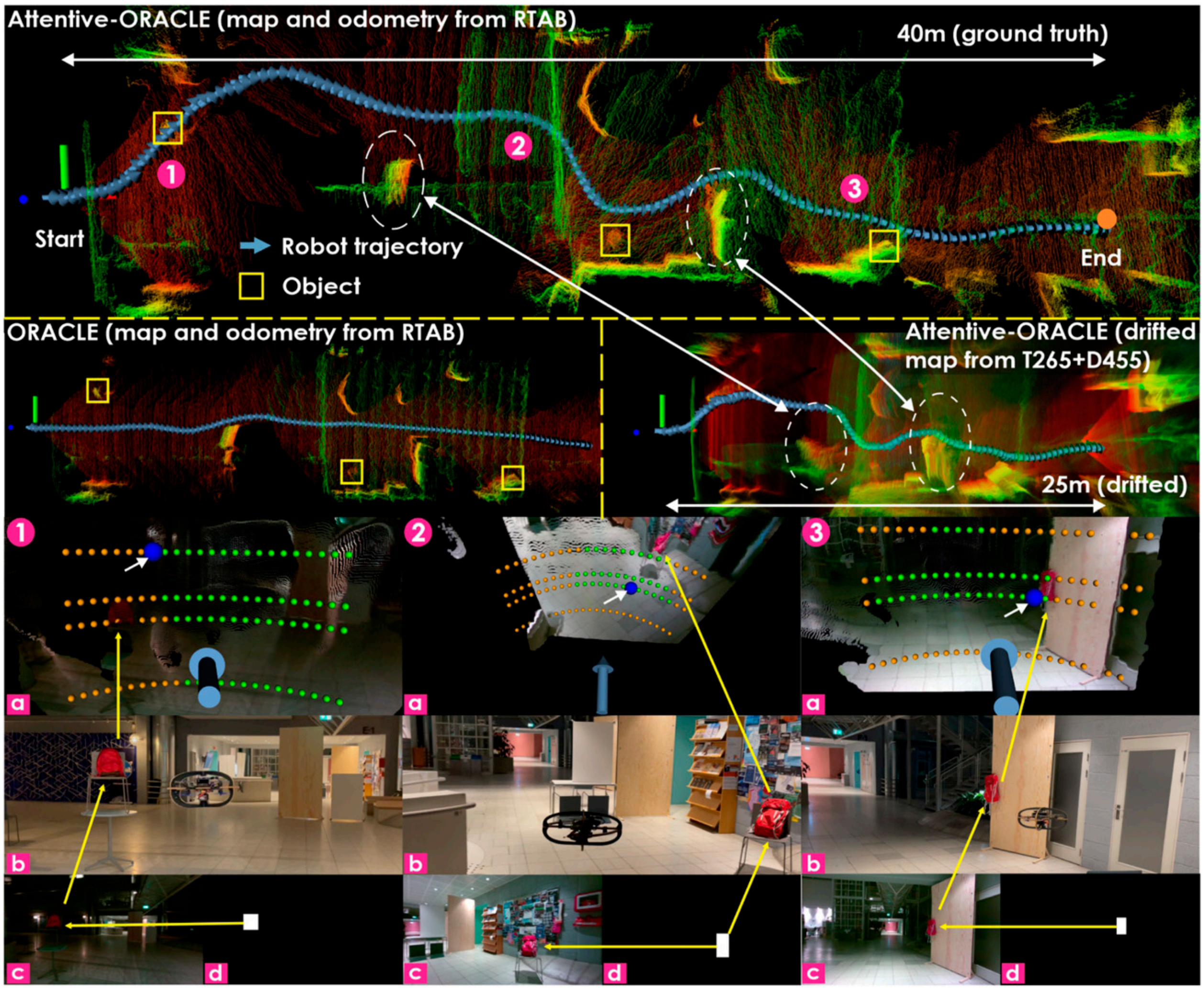
Attentive-ORACLE (map and odometry from RTAB)
40m (ground truth)
Start
End
Robot trajectory
Object
1
2
3
ORACLE (map and odometry from RTAB)
Attentive-ORACLE (drifted map from T265+D455)
25m (drifted)
a
b
c
d

Figure 29. Experiment 4: experiments with both A-ORACLE and ORACLE in a hall inside a building on the campus of NTNU. The maps of the environment and the odometry estimates of the robot, derived by RTAB Labbé and Michaud (2019), are shown in the first row (A-ORACLE) and the left plot in the second row (ORACLE). On the other hand, the drifted map of the environment, reconstructed from the Realsense T265's odometry, and the Realsense T265's odometry solution are shown in the right plot in the second row. Some instances of the experiment with A-ORACLE are shown in 1-3. The predictions from the CPN are illustrated in 1-3a where the green markers correspond to the estimated trajectory endpoints of safe action sequences, and the blue marker with an arrow corresponds to the estimated trajectory endpoint of chosen action sequence (determined using both the prediction results from the CPN and the IPN). The third-person views are displayed in 1-3b, while the onboard RGB images and detection results from YOLO are visualized in 1-3c, and d, respectively. Owing to its design, A-ORACLE and ORACLE can still avoid obstacles and additionally, A-ORACLE can pay attention to interesting objects, marked with yellow boxes, despite the significant drift of the position estimation of Realsense T265. The presented results relate to contributions 1-4.

Figure 29. A-ORACLE and ORACLE could still avoid obstacles and additionally, A-ORACLE could pay attention to interesting objects in this case despite the significant drift of the position estimates.

5.3. Onboard running time

The running time of the ORACLE/A-ORACLE methods consists of three computational components, namely a) the depth image pre-processing step ("Pre-processing"), b) multiple forward passes through the CPN (ORACLE) or the CPN and the IPN (A-ORACLE) on the GPU, and c) other operations including data transfer between the CPU and the GPU as well as remaining CPU's operations ("Others"). The actual onboard running times for ORACLE and A-ORACLE with the configurations presented in Table 4 are detailed in Tables 5 and 6. The utilized Xavier NX operates in 15 W 6-core mode in all the computational evaluations and real-world experiments presented, while NVIDIA TensorRT is used to optimize the CPN and IPN. It is noted that in practice, the number of CPNs in the ensemble N_E does affect the running time, as can be seen in Table 5. We choose to use $N_E = 3$ in all real-world experiments, allowing the

Table 5. Onboard running time of different components of ORACLE (N_E is varied while the other parameters are the same as in the first experiment mentioned in Tables 4 and it is noted That $N_E = 3$ is Used in all Real-world Experiments.). All the times are in milliseconds.

N_E	Step	Mean	Percentage	Total
1	Pre-processing	15.7	36.3	43.2
	CPN	18.7	43.3	
	Others	8.8	20.4	
2	Pre-processing	15.7	32.2	48.8
	CPN	19.5	40.0	
	Others	13.6	27.8	
3	Pre-processing	15.7	28.6	**54.9**
	CPN	22.8	41.5	
	Others	16.4	29.9	
4	Pre-processing	15.7	24.3	64.5
	CPN	29.1	45.1	
	Others	19.7	30.6	
5	Pre-processing	15.7	21.2	74.0
	CPN	36.5	49.3	
	Others	21.8	29.5	

Table 6. Onboard running time of different components of A-ORACLE in the third and fourth experiments. All the times are in milliseconds.

Step	Mean	Percentage	Total
Pre-processing	15.7	18.5	85.0
CPN	19.5	22.9	
IPN	38.1	44.8	
Others	11.7	13.8	

planning rate of 15 Hz in the first experiment and 5 Hz in the two other experiments when YOLO is also running alongside A-ORALCE with the same rate. Notably, with the same $N_E = 3$, the running time of the CPN in the first experiment (when $N_{MP} = 256$) only increases by 17% compared to the other experiments (when $N_{MP} = 96$ and the action's sequence length H is almost the same). It can be seen that by exploiting the GPU's computing capability, the running time of our methods scales gracefully with the number of action sequences in the MPL (N_{MP}) and the number of CPNs in the ensemble (N_E). The overall sufficiently low computational times and the ablation study summarized in Figures 18–20 allows the appropriate selection of the key parameter value N_E for a certain robot's capabilities and mission demands.

6. Conclusions

This paper presented a learning-based method to efficiently tackle the problem of visually-attentive uncertainty-aware 3D navigation without relying on a map of the environment or the position estimate of the robot. Two neural networks are designed in this work: a Collision Prediction Network for predicting the uncertainty-aware collision costs for action sequences in a Motion Primitives Library (utilizing the Unscented Transform and an ensemble of neural networks) and an Information gain Prediction Network for estimating their associated information gain. The networks' outputs are used in addition to a unit goal vector, given by any high-level global planner, to determine the best action sequence to be executed in a receding horizon fashion. We conducted a set of simulations and real-world experiments to verify the proposed method. Extensive simulation studies involving navigation with noisy inputs including the robot's velocity estimate and the depth image demonstrate the robustness of our methods (ORACLE and A-ORACLE). Moreover, visual attention-aware navigation with different sources of visual detection input is performed to show the benefits of A-ORACLE compared to other baselines. Finally, several real-world experiments including collision-free flights with the reference forward speed of 2.5 m/s in a cluttered corridor, and visually-attentive navigation in industrial and university environments are also described, demonstrating that the method can transfer well to real systems and complex environments. The code and training datasets will be publicly released at https://github.com/ntnu-arl/ORACLE upon acceptance.

Regarding future work, five important directions are identified. First, this relates to extending the exteroceptive sensor inputs of the method to enable multimodal fusion, especially of depth and visual data. Visual data can deliver the resolution and acuity typically lacking in depth images, while co-fusing depth data allows to benefit from the more direct collision information they offer and their ability to be simulated with higher fidelity which supports successful sim-to-real transfer. This direction of future work is especially motivated by our experience from field testing within dense forests including hard-to-detect thin branches (e.g., with a cross section less than 1 cm) where the depth camera faced limitations in its ability to correctly provide range information. Second, we aim to investigate the potential of offering safety certificates in order to not only have high performance in statistical terms but guarantee the system's safety. A plausible direction is that of developing a safety filter through control barrier functions. This is a critical domain of research that aims to address core limitations of neural network-based methods within critical tasks such as robot control and navigation. Third, also related with the previous direction, we aim to investigate the online detection of situations where the robot is exposed to inferring from data significantly different from those experienced during training. The latter could be used to trigger another fallback system to safeguard the robot or its environment. Fourth, the method can be extended to use a sequence of depth images with the goal of handling dynamic obstacles. Finally, future work may focus on alleviating the limitation of hand-tuning certain parameters in the loss equations used to train the CPN and the IPN by automatically learning such weights at training time.

Acknowledgments

We appreciate the help of the employees at RelyOn Nutec Trondheim, who provided us access to their facilities for conducting experiments.

Declaration of conflicting interests

The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Funding

The author(s) disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: This work was supported by the This material was a) supported by Research Council of Norway under project SENTIENT (grant number 321435) and also b) based upon work supported by the Air Force Office of Scientific Research under award number FA8655-21-1-7033.

ORCID iDs

Huan Nguyen https://orcid.org/0000-0002-0159-1936
Rasmus Andersen https://orcid.org/0000-0001-5867-193X
Evangelos Boukas https://orcid.org/0000-0002-9919-5746

Supplemental Material

Supplemental material for this article is available online.

References

Abdar M, Pourpanah F, Hussain S, et al. (2021) A review of uncertainty quantification in deep learning: techniques, applications and challenges. *Information Fusion* 76: 243–297.

Abdelaziz A, Watanabe S, Hershey J, et al. (2015) Uncertainty propagation through deep neural networks. *Interspeech* 2015: 3561–3565.

Achtelik MW, Lynen S, Weiss S, et al. (2014) Motion-and uncertainty-aware path planning for micro aerial vehicles. *Journal of Field Robotics* 31(4): 676–698.

Agha-mohammadi A, Agarwal S, Kim SK, et al. (2018) Slap: simultaneous localization and planning under uncertainty via dynamic replanning in belief space. *IEEE Transactions on Robotics* 34(5): 1195–1214.

Ahn MS, Chae H, Noh D, et al. (2019) Analysis and noise modeling of the intel realsense d435 for mobile robots 2019 16th International Conference on Ubiquitous Robots (UR), 24-27 June 2019. Jeju, Korea (South): 707–711.

Amini A, Soleimany AP, Karaman S, et al. (2017) Spatial uncertainty sampling for end-to-end control 31st International Conference on Neural Information Processing Systems (NeurIPS), Bayesian Deep Learning Workshop. Long Beach, California, USA. 4-9 December 2017.

Astudillo RF and Neto JP (2011) Propagation of uncertainty through multilayer perceptrons for robust automatic speech recognition. *Interspeech* 2011: 461–464.

Bartolomei L, Teixeira L and Chli M (2023) Fast multi-UAV decentralized exploration of forests. *IEEE Robotics and Automation Letters* 8(9): 5576–5583.

Brescianini D, Hehn M and D'Andrea R (2013) *Nonlinear Quadrocopter Attitude Control*. Zürich, Switzerland: ETH Zurich. Technical report.

Bry A and Roy N (2011) Rapidly-exploring random belief trees for motion planning under uncertainty 2011 IEEE International Conference on Robotics and Automation (ICRA), Shanghai, China. 9-13 May 2011: 723–730.

Bucki N, Lee J and Mueller MW (2020) Rectangular pyramid partitioning using integrated depth sensors (rappids): a fast planner for multicopter navigation. *IEEE Robotics and Automation Letters* 5(3): 4626–4633.

Cadena C, Carlone L, Carrillo H, et al. (2016) Past, present, and future of simultaneous localization and mapping: toward the robust-perception age. *IEEE Transactions on Robotics* 32: 1309–1332.

Castro MG, Triest S, Wang W, et al. (2023) How does it feel? self-supervised costmap learning for off-road vehicle traversability 2023 IEEE International Conference on Robotics and Automation (ICRA), London, UK. 29 May 2023 - 02 June 2023: 931–938.

Chen F, Martin JD, Huang Y, et al. (2020) Autonomous exploration under uncertainty via deep reinforcement learning on graphs 2020 IEEE/RSJ International Conference on

Intelligent Robots and Systems (IROS), Las Vegas, NV, USA. 24 October 2020 - 24 January 2021: 6140–6147.

Choudhury S, Kapoor A, Ranade G, et al. (2017) Learning to gather information via imitation. In: *2017 IEEE International Conference on Robotics and Automation (ICRA)*. Singapore. 29 May 2017 - 03 June 2017: 908–915.

Chua K, Calandra R, McAllister R, et al. (2018) Deep reinforcement learning in a handful of trials using probabilistic dynamics models 32nd International Conference on Neural Information Processing Systems (NeurIPS). Montreal, Canada. 2-8 December 2018: 4759–4770.

Cipolla R, Gal Y and Kendall A (2018) Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Salt Lake City, Utah, USA. 18 - 22 June 2018: 7482–7491.

Dang T, Papachristos C and Alexis K (2018) Visual saliency-aware receding horizon autonomous exploration with application to aerial robotics 2018 IEEE International Conference on Robotics and Automation (ICRA), Brisbane, QLD. 21-25 May 2018: 2526–2533.

Dang T, Tranzatto M, Khattak S, et al. (2020) Graph-based subterranean exploration path planning using aerial and legged robots. *Journal of Field Robotics* 37(8): 1363–1388.

De Petris P, Nguyen H, Dang T, et al. (2020) Collision-tolerant autonomous navigation through manhole-sized confined environments 2020 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR), Abu Dhabi, United Arab Emirates. 04-06 November 2020: 84–89.

Dugas D, Nieto J, Siegwart R, et al. (2021) Navrep: unsupervised representations for reinforcement learning of robot navigation in dynamic human environments 2021 IEEE International Conference on Robotics and Automation (ICRA). Xian, China. 30 May 2021 - 5 June 2021: 7829–7835.

Ebadi K, Bernreiter L, Biggie H, et al. (2022) Present and future of slam in extreme underground environments. arXiv preprint arXiv:2208.01787.

Florence PR, Carter J, Ware J, et al. (2018) Nanomap: fast, uncertainty-aware proximity queries with lazy search over local 3d Data. In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. Brisbane, Australia. 21 May 2018 - 25 May 2018: 7631–7638.

Foehn P, Brescianini D, Kaufmann E, et al. (2022) Alphapilot: autonomous drone racing. Robotics: Science and Systems 2020. Corvalis, Oregon, USA. July 12-16, 2020: 307–320.

Forssen PE, Meger D, Lai K, et al. (2008) Informed visual search: combining attention and object recognition 2008 IEEE International Conference on Robotics and Automation (ICRA), Pasadena, CA, USA. 19-23 May 2008: 935–942.

Fort S, Hu H and Lakshminarayanan B (2019) Deep Ensembles: A Loss Landscape Perspective. arXiv preprint arXiv:1912.02757.

Fox D, Burgard W and Thrun S (1998) Active markov localization for mobile robots. *Robotics and Autonomous Systems* 25(3): 195–207.

Francis A, Faust A, Chiang HTL, et al. (2020) Long-range indoor navigation with prm-rl. *IEEE Transactions on Robotics* 36(4): 1115–1134.

Frey J, Hoeller D, Khattak S, et al. (2022) Locomotion policy guided traversability learning using volumetric representations of complex environments. In: *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Kyoto, Japan: 23-27 October 2022: 5722–5729.

Frintrop S, Werner T and Martin Garcia G (2015) Traditional saliency reloaded: a good old model in new shape. In: 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Boston, MA, USA. 07-12 June 2015: 82–90.

Funk N, Tarrio J, Papatheodorou S, et al. (2021) Multi-resolution 3D mapping with explicit free space representation for fast and accurate mobile robot motion planning. *IEEE Robotics and Automation Letters* 6(2): 3553–3560.

Furrer F, Burri M, Achtelik M, et al. (2016) Rotors-a modular gazebo mav simulator framework *Robot Operating System (ROS)*. Berlin: Springer International Publishing: 595–625.

Gal Y and Ghahramani Z (2016) Dropout as a bayesian approximation: representing model uncertainty in deep learning. *33rd International Conference on Machine Learning* 48: 1050–1059.

Galceran E and Carreras M (2013) A survey on coverage path planning for robotics. *Robotics and Autonomous Systems* 61(12): 1258–1276.

Gandhi D, Pinto L and Gupta A (2017) Learning to fly by crashing. In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Vancouver, BC, Canada. 24-28 September 2017: 3948–3955.

Gao F, Wu W, Gao W, et al. (2019) Flying on point clouds: online trajectory generation and autonomous navigation for quadrotors in cluttered environments. *Journal of Field Robotics* 36(4): 710–733.

Gast J and Roth S (2018) Lightweight probabilistic deep networks. In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Salt Lake City, Utah, USA. 18 - 22 June 2018: 3369–3378.

Georgakis G, Bucher B, Arapin A, et al. (2022) Uncertainty-driven planner for exploration and navigation 2022 International Conference on Robotics and Automation (ICRA). Philadelphia, PA, USA. 23-27 May 2022: 11295–11302.

Ghosh S, Delle Fave F and Yedidia J (2016) *Assumed density filtering methods for learning bayesian neural networks 30th AAAI Conference on Artificial Intelligence*. Phoenix, Arizona, USA. 12 Feb 2016-17 Feb 2016: 1589–1595.

Goel K, Corah M, Boirum C, et al. (2021) Fast exploration using multirotors: analysis, planning, and experimentation Field and Service Robotics (FSR). Tokyo, Japan. 29 - 31 August 2019: 291–305.

Guo C, Pleiss G, Sun Y, et al. (2017) On calibration of modern neural networks. *34th International Conference on Machine Learning* 70: 1321–1330.

Gustafsson FK, Danelljan M and Schon TB (2020) Evaluating scalable bayesian deep learning methods for robust computer vision 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW): Seattle, WA, USA. 14-19 June 2020. 1289–1298.

Han L, Gao F, Zhou B, et al. (2019) Fiesta: fast incremental euclidean distance fields for online motion planning of aerial robots. In: *2019 IEEE/RSJ International Conference on*

Intelligent Robots and Systems (IROS). Macau, China. 03-08 November 2019: 4423–4430.

Hernández-Lobato JM and Adams RP (2015) Probabilistic backpropagation for scalable learning of bayesian neural networks. *32nd International Conference on Machine Learning* 37: 1861–1869.

Hoeller D, Wellhausen L, Farshidian F, et al. (2021) Learning a state representation and navigation in cluttered and dynamic environments. *IEEE Robotics and Automation Letters* 6(3): 5081–5088.

Hollinger GA and Sukhatme GS (2014) Sampling-based robotic information gathering algorithms. *The International Journal of Robotics Research* 33: 1271–1287.

Hornung A, Wurm KM, Bennewitz M, et al. (2013) OctoMap: an efficient probabilistic 3D mapping framework based on octrees. *Autonomous Robots* 34: 189–206.

Ichter B and Pavone M (2019) Robot motion planning in learned latent spaces. *IEEE Robotics and Automation Letters* 4(3): 2407–2414.

Julier SJ and Uhlmann JK (1997) New extension of the Kalman filter to nonlinear systems. *Signal Processing, Sensor Fusion, and Target Recognition VI* 3068: 182–193.

Kahn G, Villaflor A, Pong V, et al. (2017) Uncertainty-aware Reinforcement Learning for Collision Avoidance. arXiv preprint arXiv:1702.01182.

Kahn G, Abbeel P and Levine S (2021a) Land: learning to navigate from disengagements. *IEEE Robotics and Automation Letters* 6(2): 1872–1879.

Kahn G, Abbeel P and Levine S (2021b) Badgr: an autonomous self-supervised learning-based navigation system. *IEEE Robotics and Automation Letters* 6(2): 1312–1319.

Kaufmann E, Loquercio A, Ranftl R, et al. (2020) Deep drone acrobatics Robotics: Science and Systems (RSS). Oregon, USA. 12-17 July 2020.

Kendall A and Gal Y (2017) What uncertainties do we need in bayesian deep learning for computer vision? 31st International Conference on Neural Information Processing Systems (NeurIPS). Long Beach, California, USA. 4-9 December 2017: 5580–5590.

Kew JC, Ichter B, Bandari M, et al. (2021) Neural collision clearance estimator for batched motion planning Workshop on the Algorithmic Foundations of Robotics. Oulu, Finland. 21 June 2021 - 23 June 2021: 73–89.

Kim J and Ostrowski JP (2003) Motion planning a aerial robot using rapidly-exploring random trees with dynamic constraints. *IEEE international conference on robotics and automation (ICRA)* 2: 2200–2205.

Kingma DP and Ba J (2015) Adam: a method for stochastic optimization 3rd International Conference on Learning Representations (ICLR). San Diego, California, USA. 7 May *2015* - 9 May 2015.

Ku J, Harakeh A and Waslander SL (2018) *In Defense of Classical Image Processing: Fast Depth Completion on the Cpu*. 2018 15th Conference on Computer and robot vision (CRV). Toronto, Canada. 8-10 May 2018: 16–22.

Kümmerer M, Wallis TSA and Bethge M (2018) Saliency benchmarking made easy: separating models, maps and metrics. In: *European Conference on Computer Vision (ECCV). Munich, Germany.* 8 September - 14 September 2018: 798–814.

Kulbacki M, Segen J, Knieć W, et al. (2018) Survey of drones for agriculture automation from planting to harvest. In: *2018 IEEE 22nd International Conference on Intelligent Engineering Systems (INES)*. Las Palmas de Gran Canaria, Spain. 21-23 June 2018: 353–358.

Labbé M and Michaud F (2019) Rtab-map as an open-source lidar and visual simultaneous localization and mapping library for large-scale and long-term online operation. *Journal of Field Robotics* 36(2): 416–446.

Lakshminarayanan B, Pritzel A and Blundell C (2017) Simple and scalable predictive uncertainty estimation using deep ensembles 31st International Conference on Neural Information Processing Systems (NeurIPS). Long Beach, California, USA. 4-9 December 2017: 6405–6416.

Liu Z, Amini A, Zhu S, et al. (2021) Efficient and robust lidar-based end-to-end navigation 2021 IEEE International Conference on Robotics and Automation (ICRA). Xian, China. 30 May 2021 - 5 June 2021: 13247–13254.

Lopez BT and How JP (2017) Aggressive 3-d collision avoidance for high-speed navigation. In: *2017 IEEE International Conference on Robotics and Automation (ICRA)*. Singapore. 29 May 2017 - 03 June 2017: 5759–5765.

Loquercio A, Maqueda AI, del Blanco CR, et al. (2018) Dronet: learning to fly by driving. *IEEE Robotics and Automation Letters* 3(2): 1088–1095.

Loquercio A, Segu M and Scaramuzza D (2020) A general framework for uncertainty estimation in deep learning. *IEEE Robotics and Automation Letters* 5(2): 3153–3160.

Loquercio A, Kaufmann E, Ranftl R, et al. (2021) Learning high-speed flight in the wild. *Science Robotics* 6(59): eabg5810.

Lütjens B, Everett M and How JP (2019) Safe reinforcement learning with model uncertainty estimates. In: *2019 International Conference on Robotics and Automation (ICRA)*. Montreal, QC, Canada. 20-24 May 2019: 8662–8668.

Matthies L, Brockers R, Kuwata Y, et al. (2014) Stereo vision-based obstacle avoidance for micro air vehicles using disparity space. In: *2014 IEEE International Conference on Robotics and Automation (ICRA)*. Hong Kong, China. 31 May 2014 - 07 June 2014: 3242–3249.

Millane A, Taylor Z, Oleynikova H, et al. (2018) C-blox: a scalable and consistent tsdf-based dense mapping approach. In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Madrid, Spain. 01-05 October 2018: 995–1002.

Museth K (2013) Vdb: high-resolution sparse volumes with dynamic topology. *ACM Transactions on Graphics* 32(3): 1–22.

Nguyen H, Fyhn SH, De Petris P, et al. (2022) Motion primitives-based navigation planning using deep collision prediction. In: *2022 International Conference on Robotics and Automation (ICRA)*. Philadelphia, PA, USA. 23-27 May 2022: 9660–9667.

Niroui F, Zhang K, Kashino Z, et al. (2019) Deep reinforcement learning robot for search and rescue applications: exploration in unknown cluttered environments. *IEEE Robotics and Automation Letters* 4(2): 610–617.

Oleynikova H, Taylor Z, Fehr M, et al. (2017) Voxblox: incremental 3d euclidean signed distance fields for on-board mav planning. In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Vancouver, Canada. 24 September 2017 - 28 September 2017: 1366–1373.

Oleynikova H, Taylor Z, Siegwart R, et al. (2018) Safe local exploration for replanning in cluttered unknown environments for microaerial vehicles. *IEEE Robotics and Automation Letters* 3(3): 1474–1481.

Ovadia Y, Fertig E, Ren J, et al. (2019) Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift 33rd International Conference on Neural Information Processing Systems (NeurIPS): 13969–13980.

Petracek P, Kratky V, Petrlik M, et al. (2021) Large-scale exploration of cave environments by unmanned aerial vehicles. *IEEE Robotics and Automation Letters* 6(4): 7596–7603.

Pop R and Fulop P (2018) *Deep Ensemble Bayesian Active Learning: Addressing the Mode Collapse Issue in Monte Carlo Dropout via Ensembles*. arXiv preprint arXiv: 1811.03897.

Popovic M, Vidal-Calleja T, Hitz G, et al. (2018) An informative path planning framework for uav-based terrain monitoring. *Autonomous Robots* 44: 889–911.

Qureshi AH, Miao Y, Simeonov A, et al. (2021) Motion planning networks: bridging the gap between learning-based and classical motion planners. *IEEE Transactions on Robotics* 37(1): 48–66.

Redmon J and Farhadi A (2018) *Yolov3: An Incremental Improvement*. arXiv preprint arXiv:1804.02767.

Richter C and Roy N (2017) Safe visual navigation via deep learning and novelty detection Robotics: Science and Systems (RSS). Massachusetts, USA. 12-16 July 2017.

Rückin J, Jin L and Popović M (2022) Adaptive informative path planning using deep reinforcement learning for uav-based active sensing 2022 International Conference on Robotics and Automation (ICRA). Pennsylvania, USA. 23-27 May 2022: 4473–4479.

Sa I, Kamel M, Khanna R, et al. (2017) Dynamic system identification, and control for a cost-effective open-source vtol mav Field and Service Robotics (FSR). Zurich, Switzerland. 12-15 September 2017: 605–620.

Schmid L, Pantic M, Khanna R, et al. (2020) An efficient sampling-based method for online informative path planning in unknown environments. *IEEE Robotics and Automation Letters* 5(2): 1500–1507.

Selin M, Tiger M, Duberg D, et al. (2019) Efficient autonomous exploration planning of large-scale 3-d environments. *IEEE Robotics and Automation Letters* 4(2): 1699–1706.

Shah D and Levine S (2022) *ViKiNG: vision-based kilometer-scale navigation with geographic hints Robotics: Science and Systems (RSS)*. New York, USA. 27 June 2022 - 1 July 2022.

Shi X, Chen Z, Wang H, et al. (2015) Convolutional lstm network: a machine learning approach for precipitation nowcasting 28th International Conference on Neural Information Processing Systems (NeurIPS). Montreal, Canada. 7 December 2015 - 10 December 2015: 802–810.

Song Y, Naji S, Kaufmann E, et al. (2020) *Flightmare: a flexible quadrotor simulator 4th Conference on Robot Learning (CoRL). Cambridge MA, USA. 16-18* November 2020: 1147–1157.

Srinivas A, Jabri A, Abbeel P, et al. (2018) Universal planning networks: learning generalizable representations for visuomotor control. *35th International Conference on Machine Learning* 80: 4732–4741.

Sun K, Schlotfeldt B, Pappas GJ, et al. (2021) Stochastic motion planning under partial observability for mobile robots with continuous range measurements. *IEEE Transactions on Robotics* 37(3): 979–995.

Tabib W, Goel K, Yao J, et al. (2022) Autonomous cave surveying with an aerial robot. *IEEE Transactions on Robotics* 38(2): 1016–1032.

Tabuada P and Gharesifard B (2022) Universal approximation power of deep residual neural networks through the lens of control. *IEEE Transactions on Automatic Control* 68(5): 2715–2728.

Tao Y, Wu Y, Li B, et al. (2023) Seer: safe efficient exploration for aerial robots using learning to predict information gain 2023 IEEE International Conference on Robotics and Automation (ICRA). London, UK. 29 May 2023 - 2 June 2023: 1235–1241.

Tolani V, Bansal S, Faust A, et al. (2021) Visual navigation among humans with optimal control as a supervisor. *IEEE Robotics and Automation Letters* 6(2): 2288–2295.

Tranzatto M, Miki T, Dharmadhikari M, et al. (2022) Cerberus in the darpa subterranean challenge. *Science Robotics* 7(66): eabp9742.

Tsotsos JK (2011) *A Computational Perspective on Visual Attention*. Cambridge, MA: MIT Press.

Usenko V, von Stumberg L, Pangercic A, et al. (2017) Real-time trajectory replanning for mavs using uniform b-splines and a 3d circular buffer. In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Vancouver, BC, Canada. 24-28 September 2017: 215–222.

Veer S and Majumdar A (2020) Probably approximately correct vision-based planning using motion primitives. *4th Conference on Robot Learning (CoRL)* 155: 1001–1014.

Wagter CD, Paredes-Vallés F, Sheth N, et al. (2021) *Artificial Intelligence behind the Winning Entry to the 2019 AI Robotic Racing Competition*. arXiv preprint arXiv:2109.14985.

Wang H, Shi X and Yeung D (2016) Natural-parameter networks: a class of probabilistic neural networks 30th International Conference on Neural Information Processing Systems (NeurIPS). Barcelona, Spain. 5-10 December 2016: 118–126.

Wang L, Ye H, Wang Q, et al. (2021) Learning-based 3d occupancy prediction for autonomous navigation in occluded environments 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). Prague, Czech Republic. 27 September 2021 - 01 October 2021: 4509–4516.

Weiss SM (2012) *Vision Based Navigation for Micro Helicopters*. PhD Thesis. Zürich, Switzerland: ETH Zurich.

Zeng W, Luo W, Suo S, et al. (2019) End-to-end interpretable neural motion planner. In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Long Beach, California, USA. 15 - 20 June 2019: 8652–8661.

Zhang Z, Henderson T, Karaman S, et al. (2020) Fsmi: fast computation of shannon mutual information for information-theoretic mapping. *The International Journal of Robotics Research* 39(9): 1155–1177.

Zhou S and Gheisari M (2018) Unmanned aerial system applications in construction: a systematic review. *Construction Innovation* 18(14): 453–468.

Zhou B, Pan J, Gao F, et al. (2021) Raptor: robust and perception-aware trajectory replanning for quadrotor fast flight. *IEEE Transactions on Robotics* 37(6): 1992–2009.

Zhu D, Li T, Ho D, et al. (2018) Deep reinforcement learning supervised autonomous exploration in office environments. In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. Brisbane, QLD. 21-25 May 2018: 7548–7555.

Appendix

Appendix A: Index to multimedia extensions

The index to the multimedia extensions of this manuscript is given in Table 7.

Appendix B: Proof of lemma

The proof of Lemma IV.1 is as follows

Proof. Let $\mathbf{q} = (u, v)$ be an arbitrary pixel in the depth image $\mathbf{o}_t$, $Z = \mathbf{o}_t\,(u, v)$ the depth value of pixel $\mathbf{q}$, and (c_x, c_y),f_x^d,f_y^d the pixel coordinates of the optical center as well as the focal length of the depth camera, respectively. Then the corresponding pixel in the horizontal flip depth image $\mathbf{o}_t^{flip}$ is approximately $\mathbf{q}^{flip} = (2c_x - u, v)$. We denote the 3D projected points of the pixels $\mathbf{q}$ and $\mathbf{q}^{flip}$ as $\mathbf{Q}$ and $\mathbf{Q}^{flip}$, respectively. Given assumption (1) above, the coordinates of $\mathbf{Q}$ and $\mathbf{Q}^{flip}$ expressed in $\mathcal{C}$ are:

Table 7. Table of multimedia extensions.

Extension	Media type	Description
1	Video	This video presents the overview of the method, the setup, and the data from experiment 1 with ORACLE in a cluttered corridor
2	Video	This video presents the setup and the data from experiment 2 with ORACLE in a dense forest during under canopy flight
3	Video	This video presents the setup and the data from experiment 3 with A-ORACLE and ORACLE in an Industrial silo tank
4	Video	This video presents the setup and the data from experiment 4 with A-ORACLE and ORACLE in a university's hall

$$^{\mathcal{C}}\mathbf{Q} = \left[{}^{\mathcal{C}}\mathbf{Q}_x, {}^{\mathcal{C}}\mathbf{Q}_y, Z\right]^T \tag{31}$$

$$^{\mathcal{C}}\mathbf{Q}^{flip} = \left[{}^{\mathcal{C}}\mathbf{Q}_x^{flip}, {}^{\mathcal{C}}\mathbf{Q}_y^{flip}, Z\right]^T \tag{32}$$

where

$$^{\mathcal{C}}\mathbf{Q} = \frac{u - c_x}{f_x^d} Z \tag{33}$$

$$^{\mathcal{C}}\mathbf{Q}_y = \frac{v - c_y}{f_y^d} Z \tag{34}$$

$$^{\mathcal{C}}\mathbf{Q}_x^{flip} = \frac{c_x - u}{f_x^d} Z = -{}^{\mathcal{C}}\mathbf{Q} \tag{35}$$

$$^{\mathcal{C}}\mathbf{Q}_y^{flip} = \frac{v - c_y}{f_y^d} Z = {}^{\mathcal{C}}\mathbf{Q}_y \tag{36}$$

From the assumption (2) above, the transformation from $\mathcal{C}$-frame to $\mathcal{B}$-frame has the form:

$$\mathbf{R}_{BC} = \mathbf{R}_y(\theta_c)\begin{pmatrix} 0 & 0 & 1 \\ -1 & 0 & 0 \\ 0 & -1 & 0 \end{pmatrix} \tag{37}$$

$$\mathbf{t}_{BC} = [\mathbf{t}_{BC,x}, 0, \mathbf{t}_{BC,z}]^T \tag{38}$$

where $\mathbf{R}_j(\eta)$ $(j = x, y, z)$ denotes the rotation matrix for a rotation around the j-axis by η degrees and θ_c is the rotation angle of $\mathcal{C}$ around the $y - $ axis of $\mathcal{B}$. The 3D coordinates of the projected points $\mathbf{Q}$ and $\mathbf{Q}^{flip}$ are given in $\mathcal{B}$ and $\mathcal{V}$ as:

$$^{\mathcal{B}}\mathbf{Q} = \mathbf{R}_{BC}\,{}^{\mathcal{C}}\mathbf{Q} + \mathbf{t}_{BC} \tag{39}$$

$$^{\mathcal{V}}\mathbf{Q} = \mathbf{R}_y(\theta_t)\mathbf{R}_x(\phi_t)\,{}^{\mathcal{B}}\mathbf{Q} \tag{40}$$

$$^{\mathcal{B}}\mathbf{Q}^{flip} = \mathbf{R}_{BC}\,{}^{\mathcal{C}}\mathbf{Q}^{flip} + \mathbf{t}_{BC} = [{}^{\mathcal{B}}\mathbf{Q}_x, -{}^{\mathcal{B}}\mathbf{Q}_y, {}^{\mathcal{B}}\mathbf{Q}_z]^T \tag{41}$$

$$^{\mathcal{V}}\mathbf{Q}^{flip} = \mathbf{R}_y(\theta_t)\mathbf{R}_x(-\phi_t)\,{}^{\mathcal{B}}\mathbf{Q}^{flip} = [{}^{\mathcal{V}}\mathbf{Q}_x, -{}^{\mathcal{V}}\mathbf{Q}_y, {}^{\mathcal{V}}\mathbf{Q}_z]^T \tag{42}$$

Plugging $\mathbf{s}_t^{flip}, \mathbf{a}_{t:t+H}^{flip}$ into (6), it can be deduced that the flipped position and relative yaw labels can be obtained by reverting the signs of ${}^{\mathcal{V}}\mathbf{p}_{t+i,y}, \boldsymbol{\delta}_{t+i}\,(i = 1, ..., H)$ in ${}^{\mathcal{V}}\mathbf{p}_{t+1:t+H+1}$ and $\boldsymbol{\delta}_{t+1:t+H+1}$, respectively. From (42), we can infer that the y-coordinate of the obstacles in $\mathcal{V}$-frame is also reverted; hence, the collision labels of the augmented data point $\mathbf{d}_{CPN}^{flip}$ are the same as $\mathbf{d}_{CPN}$.