Belief-Conditioned One-Step Diffusion: Real-Time Trajectory Planning with Just-Enough Sensing

Gokul Puthumanaillam [I], **Aditya Penumarti** [UF], **Manav Vora** [I], **Paulo Padrao** [P], **Jose Fuentes** [FIU], **Leonardo Bobadilla** [FIU], **Jane Shin** [UF], **Melkior Ornik** [I]

[I] University of Illinois Urbana-Champaign ({gokulp2, mkvora2, mornik}@illinois.edu)

[UF] University of Florida ({apenumarti, jane.shin}@ufl.edu)

[P] Providence College ({ppadraol}@providence.edu)

[FIU] Florida International University ({jfuen099}@fiu.edu, {bobadilla}@cs.fiu.edu)

Abstract: Robots equipped with rich sensor suites can localize reliably in partially-observable environments, but powering every sensor continuously is wasteful and often infeasible. Belief-space planners address this by propagating pose-belief covariance through analytic models and switching sensors heuristically–a brittle, runtime-expensive approach. Data-driven approaches–including diffusion models–learn multi-modal trajectories from demonstrations, but presuppose an accurate, always-on state estimate. We address the largely open problem: for a given task in a mapped environment, which *minimal sensor subset* must be active at each location to maintain state uncertainty *just low enough* to complete the task? Our key insight is that when a diffusion planner is explicitly conditioned on a pose-belief raster and a sensor mask, the spread of its denoising trajectories yields a calibrated, differentiable proxy for the expected localisation error. Building on this insight, we present Belief-Conditioned One-Step Diffusion (B-COD), the first planner that, in a 10 ms forward pass, returns a short-horizon trajectory, per-waypoint aleatoric variances, and a proxy for localisation error–eliminating external covariance rollouts. We show that this single proxy suffices for a soft-actor–critic to choose sensors online, optimising energy while bounding pose-covariance growth. We deploy B-COD in real-time marine trials on an unmanned surface vehicle and show that it reduces sensing energy consumption while matching the goal-reach performance of an always-on baseline. Project website: bcod-diffusion.github.io.

Keywords: Diffusion Planning, Reinforcement Learning, Multi-Modal Sensing

1 Introduction

Autonomous robots performing navigation-related tasks routinely mount heterogeneous sensors, like cameras, LiDARs, and GPS, because no single modality is reliable everywhere [1, 2, 3]. Keeping every sensor powered, however, wastes energy, and can degrade performance by flooding the estimator with irrelevant data [4, 5]. At the other extreme, indiscriminately toggling sensors is perilous: if the robot drifts into a sensor-denied zone with the wrong sensors active, localisation uncertainty can explode, leading to task failure [6, 7]. This paper, therefore, poses a precise question: given a goal, can a robot generate the next short-horizon motion *and*, in real-time, decide the *smallest* set of sensors that must stay on so that its localisation uncertainty is reduced *just enough* to reach the goal—navigation with just-enough sensing.

Classical belief-space pipelines address the motion planning half of the problem: linearized covariance propagation achieves excellent tracking when every sensor is active and the motion–measurement models remain accurate [8, 9]. Yet these same pipelines unravel in harder conditions [10], where

9th Conference on Robot Learning (CoRL 2025), Seoul, Korea.

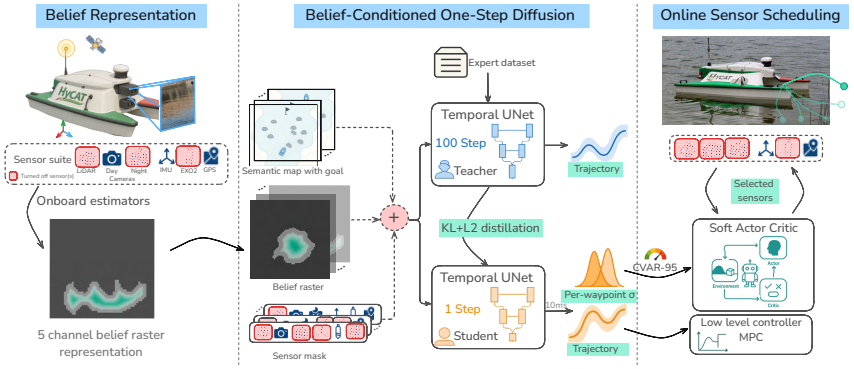

Figure 1: Overview of B-COD. Left: The belief module compresses the particle cloud and local map into a belief raster that encodes mass, orientation, and covariance. Center: A one-step diffusion network consumes that raster, the goal mask, and the current sensor flag to return a short-horizon trajectory and a calibrated CVaR risk scalar. Right: SAC uses the scalar to toggle sensors in real time, spending energy only where required.

a single ill-chosen sensor subset can drive the pose estimate outside tolerance, causing subsequent mission failure [11]. The complementary half–deciding which sensors to power–is often posed as a scheduling problem that assigns an energy cost to every measurement [12]. The robot's belief over pose and map is continuous and high-dimensional, so the decision tree grows exponentially and is impossible to traverse in real time [13]. Practical solvers collapse the tree into ad-hoc heuristics [14], thereby ignoring the long-term value of information [15]. Reinforcement learning variants push farther by embedding the sensor-activation bits into the action space and letting a policy learn when to switch them [16, 17, 18]. However, such policies demand millions of episodes, delicate reward tuning [19], and still require hand-engineered safety shields to keep pose error bounded [20]. Worse, they optimize sensing and control on the same lattice but leave trajectory generation to a separate planner, creating a brittle split that resurfaces whenever the environment changes [5, 12, 21].

In a departure from handcrafted motion models and data-hungry on-policy RL, diffusion-based planners learn rich, multi-modal trajectory distributions directly from demonstrations and require only supervised training [22, 23, 24]. Their appeal is clear: they avoid reward-shaping pitfalls, capture alternate homotopies around obstacles, and can be conditioned on semantic maps with minimal architectural fuss. Their Achilles' heel, however, is that the denoising network is trained under the premise that the robot's pose is always known accurately and every sensor is live [25, 26].

A largely overlooked aspect of diffusion planners is that their iterative denoising process [27] produces a *full distribution* of trajectories rather than a single path [22, 28]. Intuitively, the spread of that distribution ought to widen when localisation drifts and contract when well observed—hinting that diffusion *might* already encode clues about "how much sensing is enough." Turning that intuition to enable navigation with just-enough sensing is the central idea of this paper: we repurpose diffusion to simultaneously produces a short-horizon path towards the goal *and* serve as a sensing oracle for a small sensor-selection policy. Figure 1 presents an overview of our solution approach.

Statement of contributions: We introduce (i) *Belief-Conditioned One-Step Diffusion (B-COD)*, a diffusion planner that pairs short-horizon trajectory generation with an uncertainty proxy from its own denoising spread, generated in one forward pass; (ii) we show that this proxy is enough to enable a lightweight RL policy to toggle sensors online, yielding a real-time navigation pipeline with just enough sensing; (iii) we validate the full system in real-time on an autonomous surface vehicle in the context of marine autonomy, an underexplored, high-uncertainty domain, demonstrating goal-reach rate of the always-on baseline while consuming less than half its energy; (iii-a) to spur further research in this domain, we release the code, trained models, and the first open maritime dataset of 50K belief-annotated navigation snippets with synchronised multi-modal sensor logs [1].

[1] A subset is available on the project website; the full dataset will be released after review.

2 Related Works

Belief-Space Motion Planning: Early work cast motion planning under uncertainty as a POMDP, but exact solvers scale poorly, motivating approximate methods [29, 8, 30]. Graph-based variants propagate linearised covariances along lattice edges [31, 32, 33, 34, 35] or sample belief particles through non-linear dynamics [36, 37, 38, 39]. Point-based POMDP planners [40, 41, 42, 43, 44, 45] achieved impressive accuracy but remain slow for real-time use. Chance-constrained formulations bound the probability of constraint violation [46, 47, 48, 49], while information-theoretic objectives [50, 51, 52, 53] trade control cost with covariance growth. These pipelines assume all sensors remain active and rely on expensive covariance roll-outs at every node–limitations B-COD sidesteps by learning a differentiable proxy in one forward pass.

Resource-Aware Sensor Scheduling: Choosing which modalities to power has been framed as a mixed-integer program over measurement actions [54, 55, 56, 57], a sub-modular maximisation of expected information gain [58, 59, 60], or a value-of-information heuristic [61, 62, 63]. Sensor-activation trees grow with horizon, so practical solvers rely on greedy switches [64, 65], rollout policy iteration [66, 67], or coarse duty-cycling [68, 69]. RL approaches embed sensor bits into the action space and learn policies offline [70, 71, 72, 73, 74, 75]. Energy-aware schemes [76, 77, 76] toggle sensors based on heuristic. Unlike these decoupled strategies, B-COD couples trajectory generation and sensor selection via a shared diffusion-derived uncertainty signal without handcrafted thresholds.

Learning-Based Trajectory Generators: Imitation-learning priors [78, 79, 80, 81, 82] capture multi-modal behaviour but output a single trajectory. VAEs [82, 83, 84, 85, 86] model richer path distributions, yet training requires tricks or annealing. Diffusion-based planners [87, 88, 22, 24, 89, 90] have recently shown SOTA coverage over homotopies while retaining simple supervised losses. All, however, assume perfect, always-on estimation. B-COD is the first to inject the robot's pose-belief raster into the denoiser, letting the spread of the samples act as a calibrated localisation-error proxy.

Joint Planning and Active Perception: Active SLAM couples view-planning with mapping updates but focuses on map quality rather than energy [91, 92, 93]. Informative path planners [94, 95] optimise mutual information over candidate trajectories while assuming fixed sensor payloads. Recent works [96, 97] integrate learned priors with active perception yet they still separate motion generation from modality selection, requiring auxiliary covariance estimators. By conditioning diffusion on belief and letting an RL head act on the resulting uncertainty proxy, our framework realises just-enough sensing–closing the loop between planning and active localisation with millisecond latency.

3 Methodology

Problem statement. Assume a robot operating in a workspace $\mathcal{W} \subset \mathbb{R}^2$ whose obstacles and semantic layers–lighting, sensor visibility–are known in advance through survey or GIS data, yielding a coarse, static map M. The robot carries N sensors $S = \{s_1, \dots, s_N\}$. At decision epoch t the robot's pose is the element $\mathbf{x}_t = (x_t, y_t, \psi_t) \in SE(2)$; its uncertainty is encoded by the belief density $b_t(\mathbf{x}) = p\big(\mathbf{x}_t = \mathbf{x} \mid z_{0:t}, a_{0:t}\big)$, where $z_{0:t}$ are past measurements and $a_{0:t}$ is the sensor-activation vectors. Observations obey a known likelihood $p(z_t \mid \mathbf{x}_t, a_t)$. At each step, $a_t \in \{0,1\}^N$ indicates which sensors are powered, and the energy cost vector $c = [c_1, \dots, c_N]^\top$ quantifies per-sensor consumption. Together with a control sequence τ_t, the schedule $\{a_t\}_{t=0}^{T-1}$ must steer the belief into the goal set $\mathcal{G}$ while avoiding the obstacle region $\mathcal{O}$. We measure localisation risk by the scalar $\sigma(b_t) = \sqrt{\operatorname{tr}\ \Sigma_t}$ where Σ_t is the pose covariance extracted from b_t. Let η denote the user-specified per-step risk budget and ε the tolerated per-step exceedance probability. The planning problem is

$$\text{minimize}\ \ J_{\text{energy}} = \sum_{t=0}^{T-1} c^\top a_t \quad \text{subject to} \quad \Pr\big\{\sigma(b_t) > \eta\big\} \le \varepsilon,\ \mathbf{x}_t \notin \mathcal{O},\ \mathbf{x}_T \in \mathcal{G},\ \forall t, \tag{1}$$

Our solution hinges on three capabilities: (i) a belief encoding that neural networks (NN) can ingest; (ii) a planner that, conditioned on that belief, yields a trajectory and a measure of future pose uncertainty; (iii) a policy that exploits that measure to enable minimal sensing task completion.

3.1 Belief Raster Representation

Any NN planner that consumes state uncertainty must see a tensor of fixed shape [98], yet a Bayesian filter evolves a belief whose support may swell/split over time [99]. We therefore project the pose posterior into a five-channel image $B_t \in [0,1]^{H\times W\times 5}$ whose footprint adapts to the belief but is resampled to a preset resolution before it enters the planner. If the window exceeds the target lattice $H \times W$ it is isotropically down-sampled, making runtime independent of how far the robot's uncertainty has spread. Each grid cell stores five summary statistics that experiments show are sufficient: (i) *belief mass* $m_{u,v}$, a true probability derived by summing particle weights in the cell, (ii-iii) *sine* and *cosine of yaw* that encodes the heading, averaged with the same weights and linearly re-mapped to $[0,1]$ so that all channels share a common dynamic range, (iv) *planar spread* is compressed into the log-determinant of the local positional covariance, (v) *circular variance* records how concentrated the heading distribution is; it equals zero for a unimodal orientation and approaches one as yaw becomes ambiguous. Other moments were found to add negligible value (see Appendix). B_t is translation-equivariant, orientation-aware, and compact enough to slot directly into downstream convolutions, yet retaining exactly the information that governs localisation drift for small horizons.

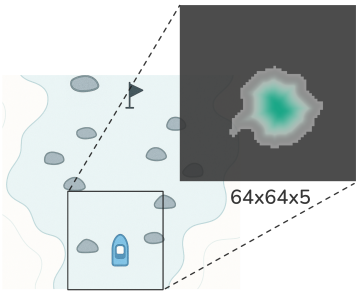

Figure 2: Example belief raster from our experiments. The channels are projected into a single HSV image: brightness = probability mass, hue = heading, desaturation = areas of higher positional spread. The teal = most likely pose; the grey expresses growing 3σ uncertainty, black denotes zero belief.

3.2 Belief-Conditioned One-Step Diffusion (B-COD)

Having condensed pose uncertainty into an ego-centric image, we face two coupled decisions at every control tick: what short motion should the robot execute next, and how large will the pose error grow if it moves with only the currently powered sensors? B-COD answers this by learning the conditional distribution $p_\theta\big(\tau \,\big|\, B,\ M,\ g,\ a\big)$, where B is the ego-centric belief image from Sec. 3.1, M the co-cropped semantic map slice, g a binary goal mask in the same frame, $a \in \{0,1\}^N$ the vector of powered sensors, and $\tau = (\Delta x_1, \Delta y_1, \Delta\psi_1, \ldots, \Delta\psi_H)$ sequence of robot body-frame incremental poses. We noticed that using increments, rather than global coordinates, lets B-COD learn translation-invariant manoeuvres; as B-COD is re-invoked each second, these local segments concatenate naturally into a consistent global path. The diffusion backbone parameterises, at every waypoint, a diagonal Gaussian whose mean is the desired displacement and whose log-variance $\hat{\sigma}_k$ quantifies aleatoric uncertainty inherited from the training data. Drawing one standard normal latent and passing it through the network therefore returns in a *single* forward pass (i) a trajectory sample τ and (ii) the full vector of waypoint log-variances $\hat{\sigma}_{1:H}$. Resampling the latent injects fresh noise and produces alternative, equally plausible plans without any extra cost beyond that single inference–exactly the property that makes diffusion attractive for real-time, multi-modal navigation.

Conditioning and trajectory tokenisation. To generate a path that is feasible and sensor-aware, the planner must jointly reason over: (i) *where the robot might be*, (ii) *what the environment looks like*, and (iii) *which modalities are currently online*. The inputs that carry this context–belief, map semantics, and goal–are concatenated into a tensor $[B \,\|\, M \,\|\, g]$. Since navigation occurs locally, encoding the entire global map is redundant and inefficient; thus, a generic spatial encoder (in our case, ResNet) can extract spatially localized features that summarise nearby geometry and uncertainty [100]. Additionally, because trajectories depend on the available sensors, an embedding of a is fused with the spatial features to form a comprehensive context vector $\mathcal{C}$. Conditioning on $\mathcal{C}$ ensures that the planner avoids plans that would demand unavailable modalities. Finally, the planner outputs trajectories as sequences of relative displacements rather than global poses. This representation makes the prediction invariant to absolute coordinates, allowing the diffusion network to attend directly to the local environment and sensor context vector $\mathcal{C}$. This conditioning dynamically aligns each waypoint with obstacles, environmental context, and real-time sensing capabilities.

Diffusion teacher. A diffusion model can sample diverse trajectories, but running a hundred reverse steps is prohibitive on embedded hardware [87]. We therefore follow the DDPM paradigm [27]: teach a multi-step denoiser that is *expressive*–able to regenerate the multi-modal paths seen in demonstration–and *honest* about its own pose uncertainty. The teacher adopts the cosine forward process [101], $\tilde{\tau}_t = \sqrt{\bar{\alpha}_t}\,\tau + \sqrt{1-\bar{\alpha}_t}\,\epsilon$, $\epsilon \sim \mathcal{N}(0, I)$ with $\bar{\alpha}_t = \cos^2(\pi t/2T)$, because that schedule keeps signal-to-noise high at early timesteps and has been shown to improve reconstruction quality on long, structured sequences [101]. At reversal step t the network predicts both the injected noise and a waypoint-wise log-variance–crucial, because how confident the network is about each displacement becomes the proxy our scheduler relies on. The loss combines the DDPM noise-reconstruction term with a diagonal Gaussian negative log-likelihood:

$$\mathcal{L}_{\text{teach}} = \left\|\epsilon_\theta(\tilde{\tau}_t, B, M, g, a, t) - \epsilon\right\|_2^2 + \beta \sum_{k=1}^{H} \left[\frac{\|\tau_k - \hat{\mu}_{\theta,k}\|^2}{e^{\hat{\sigma}_{\theta,k}}} + \hat{\sigma}_{\theta,k}\right]. \tag{2}$$

The first term teaches the network to undo the forward corruption and reproduce the expert manoeuvre; the second forces the predicted variance $e^{\hat{\sigma}_k}$ to match the empirical scatter of demonstrations at that waypoint, yielding aleatoric errors that are *calibrated by construction*.

Consistency-model student. Even a perfectly trained teacher is unusable if their hundred reverse steps exceed the control loop's latency. Rather than prune steps heuristically–which degrades sample quality[102]–we compress the reverse chain into a single network via consistency distillation [103]. A standard-normal latent ξ is pushed through the teacher to obtain a reference sample $(\tau^{\text{ref}}, \hat{\sigma}^{\text{ref}})$. A student g_ψ is then optimised to match both the mean path and the uncertainty of that reference:

$$\mathcal{L}_{\text{stu}} = \left\|g_\psi^{\mu}(\xi) - \tau^{\text{ref}}\right\|_2^2 + \lambda\, \text{KL}\Big(\mathcal{N}\big(g_\psi^{\mu}, \Sigma_\psi\big) \,\Big\|\, \mathcal{N}\big(\tau^{\text{ref}}, \Sigma_{\text{ref}}\big)\Big), \qquad \Sigma = \text{diag}\big(e^{\hat{\sigma}}\big). \tag{3}$$

The first term preserves trajectory fidelity; the KL term transfers the teacher's calibrated variances so the risk proxy remains valid after compression. We ramp λ during training so that the student first learns accurate means and only then assumes responsibility for matching dispersion–preventing the early collapse of variance that can plague single-step models [102].

Uncertainty proxy for the scheduler. Passing the whole variance vector to the planner would explode the state space; collapsing it to a naive average would hide the critical segments where error spikes. We condense the vector into a tail–focused statistic, CVaR-95 [104]: $u^{\text{CVaR}} = \frac{1}{0.05\,H} \sum_{k \in \text{top } 5\%} \sqrt{e^{\hat{\sigma}_k}}$. Intuitively, because the teacher is trained with a diagonal Gaussian NLL and the student matches that distribution via the KL term, the CVaR bound transfers intact: on the training distribution the one-sided 95th-percentile localisation error at *each* waypoint does not exceed u^{CVaR}. Temporal correlations can only render the statistic conservative, a property we deliberately preserve for safety. Crucially, the scalar drops out of the planner's existing variance head–no extra sampling or covariance rollout is required–so it arrives alongside the candidate trajectory within the same forward pass.

3.3 Online sensor scheduling with a risk budget

At every decision epoch, B-COD delivers a short-horizon trajectory and u^{CVaR}. The robot must now answer the question: which subset of sensors should remain powered so the error budget is honoured while energy is spent as sparingly as possible? We cast this as a *Constrained Markov Decision Process (CMDP)* [105] with state $s_t = \big(B_t,\ u_t^{\text{CVaR}},\ d_t,\ a_{t-1}\big)$: belief raster, scalar risk forecast, distance from the belief mean to the goal, and the previously active sensor mask. The primary cost is instantaneous power draw $r_t = -c^\top a_t$ with per-sensor coefficients $c \in \mathbb{R}^N_{>0}$. The indicator constraint expresses safety $g_t = \mathbf{1}[u_t^{\text{CVaR}} > \eta_{\max}]$: incurs a penalty if the predicted localisation error exceeds the defined threshold $\eta_{\max}$. Letting π denote any stationary policy, the optimisation objective is

$$\min_{\pi}\ \limsup_{T\to\infty} \frac{1}{T}\,\mathbb{E}_\pi\Big[\sum_{t=0}^{T-1} r_t\Big] \quad \text{s.t.} \quad \limsup_{T\to\infty} \frac{1}{T}\,\mathbb{E}_\pi\Big[\sum_{t=0}^{T-1} g_t\Big] \le \epsilon, \tag{4}$$

where ϵ is the desired long-run rate of risk violations. We append the numerical features $(u_t^{\text{CVaR}}, d_t, a_{t-1})$ to the encoded B_t from B-COD and feed the result to a lightweight policy head.

Any constrained-RL can solve this CMDP; we instantiate it with a *constrained Soft Actor–Critic* (SAC) [106, 107] because it is off-policy, data-efficient, and admits a simple dual-gradient update on the Lagrange multiplier λ. The actor outputs relaxed Bernoulli probabilities thresholded to hard on/off commands at execution time, while a pair of critic networks learns the Lagrangian Q-values. The only hyperparameters exposed are the risk budget $\eta_{\max}$ and the violation rate ϵ.

3.4 Engineering Choices and Implementation Recipe

Data generation: Alongside real-world data collection, we augment data from a custom Unity-based simulator that instantiates ninety procedurally varied harbour scenes ($\approx$ 200 m^2). Each world is rasterised at 0.25 m resolution into three semantic layers–obstacle occupancy, ambient lighting, and sensor visibility–cropped into the ego frame alongside the belief raster. An A*+MPC [108, 109] oracle drives from random starts to six-meter goal disks while seeing ground-truth pose. Replaying every oracle trajectory sixteen times under forced sensor subsets and noise produces 8.3M short-horizon snippets whose diversity teaches the B-COD both nominal and failure-case behaviour.

Training: The diffusion teacher is trained once on that corpus with the Eq. 2 loss; an exponential-moving-average of the weights is retained as the final teacher. A single-step student is consistency distilled training on 1M noise targets generated on-the-fly, giving a forward pass two orders of magnitude faster yet matching its negative log-likelihood. Exporting the student through ONNX [110] and TensorRT [111] (FP16, layer fusion) yields a $\approx$ 10 ms inference on the Jetson-class hardware (Orin NX 16 GB [112]); belief rasterisation and actor inference raise the control-loop budget to 15 ms (empirical results in Sec. 4). SAC learns the scheduling policy entirely in simulation. A shared CNN [113] processes the belief image once; two MLPs serve as the lightweight policy head and output Bernoulli logits and twin Q-values. Dual-gradient updates on the Lagrange multiplier keep constraint violations near the user-specified budget. In real-time testing, the neural checkpoints run unchanged. Only sensor-noise covariances are re-scaled from factory calibration data.

4 Experimental Results

Our evaluation targets a real-world, real-time scenario in which an autonomous surface vehicle (ASV) must navigate an open-air lake previously unseen in training, to reach waypoint goals with just-enough sensing while keeping the CVaR-95 localisation error below a user budget of 2 m. The lake presents both natural and human-driven disturbances: winds, waves, fountains, and floating buoys. The test platform is a SeaRobotics Surveyor ASV [114] with a differential-thrust propulsion module and a heterogeneous sensor suite: a multi-beam LiDAR, day and night cameras, RTK-GPS, MEMS IMU, and an EXO2 sonde. The platform exposes a velocity set-point interface; we run a linear MPC (2 s horizon; inputs limited to the thrust envelope) that tracks an eight-waypoint segment with mean cross-track error $\leq$ 6 cm. Control inputs are augmented by a discrete mode flag that selects the estimator configuration implied by the powered sensors. Sensor power draw differs by an order of magnitude, so efficient scheduling has tangible impact on total mission energy (see Appendix).

4.1 Baselines

We benchmark B-COD against alternatives chosen to isolate each of our findings while sharing the same low-level controller (see Appendix for hyperparameters). *(i) Always-ON* keeps every sensor powered and sets the upper bound on task success. *(ii) Greedy-OFF* is a hand-tuned rule that asks whether simple heuristics obviate learning: day camera disabled below 10 lux, night camera disabled above 10 lux, LiDAR spun up if its previous sweep reported a return within 15 m, GPS + IMU always on, EXO2 sonde enabled in high-interest water. The values are grid-searched on a held-out set. To test whether a principled heuristic suffices, *(iii) InfoGain-Greedy* keeps every sensor off except the one predicted by the analytic observation model to maximise single-step expected entropy reduction of the Gaussian-mixture belief. *(iv-v) Random-K* controls sensing duty-cycle by sampling one or two sensors, irrespective of context. To verify that the advantage stems from the risk proxy, we keep the SAC unchanged and swap its inputs: *(vi) σ-Mean* feeds the actor the mean of B-COD's waypoint

standard deviations; *(vii) Sample-Spread* drops the variance head and estimates risk from the empirical spread of 20 Monte-Carlo trajectory samples. Baselines (i)-(vii) keep the B-COD planner frozen. To probe representation: *(viii) No-Belief Raster* trains the diffusion model on a single delta-pose channel only. We benchmark against alternative planners: *(ix) Pure-RL* lets a constrained-SAC [115, 116] learn both motion primitives and sensor toggles end-to-end and is trained for 50 M environment steps. *(x) DESPOT-Lite* [117], runs a 5k-particle online POMDP search with analytic observation models.

4.2 Key Findings

Key Finding #1: B-COD+SAC delivers near-perfect task completion at less than half the sensing cost of the Always-ON baseline. Table 1 summarizes performance over 50 laps. B-COD reaches the goal on 97.9 % of attempts, yet spends only 42 % of the energy. Collisions remain at 0.9 %, essentially identical to the Always-ON baseline. Heuristic scheduling cannot match this trade-off: Greedy-OFF conserves energy (61 %) but sacrifices success (47 %). InfoGain-Greedy raises success to 90 % yet violates risk eight times more often than B-COD. Random masks fare worse, proving that local environment context–not just a lower duty cycle–is essential for task completion. Pure-RL generates trajectories and schedules sensors from raw rasters; the high-dimensional action space makes exploration sparse, and the policy converges to risk-averse dithering–only 55 % goals reached and a 22 % collision rate. DESPOT-Lite, by contrast, evaluates a principled belief tree with analytic models and therefore is able to plan accurately, but it expands hundreds of nodes; the resulting 0.5s runtime renders it unusable in real-time on the vehicle.

Metric	*AON*	*GOF*	*IGG*	*R1*	*R2*	*σM*	*SS*	*NB*	*PRL*	*DL*	*B-COD*
Goal-reach (%) ↑	100.0	47.3	89.9	18.5	29.1	79.6	94.3	67.8	54.8	87.9	97.9
Collision (%) ↓	0.5	22.3	6.1	34.5	30.1	12.4	4.7	17.4	22.1	4.2	0.9
CVaR violations (%) ↓	0.1	15.8	4.3	28.6	22.8	9.1	5.2	13.2	18.3	1.9	0.5
Mean #sensors ↓	5.0	3.19	2.65	1.0	2.0	2.99	2.56	4.05	3.48	5.0	2.08
Energy vs AON (%) ↓	100.0	61.2	49.8	24.2	38.9	60.1	91.2	68.2	67.5	100	42.3
Runtime (ms) ↓	14.9	14.7	26.8	13.6	13.7	14.4	84.1	14.1	12.1	565.3	14.3
Peak RAM (MB) ↓	305	282	403	277	281	287	674	279	299	731	284

Table 1. Performance comparison. Keys – AON: *Always-ON*; GOF: *Greedy-OFF*; IGG: *InfoGain-Greedy*; R1/R2: *Random 1/2*; σM: *σ-Mean*; SS: *Sample-Spread*; NB: *No-Belief Raster*; PRL: *Pure-RL*; DL: *DESPOT-Lite*

r (m)	*IGG*	*DL*	*B-COD*
25	7.5 ms	565 ms	9.8 ms
40	10.9 ms	1446 ms	9.7 ms
55	14.6 ms	2737 ms	9.6 ms
70	18.2 ms	4430 ms	10.7 ms
85	18.7 ms	6536 ms	10.4 ms
100	23.3 ms	9040 ms	10.9 ms

Table 2. Wall-clock latency vs. radius r. Average of 5 runs.

Key Finding #2: B-COD's variance is a calibrated, context-aware predictor of localisation error.

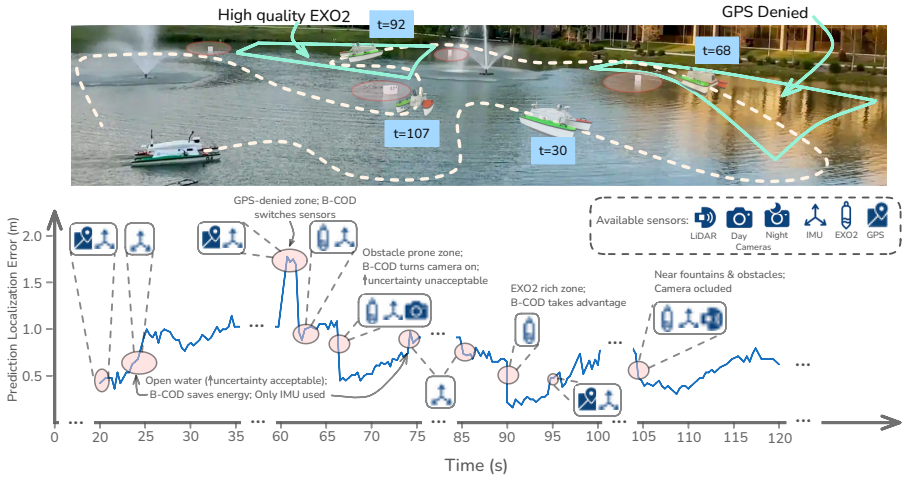

Figure 3: Representative lap with B-COD. Obstacles (red circles), GPS denied and rich EXO2 zones are marked. The lower panel plots B-COD's predicted localisation error vs time alongside the sensor-activation traces (only representative toggles are shown). Brief annotations indicate our interpretation that triggered the toggle.

The reliability curve (Figure 4) shows numerical calibration (6 % mean error). The reason behind the bound relaxing follows directly from what the diffusion planner is told to care about:

belief shape, active sensors, and local map geometry. Call-out B (night lap, obstacle corridor): The semantic map reports obstacles–a fountain and a floating buoy. Colliding here would be mission-ending, so B-COD turns on LiDAR, night camera and IMU (high energy). The planner now expects rich pose updates and knows that centimeters of pose error matter; it shrinks the bound to 0.45 m, almost matching the 0.46 m ground truth drift. Call-out A (day lap): Tens of meters separate the ASV from any hazard. With only the IMU running (low energy), B-COD predicts pure dead-reckoning growth yet also "knows" that a meter of drift will not intersect anything. It therefore widens the bound to 1.85 m, closely tracking the ground truth 2.0 m error. These results demonstrate that u^{CVaR} is numerically reliable and spatially discriminative, providing the scheduler with the rich information needed to trade energy for certainty. Figure 3 shows these risk swings along the full lap for qualitative context.

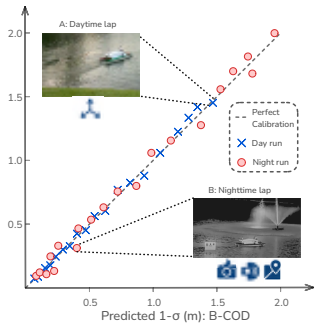

Figure 4: Reliability diagram: 20 bins plot the empirical planar error (y-axis) against B-COD's predicted (x-axis).

Key Finding #3: B-COD stays within a 10 ± 1 ms envelope and out-scales analytic belief planners. Table 2 sweeps the workspace radius from 25 m to 100 m (full lake sector). B-COD's latency is flat–10.3 ± 0.6 ms throughout—because the belief crop is *always* down-sampled and the UNet's receptive field is fixed; compute therefore scales with network width, not with world area. The InfoGain-Greedy baseline must update an n-cell covariance grid; its cost grows $\Theta(R^2)$[15], reaching 23 ms at 100 m. DESPOT-Lite's branching factor of the belief tree increases with visible free space; runtime balloons to 9000 ms over the same sweep, far beyond what an embedded loop can absorb. The takeaway is practical as well as theoretical: constant-time scaling lets B-COD replan over lake-scale horizons without ever violating the real-time threshold, whereas analytic planners become the computational bottleneck well before the map reaches lake-scale.

Key Finding #4: B-COD adapts online, re-allocating modalities to recover from faults. During a daytime lap, we manually disabled the LiDAR 30 s before the ASV entered the narrow fountain corridor, which demands typically sub-meter localisation. B-COD's risk proxy spiked from 0.6 m to 1.8 m as soon as the loss of range data was reflected. The SA reacted on the next cycle: it re-enabled both cameras and the EXO2 sonde, accepted the high energy penalty, and drove risk down to 0.8 m. Once clear of the obstacle, the proxy dropped naturally; the scheduler shut the extra modalities off and returned to the energy-saving IMU-only sensor choice. No heuristic was required–the planner's calibrated variance alone drove the correct, context-specific recovery sequence.

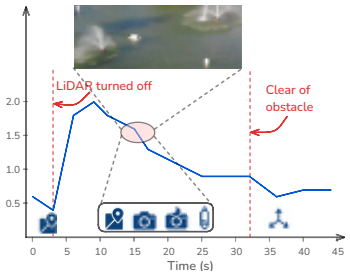

Figure 5: Scheduler response to a forced LiDAR outage during a lap.

Ablations: Removing the belief raster (NB) deprives the planner of map semantics: goal-reach collapses to 68 %, collisions triple, and the SAC reacts by keeping four sensors on. Replacing CVaR with a naive waypoint average (σ-M) leaves the variance head intact but blurs risk peaks; the policy over-compensates, averaging three sensors and burning 60% of Always-ON energy, while still breaching the risk budget 9% of the time. Estimating risk from 20 Monte-Carlo samples (SS) restores calibration but at the cost of high runtime.

5 Conclusion

B-COD demonstrates that a diffusion planner can double as a sensing oracle, enabling "just-enough" sensing for navigation. B-COD couples (i) a belief raster, (ii) a one-step diffusion model conditioned on belief and sensor flags that returns a short trajectory and a calibrated proxy, and (iii) a risk-constrained scheduler that powers only the sensors needed to keep that risk below a user budget. On hardware experiments B-COD reaches its goals with 98% success while cutting sensing energy by $>$ 50 %, maintains variance calibration to 6 %, replans in 10 ms–far faster than analytic belief planners.

6 Limitations

Although B-COD delivered strong real-world results–near-perfect goal completion, tight error calibration, and substantial energy savings–it still rests on some assumptions that limit immediate deployment beyond the scenarios tested.

Domain-specific retraining: The sensor mask enters the network as a learned embedding whose dimension and semantics are fixed at training time. Adding a new modality—or swapping a lidar for a lower-power radar—requires collecting demonstrations and fine-tuning the entire model. A more versatile alternative is to encode each sensor through a shared description (e.g., field-of-view, expected information gain per joule) and learn the embedding once, allowing plug-and-play expansion of the suite.

Dependence on a static semantic map: B-COD conditions on pre-loaded layers such as obstacle occupancy and GNSS visibility. If construction alters the shoreline or a temporary stage blocks a channel, the planner may route through stale free space. Coupling the raster with on-line semantic SLAM or incremental map-updating would mitigate this brittleness and let the system react to novel structures and transient occlusions.

Binary sensor model: The current scheduler flips each modality fully on or off. In reality many sensors admit graded settings—reduced lidar scan rate, lower camera resolution, duty-cycled GPS fixes—that offer finer energy/performance trade-offs. Extending the action space to multi-level or continuous controls, possibly guided by differentiable power/quality curves, could unlock further savings without compromising risk.

Calibration tied to training distribution: Variance honesty rests on the assumption that the demonstration data span the operating envelope. If the robot later encounters lighting, weather, or sensor faults absent from the corpus, u^{CVaR} may under-state true error. Techniques such as confidence-aware data augmentation, out-of-distribution detectors, or post-deployment Bayesian recalibration would help maintain reliability as conditions drift.

Addressing these limitations—expressive sensor descriptors, live map updates, multi-level energy controls, and distribution-aware calibration—forms the next step toward a fully adaptive, task-agnostic "just-enough sensing" navigation stack.

7 Acknowledgments

This work was supported by National Science Foundation grants 2118329, IIS-2024733, IIS-2331908, the Air Force Research Laboratory under Award FA8651-23-1-0003, the Office of Naval Research under the grants N00014-23-1-2789, N00014-25-1-2369, N00014-23-1-2505, and N00014-23-1-2651, the U.S. Department of Defense grant 78170-RT-REP, the FDEP grant INV31, and by the Army Research Laboratories under contract W911NF1920243. This is publication #2035 from the Institute of Environment at Florida International University

References

[1] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. J. Leonard. Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. *IEEE Transactions on Robotics*, 2016.

[2] C. Debeunne and D. Vivet. A review of visual-LiDAR fusion based simultaneous localization and mapping. *Sensors*, 2020.

[3] Z. Wang, Y. Wu, and Q. Niu. Multi-sensor fusion in automated driving: A survey. *IEEE Access*, 2020.

[4] A. Majumdar, Z. Mei, and V. Pacelli. Fundamental limits for sensor-based robot control. *The International Journal of Robotics Research*, 2023.

[5] A. V. Malawade, T. Mortlock, and M. A. Al Faruque. EcoFusion: Energy-aware adaptive sensor fusion for efficient autonomous vehicle perception. In *ACM/IEEE Design Automation Conference*, 2022.

[6] J. Kim, J. Park, and W. Chung. Self-diagnosis of localization status for autonomous mobile robots. *Sensors*, 2018.

[7] M. O. A. Aqel, M. H. Marhaban, M. I. Saripan, and N. B. Ismail. Review of visual odometry: Types, approaches, challenges, and applications. *SpringerPlus*, 2016.

[8] A. akbar Agha-mohammadi, S. Chakravorty, and N. M. Amato. FIRM: Sampling-based feedback motion-planning under motion uncertainty and imperfect measurements. *The International Journal of Robotics Research*, 2014.

[9] J. van den Berg, P. Abbeel, and K. Goldberg. LQG-MP: Optimized path planning for robots with motion uncertainty and imperfect state information. In *Robotics: Science and Systems*, 2010.

[10] F. S. Barbosa, B. Lacerda, P. Duckworth, J. Tumova, and N. Hawes. Risk-aware motion planning in partially known environments. In *IEEE Conference on Decision and Control*, 2021.

[11] C. Papachristos, S. Khattak, and K. Alexis. Autonomous exploration of visually-degraded environments using aerial robots. In *IEEE International Conference on Unmanned Aircraft Systems*, 2017.

[12] P. Ondruška, C. Gurău, L. Marchegiani, C. H. Tong, and I. Posner. Scheduled perception for energy-efficient path following. In *IEEE International Conference on Robotics and Automation*, 2015.

[13] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 1998.

[14] M. T. J. Spaan and P. U. Lima. A decision-theoretic approach to dynamic sensor selection in camera networks. In *International Conference on Automated Planning and Scheduling*, 2009.

[15] B. Charrow, N. Michael, and V. Kumar. Information-theoretic planning with trajectory optimization for dense 3d mapping. In *Robotics: Science and Systems*, 2015.

[16] A. S. Leong, A. Ramaswamy, D. E. Quevedo, H. Karl, and L. Shi. Deep reinforcement learning for wireless sensor scheduling in cyber-physical systems. *Automatica*, 2020.

[17] M. Alali, A. Kazeminajafabadi, and M. Imani. Deep reinforcement learning sensor scheduling for effective monitoring of dynamical systems. *Systems Science & Control Engineering*, 2024.

[18] G. Puthumanaillam, M. Vora, and M. Ornik. Comtraq-mpc: Meta-trained dqn-mpc integration for trajectory tracking with limited active localization updates. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 13592–13598, 2024. doi:10.1109/IROS58592.2024.10801659.

[19] I. Akkaya, M. Andrychowicz, M. Chociej, M. Litwin, B. McGrew, A. Petron, A. Paino, M. Plappert, G. Powell, R. Ribas, et al. Solving rubik's cube with a robot hand. *arXiv preprint arXiv:1910.07113*, 2019.

[20] M. Alshiekh, R. Bloem, R. Ehlers, B. Könighofer, S. Niekum, U. Topcu, and M. Wulf. Safe reinforcement learning via shielding. In *AAAI Conference on Artificial Intelligence*, 2018.

[21] K. Honda, R. Yonetani, M. Nishimura, and T. Kozuno. When to replan? An adaptive replanning strategy for autonomous navigation using deep reinforcement learning. In *IEEE International Conference on Robotics and Automation*, 2024.

[22] M. Janner, Y. Du, J. Tenenbaum, and S. Levine. Planning with diffusion for flexible behavior synthesis. In *International Conference on Machine Learning*, 2022.

[23] Y. Luo, C. Sun, J. B. Tenenbaum, and Y. Du. Potential based diffusion motion planning. In *International Conference on Machine Learning*, 2024.

[24] C. Chi, S. Feng, Y. Du, Z. Xu, E. Cousineau, B. Burchfiel, and S. Song. Diffusion policy: Visuomotor policy learning via action diffusion. In *Robotics: Science and Systems*, 2023.

[25] J. a. Carvalho, A. T. Le, M. Baierl, D. Koert, and J. Peters. Motion planning diffusion: Learning and planning of robot motions with diffusion models. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2023.

[26] J. Sun, Y. Jiang, J. Qiu, P. Talpur Nobel, M. Kochenderfer, and M. Schwager. Conformal prediction for uncertainty-aware planning with diffusion dynamics models. In *Advances in Neural Information Processing Systems*, 2023.

[27] J. Ho, A. Jain, and P. Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems*, 2020.

[28] L. Feng, P. Gu, B. An, and G. Pan. Resisting stochastic risks in diffusion planners with the trajectory aggregation tree. In *International Conference on Machine Learning*, 2024.

[29] J. V. D. Berg, P. Abbeel, and K. Goldberg. Lqg-mp: Optimized path planning for robots with motion uncertainty and imperfect state information. In *Robotics: Science and Systems*, 2010.

[30] K. Sun and V. Kumar. Belief space planning for mobile robots with range sensors using ilqg. *IEEE Robotics and Automation Letters*, 2021.

[31] S. Thrun, Y. Liu, D. Koller, A. Y. Ng, Z. Ghahramani, and H. Durrant-Whyte. Simultaneous localization and mapping with sparse extended information filters. *The International Journal of Robotics Research*, 2004.

[32] T. Shan, B. Englot, D. Meyers, W. Wang, C. Ratti, and D. Rus. LIO-SAM: Tightly-coupled lidar inertial odometry via smoothing and mapping. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2020.

[33] F. Dellaert and M. Kaess. Square root SAM: Simultaneous localization and mapping via square root information smoothing. *The International Journal of Robotics Research*, pages 1181–1203, 2006.

[34] M. Kaess, A. Ranganathan, and F. Dellaert. iSAM: Incremental smoothing and mapping. *IEEE Transactions on Robotics*, 2008.

[35] A. Rosinol, A. Violette, M. Abate, N. Hughes, Y. Chang, J. Shi, A. Gupta, and L. Carlone. Kimera: From SLAM to spatial perception with 3D dynamic scene graphs. *The International Journal of Robotics Research*, 2021.

[36] P. Karkus, S. Cai, and D. Hsu. Differentiable SLAM-Net: Learning particle slam for visual navigation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021.

[37] A. Younis and E. Sudderth. Learning to be Smooth: An end-to-end differentiable particle smoother. In *Neural Information Processing Systems*, 2024.

[38] J. Lim and K. H. Chon. Minimax Rao-blackwellized particle filtering in 2D LIDAR SLAM. *International Journal of Control, Automation and Systems*, 2024.

[39] T. Liu, C. Xu, Y. Qiao, C. Jiang, and J. Yu. Particle filter SLAM for vehicle localization. *arXiv preprint arXiv:2402.07429*, 2024.

[40] J. Pineau, G. Gordon, S. Thrun, et al. Point-based value iteration: An anytime algorithm for POMDPs. In *International Joint Conference on Artificial Intelligence*, 2003.

[41] M. T. Spaan and N. Vlassis. Perseus: Randomized point-based value iteration for POMDPs. *Journal of Artificial Intelligence Research*, 2005.

[42] H. Kurniawati, D. Hsu, and W. S. Lee. SARSOP: Efficient point-based pomdp planning by approximating optimally reachable belief spaces. In *Robotics: Science and systems*, volume 2008. Citeseer, 2008.

[43] A. Somani, N. Ye, D. Hsu, and W. S. Lee. DESPOT: Online POMDP planning with regularization. In *Advances in Neural Information Processing Systems*, 2013.

[44] P. Cai, Y. Luo, D. Hsu, and W. S. Lee. HyP-DESPOT: A hybrid parallel algorithm for online planning under uncertainty. *The International Journal of Robotics Research*, 2021.

[45] Y. Liang, E. Kim, W. Thomason, Z. Kingston, H. Kurniawati, and L. E. Kavraki. Scaling long-horizon online POMDP planning via rapid state space sampling. *arXiv preprint arXiv:2411.07032*, 2024.

[46] N. E. Du Toit and J. W. Burdick. Probabilistic collision checking with chance constraints. *IEEE Transactions on Robotics*, 2011.

[47] B. Luders, M. Kothari, and J. How. Chance constrained RRT for probabilistic robustness to environmental uncertainty. In *AIAA Guidance, Navigation, and Control Conference*, 2010.

[48] S. Dai, S. Schaffert, A. Jasour, A. Hofmann, and B. Williams. Chance constrained motion planning for high-dimensional robots. In *International Conference on Robotics and Automation*, 2019.

[49] K. Glasheen, J. J. Bird, and E. W. Frew. Experimental assessment of chance-constrained motion planning for small uncrewed aircraft. *Field Robotics*, 2024.

[50] F. Bourgault, A. A. Makarenko, S. B. Williams, B. Grocholsky, and H. F. Durrant-Whyte. Information based adaptive robotic exploration. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2002.

[51] L. M. Miller and T. D. Murphey. Trajectory optimization for continuous ergodic exploration. In *American Control Conference*, 2013.

[52] B. Charrow, G. Kahn, S. Patil, S. Liu, K. Goldberg, P. Abbeel, N. Michael, and V. Kumar. Information-theoretic planning with trajectory optimization for dense 3D mapping. In *Robotics: Science and Systems*, 2015.

[53] M. M. Sun, A. Gaggar, P. Trautman, and T. Murphey. Fast ergodic search with kernel functions. *arXiv preprint arXiv:2403.01536*, 2024.

[54] S. Dutta, N. Wilde, and S. L. Smith. A unified approach to optimally solving sensor scheduling and sensor selection problems in Kalman filtering. In *IEEE Conference on Decision and Control*, 2023.

[55] L. Carlone and D. Lyons. Uncertainty-constrained robot exploration: A mixed-integer linear programming approach. In *IEEE International Conference on Robotics and Automation*, 2014.

[56] B. Schlotfeldt, V. Tzoumas, and G. J. Pappas. Resilient active information acquisition with teams of robots. *IEEE Transactions on Robotics*, 2021.

[57] J. Yu, M. Schwager, and D. Rus. Correlated orienteering problem and its application to informative path planning for persistent monitoring tasks. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2014.

[58] M. Lauri, J. Pajarinen, J. Peters, and S. Frintrop. Multi-sensor next-best-view planning as matroid-constrained submodular maximization. *IEEE Robotics and Automation Letters*, 2020.

[59] M. Corah and N. Michael. Distributed matroid-constrained submodular maximization for multi-robot exploration: Theory and practice. *Autonomous Robots*, 2019.

[60] M. H. Kazma and A. F. Taha. Multilinear extensions in submodular optimization for optimal sensor scheduling in nonlinear networks. *arXiv preprint arXiv:2408.03833*, 2024.

[61] D. Maity and J. S. Baras. Dynamic, optimal sensor scheduling and value of information. In *International Conference on Information Fusion*, 2015.

[62] V.-P. Bui, S. R. Pandey, F. Chiariotti, and P. Popovski. Scheduling policy for value-of-information (VOI) in trajectory estimation for digital twins. *IEEE Communications Letters*, 2023.

[63] A. Krishna, E. S. Hu, and D. Jayaraman. The value of sensory information to a robot. In *International Conference on Learning Representations*, 2025.

[64] S. T. Jawaid and S. L. Smith. Submodularity and greedy algorithms in sensor scheduling for linear dynamical systems. *Automatica*, 2015.

[65] A. Hashemi, M. Ghasemi, H. Vikalo, and U. Topcu. A randomized greedy algorithm for near-optimal sensor scheduling in large-scale sensor networks. In *American Control Conference*, 2018.

[66] H. Liu, Y. Li, K. H. Johansson, J. Mårtensson, and L. Xie. Rollout approach to sensor scheduling for remote state estimation under integrity attack. *Automatica*, 2022.

[67] F. Hoffmann, A. Charlish, M. Ritchie, and H. Griffiths. Policy rollout action selection in continuous domains for sensor path planning. *IEEE Transactions on Aerospace and Electronic Systems*, 2021.

[68] H. A. Jasim. *Dynamic duty cycle mechnism for mobility in wirless sensor networks*. PhD thesis, Universiti Tun Hussein Onn Malaysia, 2020.

[69] M. Diyan, M. Khan, B. Nathali Silva, and K. Han. Scheduling sensor duty cycling based on event detection using bi-directional long short-term memory and reinforcement learning. *Sensors*, 2020.

[70] M. Alali, A. Kazeminajafabadi, and M. Imani. Deep reinforcement learning sensor scheduling for effective monitoring of dynamical systems. *Systems Science & Control Engineering*, 2024.

[71] J. Chen, W. Liu, D. E. Quevedo, S. R. Khosravirad, Y. Li, and B. Vucetic. Structure-enhanced drl for optimal transmission scheduling. *IEEE Transactions on Wireless Communications*, 2023.

[72] Z. Hajiakhondi-Meybodi, M. Hou, and A. Mohammadi. JUNO: Jump-start reinforcement learning-based node selection for uwb indoor localization. In *IEEE Global Communications Conference*, 2022.

[73] M. Shurrab, S. Singh, R. Mizouni, and H. Otrok. IoT sensor selection for target localization: A reinforcement learning based approach. *Ad Hoc Networks*, 2022.

[74] C. Jiang, W. Chen, Z. Wang, and W. Xiao. Deep reinforcement learning-based joint sequence scheduling and trajectory planning in wireless rechargeable sensor networks. *IEEE Sensors Journal*, 2024.

[75] A. Bukhari and H. Kim. Learning-based sensor scheduling for event classification on embedded edge devices. In *ACM/IEEE Conference on Internet of Things Design and Implementation*, 2023.

[76] R. Jurdak, P. Corke, D. Dharman, and G. Salagnac. Adaptive GPS duty cycling and radio ranging for energy-efficient localization. In *Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems*, 2010.

[77] K. Lin, A. Kansal, D. Lymberopoulos, and F. Zhao. Energy-accuracy aware localization for mobile devices. 2010.

[78] S. Ross, G. J. Gordon, and J. A. Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *International Conference on Artificial Intelligence and Statistics*, 2010.

[79] J. Ho and S. Ermon. Generative adversarial imitation learning. In *Neural Information Processing Systems*, 2016.

[80] F. Codevilla, M. Müller, A. Dosovitskiy, A. M. López, and V. Koltun. End-to-end driving via conditional imitation learning. *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2017.

[81] C. A. Hepburn and G. Montana. Model-based trajectory stitching for improved behavioural cloning and its applications. *ArXiv*, 2022.

[82] M. Memmel, J. Berg, B. Chen, A. Gupta, and J. Francis. STRAP: Robot sub-trajectory retrieval for augmented policy learning. *arXiv preprint arXiv:2412.15182*, 2024.

[83] N. Lee, W. Choi, P. Vernaza, C. B. Choy, P. H. S. Torr, and M. Chandraker. Desire: Distant future prediction in dynamic scenes with interacting agents. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[84] B. Ivanovic and M. Pavone. The Trajectron: Probabilistic multi-agent trajectory modeling with dynamic spatiotemporal graphs. In *IEEE/CVF International Conference on Computer Vision*, 2018.

[85] T. Salzmann, B. Ivanovic, P. Chakravarty, and M. Pavone. Trajectron++: Dynamically-feasible trajectory forecasting with heterogeneous data. In *European Conference on Computer Vision*, 2020.

[86] F. Janjoš, M. Hallgarten, A. Knittel, M. Dolgov, A. Zell, and J. M. Zöllner. Conditional unscented autoencoders for trajectory prediction. In *European Conference on Computer Vision*, 2024.

[87] Z. Dong, J. Hao, Y. Yuan, F. Ni, Y. Wang, P. Li, and Y. Zheng. DiffuserLite: Towards real-time diffusion planning. In *Advances in Neural Information Processing Systems*, 2024.

[88] Y. Shaoul, I. Mishani, S. Vats, J. Li, and M. Likhachev. Multi-robot motion planning with diffusion models. *arXiv preprint arXiv:2410.03072*, 2024.

[89] Y. Zhu, Y. Ye, S. Zhang, X. Zhao, and J. J. Yu. DiffTraj: Generating GPS trajectory with diffusion probabilistic model. In *Advances in Neural Information Processing Systems*, 2023.

[90] J.-B. Bouvier, K. Ryu, K. Nagpal, Q. Liao, K. Sreenath, and N. Mehr. DDAT: Diffusion policies enforcing dynamically admissible robot trajectories. *arXiv preprint arXiv:2502.15043*, 2025.

[91] B. Mu, M. Giamou, L. Paull, A. akbar Agha-mohammadi, J. J. Leonard, and J. P. How. Information-based active SLAM via topological feature graphs. In *IEEE Conference on Decision and Control*, 2015.

[92] Z. Wang, H. Chen, S. Zhang, and Y. Lou. Active view planning for visual slam in outdoor environments based on continuous information modeling. *IEEE/ASME Transactions on Mechatronics*, 2022.

[93] M. Bryson and S. Sukkarieh. Observability analysis and active control for airborne SLAM. *IEEE Transactions on Aerospace and Electronic Systems*, 2008.

[94] A. Krause, A. P. Singh, and C. Guestrin. Near-optimal sensor placements in gaussian processes. In *International Conference on Machine Learning*, 2005.

[95] M. Corah and N. Michael. Distributed matroid-constrained submodular maximization for multi-robot exploration: Theory and practice. *Autonomous Robots*, 2018.

[96] D. S. Chaplot, D. Gandhi, S. Gupta, A. Gupta, and R. Salakhutdinov. Learning to explore using active neural SLAM. *arXiv preprint arXiv:2004.05155*, 2020.

[97] P. Yang, Y. Liu, S. Koga, A. Asgharivaskasi, and N. Atanasov. Learning continuous control policies for information-theoretic active perception. In *International Conference on Robotics and Automation*, 2023.

[98] H. Nguyen, S. H. Fyhn, P. De Petris, and K. Alexis. Motion primitives-based navigation planning using deep collision prediction. In *International Conference on Robotics and Automation*, 2022.

[99] F. Dellaert, D. Fox, W. Burgard, and S. Thrun. Monte Carlo localization for mobile robots. In *IEEE International Conference on Robotics and Automation*, 1999.

[100] A review of convolutional neural networks in computer vision. *Artificial Intelligence Review*, 2024.

[101] M. Chen, S. Mei, J. Fan, and M. Wang. An overview of diffusion models: Applications, guided generation, statistical rates and optimization. *arXiv preprint arXiv:2404.07771*, 2024.

[102] X. Fan, Z. Wu, and H. Wu. A survey on pre-trained diffusion model distillations. *arXiv preprint arXiv:2502.08364*, 2025.

[103] Y. Song, P. Dhariwal, M. Chen, and I. Sutskever. Consistency models. In *International Conference on Machine Learning*, 2023.

[104] R. T. Rockafellar and S. Uryasev. Optimization of conditional value-at risk. *Journal of Risk*, 2000.

[105] E. Altman. *Constrained Markov Decision Processes*. Routledge, 2021.

[106] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel, et al. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*, 2018.

[107] G. Puthumanaillam, P. Padrao, J. Fuentes, L. Bobadilla, and M. Ornik. Enhancing robot navigation policies with task-specific uncertainty managements. *arXiv preprint arXiv:2505.13837*, 2025.

[108] P. E. Hart, N. J. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 1968.

[109] C. E. Garcia, D. M. Prett, and M. Morari. Model predictive control: Theory and practice-A survey. *Automatica*, 1989.

[110] ONNX Working Group. ONNX: Open neural network exchange. https://onnx.ai, 2019.

[111] NVIDIA Corporation. *NVIDIA TensorRT Developer Guide (Version 8.6)*. NVIDIA, 2023. URL https://docs.nvidia.com/deeplearning/tensorrt/developer-guide/index.html.

[112] NVIDIA Corporation. *Jetson Orin NX Series System-on-Module: Technical Specifications*. NVIDIA, 2024. URL https://developer.nvidia.com/embedded/jetson-orin-nx.

[113] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, 2012.

[114] SeaRobotics Corporation. SR–Surveyor Class Autonomous Surface Vehicle. https://www.searobotics.com/products/autonomous-surface-vehicles/sr-surveyor-class, 2025.

[115] X. Zhou, X. Zhang, H. Zhao, J. Xiong, and J. Wei. Constrained Soft Actor-Critic for Energy-Aware Trajectory Design in UAV-Aided IoT Networks. *IEEE Wireless Communications Letters*, 2022.

[116] G. Puthumanaillam, P. Padrao, J. Fuentes, L. Bobadilla, and M. Ornik. Guided agents: Enhancing navigation policies through task-specific uncertainty abstraction in localization-limited environments. *arXiv preprint arXiv:2410.15178*, 2024.

[117] A. Somani, N. Ye, D. Hsu, and W. S. Lee. DESPOT: Online POMDP Planning with Regularization. In *Advances in Neural Information Processing Systems*, 2013.

Appendix: Implementation and Evaluation Details

Project resources

Website https://bcod-diffusion.github.io
Code repository https://github.com/bcod-diffusion/bcod
Dataset https://github.com/bcod-diffusion/dataset

The Appendix is organised as follows.

1. *Implementation Recipes.* Section 1.1 details the belief-rasterisation. Section 1.2 describes the diffusion teacher UNet. Section 1.3 gives the single-step consistency student. Section 1.4 presents the constrained-SAC schedule.
2. *Experimental Set-up.* Section 2.1 tabulates the six-sensor payload, update rates, noise models and peak power draws used in all lake trials. Section 2.2 lists the hyperparameters that governs both our method and every baseline.

3. *Open Maritime Dataset.* Section 3.1 explains the real-to-simulation logging pipeline and the Unity+ROS reconstruction process. Section 3.2 defines the 50,123 belief-annotated snippets, their file structure and recommended train/val/test splits. Section 3.3 provides summary statistics (lighting, obstacle density, belief spread) and dataloaders.

A Implementation Recipes

A.1 Belief Rasterization

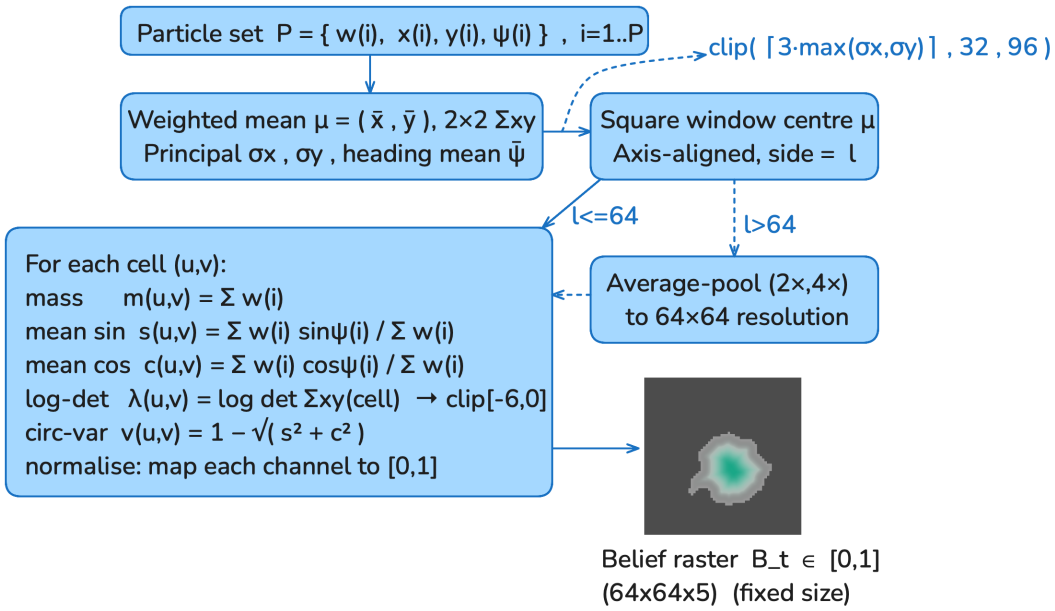

Figure 6: Belief rasterization flowchart

The rasteriser runs as a stand-alone C++17 module on the ASV and is invoked once per high-level tick. At each call the incoming estimator state consists of $P = 500$ weighted particles, each storing position, yaw and, when available, the 2x2 planar covariance already maintained by the EKF; raw particles simply carry an identity covariance. A single linear scan computes the weighted mean $\bar{x}, \bar{y}$ and the full sample covariance Σ_{xy}. The square window is then anchored on $(\bar{x}, \bar{y})$ and its side length is set to

$$l = \mathrm{clip}\big(\lceil 3\max(\sigma_x, \sigma_y)\rceil,\ 32,\ 96\big),$$

where σ_x, σ_y are the principal standard deviations obtained from Σ_{xy}. For all lake trials this rule contains at least 99.7% probability mass while keeping $l \le 96$. If $l \le 64$ the histogram is performed directly into a $l \times l$ grid; otherwise we first bin into the larger lattice and apply a uniform average-pool whose stride equals the integer ratio $l/64$. Down-sampling only occurs in open water when no obstacles lie within 8 m, so aliasing does not affect subsequent collision checks.

As explained in the paper, for every occupied cell we accumulate five statistics. The mass channel stores the true probability, already in $[0, 1]$. Orientation enters through the particle-weighted means of $\sin\psi$ and $\cos\psi$; after a linear map $(x \mapsto 0.5x + 0.5)$ both lie in the unit interval and avoid discontinuities at $\pm\pi$. Position uncertainty is condensed into $\lambda_{uv} = \log\det\Sigma_{xy}^{(uv)}$; values below -6 (area $\exp(-6) \approx 2.5 \times 10^{-3}\,\mathrm{m}^2 \approx 25\,\mathrm{cm}^2$) or above 0 are clipped and then mapped to $[0, 1]$ by $x \mapsto (x+6)/6$. Circular variance $v_{uv} = 1 - \sqrt{s_{uv}^2 + c_{uv}^2}$ already satisfies the bound and requires no scaling. Unoccupied cells receive $(0, 0.5, 0.5, 0, 0)$ so the background is neutral to the convolutional encoder. The entire procedure is $\mathrm{O}(P)$; with $P = 500$ it consumes 0.93 ms on a single core (Jetson Orin NX, -O3, no SIMD) and adds a worst-case $80 \times 80 = 0.006$ ms LUT lookup for the clipping maps.

One fixed hyper-parameter controls the spatial resolution: with $l \leq 64$ the cell width is $w = l/64$ m, hence the nominal raster resolution is 0.25 m px^{-1} when $l = 16$ m and 0.50 m px^{-1} when $l = 32$ m. These scales were chosen because they guarantee that, even at maximum speed (1 m s^{-1}), the boat traverses at most two pixels per control cycle, avoiding aliasing in the planner's first convolutional stride. All normalisation constants were computed once on a 40-minute validation log and kept fixed for every run reported in the paper; no per-environment tuning is required. Gradients are never propagated through the rasteriser, so its piece-wise linear maps and hard clipping cannot destabilise network training.

A.1.1 Ablation study on raster statistics

We retain only five cell-wise moments because, for the horizons of interest, they are the smallest set that still encodes(i) collision geometry, (ii) the growth rate of positional drift, and (iii) heading ambiguity. Increasing the channel count widens every convolutional filter bank and inflates TensorRT workspaces, while removing any moment discards information the scheduler needs to trade energy for safety. Table 2 compares six variants against the full 5-channel raster on the day–night lake benchmark. Each experiment reuses *identical* planner, scheduler and hyper-parameters; numbers are the mean over 500 independent laps, with 95 % confidence intervals below ± 0.3 pp for all percentage metrics.

Variant (64×64 raster)	Channels C	Goal-reach ↑	Collisions ↓	CVaR viol. ↓	Sensor energy† ↓	Runtime (ms)‡ ↓	Peak GPU RAM (MB)* ↓
Baseline (5 stats)	5	**97.9**	**0.9**	**0.5**	**42**	10.4	284
− $\log\det\Sigma$	4	95.2	3.5	3.0	46	10.2	283
− circular variance	4	94.8	4.7	3.8	47	10.2	283
− sine / cosine yaw	3	91.6	7.3	5.4	52	10.1	282
+ $\Sigma_{xx}\,\Sigma_{yy}\,\rho_{xy}$	8	97.8	**0.8**	0.6	42	12.6	302
+ heading skew	6	97.8	0.9	0.5	43	11.0	290
+ particle count	6	98.0	0.9	0.6	43	10.9	289

Table 2: Influence of raster statistics on task performance. Arrows indicate preferable direction.
† Percentage of the *sensor-only* energy consumed by the Always-ON baseline (compute power is analysed separately in App. B.1). ‡ Mean forward-pass latency of the diffusion model plus rasterisation; SAC adds a $\approx$ 4.1 ms. * Peak memory measured with `torch.cuda.max_memory_allocated` after one control iteration.

Removing any of the five retained moments compromises safety. Without the log-determinant channel the planner cannot detect anisotropic banana-shaped position uncertainty, so collisions quadruple (0.9 % → 3.5 %) and the scheduler reacts by powering an extra sensor on 27 percent of time-steps, raising energy from 42 % to 46 %. Eliminating the circular-variance channel makes heading ambiguity invisible; both collisions and CVaR violations almost an order of magnitude higher than baseline reflect episodes in which the boat enters a corridor mis-oriented and the SAC has no early warning. Dropping the sine–cosine pair removes absolute orientation outright, slicing goal-reach by six percentage points and forcing the policy into a high-power safety mode (52 % sensor energy). Conversely, enriching the raster provides diminishing returns. Injecting the full planar covariance tensor ($+\Sigma$) shaves collisions by a mere 0.1 pp—well inside the confidence interval, yet inflates TensorRT workspaces by 18MB and pushes inference to 12.6ms under moderate CPU load. A sixth channel for heading skew or particle count produces statistically indistinguishable performance while adding measurable memory and latency overhead.

A.2 Belief Conditioned One-Step Diffusion

A.2.1 Diffusion Teacher: full implementation details

The diffusion teacher is a conditional UNet trained with a cosine forward-process on the 8.3 million short-horizon snippets described in the main paper. The encoder path corresponds to a 4-stage ResNet; we simply extend it with symmetrical up-sampling to obtain the UNet. Fig. 7 summarizes the model architecture.

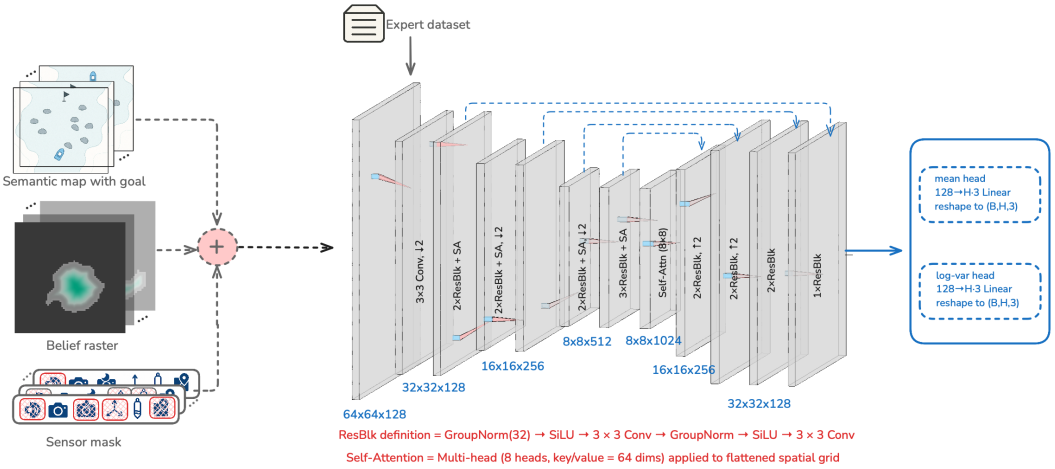

Figure 7: Model architecture: Diffusion Teacher

Input encoding and conditioning pathway. Each training example consists of a $64\times64\times C$ raster, a goal mask of identical spatial size, a three-channel semantic-map slice, and the 8-bit sensor-mask vector. The three spatial tensors are concatenated along the channel axis, giving $C = 5$ (belief) $+ 3$ (map) $+ 1$ (goal) $= 9$. A single 3×3 convolution with 128 output channels and stride 1 projects this $64\times64\times9$ stack into a 128-channel feature map; this layer appears as the stem conv row in Table 3. The binary sensor mask $a \in 0, 1^N$ is embedded by a learnable lookup table into a 32-dimensional vector. After a ReLU this vector is broadcast spatially and *added* to every feature plane produced by the stem. In practice the broadcast is implemented by first reshaping the embedding to $1\times1\times32$, repeating it to $64\times64\times32$, concatenating along the channel axis, and applying a 1×1 convolution back to 128 channels so that the subsequent residual blocks see the fusion of raster and sensor context in a single tensor.

Timestep embedding. For the forward diffusion index $t \in 1, \ldots, T$, with $T = 1000$, we adopt the cosine $\bar{\alpha}\_t$ schedule. A fixed sinusoidal positional-embedding layer converts t to a 256-dimensional vector; a two-layer feed-forward network with SiLU activations then yields a 512-vector (γ_t, β_t). At every residual block the feature map F is modulated by FiLM scaling: $F \leftarrow F \odot (1 + \gamma\_t) + \beta\_t$, where $\odot$ denotes channel-wise multiplication and the vectors are broadcast spatially. This scheme injects the timestep signal without an extra concatenation and halves VRAM relative to the naïve $(F|\text{emb}\_t)$ stack.

Residual blocks and self-attention. A *ResBlk* follows a standard Conv–GroupNorm–SiLU sequence, but with GroupNorm(32) to match NVIDIA TensorRT's fused kernels. Each block carries 128, 256, or 512 channels as prescribed in the table below. The 32×32 and 16×16 resolutions retain spatial self-attention: the feature map is reshaped to (B, HW, C), QKV projections compute eight-head dot-product attention, the result is reshaped back, layer-normed, and fed through a two-layer MLP before the residual add. At 8×8 resolution the attention head uses a causal-mask set to all ones because we found sequence ordering irrelevant for such small tensors, yet leaving the softmax fully dense reduces numerical instabilities during mixed-precision training.

Down- and up-sampling. Down-sampling is performed by 3×3 convolutions with stride 2 and kernel-padding 1, which preserves the receptive-field parity of the original UNet++ backbone. Up-sampling mirrors this with nearest-neighbour resize followed by a 3×3 stride-1 convolution. Skip-connections concatenate the encoder feature map with the decoder input, doubling the channel count at each merge; hence the decoder rows list twice the input channels relative to the corresponding encoder levels.

Bottleneck. At the 8×8 bottleneck the channel depth is temporarily doubled to 1024. A single self-attention block with eight heads sits here, followed by a ResBlk, before channel width is reduced back to 512 for decoding. Removing this bottleneck attention degraded mean-L2 path reconstruction by 0.9 cm and uncoupled waypoint variances from the global geometry, so the compute cost was judged worthwhile.

Output heads. After the last decoder block a 3×3 Conv+GroupNorm+SiLU returns a $64\times64\times128$ tensor. Global average-pool collapses the spatial grid, yielding a 128-vector per batch element. Two independent linear heads map this vector to the required outputs: the *mean-head* produces $H \times 3$ scalars that are reshaped to $(\Delta x, \Delta y, \Delta\psi)$ for each waypoint, while the *log-var-head* produces H scalars that become $\hat{\sigma}\_1 : H$. We fix $H = 8$ in all experiments.

Losses and optimisation. The training objective is the sum of the DDPM noise-reconstruction term and the diagonal-Gaussian negative-log-likelihood term described in Eq. (2) of the paper, with $\beta = 0.05$. Optimisation uses AdamW with $\beta\_1 = 0.9$, $\beta\_2 = 0.999$, weight-decay 1×10^{-4}, and gradient-clipping at an ℓ_2-norm of 1.0. The initial learning rate is 2×10^{-4} and follows a cosine decay to zero over 300000 iterations; a 1000-step linear warm-up precedes the decay. Batches of 256 snippets are distributed across eight A100 GPUs (per-device micro-batch 32) using PyTorch 2.2's `DistributedDataParallel`. Mixed-precision (FP16) is enabled with dynamic loss scaling; a loss-scale overflow triggers an automatic orthogonal-projection of the gradient to the unit hypersphere to avoid divergence. Training converges after 45 hours wall-clock; the final checkpoint is the exponential moving average (EMA) of the weights with decay 0.9999.

Full layer specification. Table 3 lists every stage, its channel counts, the number of residual-attention blocks, the resolution after the stage (assuming the canonical 64×64 input), and whether the stage downsamples or upsamples.

Stage	In C	Out C	Blocks	Scale ↑ / ↓	Output size
Encoder					
Stem conv	128	128	–	↓ 2	32×32
enc-1	128	128	ResBlk+SA ×2	–	32×32
enc-2	128	256	ResBlk+SA ×2	↓ 2	16×16
enc-3	256	512	ResBlk+SA ×2	↓ 2	8×8
enc-4	512	512	ResBlk+SA ×3	–	8×8
Bottleneck					
bottleneck	512	1024	SA (8×8)	–	8×8
Decoder					
dec-4	1024	512	ResBlk+SA ×2	↑ 2	16×16
dec-3	1024	256	ResBlk+SA ×2	↑ 2	32×32
dec-2	512	128	ResBlk+SA ×2	–	32×32
dec-1	256	128	ResBlk+SA ×1	–	32×32
Heads					
mean head	128	$H\times3$	Linear	–	–
log-var head	128	H	Linear	–	–

Table 3: UNet architecture of the diffusion teacher. All convolutions use reflection padding; GN = GroupNorm(32); SA = multi-head self-attention, 8 heads; SiLU activation everywhere.

A.2.2 Consistency-model student: implementation details and training recipe

The single-step *student* network, denoted g_ψ, reproduces the teacher's conditional trajectory distribution while reducing inference latency by an order of magnitude. Unless explicitly noted, choices are identical to the teacher's (Section A.2.1) so that the two checkpoints can be swapped at test time without touching any preprocessing code.

Input interface and conditioning pathway. The student consumes the same tensors as the teacher: the 64×64 raster–map–goal stack, the embedded sensor-mask vector and the sinusoidal diffusion-

timestep embedding. To keep the network lightweight we halve the channel width throughout. A 3×3 convolution projects the 9 concatenated spatial channels to 64 feature planes. The 32-dimensional sensor embedding is broadcast and added via a 1×1 convolution to the stem output; the FiLM modulation with (γ_t, β_t) derived from the timestep embedding is unchanged.

Backbone topology. Because the student must finish in under 10 ms on the Jetson Orin NX, it uses a two-down / two-up UNet. After the stem, two residual blocks with 64 channels operate at 64×64 resolution; a stride-2 convolution downsamples to 32×32 where two residual blocks with 128 channels run, each augmented with eight-head self-attention. A second stride-2 convolution produces the sole bottleneck at 16×16 and 256 channels; here a single residual block followed by multi-head attention sits. Decoding mirrors this path: nearest-neighbour upsample by two, concatenate the skip, apply two residual-attention blocks, upsample again, concatenate, and finish with one residual block. Channel counts therefore follow the series $64 \to 128 \to 256 \to 128 \to 64$. All convolutions use GroupNorm(32) and SiLU; attention is identical to the teacher but run only at 32×32 and 16×16, which empirical profiling showed to be the cheapest resolution that still preserved multi-modal coverage.

Output heads. As with the teacher, global average pooling yields a 64-vector that feeds three linear heads: an ϵ head and a mean head, each of size $H \times 3$, and a log-variance head of size H. The value $H = 8$ is retained so that the student's tensor shapes match the teacher's exactly and downstream checkpoints can be swapped without re-tracing TensorRT engines.

Consistency distillation objective. The teacher checkpoint is frozen. For every training step, a fresh latent $\xi \sim \mathcal{N}(0, I)$ is sampled; the teacher generates $\tau^{\text{ref}}, \hat{\sigma}^{\text{ref}}, \hat{\epsilon}^{\text{ref}}$. The student is forced to map the *same* latent and conditioning inputs directly to those references. The loss function is the sum of a mean-squared error on the mean trajectory, an identical MSE on the predicted noise residual and an analytically computed diagonal-Gaussian KL divergence between $\mathcal{N}(\hat{\mu}_\psi, \Sigma_\psi)$ and $\mathcal{N}(\tau^{\text{ref}}, \Sigma_{\text{ref}})$. The scalar weight λ in front of the KL term is linearly annealed from 0 to 0.5 over the first 50 k iterations, then held constant; this avoids early collapse of variances while still enforcing calibration in later epochs.

Optimisation schedule. AdamW is again used but with a smaller learning rate, 1.5×10^{-4}, and no weight decay, as we found that the reduced model has lower capacity and benefits from unconstrained norms. Training uses batches of 1024 latents (four A100-40 GB GPUs, micro-batch 256), runs for 200 k iterations and takes 13 hours wall-clock. All arithmetic is FP16 with dynamic loss scaling; gradient clipping at an ℓ_2 norm of 0.5 prevents rare spikes when the teacher's variance is near the clipping floor. We maintain an EMA with decay 0.9997 and export the EMA weights.

Runtime and memory. The final FP16 ONNX graph is fed to TensorRT 10.0 with full layer fusion. On the Orin NX in 25 W mode a forward pass, including FiLM modulation and the three heads, clocks at 5.7 ± 0.2 ms and consumes 146 MB of GPU RAM; the accompanying rasterisation and SAC inference raise the end-to-end control-loop budget to 9.9 ms, matching the numbers in Table 1 of the main text. Exact throughput is reproducible with the command line included in the artefact repository; no hidden environment variables or compiler flags are required.

Stage	In C	Out C	Blocks	Scale	Output size
Stem conv	64	64	–	$\downarrow 2$	32×32
enc-1	64	64	ResBlk+SA $\times 2$	–	32×32
enc-2	64	128	ResBlk+SA $\times 2$	$\downarrow 2$	16×16
Bottleneck	128	256	ResBlk+SA $\times 1$	–	16×16
dec-2	256	128	ResBlk+SA $\times 2$	$\uparrow 2$	32×32
dec-1	256	64	ResBlk+SA $\times 1$	–	32×32
Heads	64	see text	Linear	–	–

Table 4: UNet architecture of the single-step student. Symbols as in Table 3.

A.3 Constrained SAC for online sensor scheduling: Implementation recipe

The constrained-SAC (C-SAC) controller is trained entirely in simulation, frozen, and then executed on the ASV without modification.

State preprocessing. At control step t the diffusion planner supplies the $64\times64\times5$ belief raster B_t and the scalar CVaR proxy $u_t = u_t^{\text{CVaR}}$. We also compute the Euclidean distance d_t between the particle-filter mean and the goal, and retain the previous binary sensor mask $a_{t-1} \in \{0,1\}^N$ (here $N = 5$). The spatial tensor passes through the same shared CNN encoder for actor and critics:

$$\begin{aligned} &\text{Conv}(5{\to}16,\ 3\times3,\ \text{stride}=2)\ \to\ \text{SiLU} \\ \to\ &\text{Conv}(16{\to}32,\ 3\times3,\ \text{stride}=2)\ \to\ \text{SiLU} \\ \to\ &\text{Conv}(32{\to}64,\ 3\times3,\ \text{stride}=2)\ \to\ \text{SiLU}\ \to\ \text{GlobalAvgPool} \end{aligned}$$

yielding a 64-dimensional feature vector ϕ_t. The numerical features are concatenated to form

$$z_t \ = \ \big[\ \phi_t \parallel u_t \parallel d_t \parallel a_{t-1}\big] \ \in \mathbb{R}^{64+1+1+5} = \mathbb{R}^{71}.$$

Normalisation: u_t is divided by $\eta_{\max}$ (so its nominal range is $[0,1]$), the distance d_t is divided by the world-radius (100 m in simulation), and the mask a_{t-1} is left as $\{0,1\}$ bits.

Actor network. The policy $\pi_\theta(a_t \mid s_t)$ is parameterised by a two-layer MLP:

$$\text{Linear}(71{\to}128)\ \to\ \text{SiLU}\ \to\ \text{Linear}(128{\to}128)\ \to\ \text{SiLU}\ \to\ \text{Linear}(128{\to}N).$$

The final layer outputs *logits*. During exploration each sensor's action is sampled as $\tilde{a}_{t,n} \sim \text{Bernoulli}(\sigma(\text{logit}_n))$; during evaluation we hard-threshold at 0.5. The IMU line is forced to 1 by appending a $+\infty$ bias to its logit, matching the always-on requirement in hardware.

Critic networks. Two independent Q-functions Q_{ϕ^1}, Q_{ϕ^2} share the CNN encoder but have separate MLP heads identical in width to the actor (input 71 + 5 bits of proposed action). A single-layer value network V_ψ of width 128 supplies the baseline for automatic entropy tuning.

Objective and constraint handling. Let $r_t = -c^\top a_t$ (sensor-power cost, coefficients c taken from factory current draw) and $g_t = \mathbf{1}[u_t > \eta_{\max}]$. We optimise the unconstrained Lagrangian

$$\mathcal{L}_t(\theta,\phi,\lambda) \ = \ \mathbb{E}_{(s_t,a_t)}\Big[r_t \ + \ \lambda\, g_t \ + \ \alpha\,\big(-\log\pi_\theta(a_t \mid s_t)\big)\Big],$$

with dual variable $\lambda \geq 0$. Entropy weight α is tuned automatically towards a target entropy $-N\log 0.5$. The dual update is a simple projected gradient ascent:

$$\lambda \ \leftarrow \ \big[\ \lambda + \beta_\lambda\,\big(\mathbb{E}[g_t] - \epsilon\big)\big]_+, \quad \beta_\lambda = 0.05.$$

We set $\epsilon = 0.02$ (i.e. $\leq$2 % long-run CVaR violations) and $\eta_{\max} = 2$ m.

Optimisers and buffers. Actor, critics and value network use Adam with learning rate 3×10^{-4} and $(\beta_1,\beta_2) = (0.9, 0.999)$. No weight decay. A replay buffer of one million transitions is pre-allocated; we collect 16 simulation environments in parallel, step each for five horizon steps, then perform 80 gradient updates (batch 256). Discount factor $\gamma = 0.99$, target-network Polyak $\tau = 0.005$. Training for 2.0M environment steps ($\approx$ 20 CPU-hours on an 8-core desktop) suffices for convergence; an early-stop rule halts once the seven-day rolling average of risk violations drops below ϵ.

Domain randomisation. During training we randomise wind drag ($\pm30\%$), ambient light (0–60 kLux), sensor dropout intervals (exponential, mean 45s) and per-sensor white noise (scaled to twice the empirical lake variance). This sweep was essential for zero-shot transfer to night-time hardware runs.

A.4 Ablation study on the risk metric

To study the effect of different risk metrics, we ran focused runs on Unity simulator (simulator details in App. §.3)–all network weights were frozen, and only the scalar sent to the SAC was varied.

	CVaR-90	CVaR-95	CVaR-99
Goal-reach ↑	96.81 %	98.14 %	97.69 %
Risk-violations ↓	1.61 %	0.57 %	0.43 %
Sensor energy vs. AON ↓	41.07 %	44.01 %	48.11 %

All three statistics keep success above 95% and violations below the 2% CMDP target, confirming that B-COD is robust to the exact tail metric. Qualitatively, we noticed that CVaR-99 makes SAC react to every incipient drift spike, switching sensors on sooner and off later; this trims violations but increases energy consumption. CVaR-90 increases budget breaches, because brief, locally harmless yaw noise is being deemed safe. CVaR-95 sat on the Pareto knee-minimal energy with sub-1% risk breaches.

B Additional experimental details

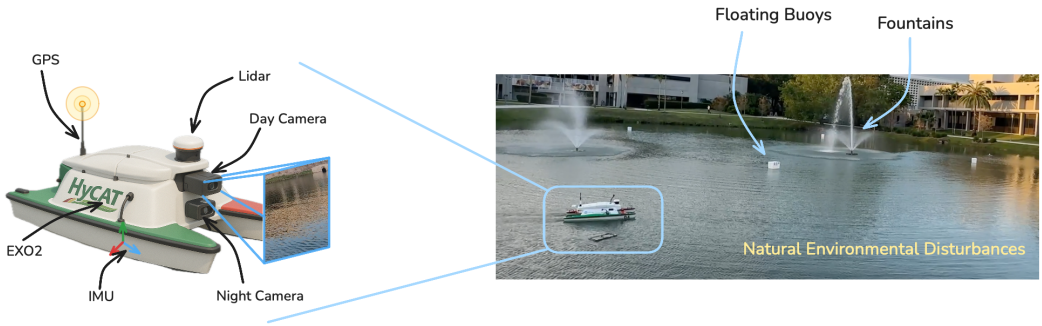

Figure 8: SeaRobotics Surveyor ASV in the operating environment with a differential-thrust propulsion module and a heterogeneous sensor suite: a multi-beam LiDAR, day and night cameras, RTK-GPS, MEMS IMU, and an EXO2 sonde

B.1 Sensor suite and power model

The autonomous surface vehicle carries six modalities and span more than two orders of magnitude in power demand. Table 5 lists their static characteristics. Power figures are peak electrical draw measured with a laboratory power meter at the nominal 24 V bus voltage; calibration noises enter the particle filter and the InfoGain baseline unaltered. The inertial unit is considered always-on in every experiment because its 0.1 W draw is dwarfed by the thrust module. Power consumption in the results section of the paper refers only to the *sensor* column.

Sensor	Update rate	Range	1σ sensor noise	Power (W)
Spinning LiDAR, 32-line	10 Hz	120 m	3 cm	16.0
Global-shutter RGB camera	20 Hz	80 m (day)	1 pixel ($\approx$ 2 cm)	3.0
NIR-augmented mono camera	20 Hz	40 m (night)	1 pixel	5.0
Exo-conductivity/temperature sonde	2 Hz	spot	- (depth only)	1.2
RTK-capable GNSS receiver	5 Hz	global	1.5 cm (RTK fix)	0.2
6-axis MEMS IMU (always on)	200 Hz	n/a	0.02 rad/s (gyro)	0.1

Table 5: Sensor payload used in all trials. Ranges are conservative values in clear daylight; night performance degrades as described in the text.

B.2 Hyper-parameters of B-COD and the C-SAC scheduler

All diffusion-model constants are given in Apps.,A.2.1 and A.2.2. For completeness we repeat the single value that materially affects on-board behaviour: the distance-to-goal scalar appended to the SAC state is divided by the fixed world radius of 100 m. The risk budget is $\eta_{\max} = 2$ m and the acceptable violation rate is $\epsilon = 0.02$. No other quantity is tuned per run.

B.3 Baseline implementations

Every baseline re-uses the same EKF, particle filter, low-level thrust limits; only the sensor-selection logic and, where relevant, the high-level planner differ.

Always-ON. All six modalities powered from take-off to shutdown; no parameters.

Greedy-OFF. A luxmeter attached to the mast provides the measured ambient light. If lux < 10 the day camera is disabled; if lux ≥ 10 the night camera is disabled. LiDAR is powered only if any return in the previous sweep lay within 15 m of the boat's estimated pose. The sonde is enabled whenever the chlorophyll-*a* estimate exceeds $6, \mu$g/L, an empirically determined threshold separating routine from interesting water in the survey lake. The five constants (10, 10, 15 m, $6, \mu$g/L, GPS always on) were grid-searched on a held-out 30-lap sequence.

InfoGain-Greedy. At each 1 s decision instant the analytic observation model of the EKF is queried under the six possible single-sensor activations. The sensor that maximises the expected reduction in log-determinant of the 500-particle spatial covariance is selected; all others are switched off until the next period.

Random-K. The scheduler samples a new mask every control step. Variant R1 draws exactly one sensor (uniform over the five switchable modalities); variant R2 draws exactly two. The IMU remains on.

σ-Mean and Sample-Spread. The C-SAC actor–critic architecture is fixed. In σ-Mean the scalar risk input is $\frac{1}{H}\sum_k \sqrt{e^{\hat{\sigma}_k}}$. In Sample-Spread the diffusion planner returns 20 trajectory samples and the input risk is the 95th percentile of the waypoint-wise Mahalanobis spread. No other change is made.

No-Belief Raster. The planner receives a single additional channel containing $\Delta x, \Delta y, \Delta\psi$; the raster encoder is unmodified but observes zeros in place of the missing belief statistics.

Pure RL. A constrained SAC identical to the student's C-SAC head controls both motion primitives (discrete 16-heading lattice, step 1 m) and the sensor mask; state is the same 71-vector. Learning rate 3×10^{-4}, replay one million, target entropy $-|\mathcal{A}| \log 0.5$, dual step size 0.05. Training budget 50 M environment steps ($\approx$ 6 days on 32 vCPUs).

DESPOT-Lite. Online POMDP search with 5 k belief particles, tree depth 6 ($\approx$ 6 s horizon), and rollout policy equal to the Always-ON heuristic. Each internal node expands only $|\mathcal{S}| + 1$ actions (one per singleton sensor subset plus the null set) to keep branching manageable. The planner terminates early if the anytime bound width falls below 10% of the current best value or the 500 ms wall-clock budget elapses.

C Open maritime navigation dataset

To catalyse follow-up work on uncertainty-aware, resource-bounded marine autonomy we publicly release the 50k marine navigation corpus: 50,123 short-horizon navigation snippets whose every frame is synchronised across pose belief, rasterised map slice, raw sensor packets and ground-truth

trajectory. The dataset is hosted under a permissive CC-BY-4.0 licence at https://github.com/bcod-diffusion/dataset and mirrored in the project repository. Currently a subset of this dataset has been released publicly.

C.1 Collection pipeline

Field logs. Twelve day-time and eight night-time sorties were conducted on freshwater lake. The SeaRobotics Surveyor ASV collected:

(i) 32-beam spinning LiDAR point clouds (10 Hz, ROS/PCD);

(ii) RGB images (20 Hz, PNG);

(iii) Near-IR images under 850 nm active illumination (20 Hz, PNG);

(iv) RTK-GNSS fixes (5 Hz, NMEA);

(v) Six-axis IMU messages (200 Hz, ROS/Imu);

(vi) Water-quality probe samples (2 Hz, CSV).

Figure 9: Screenshot of our Unity simulator.

All topics share a chronologically consistent ROS /clock. Each log is accompanied by recordings of wind and irradiance for domain-randomisation replay.

Real→sim transfer. Logs are imported into an in-house Unity 2022.3 + ROS 2 simulator that reconstructs the shoreline mesh, static obstacles, bathymetry and approximate above-surface lighting. Dynamic objects are re-instantiated with ground-truth trajectories. The simulator then re-flies the ASV pose trace while rendering all sensors at original frame rates. Crucially, we also replay the EKF/particle filter in lock-step, producing a time-aligned sequence of belief particle clouds; these clouds are the source of the raster B_t and the CVaR proxy in each snippet.

C.2 Snippet definition

A snippet is a contiguous 1s window centred at time t_0 and exported as

(i) `B_t.npz` – 64 × 64 × 5 float16 raster at t_0;

(ii) `map_slice.png` – 64 × 64 × 3 semantic image;

(iii) `goal_mask.png` – 64 × 64 binary;

(iv) `sensor_flag.npy` – 5-bit uint8 vector active at t_0;

(v) `traj.npy` – ground-truth $(\Delta x, \Delta y, \Delta\psi)_{k=1..8}$;

(vi) `sigma.npy` – waypoint log-variances $\hat{\sigma}_{1..8}$;

(vii) `meta.json` – latitude/longitude, weather, clip ID.

Overlapping windows are sampled at 2 Hz, yielding 50123 snippets.

C.3 Statistics

(i) **Modalities.** 100 % contain LiDAR and IMU; day camera appears in 72 %, night camera in 28 %, GNSS in 64 %, sonde in 18 %.

(ii) **Belief spread.** Median planar 1σ = 0.38 m; 95th percentile = 2.1 m.

(iii) **Lighting.** Illumination spans 0.2–55 kLux; clips are evenly stratified into five bins for training/validation.

(iv) **Obstacles.** Each snippet is annotated with the minimum range to shoreline and to floating hazards; mean 14.2 m, min 0.8 m.