

Appendix

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

Anonymous Author(s)

Affiliation

Address

email

Project resources

Website <https://bcod-diffusion.github.io>

Code repository <https://github.com/bcod-diffusion/bcod>

Dataset (anonymised subset) <https://github.com/bcod-diffusion/dataset>

Appendix: Implementation and Evaluation Details

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 hyper-parameter 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.

1 Implementation Recipes

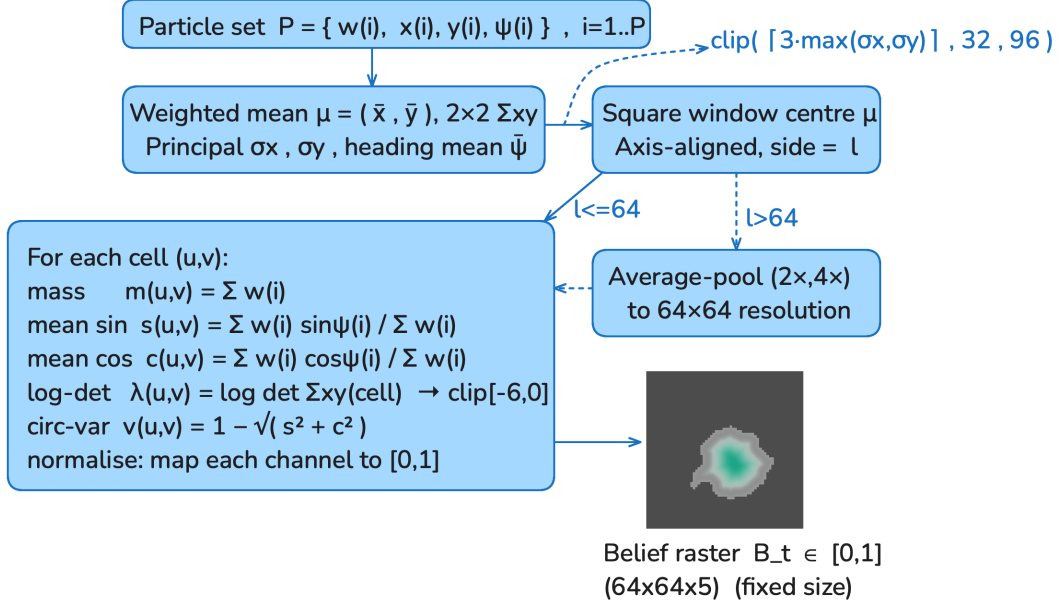
1.1 Belief Rasterization

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

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

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 $l \leq 96$. If $l \leq 64$ 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



Variant (64×64 raster)	Channels C	Goal-reach \uparrow	Collisions \downarrow	CVaR viol. \downarrow	Sensor energy † \downarrow	Runtime (ms) ‡ \downarrow	Peak GPU RAM (MB) * \downarrow
Baseline (5 stats)	5	97.9	0.9	0.5	42	10.4	284
— log det Σ	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 1: 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. 2.1). ‡ Mean forward-pass latency of the diffusion model plus rasterisation; SAC adds a ≈ 4.1 ms. * Peak memory measured with `torch.cuda.max_memory_allocated` after one control iteration.

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 (+ Σ) 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.

1.2 Belief Conditioned One-Step Diffusion

1.2.1 Diffusion Teacher: full implementation details

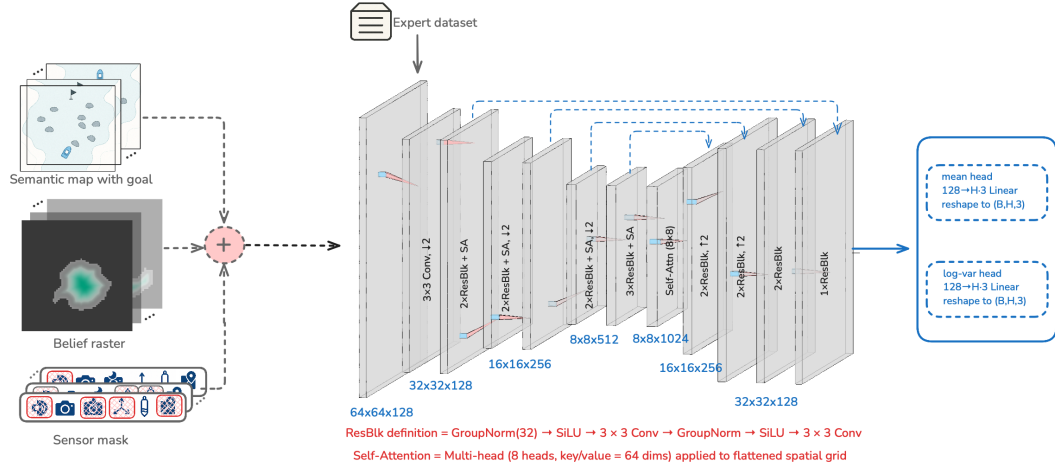


Figure 2: Model architecture: Diffusion Teacher

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. 2 summarizes the model architecture.

Input encoding and conditioning pathway. Each training example consists of a $64 \times 64 \times C$ 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 $64 \times 64 \times 9$ stack into a 128-channel feature map; this layer appears as the stem conv row in Table 2. 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 $1 \times 1 \times 32$, repeating it to $64 \times 64 \times 32$, 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 $t \in 1, \dots, T$, 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 $64 \times 64 \times 128$ 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 2 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 \uparrow / \downarrow	Output size
<i>Encoder</i>					
Stem conv	128	128	–	$\downarrow 2$	32×32
enc-1	128	128	ResBlk+SA $\times 2$	–	32×32
enc-2	128	256	ResBlk+SA $\times 2$	$\downarrow 2$	16×16
enc-3	256	512	ResBlk+SA $\times 2$	$\downarrow 2$	8×8
enc-4	512	512	ResBlk+SA $\times 3$	–	8×8
<i>Bottleneck</i>					
bottleneck	512	1024	SA (8×8)	–	8×8
<i>Decoder</i>					
dec-4	1024	512	ResBlk+SA $\times 2$	$\uparrow 2$	16×16
dec-3	1024	256	ResBlk+SA $\times 2$	$\uparrow 2$	32×32
dec-2	512	128	ResBlk+SA $\times 2$	–	32×32
dec-1	256	128	ResBlk+SA $\times 1$	–	32×32
<i>Heads</i>					
mean head	128	$H \times 3$	Linear	–	–
log-var head	128	H	Linear	–	–

Table 2: UNet architecture of the diffusion teacher. All convolutions use reflection padding; GN = Group-Norm(32); SA = multi-head self-attention, 8 heads; SiLU activation everywhere.

1.2.2 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 1.2.1) 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 $64 \rightarrow 128 \rightarrow 256 \rightarrow 128 \rightarrow 64$. 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.

Table 3: UNet architecture of the single-step student. Symbols as in Table 2.

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	–	–

1.3 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 $64 \times 64 \times 5$ 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:

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

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

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

Normalisation: u_t is divided by η_{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 $\pi_\theta(a_t | s_t)$ is parameterised by a two-layer MLP:

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

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

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

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:

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

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 (≈ 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 ($\pm 30\%$), 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.

2 Additional experimental details

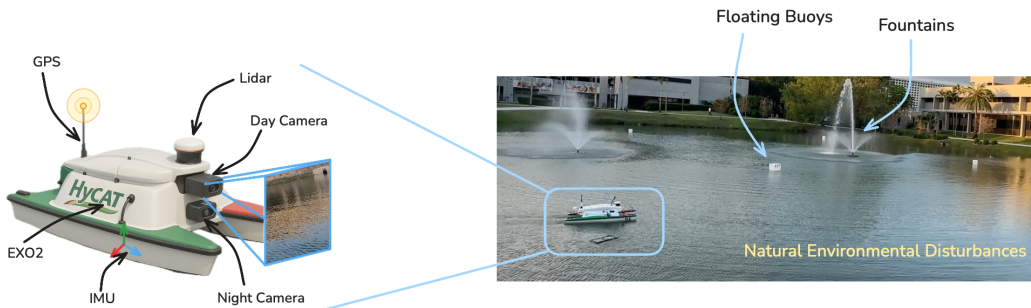


Figure 3: SeaRobotics Surveyor ASV in the operating environment with a differential-thrust propulsion module and a heterogeneous sensor suite: a multi-beam LiDAR, day and night cameras, RTK-GPS, MEMS IMU, and an EXO2 sonde

2.1 Sensor suite and power model

The autonomous surface vehicle carries six modalities and span more than two orders of magnitude in power demand. Table 4 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 (≈ 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 4: Sensor payload used in all trials. Ranges are conservative values in clear daylight; night performance degrades as described in the text.

2.2 Hyper-parameters of B-COD and the C-SAC scheduler

All diffusion-model constants are given in Apps., 1.2.1 and 1.2.2. 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.

2.3 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 $\text{lux} < 10$ the day camera is disabled; if $\text{lux} \geq 10$ 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, $\mu\text{g/L}$, an empirically determined threshold separating routine from interesting water in the survey lake. The five constants (10, 10, 15 m, 6, $\mu\text{g/L}$, 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.

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

237 **Pure RL.** A constrained SAC identical to the student’s C-SAC head controls both motion primitives
 238 (discrete 16-heading lattice, step 1 m) and the sensor mask; state is the same 71-vector. Learning rate
 239 3×10^{-4} , replay one million, target entropy $-|\mathcal{A}| \log 0.5$, dual step size 0.05. Training budget 50 M
 240 environment steps (≈ 6 days on 32 vCPUs).

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

246 3 Open maritime navigation dataset

247 To catalyse follow-up work on uncertainty-aware, resource-bounded marine autonomy we publicly
 248 release the 50k marine navigation corpus: 50,123 short-horizon navigation snippets whose every
 249 frame is synchronised across pose belief, rasterised map slice, raw sensor packets and ground-truth
 250 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
 251 repository. Currently a subset of this dataset has been anonymized and released publicly.
 252

253 3.1 Collection pipeline

254 **Field logs.** Twelve day-time and eight night-time sor-
 255 ties were conducted on freshwater lake. The SeaRobotics
 256 Surveyor ASV collected:

- 257 • 32-beam spinning LiDAR point clouds (10 Hz,
258 ROS/PCD);
- 259 • RGB images (20 Hz, PNG);
- 260 • Near-IR images under 850 nm active illumination
261 (20 Hz, PNG);
- 262 • RTK-GNSS fixes (5 Hz, NMEA);
- 263 • Six-axis IMU messages (200 Hz, ROS/Imu);
- 264 • Water-quality probe samples (2 Hz, CSV).



Figure 4: Screenshot of our Unity simulator.

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

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

273 3.2 Snippet definition

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

- 275 • `B.t.npz` – $64 \times 64 \times 5$ float16 raster at t_0 ;

- 276 • `map_slice.png` – $64 \times 64 \times 3$ semantic image;
- 277 • `goal_mask.png` – 64×64 binary;
- 278 • `sensor_flag.npy` – 5-bit uint8 vector active at t_0 ;
- 279 • `traj.npy` – ground-truth $(\Delta x, \Delta y, \Delta \psi)_{k=1..8}$;
- 280 • `sigma.npy` – waypoint log-variances $\hat{\sigma}_{1..8}$;
- 281 • `meta.json` – latitude/longitude, weather, clip ID.

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

283 3.3 Statistics

- 284 • **Modalities.** 100 % contain LiDAR and IMU; day camera appears in 72 %, night camera in
285 28 %, GNSS in 64 %, sonde in 18 %.
- 286 • **Belief spread.** Median planar $1\sigma = 0.38$ m; 95th percentile = 2.1 m.
- 287 • **Lighting.** Illumination spans 0.2–55 kLux; clips are evenly stratified into five bins for
288 training/validation.
- 289 • **Obstacles.** Each snippet is annotated with the minimum range to shoreline and to floating
290 hazards; mean 14.2 m, min 0.8 m.

291 3.4 Recommended splits

292 The download will include `train.txt` (40 k IDs), `val.txt` (5 k) and `test.txt` (5 k). Splits are
293 stratified by lake, lighting bin and obstacle density; they guarantee that no test snippet shares a parent
294 log with any train snippet.

295 3.5 Baseline loaders

296 We provide PyTorch and JAX dataloaders that:

- 297 1. lazily mmap `.npz/.npy` to minimise RAM;
- 298 2. convert rasters to FP32 tensors;
- 299 3. optionally augment with random yaw and mirroring (flags stored in `meta.json`).