Mathematical Formulation of the Multi-Fidelity Air–Ground Coverage Simulator

ARSControl

Abstract

This document is the typeset mathematical reference for the coupled multi-fidelity coverage simulator. It specifies the implemented scalar fields, sensor model, central autoregressive Gaussian process, controller densities, aerial and ground controllers, dynamics, event ordering, and evaluation metrics.

1 Scope and notation

The equations are marked conceptually as follows:

- **Model** describes the mathematical model or control objective.
- **Implementation** gives the discrete equation evaluated by the current code.
- **Compatibility path** describes retained legacy behavior that is not active when the central multi-fidelity estimator supplies the controllers.

The main symbols are:

Symbol	Meaning
$\mathbf{q} = (x, y)^\top$	A point in the two-dimensional workspace Ω
$\mathbf{p}_i$	Position of robot i
θ_i	Heading of robot i
f_L, f_H	Latent LOW- and HIGH-fidelity scalar fields
δ	Discrepancy between the scaled LOW field and the HIGH field
ρ	Fixed autoregressive scaling coefficient
μ_H, σ_H^2	Posterior mean and marginal variance of f_H
ϕ	Nonnegative, normalized controller importance density
w_j	Numerical quadrature weight associated with query point $\mathbf{q}_j$
V_{ij}	Indicator that query point j belongs to robot i's Voronoi cell
Δt	Simulation or controller time step
$\mathrm{sat}_{[a,b]}(z)$	Clipping of z to the interval $[a, b]$

Unless stated otherwise, the current multi-fidelity path uses one central posterior for both teams:

- aerial robots use a normalized combination of posterior interest and uncertainty;

- ground robots use posterior interest only;
- ground positions affect aerial motion only indirectly, through the locations at which ground robots acquire HIGH-fidelity samples;
- ground Voronoi cells are built from ground-robot positions only.

2 Simulator ground truth

2.1 HIGH-fidelity field

The simulator constructs a smooth HIGH-fidelity field from a Gaussian mixture. For K components,

$$g(\mathbf{q}) = \sum_{k=1}^{K} \pi_k \frac{\exp\left[-\frac{1}{2}(\mathbf{q}-\boldsymbol{\mu}_k)^\top \boldsymbol{\Sigma}_k^{-1}(\mathbf{q}-\boldsymbol{\mu}_k)\right]}{2\pi\sqrt{\det(\boldsymbol{\Sigma}_k)}}, \sum_{k=1}^{K} \pi_k = 1 \tag{1}$$

On the raster, this field is min-max scaled and obstacles are masked:

$$f_H(\mathbf{q}_j) = \mathbf{1}_{\text{free}}(\mathbf{q}_j) \frac{g(\mathbf{q}_j) - g_{\min}}{} g_{\max} - g_{\min} + \varepsilon \tag{2}$$

This is the simulator's hidden truth. The estimator receives noisy samples, not the mixture parameters or the complete raster.

2.2 Simulator LOW field and discrepancy

The LOW truth is a normalized Gaussian smoothing of the masked HIGH truth. If G_s is a Gaussian filter with standard deviation s cells and M is the free-space mask, the implemented normalized convolution is

$$f_L = \frac{G_s * (M f_H)}{G_s * M} \tag{3}$$

where division is pointwise wherever the denominator is nonzero, and the result is zero outside the mask. The simulator then defines

$$\delta(\mathbf{q}) = f_H(\mathbf{q}) - \rho f_L(\mathbf{q}) \tag{4}$$

so the autoregressive identity is exact for the simulated fields:

$$f_H(\mathbf{q}) = \rho f_L(\mathbf{q}) + \delta(\mathbf{q}) \tag{5}$$

The simulator uses this identity to generate a coherent test problem. The estimator does not observe the simulator's δ field directly.

3 Sensor model

An observation from fidelity $s \in \{L, H\}$ at position $\mathbf{q}_n$ is

$$y_n = f_s(\mathbf{q}_n) + \epsilon_n, \qquad \epsilon_n \sim \mathcal{N}(0, \tau_{s,n}^2) \tag{6}$$

The configured sensor noise values are variances, so the noise standard deviation used to draw a sample is $\sqrt{\tau_{s,n}^2}$. Field values are read from the nearest raster cell after the continuous sample position is generated.

3.1 Uniform-sector sampling

For a robot at $\mathbf{p}$ with heading θ, sensor range R, and total field-of-view angle Ψ, the sensor's only sampling law draws

$$\alpha \sim \mathcal{U}\left(\theta - \frac{\Psi}{2}, \theta + \frac{\Psi}{2}\right), \qquad U \sim \mathcal{U}(0,1), \qquad r = R\sqrt{U} \tag{7}$$

and returns

$$\mathbf{q} = \mathbf{p} + r \begin{bmatrix} \cos\alpha \\ \sin\alpha \end{bmatrix} \tag{8}$$

The square root makes sample locations uniform with respect to area inside the sector because the polar area element is $r\,dr\,d\alpha$. Out-of-bounds draws are resampled up to the configured attempt limit. There is no sampling-mode parameter or alternative fixed-ray branch in the production sensor.

4 Autoregressive multi-fidelity Gaussian process

4.1 Prior

The central estimator assumes independent zero-mean Gaussian processes

$$f_L \sim \mathcal{GP}(0, k_L), \qquad \delta \sim \mathcal{GP}(0, k_\delta), \qquad f_L \perp \delta \tag{9}$$

with

$$f_H(\mathbf{q}) = \rho f_L(\mathbf{q}) + \delta(\mathbf{q}) \tag{10}$$

Both covariance functions are isotropic squared-exponential kernels:

$$k_a(\mathbf{q}, \mathbf{q}') = \sigma_a^2 \exp\left(-\frac{\|\mathbf{q} - \mathbf{q}'\|_2^2}{2\ell_a^2}\right), \qquad a \in \{L, \delta\} \tag{11}$$

Here ℓ_a is a length scale and σ_a^2 is the configured kernel variance.

4.2 What is known and what is learned

Quantity	Current treatment
Kernel family	Known: isotropic squared-exponential
Autoregressive structure	Known: $f_H = \rho f_L + \delta$
ρ	Fixed from configuration
Sensor noise variances	Fixed from configuration and stored per observation
LOW and HIGH sample positions and values	Observed
$\ell_L, \sigma_L^2, \ell_\delta, \sigma_\delta^2$	Optionally fitted by marginal likelihood
Latent field values away from samples	Inferred as a posterior distribution
$\mu_H(\mathbf{q}), \sigma_H^2(\mathbf{q})$	Recomputed on the query grid after a successful update
Simulator truth, GMM parameters, and true discrepancy	Unknown to the estimator

Fitting hyperparameters is empirical Bayes: it produces one bounded maximum-marginal-likelihood point estimate. It does not maintain a posterior distribution over the hyperparameters.

4.3 Joint observation covariance

Let

$$\mathbf{y} = \begin{bmatrix} \mathbf{y}_L \\ \mathbf{y}_H \end{bmatrix} \tag{12}$$

with LOW inputs X_L, HIGH inputs X_H, and diagonal observation-noise matrices D_L and D_H. In LOW-then-HIGH order, the training covariance is

$$K_y = \begin{bmatrix} K_L(X_L, X_L) + D_L & \\ \rho K_L(X_L, X_H) & \\ \rho K_L(X_H, X_L) & \\ \rho^2 K_L(X_H, X_H) + K_\delta(X_H, X_H) + D_H & \end{bmatrix} \tag{13}$$

The implementation adds adaptive diagonal jitter:

$$\widetilde{K}_y = K_y + \eta I \tag{14}$$

Starting from the configured η, failed Cholesky attempts multiply it by the configured jitter multiplier. No explicit matrix inverse is formed.

4.4 Cholesky solve

For

$$LL^\top = \widetilde{K}_y \tag{15}$$

the implementation solves

$$L\mathbf{z} = \mathbf{y}, L^\top \boldsymbol{\alpha} = \mathbf{z} \tag{16}$$

Thus

$$\boldsymbol{\alpha} = \widetilde{K}_y^{-1}\mathbf{y} \tag{17}$$

conceptually, while the numerical computation uses triangular solves.

4.5 HIGH-fidelity posterior

For query points X_*, the covariance between training observations and the latent HIGH field is

$$K_{y*} = \begin{bmatrix} \rho K_L(X_L, X_*) \\ \rho^2 K_L(X_H, X_*) + K_\delta(X_H, X_*) \end{bmatrix} \tag{18}$$

The prior HIGH covariance is

$$K_{**} = \rho^2 K_L(X_*, X_*) + K_\delta(X_*, X_*) \tag{19}$$

The posterior mean is

$$\boldsymbol{\mu}_H = K_{y*}^\top \boldsymbol{\alpha} \tag{20}$$

If V solves

$$LV = K_{y*} \tag{21}$$

then the posterior marginal variance at query point j is

$$\sigma_{H,j}^2 = [K_{**}]_{jj} - \sum_r V_{rj}^2 \tag{22}$$

Only tiny negative values attributable to floating-point roundoff are clipped to zero. Materially negative variances cause the candidate update to fail.

4.6 Hyperparameter fitting

The four positive kernel parameters are optimized in log space:

$$\boldsymbol{\xi} = \log \begin{bmatrix} \ell_L \\ \sigma_L^2 \\ \ell_\delta \\ \sigma_\delta^2 \end{bmatrix}^\top \tag{23}$$

The bounded L-BFGS-B objective is the negative log marginal likelihood

$$\mathcal{L}(\boldsymbol{\xi}) = \frac{1}{2}\mathbf{y}^\top \boldsymbol{\alpha} + \sum_{r=1}^{N} \log L_{rr} + \frac{N}{2}\log(2\pi) \tag{24}$$

Fitting occurs only when optimization is enabled, the retained sample count reaches its configured minimum, and the current posterior version satisfies the configured update interval. The value of ρ and both sensor-noise models remain fixed.

4.7 Query-grid quadrature

For each axis, the estimator uses equally spaced query coordinates and trapezoidal weights. With spacing h_x,

$$w_j^x = \begin{cases} h_x/2, & j \text{ is an endpoint,} \\ h_x, & \text{otherwise,} \end{cases} \tag{25}$$

and similarly for w_k^y. The two-dimensional weight is

$$w_{jk} = w_j^x w_k^y \tag{26}$$

These weights approximate integrals and are the w_j used in density normalization and Lloyd centroid calculations.

5 From the GP posterior to controller densities

5.1 Positive posterior interest

The shared posterior interest field is the positive part of the HIGH posterior mean:

$$\mu_H^+(\mathbf{q}_j) = \max(\mu_H(\mathbf{q}_j), 0) \tag{27}$$

For a free-space mask $M_j \in \{0, 1\}$ and quadrature weights w_j, the published density is

$$\phi_j = \frac{M_j \mu_H^+(\mathbf{q}_j)}{\sum_k M_k \mu_H^+(\mathbf{q}_k) w_k} \tag{28}$$

Therefore

$$\phi_j \geq 0, \sum_j \phi_j w_j = 1 \tag{29}$$

The clipping preserves all positive posterior contrast. It does not exponentiate the field, shift negative values upward, or retain the removed softplus transformation. If every posterior mean value is nonpositive, normalization has zero mass and the estimator transaction preserves the last valid snapshot.

5.2 Aerial interest-plus-uncertainty target

Let

$$\bar{\sigma}_{H,j} = \begin{cases} \sigma_{H,j} / \max_k \sigma_{H,k}, & \max_k \sigma_{H,k} > 0, \\ 0, & \text{otherwise.} \end{cases} \tag{30}$$

The unnormalized aerial importance is

$$\widetilde{\phi}_j^A = \lambda_I \phi_j + \lambda_U \bar{\sigma}_{H,j} \tag{31}$$

where $\lambda_I \geq 0$ is the interest weight and $\lambda_U \geq 0$ is the uncertainty weight. After masking and quadrature normalization,

$$\phi_j^A = \frac{M_j \widetilde{\phi}_j^A}{\sum_k M_k \widetilde{\phi}_k^A w_k} \tag{32}$$

This density is bilinearly resampled onto the aerial HEDAC raster and normalized again there. All aerial robots see the same target; HEDAC coverage history and their different positions make their individual motions differ.

5.3 Ground importance

The ground controllers use

$$\phi_j^G = \phi_j \tag{33}$$

Posterior variance is intentionally absent from the current ground objective. If the controller grid differs from the posterior grid, the density is bilinearly interpolated and quadrature-normalized again.

6 HEDAC aerial coverage

6.1 Coverage footprint

For an offset $\mathbf{r}$ from an aerial robot, the discrete coverage footprint is

$$\kappa(\mathbf{r}) = \exp\left(- \frac{\|\mathbf{r}\|_2^2}{R_a} \right) \tag{34}$$

because the implementation uses the RBF shape parameter $1/R_a$. The square footprint is truncated using the configured minimum kernel value $\kappa_{\min}$. With workspace dimension $d = 2$, its per-axis half-width is

$$h_\kappa = \left\lceil \sqrt{\frac{-R_a \log \kappa_{\min}}{d}} \right\rceil \tag{35}$$

At HEDAC step n, the cumulative raster coverage is updated by adding one footprint for every aerial robot:

$$C_j^{n+1} = C_j^n + \sum_i \kappa(\mathbf{q}_j - \mathbf{p}_i^n) \tag{36}$$

with each footprint cropped at map boundaries.

6.2 Coverage deficit and source

HEDAC uses sum normalization on its raster:

$$\widehat{C}_j = \frac{M_j C_j}{\sum_k M_k C_k + \varepsilon} \tag{37}$$

The current target ϕ^A is also sum-normalized on this raster. The positive coverage deficit and source are

$$d_j = \max(\phi_j^A - \widehat{C}_j, 0), \widetilde{S}_j = d_j^2 \tag{38}$$

$$S_j = A_\Omega \frac{M_j \widetilde{S}_j}{\sum_k M_k \widetilde{S}_k + \varepsilon} \tag{39}$$

where A_Ω is the free map area.

6.3 Heat model

The intended heat-equation form is

$$\frac{\partial T}{\partial t} = \alpha \nabla^2 T + sS - \beta T - \gamma L_c \tag{40}$$

where α is diffusion, s is source strength, β is global cooling, and L_c is local cooling.

The current non-optimized implementation evaluates the following five-point interior stencil exactly:

$$\mathcal{D}_\alpha[T]_{r,c} = \alpha \left(T_{r+1,c} + T_{r-1,c} + T_{r,c+1} + T_{r,c-1}\right) - 4T_{r,c} \tag{41}$$

Then

$$T_{r,c}^{n+1} = T_{r,c}^n + \Delta t_H \left[\frac{\mathcal{D}_\alpha[T^n]_{r,c}}{\Delta x^2} + sS_{r,c} - \frac{\beta}{A_\Omega} T_{r,c}^n - \frac{\gamma}{A_\Omega}(L_c)_{r,c}\right] \tag{42}$$

This discrete stencil is recorded exactly as implemented: the central $-4T_{r,c}$ term is not multiplied by α. It therefore differs from the usual discretization $\alpha(T_{\text{neighbors}} - 4T_{r,c})/\Delta x^2$ when $\alpha \neq 1$.

The active local-cooling raster is reset to zero every step and footprint accumulation into it is disabled, so the local-cooling contribution is currently zero. The non-optimized update modifies

interior cells only, leaves outer boundary cells fixed, and does not explicitly skip obstacle cells during diffusion.

The explicit heat step is limited by

$$\Delta t_H = \min\left(\Delta t, \frac{c_{\mathrm{CFL}}\Delta x^2}{4\alpha}\right) \tag{43}$$

for $\alpha > 0$, and $\Delta t_H = \Delta t$ for $\alpha = 0$.

6.4 Heat-gradient direction

The raster gradients are computed by finite differences. They are first globally scaled by

$$g_{\mathrm{rms}} = \sqrt{\mathrm{mean}(T_x^2) + \mathrm{mean}(T_y^2)} \tag{44}$$

when this value is nonzero. The scaled gradient is bilinearly interpolated at the robot position. Close to the outer boundary, the relevant component is replaced by a fixed inward value of magnitude 10. Obstacle-wall avoidance is not active in this gradient function.

Finally, only the direction is retained:

$$\mathbf{d}_i = \begin{cases} \nabla T(\mathbf{p}_i)/\|\nabla T(\mathbf{p}_i)\|_2, & \|\nabla T(\mathbf{p}_i)\|_2 > 0, \\ \mathbf{0}, & \text{otherwise.} \end{cases} \tag{45}$$

The target velocity and heading are

$$\mathbf{v}_i^\star = v_{\max}\mathbf{d}_i, \theta_i^\star = \mathrm{atan2}(d_{i,y}, d_{i,x}) \tag{46}$$

7 Robot dynamics

7.1 Aerial Dubins dynamics

The canonical aerial configuration uses fixed-speed Dubins dynamics:

$$\dot{x}_i = v_A\cos\theta_i, \dot{y}_i = v_A\sin\theta_i, \dot{\theta}_i = u_i \tag{47}$$

The coordinated-turn limit derived from maximum bank angle $\varphi_{\max}$ is

$$u_{\max} = \frac{g\tan\varphi_{\max}}{v_A} \tag{48}$$

The HEDAC heading tracker uses

$$e_{\theta,i} = \mathrm{atan2}\left(\sin(\theta_i^\star - \theta_i), \cos(\theta_i^\star - \theta_i)\right) \tag{49}$$

$$u_i = \mathrm{sat}_{[-u_{\max}, u_{\max}]}\left(2e_{\theta,i}\right) \tag{50}$$

The implementation updates heading first and then position:

$$\theta_i^{n+1} = \mathrm{wrap}\left(\theta_i^n + u_i^n \Delta t_A\right) \tag{51}$$

$$\mathbf{p}_i^{n+1} = \mathbf{p}_i^n + v_A \Delta t_A \begin{bmatrix} \cos\theta_i^{n+1} \\ \sin\theta_i^{n+1} \end{bmatrix} \tag{52}$$

Positions are clipped to the map after the controller step.

7.2 Ground unicycle dynamics

Both current ground controllers require or predict the unicycle model

$$\dot{x}_i = v_i \cos\theta_i, \dot{y}_i = v_i \sin\theta_i, \dot{\theta}_i = \omega_i \tag{53}$$

The real ground agent clips (v_i, ω_i) to its configured limits, updates heading first, and then position:

$$\theta_i^{n+1} = \mathrm{wrap}\left(\theta_i^n + \omega_i^n \Delta t_G\right) \tag{54}$$

$$\mathbf{p}_i^{n+1} = \mathbf{p}_i^n + v_i^n \Delta t_G \begin{bmatrix} \cos\theta_i^{n+1} \\ \sin\theta_i^{n+1} \end{bmatrix} \tag{55}$$

The MPC prediction model uses forward Euler with the old heading instead:

$$\begin{bmatrix} x_{h+1} \\ y_{h+1} \\ \theta_{h+1} \end{bmatrix} = \begin{bmatrix} x_h + v_h \cos\theta_h \, \Delta t \\ y_h + v_h \sin\theta_h \, \Delta t \\ \theta_h + \omega_h \Delta t \end{bmatrix} \tag{56}$$

This one-step discretization difference is part of the present implementation.

8 Ground-only Voronoi partition

For ground positions $\{\mathbf{p}_1, \ldots, \mathbf{p}_R\}$, query point $\mathbf{q}_j$ is assigned by

$$i^\star(j) = \underset{i \in \{1,\ldots,R\}}{\arg\min} \, \|\mathbf{q}_j - \mathbf{p}_i\|_2^2 \tag{57}$$

$$V_{ij} = \begin{cases} 1, & i = i^\star(j) \text{ and } \|\mathbf{q}_j - \mathbf{p}_i\|_2 \le R_V, \\ \\ 0, & \text{otherwise.} \end{cases} \tag{58}$$

Only ground positions are passed to this partition. Aerial positions do not compete for ground Voronoi cells. The Lloyd controller chooses a range larger than the map diagonal, so its cells cover the entire query grid. The MPC controller currently uses $R_V = 50$ in map-coordinate units.

9 Lloyd ground controller

9.1 Continuous coverage objective

The classical locational coverage objective is

$$\mathcal{H}(\mathbf{p}_1, \ldots, \mathbf{p}_R) = \sum_{i=1}^{R} \int_{\mathcal{V}_i} \|\mathbf{q} - \mathbf{p}_i\|_2^2 \phi^G(\mathbf{q})\, d\mathbf{q} \tag{59}$$

For fixed Voronoi cells, its minimizer places each generator at the density-weighted centroid of its cell.

9.2 Discrete mass and centroid

The implemented cell mass is

$$m_i = \sum_j V_{ij} \phi_j^G w_j \tag{60}$$

The centroid is

$$\mathbf{c}_i = \frac{\sum_j \mathbf{q}_j V_{ij} \phi_j^G w_j}{m_i} \tag{61}$$

Thus w_j is the area represented by grid point j under trapezoidal quadrature. On a uniform interior grid it is approximately $\Delta x \Delta y$; boundary weights are halved per boundary axis. These weights are what make the vectorized sum approximate the slower nested area-integral calculation.

If m_i is numerically zero, the current robot position is used as the centroid.

9.3 Centroid tracking

Define

$$\mathbf{e}_i = \mathbf{c}_i - \mathbf{p}_i, r_i = \|\mathbf{e}_i\|_2, \theta_i^\star = \operatorname{atan2}(e_{i,y}, e_{i,x}) \tag{62}$$

$$e_{\theta,i} = \operatorname{wrap}(\theta_i^\star - \theta_i) \tag{63}$$

Outside the centroid tolerance, the controls are

$$v_i = \mathrm{sat}_{[-v_{\max}, v_{\max}]} \left(k_p r_i \cos e_{\theta,i} \right) \tag{64}$$

$$\omega_i = \mathrm{sat}_{[-\omega_{\max}, \omega_{\max}]} \left(k_\theta e_{\theta,i} \right) \tag{65}$$

Inside the tolerance, both controls are zero. The factor $\cos e_{\theta,i}$ reduces forward motion when the centroid is lateral and allows bounded reverse motion when it lies behind the robot.

10 MPC ground controller

10.1 Per-robot importance weights

Robot i receives the masked weights

$$W_{ij} = V_{ij} \phi_j^G \tag{66}$$

These weights do not include quadrature factors in the current MPC objective.

10.2 Implemented limited-FOV stage cost

For predicted state $\mathbf{x}_h = (x_h, y_h, \theta_h)^\top$, define

$$r_{hj}^2 = \left\| \mathbf{q}_j - \begin{bmatrix} x_h \\ y_h \end{bmatrix} \right\|_2^2 \tag{67}$$

$$a_{hj} = \mathrm{atan2}(q_{j,y} - y_h, q_{j,x} - x_h) - \theta_h \tag{68}$$

$$F_{hj} = 2 - \frac{r_{hj}^2}{R_F^2}, M_{hj} = \exp\left[- \frac{(a_{hj} - \Psi/2)^2}{2(\Psi/2)^2} \right] \tag{69}$$

The exact current stage cost is

$$\ell_i(\mathbf{x}_h) = - \sum_j F_{hj} M_{hj} W_{ij} - 3 \sum_j W_{ij} \exp\left(- \frac{r_{hj}^2}{3R_F^2} \right) \tag{70}$$

This formula is documented as implemented. In particular, the angular Gaussian is centered at $+\Psi/2$, not at zero, and the relative angle is not explicitly wrapped. The function parameter intended for mask steepness is currently unused.

10.3 Finite-horizon problem

With horizon H, the solver computes

$$\min_{\{v_h,\omega_h\}_{h=0}^{H-1}} 10 \sum_{h=0}^{H-1} \ell_i(\mathbf{x}_h \tag{71}$$

subject to the unicycle prediction dynamics and control bounds

$$-a_{\max} \leq v_h \leq a_{\max}, \; -a_{\max} \leq \omega_h \leq a_{\max} \tag{72}$$

The same configured quantity named maximum acceleration is currently used as the bound for both MPC control components. Only the terminal predicted position is constrained:

$$0 \leq x_H \leq W_{\text{map}}, 0 \leq y_H \leq H_{\text{map}} \tag{73}$$

There is currently no active collision penalty, control-effort penalty, or orientation penalty in this optimization. IPOPT is warm-started from the previous solution, and only the first optimized pair (v_0, ω_0) is sent to the real unicycle agent.

10.4 Other retained cost functions

The codebase also defines, but the current coupled MPC does not use,

$$J_{\text{coverage}}(\mathbf{p}_i) = \sum_j \|\mathbf{q}_j - \mathbf{p}_i\|_2^2 W_{ij} \tag{74}$$

and the collision penalty

$$J_{\text{collision}} = \alpha_c \exp\Big[-\beta_c \left(\|\mathbf{p}_i - \mathbf{p}_{\text{obs}}\|_2^2 - D_s^2 \right) \Big] \tag{75}$$

They are included here to distinguish available mathematical utilities from the active MPC objective.

11 Asynchronous estimation and causal order

For a periodic event with period P, start time t_0, and fire count k, the next scheduled time is computed without accumulated drift:

$$t_{\text{next}} = t_0 + kP \tag{76}$$

An event is due when

$$t + \varepsilon_t \geq t_{\text{next}} \tag{77}$$

At simulation step n, $t_n = n\Delta t$. The current coupled order is:

1. collect LOW samples from aerial robots if due;
2. read the latest already-published posterior and move aerial robots with HEDAC;

3. collect HIGH samples from ground robots if due;

4. update and publish the central GP if due;

5. read that posterior density and move ground robots.

Consequently, a HIGH sample collected at step n can influence ground control at step n after a successful update, but influences aerial control no earlier than step $n+1$.

When video recording is enabled, it is an observer of this completed state: after each outer step, it may render the already-published posterior and the current aerial and ground positions. It captures only steps satisfying $n \bmod r = 0$, where r is the configured frame interval, and also captures the final completed step if it was not already selected. Rendering and encoding do not submit observations, update the estimator, or alter either controller. For one video, the uncertainty panel uses the fixed color interval $[0, \sigma_{\max}^{(0)}]$, where $\sigma_{\max}^{(0)}$ is the largest HIGH posterior standard deviation in its first captured frame; later frames reuse that interval rather than independently rescaling their colors.

Estimator publication is transactional. A candidate update must complete retention, optional hyperparameter fitting, Cholesky factorization, prediction, positive-part conversion, and density validation before it replaces the cached posterior.

12 Evaluation quantities

12.1 Current HEDAC coverage error

The quantity reported by HEDAC as an ergodic metric is a discrete spatial L^2 difference. With sum-normalized cumulative coverage

$$\widehat{C}_j = \frac{C_j}{\sum_k C_k + \varepsilon} \tag{78}$$

the reported value is

$$E = \sqrt{\sum_j \left[M_j (\widehat{C}_j - \phi_j^{\text{reference}}) \right]^2} \tag{79}$$

This is not a Fourier-coefficient ergodic metric. In the present HEDAC step, $\phi^{\text{reference}}$ is the original normalized goal field stored when HEDAC was constructed, not necessarily the latest external multi-fidelity aerial target. It should therefore be interpreted carefully in coupled runs.

12.2 Posterior reconstruction diagnostics

Useful field diagnostics include root mean squared error

$$\text{RMSE} = \sqrt{\frac{1}{N} \sum_{j=1}^{N} \left(\mu_H(\mathbf{q}_j) - f_H(\mathbf{q}_j) \right)^2} \tag{80}$$

and mean negative log predictive density, when evaluating against simulator truth:

$$\mathrm{MNLPD} = \frac{1}{N}\sum_{j=1}^{N}\left[\frac{1}{2}\log(2\pi\sigma_{H,j}^2) + \frac{(f_H(\mathbf{q}_j) - \mu_H(\mathbf{q}_j))^2}{2\sigma_{H,j}^2}\right] \tag{81}$$

These diagnostics use hidden truth only for evaluation; they are not controller inputs.

13 Legacy compatibility path

When no central multi-fidelity posterior is supplied, the retained ground path can fuse separate aerial and ground GP estimates. Let $\bar{\sigma}_A, \bar{\sigma}_G$ be each standard-deviation field divided by its own maximum. The implemented weights are

$$d_j = \frac{1}{\bar{\sigma}_{G,j} + \varepsilon} + \frac{1}{\bar{\sigma}_{A,j} + \varepsilon} \tag{82}$$

$$\lambda_{A,j} = \frac{1/(\bar{\sigma}_{A,j} + \varepsilon)}{d_j}, \lambda_{G,j} = \frac{1/(\bar{\sigma}_{G,j} + \varepsilon)}{d_j} \tag{83}$$

and

$$\mu_{\mathrm{fused},j} = \lambda_{A,j}\mu_{A,j} + \lambda_{G,j}\mu_{G,j} \tag{84}$$

This is a compatibility fallback, not the central autoregressive GP. In multi-fidelity mode the controllers consume the shared posterior products defined in Section 5.

14 Equation-to-code map

Mathematical component	Primary implementation
HIGH truth and Gaussian mixture	`src/core/gmm.py`, `src/coupled_simulation.py`
LOW smoothing and sensor observations	`src/models/sensors.py`
Autoregressive GP and marginal likelihood	`src/core/multifidelity_gp.py`
Asynchronous estimator and posterior publication	`src/core/multifidelity_estimator.py`
Density clipping, normalization, and aerial target	`src/core/density.py`
Event schedule and controller-density caching	`src/simulation.py`
Video observer and encoding	`src/utils/multifidelity_video.py`, `examples/run_multifidelity.py`
HEDAC source, heat update, and gradient law	`src/core/hedac.py`, `src/utils/math_utils.py`
Dubins and unicycle state updates	`src/models/agents.py`
MPC prediction dynamics	`src/models/models.py`

Mathematical component	Primary implementation
Voronoi partition and Lloyd centroids	`src/utils/voronoi.py`, `src/coupled_simulation.py`
MPC objective	`src/core/costFunctions.py`, `src/coupled_simulation.py`
Coverage error	`src/core/base.py`

The associated configuration parameters and their tuning effects are documented in Multi-fidelity configuration reference.