Supplementary Materials: Coordination Without Command: Active Inference and the Route to Embodied Intelligence

Alexander D. Shaw

1 Temporally Predictive Scene-Based Drone Controller

This supplement summarises the implementation and mathematical structure of the temporally predictive scene-based drone controller used as the computational case study in the main paper. The controller is deliberately lightweight. It is not a full symbolic world model, a learned POMDP solver, or a complete active inference implementation in the most general sense. Instead, it is intended as a compact proof of principle showing how a continuous embodied agent can combine fast local belief updates, slower scene-level latent variables, and short-horizon policy valuation over temporally evolving scene structure.

The implementation is organised around four interacting components: noisy observation generation, variational-style belief updating, short-horizon policy rollout, and low-level command execution. At each control step, the agent receives noisy self-observations, sparse 3D ray observations, and an egocentric target observation when the target is directly visible or weakly glimpsed. These are used to update fast beliefs over self-state, target-state, and local ray-space structure, together with slower beliefs over scene visibility, progress, affordance, broader context, and target-memory reliability. Candidate actions are then scored under a short-horizon expected-free-energy-like objective that includes both immediate control terms and a temporally predictive scene-value term.

2 Implementation structure

The codebase is divided into a small number of modules:

- `sensing/observation_model.py`: noisy self, target, and ray observations.
- `sensing/raycast_sensor.py`: sparse 3D ray fan used as a lightweight local depth sensor.
- `belief/belief_state.py`: data structures for self, target, local-map, scene, temporal-scene, and context beliefs.
- `belief/vmp_filter.py`: approximate Bayesian / VMP-style updates for fast and slow beliefs.

- `control/autopilot.py`: short-horizon policy rollout and expected-free-energy-like action scoring.
- `env/`: simple kinematic drone, world, scene, and target setup in PyBullet.
- `main.py`: simulation loop, belief update, controller call, logging, and rendering.

The controller itself remains compact by design. The drone is treated as a controllable 3D body rather than a full quadrotor with detailed flight dynamics, allowing the case study to focus on embodied inference rather than low-level vehicle modelling.

3 Generative variables

3.1 Fast states

The fast layer tracks self-state, target-state, and local spatial structure:

$$x_t = \{x_t^{\text{self}}, x_t^{\text{target}}, x_t^{\text{map}}\}. \tag{1}$$

More concretely:

- x_t^{self}: drone position, velocity, yaw, and yaw-rate beliefs.
- x_t^{target}: target position, covariance, confidence, visibility flag, and occlusion history.
- x_t^{map}: ray-distance means, ray variances, and hit-mask structure from the sparse 3D ray sensor.

Self-state beliefs are represented with Gaussian means and covariances for position together with scalar uncertainty for yaw. Target beliefs are maintained in world coordinates, even though the raw observation is egocentric.

3.2 Slow scene states

The slower scene layer summarises what kind of situation the agent is in:

$$z_t = \{z_t^{\text{vis}}, z_t^{\text{prog}}, z_t^{\text{aff}}, z_t^{\text{ctx}}, z_t^{\text{mem}}\}. \tag{2}$$

These components correspond to:

- z_t^{vis}: visibility state, represented over {visible, soft-occluded, lost}.
- z_t^{prog}: progress state, represented over {advancing, stalled, trapped}.
- z_t^{aff}: affordance / vantage state, represented over {good-vantage, poor-vantage, blocked}.

- z_t^{ctx}: broader control context, including mode and context-specific precisions.
- z_t^{mem}: target-memory reliability, representing how much weight should still be placed on target beliefs under occlusion.

This scene layer is categorical in spirit, but implemented through smoothed probability-like belief variables rather than hard symbolic states.

4 Observation model

The observation model is intentionally sparse and noisy.

4.1 Self observation

The agent receives noisy self-position and yaw observations:

$$o_t^{\text{self,pos}} = p_t + \epsilon_t^p, \tag{3}$$

$$o_t^{\text{self,yaw}} = \psi_t + \epsilon_t^\psi, \tag{4}$$

with Gaussian noise determined by fixed configuration parameters.

4.2 Ray observations

Local spatial structure is sensed through a sparse 3D ray fan:

$$o_t^{\text{ray}} = \{d_{t,1}, \ldots, d_{t,N}\}, \tag{5}$$

where each $d_{t,i}$ is a noisy hit distance clipped at the sensor maximum range. The implementation uses a small grid of azimuth and elevation angles, yielding a lightweight local depth signal rather than a dense occupancy map.

4.3 Target observation

The target channel is egocentric, not world-coordinate. When visible, the agent receives noisy range, bearing, and elevation:

$$o_t^{\text{target}} = (r_t, \theta_t, \phi_t). \tag{6}$$

When the target is not directly visible but still broadly detectable, the observation model provides a weak "soft glimpse" with substantially increased uncertainty. This helps preserve purposeful search under partial occlusion without handing over a precise latent target state.

5 Belief updates

5.1 Fast Gaussian update

Fast self and target beliefs are updated through a Gaussian precision-weighted fusion of prior and observation. In generic form:

$$\Lambda_t^+ = \Lambda_t^- + \Lambda_t^o, \tag{7}$$

$$\mu_t^+ = (\Lambda_t^+)^{-1} \left(\Lambda_t^- \mu_t^- + \Lambda_t^o o_t \right), \tag{8}$$

where μ_t^- and Λ_t^- are the prior mean and precision, and Λ_t^o is the observation precision.

Self-state prediction uses the previous command and a simple kinematic forward model. Target-state prediction expands uncertainty under occlusion and reduces confidence when the target has not been observed for several steps.

5.2 Slow scene update

The slower scene variables are updated by combining a predicted prior over scene state with current evidence extracted from visibility quality, occlusion, open space, and progress-like signals. In stylised form:

$$q(z_t) \propto p(z_t \mid z_{t-1}, u_{t-1})^{\alpha} \, p(o_t \mid z_t)^{1-\alpha}, \tag{9}$$

where α is a tempering factor controlling the relative contribution of temporal persistence and current evidence.

The implementation uses smoothed multiplicative fusion rather than exact message passing. This is enough to maintain stable scene beliefs while allowing them to remain responsive to new evidence.

6 Temporal scene dynamics

A key component is that the slow scene is treated as temporally evolving rather than as a snapshot summary:

$$p(z_{t+1} \mid z_t, u_t). \tag{10}$$

In practice, this is implemented through transition-style persistence dynamics for visibility, progress, and affordance, together with an explicit memory-reliability variable:

$$z_{t+1}^{\text{vis}} \sim T_{\text{vis}}(z_t^{\text{vis}}, u_t), \tag{11}$$

$$z_{t+1}^{\text{prog}} \sim T_{\text{prog}}(z_t^{\text{prog}}, u_t), \tag{12}$$

$$z_{t+1}^{\text{aff}} \sim T_{\text{aff}}(z_t^{\text{aff}}, u_t), \tag{13}$$

$$z_{t+1}^{\text{mem}} = f_{\text{mem}}(z_t^{\text{mem}}, z_t^{\text{vis}}, z_t^{\text{aff}}). \tag{14}$$

These transitions are not learned from data. They are lightweight engineered dynamics that en-

courage persistence while still allowing scene state to change under action-conditioned evidence.

Conceptually, this means the drone is no longer asking only "what kind of situation am I in now?" but also "how is this situation likely to evolve if I move this way?"

7 Policy library and short-horizon rollout

The action space is discretised into a small library of motion primitives:

$$\Pi = \{\pi_1, \ldots, \pi_M\}, \tag{15}$$

where each candidate policy corresponds to a short motor primitive, such as forward, forward-left, forward-right, lateral slide, vertical adjustment, or turning behaviour.

For each candidate action, the controller rolls the simple kinematic state forward over a short horizon:

$$\hat{x}_{t+h+1} = f(\hat{x}_{t+h}, \pi), \qquad h = 0, \ldots, H-1. \tag{16}$$

Along the imagined trajectory it computes:

- target distance and progress,
- obstacle clearance and safety cost,
- altitude regularity,
- directional open-space proxy,
- visibility quality,
- occlusion proxy,
- scene-level epistemic value,
- temporally predicted future scene value.

8 Policy valuation

The controller uses an expected-free-energy-like score rather than a literal full EFE derivation. In schematic form:

$$J(\pi) = -G(\pi) + \alpha\,\mathrm{Vis}(\pi) + \beta\,\mathrm{Prog}(\pi) + \delta\,\mathrm{MemRel}(\pi), \tag{17}$$

where:

- $G(\pi)$ captures pragmatic, safety, stability, heading, uncertainty, and occlusion-related penalties,
- $\mathrm{Vis}(\pi)$ rewards predicted future visibility,

- Prog(π) rewards predicted future progress,
- MemRel(π) rewards actions that preserve useful target memory under occlusion.

A more explicit stylised decomposition is:

$$G(\pi) = G_{\text{target}} + G_{\text{risk}} + G_{\text{stability}} + G_{\text{uncertainty}} - G_{\text{scene-epistemic}} - G_{\text{temporal-scene}}. \tag{18}$$

The component, $G_{\text{scene-epistemic}}$, rewards actions expected to reduce uncertainty over the current slow scene state. The component, $G_{\text{temporal-scene}}$, rewards actions predicted to move the latent scene toward better future states, for example:

- improved visibility,
- improved progress,
- better affordance / vantage,
- lower scene entropy,
- preserved memory reliability.

This is the main point at which the controller moves from snapshot epistemics to temporally predictive scene-based valuation.

9 Why the implementation matters

Although the implementation remains deliberately lightweight, it is sufficient to illustrate the architectural claim made in the main paper.

It is *distributed* because inferential labour is not confined to a single flat state estimate. Fast local beliefs over self, target, and nearby geometry remain active at the sensorimotor interface, while slower scene variables summarise broader aspects of visibility, progress, affordance, context, and memory.

It is *hierarchical* because these components operate on different timescales and levels of abstraction. Fast variables handle immediate regulation and local geometry, whereas the slow scene layer tracks more persistent features of the situation.

It is *plural* because policy evaluation is shaped by multiple partially distinct pressures rather than a single immediate objective. Target pursuit, visibility recovery, progress maintenance, uncertainty reduction, and memory preservation can all influence action together.

10 What this controller is, and is not

The controller should be understood as a proof of principle rather than a finalised product. It is intentionally more structured than a reactive tracker, but simpler than a full learned world model

or POMDP planner.

It does *not* currently implement:

- full symbolic factor-graph (Variational) message passing,
- long-horizon optimal planning,
- a globally learned latent world model,
- a complete *generalised active inference* architecture.

What it *does* show is that once an embodied agent is given temporally evolving scene beliefs and a policy valuation scheme sensitive to how those beliefs are expected to change, behaviour begins to move in a recognisably different direction from reactive monolithic control. That is enough to make it a useful computational bridge between the conceptual claims of the main paper and the practical design of embodied active-inference-style systems.

11 Minimal pseudocode

1. Observe noisy self-state, ray distances, and target cue.
2. Predict fast self-state and target-state from previous command.
3. Update fast beliefs over self, target, and local map.
4. Predict slow scene state from previous slow state and persistence dynamics.
5. Fuse current evidence with predicted slow-state priors.
6. For each candidate action:
   - 6.1. roll out short-horizon kinematic future,
   - 6.2. estimate visibility, occlusion, open-space, and safety,
   - 6.3. predict future scene trajectory,
   - 6.4. score policy using expected-free-energy-like objective.
7. Select best-scoring action and execute command.
8. Repeat.