# AIF MuJoCo Robot

*Code Architecture & API Reference*

---

Complete code architecture, module APIs, CLI options, configuration, and MuJoCo model descriptions.

# 1. Project Overview

A simple Active Inference (AIF) controller for a 3D MuJoCo robot. The robot navigates toward a goal by minimizing Expected Free Energy (EFE).

## Project Structure

| Path | Description |
|---|---|
| src/AIFMuJoCoRobot.jl | Main module |
| src/aif/beliefs.jl | Belief state (Gaussian) |
| src/aif/generative_model.jl | Transition + observation models |
| src/aif/efe.jl | Expected Free Energy |
| src/aif/policy.jl | Policy selection (argmin EFE) |
| src/aif/action.jl | Action-to-control mapping |
| src/aif/rxinfer_filter.jl | RxInfer streaming filter |
| src/control/aif_controller.jl | AIF controller (integrator) |
| src/sim/mujoco_env.jl | MuJoCo environment wrapper |
| src/sim/sensors.jl | Sensor / observation model |
| src/utils/math.jl | Math utilities |
| src/utils/logging.jl | Logging utilities |
| scripts/run_cli.jl | CLI entry point |
| experiments/configs.jl | Configuration presets |
| models/robot.xml | MuJoCo 3D robot model |

## Dependencies (Project.toml)

| Package | Purpose |
|---|---|
| BangBang | Efficient in-place operations |
| Distributions | Probability distributions |
| LinearAlgebra | Matrix/vector operations (stdlib) |
| MuJoCo | MuJoCo.jl physics simulation |
| Plots | Plotting trajectories |
| Printf | Formatted printing (stdlib) |
| Random | RNG (stdlib) |
| Rocket | Reactive streams for RxInfer |
| RxInfer | Bayesian inference on factor graphs |

## Quick Start

```
cd aif_mujoco_robot
julia --project=. -e 'using Pkg; Pkg.instantiate()'
julia --project=. scripts/run_cli.jl \
  --goal 0.8 0.8 0.4 --init -0.5 -0.5 0.2 \
```

```
  --steps 500 --ctrl_scale 5.0 --save_plot trajectory.png
```

# 2. Main Module (AIFMuJoCoRobot.jl)

Entry point that includes all submodules and exports the public API.

## Include Order

```
utils/math.jl  ->  utils/logging.jl
aif/beliefs.jl -> aif/generative_model.jl -> aif/efe.jl
aif/action.jl  -> aif/policy.jl -> aif/rxinfer_filter.jl
sim/sensors.jl -> sim/mujoco_env.jl
control/aif_controller.jl
experiments/configs.jl
```

## Exports

| Symbol | Description |
|---|---|
| run_simulation | Run the full AIF + MuJoCo loop |
| default_model_path | Path to models/robot.xml |
| BeliefState | Gaussian belief struct |
| init_belief | Create initial belief |
| update_belief! | Bayesian update with observation |
| EnvState | MuJoCo environment struct |
| load_env | Load MuJoCo model |
| reset! | Reset environment |
| step! | Step MuJoCo simulation |
| get_position | Read current [x,y,z] |
| get_goal | Read goal [x,y,z] |

## run_simulation Signature

```
run_simulation(;
    steps::Int = 100,
    goal::Vector{Float64} = [0.8, 0.8, 0.4],
    init_pos::Vector{Float64} = [-0.5, -0.5, 0.2],
    obs_noise::Real = 0.01,
    γ::Real = 1.0,  β::Real = 0.1,
    ctrl_scale::Real = 1.0,
    nsteps_per_ctrl::Int = 5,
    model_path::String = default_model_path(),
    seed::Union{Int,Nothing} = 42,
    process_noise = 0.005,
    verbose::Bool = true,
    inference_backend::Symbol = :analytic,  # or :rxinfer
)
```

## Control Loop (Architecture)

```
+------------------+
| compute_control  |  belief + goal -> (ctrl, action, efe)
+--------+---------+
         |
+--------v---------+
| predict_belief!  |  OR  rxinfer_step! (if :rxinfer)
+--------+---------+
         |
+--------v---------+
| MuJoCo step!     |  Apply ctrl, advance physics
+--------+---------+
         |
+--------v---------+
| read_observation  |  qpos + noise -> obs
+--------+---------+
         |
+--------v---------+
| update_belief!   |  OR  rxinfer_step! (if :rxinfer)
+--------+---------+
         |
      [repeat]
```

# 3. Belief State (beliefs.jl)

Represents the agent's belief over 3D position as a diagonal Gaussian.

## BeliefState Struct

```
struct BeliefState
    mean::Vector{Float64}   # [μ_x, μ_y, μ_z]
    cov::Vector{Float64}    # [σ²_x, σ²_y, σ²_z]
end
```

## Exports

| Symbol | Description |
|---|---|
| `BeliefState` | Struct with mean and cov |
| `init_belief` | Create initial belief from mean/cov |
| `update_belief!` | Bayesian update with observation |
| `predict_belief!` | Predict forward given control |
| `entropy` | Entropy of diagonal Gaussian |

## Key Constants

- COV_MIN = 1e-5 (lower bound on variance)
- COV_MAX = 2.0 (upper bound on variance)

*When inference_backend = :rxinfer, predict_belief! and update_belief! are bypassed. The RxInfer posterior is written directly into belief.mean and belief.cov.*

# 4. Generative Model (generative_model.jl)

Transition and observation likelihood for Active Inference.

| Symbol | Description |
|---|---|
| `predict_transition` | `s' = s + u (3D additive)` |
| `observation_likelihood` | `P(o\|s) diagonal Gaussian` |
| `predict_observation` | `Expected obs from state (noiseless)` |

*These functions are used by EFE/policy selection regardless of which inference backend is active.*

# 5. Expected Free Energy (efe.jl)

Computes EFE = pragmatic - epistemic for policy selection.

| Symbol | Description |
|---|---|
| `compute_efe` | Full EFE for an action |
| `pragmatic_term` | Goal-distance cost (with ctrl_scale) |
| `epistemic_term` | Uncertainty / information term |

## compute_efe Signature

```
compute_efe(belief_mean, belief_cov, action, goal;
    γ=1.0, β=0.1, axis_weights=nothing,
    ctrl_scale=1.0, ctrl_lim=1.2)
```

# 6. Policy Selection (policy.jl)

Selects the action minimizing EFE over a discrete 5×5×5 action grid.

| Symbol | Description |
|---|---|
| `select_action` | `Choose best action for belief+goal` |
| `get_action_set` | `125-action grid (step_size=0.08)` |

## select_action Signature

```
select_action(belief, goal;
    actions=get_action_set(), γ=1.0, β=0.1,
    axis_weights=nothing, ctrl_scale=1.0, ctrl_lim=1.2)
```

# 7. Action (action.jl)

Converts abstract velocity actions into MuJoCo control signals.

| Symbol | Description |
|---|---|
| to_control | action -> scaled+clamped control |
| clamp_action | Clamp action to limits |

## to_control Signature

```
to_control(action; scale=1.0, ctrl_lim=1.2)
```

# 8. RxInfer Filter (rxinfer_filter.jl)

Alternative inference backend using RxInfer.jl's reactive message-passing. Three independent 1D engines (x, y, z) run a linear-Gaussian SSM.

| Symbol | Description |
|---|---|
| `RxInferFilter3D` | `Struct with 3 per-axis engines` |
| `init_rxinfer_filter` | `Create and start 3-axis filter` |
| `rxinfer_step!` | `Push (ctrl, obs), return posterior` |
| `rxinfer_stop!` | `Stop engines, unsubscribe` |

## Streaming Design

Each axis uses a single RecentSubject{NamedTuple{(:y,:u)}} stream. Pushing a (y, u) tuple triggers exactly one inference cycle per step, avoiding the combineLatest double-fire issue.

## init_rxinfer_filter Signature

```
init_rxinfer_filter(
    init_mean::AbstractVector,
    init_cov::AbstractVector;
    obs_noise = 0.01,
    process_noise = 0.005,
)
```

## rxinfer_step! Signature

```
rxinfer_step!(filter, ctrl, obs)
  → (post_mean::Vector, post_var::Vector)
```

# 9. AIF Controller (aif_controller.jl)

Integrates belief, policy, and action into a single control step.

| Symbol | Description |
|---|---|
| `compute_control` | `belief+goal -> (ctrl, action, efe)` |

## compute_control Flow

```
1. select_action(belief, goal; ...)
2. to_control(action; scale, ctrl_lim)
3. compute_efe(belief.mean, belief.cov, action, goal; ...)
4. return (ctrl=..., action=..., efe=...)
```

*The controller is backend-agnostic: it always reads belief.mean and belief.cov regardless of whether they came from analytic or RxInfer updates.*

# 10. MuJoCo Environment (mujoco_env.jl)

Wraps MuJoCo for the 3D point-mass robot.

| Symbol | Description |
|---|---|
| `EnvState` | `Struct: model, data, goal` |
| `load_env` | `Load XML model` |
| `reset!` | `Reset to initial position` |
| `step!` | `Apply control, advance physics` |
| `get_position` | `Current [x,y,z] from qpos` |
| `get_goal` | `Goal [x,y,z]` |

## Model Layout

- qpos[1], qpos[2], qpos[3]: x, y, z (slide joints)
- ctrl[1], ctrl[2], ctrl[3]: displacement (target = pos + ctrl)

## step! Signature

```
step!(env, ctrl; nsteps=5)
```

Applies ctrl and runs nsteps MuJoCo physics steps for stability.

# 11. Sensors (sensors.jl)

Extracts observations from MuJoCo state with optional noise.

| Symbol | Description |
|---|---|
| `read_position` | `Extract [x,y,z] from qpos` |
| `read_observation` | `Position + Gaussian noise` |

## read_observation Signature

```
read_observation(qpos; obs_noise=0.01, rng=default_rng())
```

obs_noise: scalar or [σ²_x, σ²_y, σ²_z]. Adds sqrt(σ²) * randn() per dimension when σ² > 0.

# 12. Utilities

## Math Utilities (math.jl)

| Symbol | Description |
|---|---|
| normalize! | Normalize probability vector in place |
| gaussian_pdf | Univariate/multivariate diagonal PDF |
| gaussian_entropy | Entropy of univariate Gaussian |
| softmax | Numerically stable softmax |
| clamp_vec | Clamp vector elements to bounds |

## Logging Utilities (logging.jl)

| Symbol | Description |
|---|---|
| log_step | Print step: pos, goal, belief, efe |
| log_summary | Print summary: steps, dist, converged |

# 13. CLI (run_cli.jl)

Command-line interface for running simulations and visualisations.

## Usage

```
julia --project=. scripts/run_cli.jl \
  --goal <x> <y> <z> --init <x> <y> <z> [options]
```

## All Options

| Option | Description | Default |
|---|---|---|
| `--goal` | Goal position (x y z) | `0.8 0.8 0.4` |
| `--init` | Initial position (x y z) | `-0.5 -0.5 0.2` |
| `--steps` | Max simulation steps | `500` |
| `--ctrl_scale` | Control scaling factor | `4.0` |
| `--obs_noise` | Observation variance | `0.01` |
| `--process_noise` | Process noise variance | `0.005` |
| `--backend` | analytic \| rxinfer | `analytic` |
| `--save_plot` | Path to save trajectory PNG | (none) |
| `--render` | Launch MuJoCo visualiser | `false` |
| `--renderarm` | Panda arm pick-and-place | `false` |
| `--agent_color` | Agent color r g b [a] | (default) |
| `--goal_color` | Goal color r g b [a] | (default) |
| `--seed` | Random seed | `42` |
| `--verbose` | Print step logs | `true` |

## Render Arm Sequence

With --renderarm, the AIF trajectory is replayed on a Panda arm:

1. Arm moves to init position (pickup)
2. Red ball follows arm along AIF path (carry)
3. Red ball placed on green box at goal (drop)

Panda workspace bounds: X [0.2, 0.8], Y [-0.4, 0.4], Z [0.1, 0.5]. Trajectory positions are clamped.

## Example (RxInfer + Panda Arm)

```
julia --project=. scripts/run_cli.jl \
  --backend rxinfer \
  --goal 0.8 0.8 0.4 --init 0.3 0.0 0.2 \
  --steps 500 --ctrl_scale 5.0 \
  --save_plot trajectory.png --renderarm
```

# 14. Configuration (configs.jl)

Predefined configuration presets for experiments.

## default_config()

| Parameter | Default Value |
|---|---|
| steps | 500 |
| goal | [0.8, 0.8, 0.4] |
| init_pos | [-0.5, -0.5, 0.2] |
| obs_noise | 0.01 |
| process_noise | 0.005 |
| γ | 1.2 |
| β | 0.05 |
| ctrl_scale | 4.0 |
| nsteps_per_ctrl | 5 |
| seed | 42 |
| verbose | true |
| inference_backend | :analytic |

## Presets

- config_goal_seeking(): γ=2.0, β=0.05 (emphasize goal)
- config_exploration(): γ=0.5, β=0.3 (emphasize exploration)

# 15. MuJoCo Models

## models/robot.xml

Main 3D robot scene for AIF simulation:

- World: ground plane, light, target site at (0.8, 0.8, 0.4)
- Robot: body with three slide joints (x, y, z), sphere geom
- Actuators: position actuators on slide_x/y/z, range ±10

## panda_render_scene.xml

Scene for --renderarm: Panda arm with pick-and-place visuals.

- Red sphere (mocap) at initial position
- Green box (static) at goal position
- Includes panda_mocap.xml for the arm model
- Positions substituted at runtime from --init and --goal