CAMAP · ARENA_CONFIG.YAML · PLACE-CELL PARAMETERS

What defines a place cell

Every parameter in the `behavior` / `spatial_map_2d` block plays a role in one question: does this neuron reliably fire in a specific place? Here's what each one does — and which way it pushes the final count.

THE DECISION

CaMAP doesn't have a single "place cell" knob. A unit must pass **two independent gates**, both built from its firing-rate map over the arena.

A place cell = significant spatial tuning AND a stable map

Significant — carries more spatial information than chance (shuffle test) · **Stable** — the same map reappears in independent halves of the session

708	416	210
units analyzed	spatially significant (SI shuffle test)	stable across splits
	58.8%	29.7%

200
place cells (pass both)
28.2%

Funnel from your run on the prerecorded session. The parameters below shift these numbers up or down by tightening or loosening each gate.

1 CLEAN THE BEHAVIOR

Before any map is built, the trajectory is de-noised and frames where the animal isn't really running are dropped. Garbage-in here corrupts every map downstream.

`speed_threshold` `25.0`

mm/s

Frames where the animal moves slower than this are dropped before mapping. Place fields are defined during active locomotion — including immobility folds in resting/grooming periods that smear spatial tuning. In your run this kept **17,664 of 23,705** frames.

↑ higher → only fast running counts → cleaner fields, fewer events

↓ lower → more frames kept, more noise from slow periods

`speed_window_seconds` `0.25`

s

Speed is computed/smoothed over this window before the threshold is applied. Frame-to-frame speed is jittery; smoothing over 0.25 s stops the animal from flickering in and out of the "moving" set right at the threshold.

larger → steadier speed estimate, blurs quick stops/starts

`hampel_window_frames` `7`

frames

Window of the **Hampel filter** — a sliding median-based outlier detector run over the x/y position. For each point it looks at the 7 surrounding frames to decide what "normal" is locally.

larger → smooths over longer glitches, but can erase real fast turns

hampel_n_sigmas `3.0` MAD-scaled

How far from the local median (in robust MAD units) a position sample must be to be flagged as a tracking **jump** and interpolated away — LED flicker, reflections, lost frames. ~3σ ≈ the 99.7% band. Your run interpolated **3,080 jump frames**.

↑ higher → only extreme jumps removed (more permissive)

↓ lower → aggressively scrubs the trajectory (risk: real movement)

2 BUILD THE RATE MAP

The arena is gridded into bins; each unit gets a 2D firing-rate map (activity per bin ÷ time spent there). This is the object every later test operates on.

bins `40` per axis → 40×40 = 1600

Spatial resolution of the map. With a 400 mm arena, 40 bins ≈ **10 mm/bin**. Too coarse hides small fields; too fine leaves each bin under-sampled and noisy.

↑ more bins → finer detail, but each bin has fewer samples (noisier)

↓ fewer bins → smoother, more robust, but blurs field shape

`min_occupancy` `0.05`

s per bin

A bin must be visited for at least this many seconds to be counted as valid; under-visited bins are masked out (a rate from near-zero dwell time is meaningless). Your run had all **1600/1600** bins valid — the animal covered the whole arena.

↑ higher	→ only well-sampled bins kept, smaller usable map
↓ lower	→ keeps sparse bins, risk of spurious hot spots

`spatial_sigma` `3`

bins (≈30 mm)

Width of the Gaussian smoothing applied to the rate map. Smooths out single-bin noise so a field reads as a coherent blob rather than scattered pixels.

↑ more smoothing	→ rounder fields, may merge nearby ones
↓ less smoothing	→ sharper, but noisier and more multi-peak maps

`event_threshold_sigma` `0` ADVANCED

Activity level (in SD above baseline) a sample must exceed to count as an "event" feeding the map. `0` means no hard cutoff — the deconvolved activity is used directly (paired with amplitude weighting below). Raise it to keep only strong transients.

higher → only big transients count → sparser maps

`si_weight_mode` `amplitude` ADVANCED

How each frame contributes to spatial information: `amplitude` weights by the activity magnitude (uses how much the cell fired), rather than treating every event as an equal binary tick. Matches the deconvolved-activity input from calab.

amplitude → strong transients dominate the map

3 GATE A — SIGNIFICANT SPATIAL INFORMATION

Is the map's spatial structure real, or could random timing produce it? CaMAP computes the unit's spatial information (bits), then compares it against a null built by repeatedly shifting the activity relative to position. **This gate produced 416/708 in your run.**

`n_shuffles` `500`

null draws

How many times the activity is circularly shifted to build the chance distribution. More shuffles = a smoother null and more precise p-values — and a longer run. This is the slow step (you parallelized it with `n_workers`=8).

↑ more	→ finer p-value resolution, slower
↓ fewer (e.g. 100)	→ fast pass, coarser p-values

`p_value_threshold` `0.05`

significance

The cutoff: a unit is "spatially significant" if its real spatial information beats the 95th percentile of its own shuffle null. **This is the main strictness dial for Gate A.**

↑ higher (0.1)	→ looser → MORE significant units
↓ lower (0.01)	→ stricter → FEWER, higher-confidence units

`min_shift_seconds` 20

s

Each shuffle shifts the activity by at least this much. A floor matters: tiny shifts leave activity still aligned with position, inflating the null toward the real value and hiding true tuning. 20 s guarantees the shuffled copy is genuinely decorrelated.

too small → weak/contaminated null; keep ≥ a field-crossing timescale

`random_seed` 1

reproducibility

Seeds the shuffle RNG so the same data gives the same p-values every run. Change it only to confirm a result isn't an artifact of one particular shuffle draw.

fixed → reproducible; vary → sanity-check robustness

`block_shift` 0 ADVANCED

Shuffling scheme. `0` = plain circular shift of the whole activity trace (preserves its temporal autocorrelation, just offsets it in time). Non-zero switches to block-wise shuffling — rarely needed here.

leave at 0 unless you have a specific null-model reason

4 FIND THE PLACE FIELD

A significant cell should have an actual contiguous field — a defined region of elevated firing — not just diffuse structure. These set what "a field" means geometrically.

`place_field_seed_percentile` 95

percentile

Field detection starts from the hottest bins — those above the 95th percentile of the rate map — then grows outward. The seed marks the peak of a candidate field.

↑ higher → only the very hottest peaks seed fields

↓ lower → more candidate peaks considered

`place_field_threshold` `0.35`

fraction of peak

The field's border: bins firing above 35% of the unit's peak rate belong to the field. Sets how big and inclusive each field is.

↑ higher (0.5)	→ tighter, smaller fields

↓ lower (0.2)	→ larger fields, may merge neighbors

`place_field_min_bins` `5`

bins

A region must span at least this many contiguous bins to count as a real field — rejects single-bin noise blobs that survive smoothing.

↑ higher	→ only sizeable fields qualify

↓ lower	→ tiny hot spots accepted (noise risk)

5 GATE B — STABILITY

A real place cell fires in the same spot throughout the session. CaMAP splits the session into independent chunks, makes a map from each, and correlates them. **This gate produced 210/708; units passing both A and B = 200 place cells.**

`stability_splits` `[2, 10]`

session divisions

The session is split two ways — into **2 halves** and into **10 blocks** — and the per-split rate maps are correlated. A unit must stay consistent across all splits to be "stable." Example unit from your run: split-2 r = +0.897, split-10 r = +0.958 → stable.

more / finer splits	→ stricter consistency test → fewer pass

coarser splits	→ easier to pass

QUICK MENTAL MODEL

Stages 1–2 **prepare** the rate map · Gate A (stage 3) asks is it spatial? · stage 4 asks is there a real field? · Gate B (stage 5) asks is it stable? A **place cell passes both gates**. To get more place cells, loosen `p_value_threshold` / stability; for higher-confidence cells, tighten them.

Parameters from `capstone/config/arena_config.yaml` · counts from the prerecorded-session capstone run · analysis by CaMAP (`analyze_units` + the spatial-information shuffle test). See `WORKSHOP_GUIDE.html` for the full pipeline and every other config knob.