Foveal Playground — Plan B: Compact map representations

Nov 8, 2026 · @Luiz Velho

Question and scope

Question. How far can the persistent map's memory and fusion time be cut while it keeps the fusion semantics of the baseline (12-mm association, support-weighted mean of one contribution per surfel per fixation), and what does each saving cost in reconstruction quality? TR-26-11 (Section 13) lists "compact map representations that keep fusion semantics while cutting memory and time".

The rows changed. Map representation (flat float64 surfel arrays → compact surfels, voxel or octree maps) and, in the later tiers only, fusion. Everything else is held fixed.

Why this experiment is unusually clean. The gaze policy never reads the map. A different map therefore leaves every gaze, every coverage value and every saccade exactly as in the baseline. Only the map products, fusion counts and timings can change, so map variants can be compared on identical input.

A note on the baseline. TR-26-11 reports the v0 run (16,517,811 surfels). The repository ships Playground-2 (16,943,073 surfels, 356 fixations). This plan uses Playground-2.

Starting point

`explorer.GlobalMap` stores, per surfel, `xyz` and `rgb` as float64 and `inst`, `support`, `first_fix` as int32: 60 bytes. At 16.94 M surfels that is about 1.0 GB in memory; f3d-vision measured a 991 MB `final-map.npz` and a 2.2 GB run directory (mostly per-fixation `points.npz`). In the v0 run fusion took 255 of 729 s, more than any host step, growing from about 0.1 s to 2.9 s per fixation (MEASURED in f3d-vision).

Why the map is so large. Each fixation yields about 65,000 points, and 356 fixations about 23 M. Two rules keep most of them:

- *No within-patch de-duplication.* A point matches only surfels from earlier fixations, so every first-time point becomes its own surfel. A pixel subtends about 0.82 mm per metre of range, so a frontal surface at 2 m puts about 50 points in one 12-mm cell (ESTIMATE). Those 50 points become 50 surfels.

- *Little overlap.* The sparse lattice was built so footprints barely overlap, so few points ever meet an existing surfel.

Why fusion is slow. `GlobalMap.nearest` rebuilds its index every fixation. It filters all N surfels to the patch's bounding box, packs their cell keys and sorts them, then searches. Cost grows with the map, not with the patch.

Who uses the map.

caller	uses
`run.explore`	`fuse`, `n`, `save`, `arrays` (for the PLYs)
`run.checks`	`load`, `arrays`, `nearest` (save/load round trip), and the FSG3 bitwise known answer
`visuals.overview`	`final-map.npz` keys `xyz_h`, `rgb`, `instance_id_oracle`
`compare.py`	`final-map.npz` arrays bitwise; per-fixation `fusion` counts in `trajectory.json`
an E3 evaluation	`nearest`

A detail that matters for offline work. `points.npz` stores `xyz_h` as float32, while fusion used the float64 points. Replaying a run's `points.npz` into a fresh `GlobalMap` comes close to the baseline map, but not bitwise.

Design

One map interface. Define a small protocol in a new `tools/greedy_foveal/maps.py`: `n`, `fuse(patch_id, xyz, rgb, inst, fix) -> counts`, `nearest(p) -> (index, distance)`, `arrays() -> dict`, `save(path)`, `load(path)`. `GlobalMap` is its reference implementation. `run.explore` takes `--map {global,…}` (default `global`) and records the choice in `trajectory.json`. `arrays()` keeps the key names `visuals.py` and the PLY writer expect, so the figures work for every variant.

Offline replay. Because the map does not affect control, map work needs the GPU only once. One baseline run with an extra `--save-points-f64` flag writes `points64.npz` beside each `points.npz` (an extra file, so the signature is untouched). A new

`tools/greedy_foveal/map_replay.py --run BASE --map KIND --out OUT` then fuses those 356 patches in order on any laptop and writes the map, the fusion counts and the timings. The instructor can produce this run once and share it.

Three tiers of change. Each tier is done and measured before the next. The tiers differ in what they promise about the baseline map.

tier	change	promise	expected gain (ESTIMATE)
0 — engineering	an incremental spatial hash kept across fixations (cell key → surfel indices), appended to as surfels are created; same exact search and smallest-index tie rule	bitwise identical map	fusion time per fixation stops growing with the map
1 — compact storage	float32 `xyz`; uint8 or float16 `rgb`; uint16 `support`, `first_fix`, `inst` (60 → about 21 B per surfel)	same rule, values rounded; association can flip only at the 12-mm boundary	memory about ×0.35
2a — within-patch de-duplication	unmatched points of one fixation are merged into one new surfel per 12-mm cluster, the cluster mean; a `weight` (point count) is stored beside `support`	same association and blending rule against existing surfels; new rule inside a patch	surfels ÷ 10 to ÷ 50, depending on range
2b — voxel map	one surfel per 12-mm voxel; association by cell instead of by radius	fusion by cell, support-weighted mean kept	the simplest and fastest map; slightly different neighbourhoods

tier	change	promise	expected gain (ESTIMATE)
stretch	range-adaptive cells (an octree whose cell grows with range × pixel angle), or a TSDF along the eye rays	new semantics	fixes the scale dependence of a fixed 12 mm; gives normals

What "keep fusion semantics" means here. For tiers 0–2a it means: a point is associated with the nearest existing surfel strictly within 12 mm, all points of one fixation on one surfel make one contribution, and contributions are blended with weight `support`. Tier 2b keeps the blending but replaces the radius search with the cell. The stretch tier is a different representation and is reported as such.

Comparing a variant with the baseline. `compare.py check` will fail for every variant past tier 0, because it compares the map and the per-fixation fusion counts. Add `compare.py check --control-only`, which compares gazes, decisions, coverage and points but skips the map products and `fusion` fields. It must PASS for every variant: that is the test that the map really stayed out of the loop.

Work plan

Six milestones on branch `project/compact-map`. Only M0 and M5 need the GPU.

1. **M0 — Interface and replay harness.**
   - `maps.py` protocol; `GlobalMap` behind it; `--map` on `explore`; `--save-points-f64` writing `points64.npz`.
   - `map_replay.py` and `compare.py check --control-only`.
   - One full baseline run with `--save-points-f64`.
   - *Checks that can fail:* that run PASSES the full `compare.py check` (the extra file changes nothing); replaying its `points64.npz` through `GlobalMap` reproduces its `final-map.npz` bitwise.
2. **M1 — Tier 0, incremental hash.**
   - `HashedGlobalMap`: persistent cell index, exact 27-cell search, smallest-index ties, the same two-stage search as `nearest`.

- *Checks:* bitwise equal to `GlobalMap` on the replay and on the FSG3 three-plane known answer in `run.checks`; then a GPU smoke run and full run with `--map hashed` PASS the full `compare.py check`. If it holds, propose it as the new default.
- Report fusion seconds per fixation against fixation number, for both maps.

3. **M2 — Tier 1, compact storage.**
   - `CompactMap`: float32 positions, small integer types, uint8 display RGB.
   - Count association flips against `GlobalMap` on the replay (points matched to a different surfel, or matched by one map and not the other).
   - *Check:* `--control-only` PASS; map-to-map distances (Evaluation) within 0.01 mm at the median.
4. **M3 — Tier 2a, within-patch de-duplication.**
   - `DedupMap`: cluster each fixation's unmatched points at 12 mm (by hash cell, then merge clusters closer than 12 mm, deterministically), one new surfel per cluster.
   - Replace the FSG3 known answer in `run.checks` with this rule's own: a plane at 2 m sampled at 1.6 mm must give its area ÷ cell² surfels within a stated boundary margin; a second, overlapping plane must add only its non-overlapping part. Keep the FSG3 check for `global` and `hashed`.
5. **M4 — Tier 2b, voxel map; stretch tier if time allows.**
   - `VoxelMap` at 12 mm with its own known answers, as in M3.
   - Stretch: range-adaptive octree cells, sized from range × pixel angle, which also answers Plan A's scale question.
6. **M5 — Confirmation runs and report.** One full GPU run per kept variant, `checks` with that variant's known answers, `--control-only` PASS, the Evaluation table.

Evaluation

Each variant gets one row in a table against `GlobalMap`, all from the same 356 replayed patches, so input differences cannot explain any result.

measure	definition
surfels	final count
bytes per surfel; map bytes in memory	from the array dtypes; peak RSS of the replay

measure	definition
`final-map.npz` bytes; run directory bytes	on disk
fusion seconds	total, and the mean over the last 50 fixations (the growth the hash should remove)
completeness at 12 mm	% of baseline surfels with a variant surfel within 12 mm
accuracy at 12 mm	% of variant surfels with a baseline surfel within 12 mm
distance to baseline	median and 95th percentile, mm, both directions
first-hit coverage at 12 mm	% of first-hit solid angle from the head origin with a surfel within 12 mm (exercise E3, shared with Plan A)

The criterion. A variant "keeps fusion semantics" in practice if first-hit coverage stays within 0.1 percentage points of the baseline, and completeness stays above 99 %. These thresholds are proposals: set them before the first variant is measured, and record them in the branch's README.

Figures. Fusion seconds per fixation against fixation number, one line per variant. And the Pareto view: map bytes on one axis, first-hit coverage on the other, one point per variant.

Risks and open questions

- **Bitwise is harder than it looks (tier 0).** The baseline's tie rule (smallest index), its two-stage search and its float64 arithmetic order must all be reproduced. Any reordering of a sum can change the last bit. Test on the replay before any GPU time.
- **Determinism of clustering (tier 2a).** Merging clusters must not depend on hash-iteration order. Sort by cell key and point index.
- **Support changes meaning.** After de-duplication one surfel stands for many points. Keep `support` as "fixations that contributed" and add `weight`; anything downstream that reads `support` (Plan C uses it) must say which it wants.
- **The quality measure is only as good as E3.** Without the first-hit evaluation, only map-to-map numbers are available, and those measure agreement with the baseline, not correctness.

- **Run-directory size is mostly `points.npz`, not the map.** Shrinking it (float16 offsets, or an opt-in `--points none`) would also cut storage, but those files are compared by the signature. Keep that a separate, opt-in change.
- *Open question:* should tier 0 become the default on `main` once it passes? It is a pure speed-up, and Plans C and D will read the map often.

Deliverables and effort

About 6 student-weeks, with 3–4 GPU runs in all; the rest runs on a laptop from one shared baseline run (ESTIMATE).

milestone	deliverable	effort (ESTIMATE)
M0	`maps.py`, `map_replay.py`, `--save-points-f64`, `compare.py --control-only`, one shared baseline run	1 week
M1	`HashedGlobalMap`, bitwise PASS, timing curve	1 week
M2	`CompactMap`, flip count, map-to-map numbers	0.5–1 week
M3	`DedupMap` with its known answers	1 week
M4	`VoxelMap`; stretch tier optional	1–1.5 weeks
M5	confirmation runs, comparison table, two figures, short report	1 week

The replay harness and `--control-only` are useful well beyond this plan: Plan C uses the same idea for the policy.