Yes — freeze is totally patch-able (not fundamental)

- **What you want (per)**: when an outage event is active, the "pose measurement" should become **stale** (hold last valid reading) for L steps.
- **What you currently do**: even while the event is active, you keep updating `last_valid_pose` every step to the current oracle pose, then "freeze" returns `last_valid_pose` — which is just the current pose.

So "freeze" ends up being effectively **no corruption**.
This is just a logic bug and is straightforward to fix (e.g., only update `last_valid_pose` when not in an active freeze event, or snapshot it once at event start).

Variant 1: also patch-able (not fundamental) — here's the simple problem

- **What Variant 1 intends**: for each condition (phase × duration), run episodes that *start normally*, then **trigger corruption when the episode enters that phase**.
- **What happens now**: when you begin a new condition, you **don't reset the envs**; you start stepping from whatever state each parallel env is already in. Many envs are mid-episode, so the "entry into reach/grasp/…" may have already happened **before** you started measuring.
  - That's why even `reach` (which should happen in every episode) has low `corruption_episode_rate` in your saved results.

So the Variant 1 issue is: **condition runs aren't aligned to episode starts**, which breaks the semantics of "trigger on entering phase."