Propagating Change Through World-Space Radiance Caches for Dynamic Global Illumination

Motivation. In the real world, light bounces around: Sunlight hits the floor, which reflects light onto walls, filling the room with warm indirect light. A computer simulates this *global illumination* by following random paths of light (path tracing), producing some of the most realistic images in computer graphics. However, interactive programs can only afford to follow a handful of rays per pixel each frame, which is far too few for a clean image. To account for this, modern real-time renderers reuse their previously calculated work through various techniques. Screen space resampling techniques like ReSTIR [1,2] reuse promising paths of light and share them between nearby pixels, as well as considering previously computed values from past frames. World-space caching (probe grids [3] or spatial hash tables [4]) store lighting in cells linked to the scene itself to be looked up and used in future calculations. Both techniques rely on history, but history breaks down when dynamic updates occur: when a light turns on, an object moves and casts a new shadow, or the sun repositions throughout the day. Deciding what and how much to trust from the past is "a key challenge" many methods of solving "rely on heuristics" that require "careful tuning" [5]. My research asks how every cell in a world-space cache, even ones that sparsely receive new samples, can quickly learn that its past is stale, without spending extra rays.

The Gap. Most world-space caches blend new samples into a running average over some history length N. This is like a photographer setting their shutter speed, where a long exposure is smooth but smears anything that moves, and a short shutter is crisp but noisy. EA's surfel-based GI [6] takes an adaptive approach where each surfel tracks a short-term mean and variance, which it uses to adjust its long-term blend factor so surfels with higher variance request more rays. Even with this elegant model, surfels detect change only from their own samples, so cells that receive fewer samples, or are lit indirectly through other cells, still lag behind. Screen-space filters like A-SVGF [9] detect stale history with temporal gradients, and ADGI [7] extends this idea to probe grids, but relies on dedicated pilot rays.

Proposed Approach. When a cache cell traces a ray that hits another cached surface, it already reads that surface's stored lighting to use for further bounces. That lookup can also report if the cell hit has recently changed. **I propose that cache cells share evidence of change, so each cell learns its lighting is stale from the other cells it gathers light from, instead of waiting for its own samples to drift.** A cell needs many noisy samples to notice its own average has shifted, but a single ray landing on a flagged cell is enough to learn that something it depends on has changed. If a light turns on, the cells lit directly react first, but with shared evidence, the cells that receive bounced light from those cells can also react within frames rather than having to discover the change themselves bounce by bounce.

Preliminary Work. My voxel renderer (Vulkan with hardware ray queries) maintains a radiance cache on every voxel face, a perfect example of hash-grid caches. I tested rotating the sun's azimuth 90° in a single frame and compared two fixed history lengths (Figure 1). With $N = 8$, the cache responds quickly but plateaus at a 6.6% noise floor, flickering about six times as much as $N = 64$. $N = 64$ drops to a smoother 2.8% error, but needs over 100 frames to close 95% of that gap. If every face updated equally, each halving of its remaining error would take the same number of frames. Instead, it closed the first half in just 9 frames, then slowed and took 50 more frames to go from 90% to 95%, likely because sparsely sampled and bounce-lit faces lag behind directly lit ones.

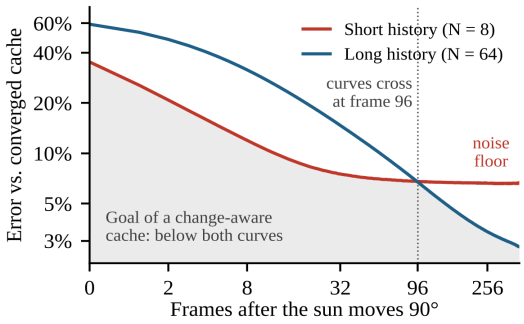


Figure 1. Preliminary measurement in my voxel renderer: relative L1 error after the sun swings 90° in one frame, for two fixed history lengths (one viewpoint, one event).

Aim 1: A Benchmark for Dynamic Lighting. The current benchmark, TauBench [8], animates the camera and objects but removed its dynamic lights and scores over average image quality rather than recovery. I will create an open set of scripted events (a light switching on, a light moving between rooms, a door opening, an occluder moving, and the camera revealing unseen areas) and record error over time against a converged path-traced reference. I will report recovery time, flicker on static frames,

milliseconds per frame, and memory for fixed-cap caching, a surfel-style per-cell variance estimator [6], and a history-free cache. I will test which cheap signals (sample count, cell age, per-cell variance, or change flags from other cells) best predict how stale a cell is. Success is a benchmark that separates these methods and that others can reuse.

Aim 2: Propagating Change Through the Cache. Using a surfel-style per-cell estimator [6] as a starting point, each cell will raise a *change* flag when its short-term statistics deviate from its long-term mean. Cells will read these flags wherever their rays land and from spatial neighbors, and a cell that gathers light from a changed cell will shorten its history and request more rays before its own statistics react. The risk with this technique is raising false flags that cascade through a static scene, so flags will be weighted by each hit's contribution, decay with every bounce, and require a threshold that I will tune using the benchmark framework created in Aim 1. The goal is faster recovery than per-cell adaptation (especially in low-sample and indirectly lit cells), at equal stability and frametime, landing in Figure 1's shaded region.

Aim 3: Warm-Starting Newly Visible Pixels. Multi-Layer Reservoir Splatting [5] recovers surfaces that were briefly hidden, but its authors warn that it "operates entirely in screen space" and that regions "visible for the first time may initially exhibit higher noise." After Aim 2 makes the cache trustworthy, I will let newly visible pixels seed their reservoirs from cache cells whose change flags have settled rather than starting from nothing. This is a stretch goal connecting my cache to the broader ReSTIR literature.

Resources and Timeline. I will prototype in my voxel engine on an RTX-class GPU, then validate on standard triangle scenes in NVIDIA's open-source Falcor framework. Year 1 will be spent creating the benchmarks and baselines. Year 2 will focus on developing the change-aware cache, and Year 3 on warm-starting and triangle-scene validation. Results will target venues such as High-Performance Graphics and SIGGRAPH. My scope will be limited to diffuse indirect light, the case most caches target. Even if Aim 2 falls short, the benchmarks and signal study will still show the field where and why current caches fail.

Intellectual Merit

Lighting changes are information that can flow through a cache rather than a discovery each cell must make on its own. I contribute (1) the first open benchmark measuring how fast real-time renderers recover from lighting events, (2) an experimental investigation into which signals actually indicate stale lighting, and (3) a cache that propagates evidence of change along the same paths light already takes. The concept wouldn't be tied to my renderer, as any cache reading other cells' lighting at ray hits, like probes, surfels, or hash grids, could share change evidence this way. This work ties together two streams of research that usually develop separately, screen-space sampling and world-space caching, through a shared question: when is the past still trustworthy?

Broader Impacts

Better light reuse means fewer rays to achieve the same image and enables realistic lighting on consumer laptops and consoles, not just high-end GPUs. It also reduces energy consumption per rendered frame. The same techniques power architectural visualization, film previsualization, and simulated worlds used to train robots and computer vision, which require light to react properly to an opening door. To ensure others can repeat and build upon my work, I will release benchmark scenes, evaluation scripts, and cache implementations as open source for researchers and engine developers. I will use my teaching experience to create tutorials and explainer videos that build from basic light caching to this research, helping students understand and imagine their own systems and solutions for coupling world-space and screen-space.

References: [1] Bitterli et al., ACM Trans. Graph. (2020). [2] Ouyang et al., Comput. Graph. Forum (2021). [3] Majercik et al., J. Comput. Graph. Tech. (2019). [4] Boissé et al., AMD Tech. Report (2022). [5] Hong et al., ACM SIGGRAPH Conf. Papers (2026). [6] Halen and Brinck, SIGGRAPH Advances in Real-Time Rendering (2021). [7] Datta et al., arXiv (2023). [8] Yazdi et al., arXiv (2023). [9] Schied et al., Proc. ACM Comput. Graph. Interact. Tech. (2018).