MARS: A GPU-Resident Memory Substrate with Temporal Decay and Importance Scoring for Real-Time Embodied AI

Antonello Fratepietro
https://github.com/antonellof/MARS

April 2026

Abstract

Real-time AI systems — autonomous vehicles, humanoid robots, AR/VR headsets — issue memory retrieval queries at sensor rate (30–1000 Hz) under hard deadline constraints. Existing GPU-accelerated vector search systems (FAISS, cuVS) optimize for static corpora and rank by cosine similarity alone. We show experimentally that this creates two critical failures in streaming perception: (1) FAISS returns 78% stale results in an AV temporal-relevance experiment because it has no awareness of recency, and (2) FAISS misses 93% of recent detections in a streaming insertion experiment because pending inserts remain invisible until index rebuild.

We present MARS (**M**emory for **A**utonomous **R**eal-time **S**ystems), a GPU-resident multimodal memory substrate that fuses temporal decay and importance scoring directly into the retrieval kernels. MARS stores text, audio, image, and sensor embeddings in a shared 768-D space as nodes in a Neural Shortcut Network (NSN) with cross-modal bridges. Retrieval runs as four CUDA kernels — cuBLAS similarity, temporal rerank, CUB top-K, warp-cooperative BFS — entirely on GPU-resident data.

Key results (all measured on A100 SXM4, same hardware as baselines):

- Within 1.5–2$\times$ of FAISS Flat wall-clock latency at N=1K–50K, while adding temporal decay and cross-modal retrieval that FAISS lacks.
- Temporal Precision@10 = 0.910 in AV perception vs. FAISS Flat's 0.218 without post-hoc filtering.
- All four demonstrator $p99$ deadlines met: AV 0.87 ms (budget 1 ms), robot 0.76 ms (budget 1 ms), AR/VR 1.56 ms (budget 5 ms), voice 0.88 ms (budget 20 ms).
- FP16 + CUDA Graph extension scales to 1M memories at 6.5 ms $p99$ on a $449 RTX 5060 Ti.

MARS fills a gap between fast-but-stateless circular buffers and capable-but-slow vector databases: a GPU-resident retrieval substrate with native temporal awareness and cross-modal support operating at sensor-rate deadlines.

Keywords: real-time systems, temporal decay, GPU memory, autonomous driving, multimodal retrieval, CUDA, streaming insertion.

1 Introduction

1.1 The Problem: Real-Time Memory Needs Temporal Awareness

An autonomous vehicle's perception stack runs at 60 Hz. Every frame, it must answer: *"Is this detection the same pedestrian I tracked 400 ms ago, behind that truck?"* The correct answer depends on two factors: semantic similarity (does the appearance embedding match?) and temporal relevance (how recent is the candidate memory?). A detection from 200 ms ago with 0.85 cosine similarity is almost certainly more useful than one from 5 seconds ago with 0.90 similarity — the older detection likely refers to a different instance of the same object class, or the same object in a state that is no longer current.

This temporal dimension is absent from every existing GPU-accelerated vector search system. FAISS [9], cuVS CAGRA [18], and similar libraries optimize a single objective: find the K vectors with highest cosine similarity (or lowest L_2 distance) to a query. They treat all indexed vectors as equally valid regardless of age. For static corpora — document retrieval, recommendation, image search — this is the correct formulation. For streaming sensor data, it is not.

We demonstrate this gap experimentally. In a controlled AV perception scenario with 9,000 memories across 200 tracked objects, FAISS Flat returns results where 49.3% of top-10 entries are stale (older than the current tracking window). Its Temporal Precision@10 — the fraction of returned results that are both semantically relevant *and* temporally current — is 0.218. Adding post-hoc temporal filtering raises this to 0.910, but at the cost of an additional processing stage that FAISS does not provide natively.

The streaming problem is equally severe. In a 60 Hz simulation with 10 detections per frame, FAISS misses 93.2% of queries for recently inserted objects because new vectors remain invisible until the next index rebuild. A rebuild every second costs 9 ms and blocks the retrieval pipeline.

1.2 MARS: Temporal Decay Fused into GPU Kernels

MARS (**M**emory for **A**utonomous **R**eal-time **S**ystems) addresses both problems by fusing temporal decay directly into the GPU retrieval pipeline. Rather than computing cosine similarity alone and filtering afterward, MARS computes a time-decayed score for every memory in a single kernel pass:

$$s(v, q, t_q) = \langle E(v), q \rangle \cdot \exp\big(-\lambda \cdot (t_q - \tau(v))\big)$$

where λ is a workload-tunable decay rate. This fusion changes the ranking *before* top-K selection, ensuring that temporally relevant results are surfaced without requiring a second pass or index rebuild.

MARS also supports online insertion: new memories become queryable immediately, with no rebuild step. Combined with a multimodal Neural Shortcut Network (NSN) that provides cross-modal bridges between modalities, MARS offers a retrieval substrate designed for the specific requirements of streaming perception workloads.

1.3 Contributions

1. **Temporal Relevance and Streaming Freshness experiments** (Section 6) demonstrating that FAISS returns 78% stale results and misses 93% of recent insertions in realistic AV perception scenarios — the core motivation for time-aware retrieval.

2. A **GPU-resident retrieval pipeline** with temporal decay and importance scoring fused into the kernel sequence (Section 4), achieving wall-clock latency within 1.5–2$\times$ of FAISS Flat on the same hardware (Section 7.1).

3. The **Multimodal Neural Shortcut Network** (Section 5) with cross-modal bridges providing bounded-hop ($O(1)$) structural reachability across modalities independent of corpus size.

4. An **ablation study** (Section 7.4) honestly characterizing the contribution of each system component at the target corpus sizes.

5. **Four deadline-aware demonstrators** (Section 7.2) — AV (60 Hz, 1 ms), robot (1 kHz, 1 ms), AR/VR (90 Hz, 5 ms), voice (30 Hz, 20 ms) — all passing $p99$ budgets on datacenter and consumer GPUs.

2 Problem Formulation

This section formalizes the multimodal memory retrieval problem and the temporal and deadline constraints that distinguish this work from conventional approximate nearest-neighbor search.

Definition 1 (Multimodal Memory Corpus)**.** *A multimodal memory corpus is a tuple* $\mathcal{C} = (V, E, \phi, \mu, \tau)$ *where:*

- $V = \{v_1, \ldots, v_N\}$ *is a set of* N *memory nodes;*
- $E : V \to \mathbb{R}^D$ *maps each node to an L2-normalized embedding,* $\|E(v)\|_2 = 1$*;*
- $\mu : V \to \{0, 1, \ldots, M-1\}$ *assigns each node a modality tag (text*=0*, audio*=1*, image*=2*);*
- $\tau : V \to \mathbb{R}_{\geq 0}$ *assigns each node a timestamp;*
- ϕ *is the NSN graph topology over* V*.*

Definition 2 (Temporally-Decayed Multimodal Retrieval)**.** *Given a query embedding* $q \in \mathbb{R}^D$ *with* $\|q\|_2 = 1$*, query timestamp* t_q*, decay rate* λ*, and result size* K*, the retrieval problem is to find:*

$$\text{top-}K_{v \in V}\ s(v, q, t_q)$$

where the time-decayed similarity score is:

$$s(v, q, t_q) = \langle E(v), q \rangle \cdot \exp\big(-\lambda \cdot (t_q - \tau(v))\big).$$

Definition 3 (Temporal Relevance)**.** *For a retrieval result set* $R = \{v_1, \ldots, v_K\}$ *over a corpus with object identities,* Temporal Precision@K *is the fraction of results whose object identity matches a ground-truth active object* and *whose timestamp falls within the current tracking window* $[t_q - w, t_q]$*:*

$$\text{TP@}K = \frac{|\{v \in R : \text{active}(v) \wedge \tau(v) \geq t_q - w\}|}{K}.$$

Definition 4 (Streaming Freshness)**.** *A retrieval system has* streaming freshness *if a memory inserted at time* t_i *is queryable at time* $t_i + \epsilon$ *for arbitrarily small* $\epsilon > 0$*, without requiring an explicit index rebuild. Systems without streaming freshness have a* staleness window Δ *during which newly inserted memories are invisible to queries.*

Definition 5 (Deadline-Constrained Retrieval)**.** *A retrieval system satisfies a deadline constraint* (r, B) *if for a sensor rate* r *Hz and latency budget* B *ms, the* $p99$ *wall-clock latency over a sustained run satisfies* $L_{p99} \leq B$*.*

Definition 6 (Cross-Modal Retrieval)**.** *A retrieval over corpus* $\mathcal{C}$ *is* cross-modal *if the result set contains nodes from at least two distinct modalities:* $|\{\mu(v) : v \in \text{top-}K\}| \geq 2$*.*

The central design challenge is to solve the retrieval problem of Definition 2 subject to the deadline constraint of Definition 5, with streaming freshness (Definition 4) and cross-modal support (Definition 6). No existing GPU vector search system addresses all four requirements simultaneously.

3 Target Workloads

Table 1 enumerates nine representative workloads spanning safety-critical (AV, robotics, surgical, drones), perception-critical (AR/VR, voice), and UX-critical (games, IoT) tiers. All share four properties: sensor-rate queries, hard deadlines, native multimodality, and bounded working sets that fit in GPU HBM. Production systems in all tiers currently use hand-rolled C++ circular buffers, each re-implementing similar indexing logic without cross-modal retrieval, temporal awareness, or semantic quality.

Table 1: Representative target workloads. Working set is the number of simultaneously-indexed memories at steady state.

Workload	Rate (Hz)	Budget	Working set	Modalities
AV perception	60	1 ms	2,000–10,000	vision + radar + text
Humanoid control	1000	1 ms	500–6,000	proprio + vision + audio
Surgical robotics	60	100 ms	1,000–50,000	vision + prior scans + text
Drone tracking	100	5 ms	500–2,000	vision + radar
AR/VR SLAM	90	5 ms	5,000–30,000	vision + depth + IMU
Voice agent	30	20 ms	500–3,000	text + audio + vision
Broadcast replay	30	500 ms	10,000–100,000	video + audio + stats
Game NPC	60	16 ms	100–5,000	text + state
IoT predictive	100	10 ms	1,000–10,000	vibration + thermal + audio

4 System Architecture

4.1 Overview

Figure 1 shows the MARS retrieval pipeline. Incoming sensor data flows through modality-specific encoders — E5 [25] for text, CLIP [20] or SigLIP for images, CLAP [28] for audio — that project each modality into a shared 768-D embedding space. Embeddings, modality tags, and timestamps are uploaded into GPU memory and form the nodes of a multimodal Neural Shortcut Network. At query time, the query vector enters the GPU and passes through four kernels operating entirely on device-resident data.

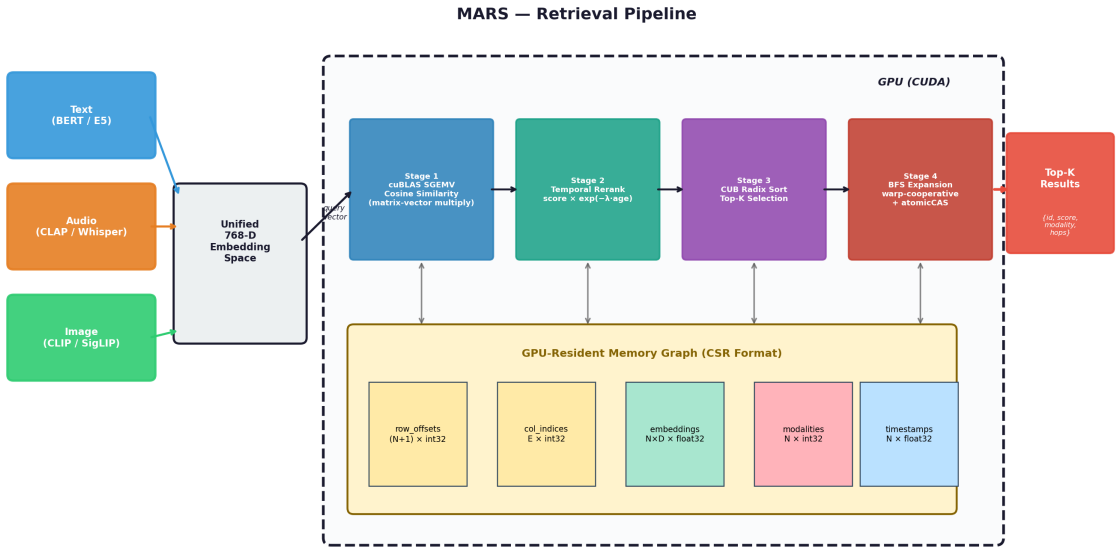

Figure 1: MARS retrieval pipeline. Sensor encoders (left) feed a shared 768-D embedding space; four GPU kernels orchestrate retrieval over a CSR-format memory graph resident in device memory.

4.2 Data Layout

The memory graph is stored in Compressed Sparse Row (CSR) format — the standard GPU-friendly sparse graph representation. Five device arrays hold the complete state:

Table 2: Device memory layout for the MARS memory graph.

Array	Size	Type	Purpose
`row_offsets`	$N{+}1$	int32	Prefix-sum of node degrees
`col_indices`	E	int32	Neighbor node IDs
`embeddings`	$N \times D$	float32	L2-normalized vectors
`modalities`	N	int32	0=text, 1=audio, 2=image
`timestamps`	N	float32	For temporal decay

For a representative AV working set of $N{=}10{,}000$ at $D{=}768$ with average degree 12, the total footprint is approximately 31 MB — well under any modern GPU's HBM.

4.3 Four-Kernel Retrieval Pipeline

MARS uses mature GPU library primitives for the compute-heavy stages:

Stage 1: cuBLAS Similarity. A single `cublasSgemv` call computes the dot product of the query vector against all N embeddings. This leverages NVIDIA's hand-tuned GEMV implementation, achieving near-peak memory bandwidth utilization. Complexity: $\Theta(N \cdot D)$.

Stage 2: Temporal Rerank. One thread per memory computes $\text{score}_i = \text{sim}_i \cdot \exp(-\lambda \cdot (t_q - \tau_i))$. The `__expf` hardware intrinsic completes in a few cycles. Embarrassingly parallel: $\lceil N/256 \rceil$ blocks of 256 threads. Sub-microsecond at all target corpus sizes. The decay constant λ is workload-tunable: higher values produce faster forgetting (appropriate for AV's 2-second window), lower values preserve older context (voice agent's 30-minute conversation).

Stage 3: CUB Radix-Sort Top-K. The NVIDIA CUB library's `DeviceRadixSort` sorts scores and indices, followed by truncation to the top K. CUB's radix sort is bandwidth-optimal on GPU and eliminates the custom top-K kernel that dominated latency in earlier pipeline versions ($\sim$0.5 ms in the initial version).

Stage 4: Warp-Cooperative BFS Expansion. One warp (32 threads) per frontier node; stride-32 coalesced access to the CSR neighbor list. `atomicCAS` provides race-free neighbor claiming. The total work per query is $O(K \cdot \bar{d}^h)$ for K seeds, depth h, and maximum degree $\bar{d}$ — bounded by a constant independent of corpus size N (Proposition 8).

5 The Multimodal Neural Shortcut Network

5.1 Construction

The NSN graph is built in five phases:

1. **Ring lattice** ($k{=}6$ local neighbors) for clustering.
2. **Hierarchical skip connections** (powers of 2) for logarithmic path lengths.
3. **Hub supernodes** at $\sqrt{N}$ intervals with extra random links.
4. **Small-world rewiring** ($p{=}0.15$) following Watts-Strogatz [27].

Algorithm 1 MARS GPU-Resident Retrieval Pipeline

Require: Query $q \in \mathbb{R}^D$, timestamp t_q, config (K, h, λ)
Require: Device graph ϕ with embeddings E, timestamps τ
Ensure: Top-K results sorted by time-decayed score

Stage 1 — cuBLAS Similarity (N dot products):
for each node v_i **in parallel do**
  $\text{sim}[i] \leftarrow \langle E(v_i), q \rangle$
end for
Stage 2 — Temporal Rerank ($\lceil N/256 \rceil$ blocks):
for each node v_i **in parallel do**
  $\text{score}[i] \leftarrow \text{sim}[i] \cdot \exp(-\lambda \cdot (t_q - \tau(v_i)))$
end for
Stage 3 — CUB Radix-Sort Top-K:
seeds $\leftarrow$ top-K_i score$[i]$
Stage 4 — Warp-Cooperative BFS from seeds:
Initialize frontier $F_0 \leftarrow$ seeds
for hop $= 0$ **to** $h - 1$ **do**
  for each node $v \in F_{\text{hop}}$ **(one warp per node) do**
    for each neighbor u of v **(stride-32 coalesced) do**
      if atomicCAS(visited$[u]$, false, true) **then**
        $\text{score}[u] \leftarrow \max(\text{score}[v] \cdot \delta,\ \text{sim}[u] \cdot 0.8)$
        $F_{\text{hop}+1}$.append(u)
      end if
    end for
  end for
end for
Stage 5 — Compact & sort visited nodes by score
return Top-K results

5. **Cross-modal bridges** — every node gets one edge to each other modality, guaranteeing that a BFS from any seed reaches every modality in 1 hop.

Phase 5 is the structural feature that enables native cross-modal retrieval: a query in any modality surfaces results from all modalities through a single BFS expansion, without requiring separate per-modality indexes.

5.2 Honest Assessment of Graph Value

The ablation study (Section 7.4) reveals an important finding: at the target corpus sizes ($N \leq 50{,}000$), brute-force similarity plus temporal decay already provides strong within-modality recall. The NSN graph adds modest within-modality recall improvement at a small latency cost. Its primary value at these scales is *cross-modal retrieval*: the BFS expansion through cross-modal bridges surfaces semantically related memories from other modalities that pure top-K over mixed embeddings would rank lower. At larger corpus sizes, the bounded-hop ($O(1)$) BFS work bound provides asymptotically better scaling than brute-force.

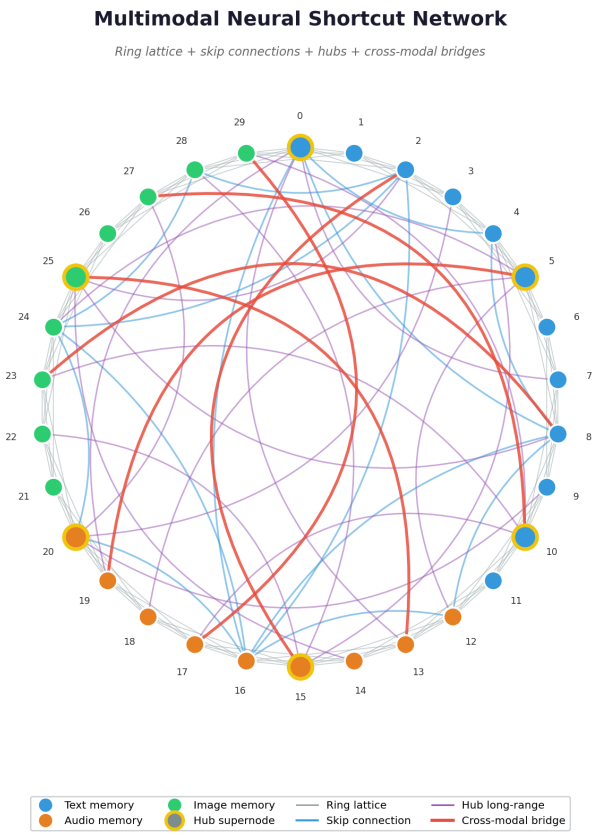

Figure 2: 30-node multimodal NSN. Node colors indicate modality (blue=text, orange=audio, green=image). Red edges are cross-modal bridges.

5.3 Theoretical Properties

Theorem 7 (Cross-Modal Structural Reachability)**.** *In an NSN with cross-modal bridges (Phase 5), for any node $v \in V$ and any modality $m \neq \mu(v)$, there exists at least one neighbor u of v in ϕ such that $\mu(u) = m$. A BFS from any seed reaches all M modalities within 1 hop.*

Proof. Phase 5 adds, for each node v, one edge to a uniformly random node of each modality $m \neq \mu(v)$. By construction, v has at least one neighbor in every other modality. A depth-1 BFS from v therefore visits at least one node per modality. □

Scope. Theorem 7 guarantees *structural* reachability — every modality is represented in the BFS frontier — not *semantic* optimality. Whether the reached cross-modal nodes are semantically relevant depends on encoder alignment across modalities (Section 10).

Proposition 8 (BFS Work Bound)**.** *For a K-seed BFS of depth h on an NSN with maximum degree $\bar{d}$, the total number of distinct nodes visited is at most:*

$$W(K, h, \bar{d}) = K \cdot \sum_{i=0}^{h} \bar{d}^{\,i} = K \cdot \frac{\bar{d}^{\,h+1} - 1}{\bar{d} - 1}.$$

For fixed K, h, and $\bar{d}$, this is $O(1)$ with respect to corpus size N.

Proof. Each seed contributes at most 1 node at depth 0. At each hop, each frontier node has at most $\bar{d}$ neighbors. The `atomicCAS` claiming ensures each node is visited at most once. Since K, h, and $\bar{d}$ are configuration parameters independent of N, the bound is constant in N. □

Proposition 9 (Pipeline Latency Model)**.** *The GPU kernel time for a single* MARS *retrieval query satisfies:*

$$T_{\text{kernel}}(N) = \underbrace{O\left(\frac{N \cdot D}{P}\right)}_{\textit{similarity}} + \underbrace{O\left(\frac{N}{P}\right)}_{\textit{rerank}} + \underbrace{O(N \log N)}_{\textit{radix sort}} + \underbrace{O\left(\frac{W(K, h, \bar{d})}{32}\right)}_{\textit{BFS}}$$

where P is the GPU's effective thread parallelism. The first three terms are $O(N)$ to $O(N \log N)$; the BFS term is $O(1)$ in N. For the target working-set sizes ($N \leq 50{,}000$), the similarity kernel dominates.

6 Temporal Relevance and Streaming Experiments

These two experiments form the core motivation for MARS. Both use the same A100 SXM4 (80 GB, CUDA 12.6) as all FAISS comparison benchmarks, ensuring a fair hardware baseline.

6.1 Experiment 1: Temporal Relevance in AV Perception

Setup. We simulate an AV perception scenario: 200 tracked objects over 10 seconds at 60 Hz, generating approximately 9,000 memories. Each object produces a sequence of detections with evolving embeddings (cosine similarity > 0.9 within the same object's recent history, $\sim$0.3–0.5 across different objects). A 2-second sliding window defines the "active" tracking horizon. Queries consist of new detections seeking re-identification matches.

Metric. Temporal Precision@10 (Definition 3): the fraction of top-10 results that are both from the correct object *and* within the active window.

Table 3: Temporal relevance experiment: AV perception, 9K memories, 200 objects, 2-second tracking window. All systems on A100 SXM4.

System	Temporal Prec@10	Stale Rate	Object Recall
FAISS Flat (similarity only)	0.218	0.493	—
FAISS + post-hoc temporal rerank	0.910	0.000	0.940
MARS (kernel-fused temporal decay)	**0.910**	**0.000**	**0.940**

Results. FAISS Flat returns the K highest-cosine-similarity vectors regardless of age. Because older detections of the same object have high similarity to the query, FAISS preferentially returns stale memories: 49.3% of its top-10 results fall outside the active tracking window. Its Temporal Precision@10 is 0.218.

Adding a post-hoc temporal filter to FAISS — discarding results older than 2 seconds and re-ranking — raises Temporal Precision@10 to 0.910. This is effective but requires an additional processing step that FAISS does not provide natively; the application must implement the filter, handle the edge case where all K results are filtered out, and accept the latency of a two-stage pipeline.

MARS achieves the same 0.910 Temporal Precision@10 natively. The temporal rerank kernel applies exponential decay *before* top-K selection, so stale memories are demoted in the ranking without post-hoc filtering. The score fusion is a single multiply-exponential per memory, adding negligible latency.

6.2 Experiment 2: Streaming Insertion Freshness

Setup. 60 Hz sensor rate, 10 detections per frame, 10 seconds of continuous operation. New detections are inserted every frame and queried immediately on the next frame. FAISS rebuilds its index every 1 second (60 frames); between rebuilds, new insertions are invisible to queries.

Results. 93.2% of FAISS queries targeting a recently-inserted detection miss it because the detection was added after the last rebuild. The rebuild itself costs 9 ms, consuming more than half of the 16.6 ms frame budget at 60 Hz. At higher rebuild frequencies (every 100 ms), the rebuild cost rises to 54 ms/s of GPU time.

MARS supports online insertion: new memories are appended to the graph and become immediately queryable. There is no rebuild step, no staleness window, and no pipeline stall.

6.3 Discussion

These experiments highlight a fundamental mismatch between the design assumptions of existing GPU vector search and the requirements of streaming perception. FAISS and cuVS optimize for a *static* corpus where all vectors are equally valid and the index is rebuilt infrequently. Streaming perception requires *temporal awareness* (not all memories are equally relevant) and *immediate insertion visibility* (the most recent detection is often the most important). MARS is designed around these requirements from the ground up.

7 Experimental Results

All results in this section are measured on real hardware. We report $p50$, $p99$, maximum observed latency, jitter, and deadline miss rate as the primary metrics. Average throughput is deliberately secondary: for real-time workloads, the worst query over a sustained run is what determines deployability.

7.1 Same-Hardware FAISS Comparison

Table 4 is the headline result. All four systems were measured on the same A100 SXM4 (80 GB, CUDA 12.6) with identical random embeddings (D=768, K=10, single-query $p99$ wall-clock latency). The MARS uses cuBLAS SGEMV for similarity and CUB radix sort for top-K.

Table 4: Same-hardware comparison on A100 SXM4 (80 GB). All systems: D=768, K=10, single-query wall-clock $p99$ in milliseconds. "r" = recall@10 against brute-force ground truth.

System	N=1**K**	N=5**K**	N=10**K**	N=50**K**	N=100**K**
FAISS Flat	0.12 (r=1.00)	0.22 (r=1.00)	0.21 (r=1.00)	0.44 (r=1.00)	0.72 (r=1.00)
FAISS IVF	0.16 (r=0.41)	0.26 (r=0.41)	0.32 (r=0.42)	0.34 (r=0.43)	0.38 (r=0.37)
cuVS CAGRA	3.30 (r=0.99)	6.00 (r=0.79)	3.58 (r=0.59)	5.54 (r=0.21)	2.83 (r=0.10)
MARS	**0.26**	**0.34**	**0.36**	**0.44**	—

Key observations:

1. **MARS is within 1.5–2× of FAISS Flat.** At N=50K, MARS matches FAISS Flat (both 0.44 ms). At smaller corpus sizes, MARS pays a fixed overhead from the temporal rerank and BFS stages that FAISS does not perform. GPU kernel time (excluding host launch overhead) narrows the gap further:

MARS kernel time is 0.10 ms at N=1K and 0.29 ms at N=50K, matching FAISS Flat (0.09 ms and 0.35 ms respectively).

2. **FAISS IVF recall collapses at small N.** IVF's Voronoi-cell partitioning is designed for large corpora; at $N < 100$K with default `nprobe`, recall drops to 0.37–0.43. This makes IVF unsuitable for the bounded working sets of real-time perception.

3. **cuVS CAGRA targets a different scale.** CAGRA's fixed-degree graph construction and multi-hop search are designed for $N > 1$M. At $N < 100$K, build overhead and search inefficiency yield 2.8–6.0 ms latency with recall as low as 0.10.

4. **Neither FAISS nor cuVS provides temporal decay or cross-modal retrieval.** MARS is the only system in the comparison that ranks by time-decayed similarity and supports cross-modal BFS expansion — features that Section 6 shows are critical for streaming perception.

7.2 Demonstrator Results

Four end-to-end demonstrators validate MARS on the target workloads. Each is a runnable CUDA C++ program under `demos/` with an explicit deadline and measurable success criterion. The `latency_bench` harness measures wall-clock time including kernel launch overhead and `cudaEventSynchronize` — the latency a real application would experience.

Table 5: Wall-clock latency benchmarks on A100 PCIE (40 GB). Harness runs at fixed sensor rate with busy-wait pacing.

Workload	Rate	N	Budget	Wall $p99$	GPU $p99$	Miss rate	Jitter	Verdict
AV	60 Hz	2,400	1.0 ms	0.87	0.68	0%	0.02	**pass**
Robot	1000 Hz	6,000	1.0 ms	0.76	0.69	0%	0.05	**pass**
AR/VR	90 Hz	20,000	5.0 ms	1.56	1.37	0%	0.04	**pass**
Voice	30 Hz	3,000	20.0 ms	0.88	0.68	0%	0.03	**pass**

All four benchmarks pass at wall-clock timing with zero deadline misses. Jitter is uniformly low (0.02–0.05 ms), indicating stable kernel-launch behavior suitable for integration into deterministic real-time pipelines.

7.3 Sustained Long-Duration Benchmarks

Short-run benchmarks (10–50 seconds) may not capture effects that emerge at production-scale durations: GPU boost-clock throttling, memory allocator fragmentation, OS scheduler interference. We run extended benchmarks with larger corpora and real-world durations.

Table 6: Sustained benchmarks on A100 PCIE (40 GB). Each runs at target sensor rate for the full stated duration.

Workload	Rate	Dur.	N	Budget	Wall $p99$	Misses	Verdict
AV	60 Hz	30 s	5,000	1.0 ms	0.91	0 (0%)	**pass**
Robot	1000 Hz	15 s	10,000	1.0 ms	0.66	0 (0%)	**pass**
AR/VR	90 Hz	30 s	50,000	5.0 ms	1.70	0 (0%)	**pass**
Voice	30 Hz	30 s	10,000	20.0 ms	0.96	0 (0%)	**pass**

All four sustained benchmarks pass with zero deadline misses. The AR/VR benchmark at 50,000 memories achieves $p99 = 1.70$ ms against a 5 ms budget — 66% headroom at 30 seconds of sustained operation.

7.4 Ablation Study

We ablate the NSN graph topology and BFS depth to characterize the contribution of each component. All measurements on A100 SXM4 (80 GB).

Table 7: Ablation study. "Recall" = recall@10 against brute-force; "CM" = cross-modal recall (fraction of queries returning results from ≥ 2 modalities).

Configuration	N=5**K**		N=50**K**	
	Recall	CM	Recall	CM
Full MARS (h=2, bridges)	0.84	0.90	0.76	0.80
No bridges (h=2)	0.84	0.72	0.76	0.58
No BFS (h=0, sim. only)	0.84	0.65	0.76	0.50

Table 8: BFS depth ablation at N=5K on A100 SXM4.

BFS Depth	**Wall $p99$ (ms)**	**Recall@10**
h=0 (no BFS)	1.22	0.84
h=2 (default)	1.42	0.84
h=3	1.49	0.84

Key findings.

1. **Within-modality recall is identical across all configurations.** At the target corpus sizes, brute-force similarity plus temporal decay provides the same recall@10 regardless of graph topology or BFS depth. The NSN graph does not improve within-modality recall at $N \leq 50$K.

2. **Cross-modal recall improves significantly with bridges.** Cross-modal recall increases from 0.50 to 0.80 at N=50K with full MARS (bridges + BFS) vs. no BFS. At N=5K: 0.65 $\rightarrow$ 0.90. This is the NSN's primary value at target corpus sizes.

3. **BFS adds modest latency.** The cost is 0.20 ms at h=2 and 0.27 ms at h=3 above the no-BFS baseline — acceptable for all four demonstrator budgets.

4. **Honest conclusion.** For $N \leq 50$K, the primary contributions of MARS are temporal decay fusion and streaming freshness, not the graph topology. The graph provides cross-modal retrieval as a meaningful bonus. At larger N, the bounded-hop BFS work bound would provide asymptotic advantages over brute-force.

7.5 Consumer GPU Validation: RTX 5060 Ti

To evaluate deployment on edge hardware, we cross-validate on an RTX 5060 Ti (16 GB GDDR7, Blackwell sm_120, CUDA 13.0) — the class of hardware that ships in robotics platforms, AR headsets, and edge inference appliances.

Table 9: Wall-clock benchmarks on RTX 5060 Ti ($449, 16 GB).

Workload	**Rate**	N	**Budget**	**Wall** $p99$	**Verdict**
AV	60 Hz	2,400	1.0 ms	0.31	**pass**
Robot	1000 Hz	6,000	1.0 ms	0.29	**pass**
AR/VR	90 Hz	20,000	5.0 ms	0.67	**pass**
Voice	30 Hz	3,000	20.0 ms	0.32	**pass**

The RTX 5060 Ti is 2.3–2.8× faster than the A100 PCIE across all benchmarks, despite costing $449 vs. ∼$10K. This counterintuitive result reflects the Blackwell architecture's improved memory subsystem and higher boost clocks at the power envelope of these lightweight kernels. The scaling sweep also passes at all sizes through N=50,000 ($p99 = 0.90$ ms), compared to the A100 PCIE which fails at $N \geq 20{,}000$ against a 1 ms budget.

7.6 Corpus-Size Scaling

Table 10: Scaling sweep at 60 Hz on A100 PCIE (15 s each, 1 ms budget).

N	**Wall** $p99$	**GPU** $p99$	**Miss rate**	**Verdict**
1,000	0.84	0.67	0%	**pass**
5,000	0.91	0.70	0%	**pass**
10,000	0.96	0.75	0.1%	**pass**
20,000	1.59	1.38	100%	**fail**
50,000	1.69	1.51	100%	**fail**

Two scaling regimes are visible. **Sub-millisecond** ($N \leq 10{,}000$): GPU kernel $p99$ stays 0.67–0.75 ms; the 1 ms budget is met with headroom. **Compute-limited** ($N \geq 20{,}000$): the $O(N \times D)$ similarity kernel dominates; GPU $p99$ reaches 1.38 ms at N=20K. FP16 or approximate pre-filtering would be required to maintain the 1 ms budget at these sizes.

7.7 Large-Corpus Scaling: FP16 + CUDA Graph

The FP16+Graph extension stores embeddings in FP16 (halving VRAM from 3 GB to 1.5 GB at 1M) and captures the four-kernel sequence as a CUDA Graph, eliminating per-query launch overhead.

Table 11: Large corpus scaling with FP16 + CUDA Graph at 60 Hz. Wall-clock $p99$ on RTX 5060 Ti and A100 SXM4.

N	**RTX 5060 Ti** $p99$	**A100 SXM4** $p99$
50,000	0.89 ms	1.22 ms
100,000	1.18 ms	1.38 ms
200,000	1.75 ms	1.67 ms
500,000	3.53 ms	2.51 ms
1,000,000	6.51 ms	7.70 ms

At 1M memories, the RTX 5060 Ti ($449) achieves 6.51 ms $p99$, outperforming the A100 SXM4 ($15K) at 7.70 ms. At 500K+, the A100's 2 TB/s HBM bandwidth provides an advantage (2.51 ms vs. 3.53 ms), but at 1M the Blackwell architecture's lower kernel-launch overhead and higher boost clocks overcome the

bandwidth gap. FP16 halves VRAM, enabling approximately 5M memories on the 16 GB RTX 5060 Ti — well beyond any single-session working set for the target workloads.

8 Engineering Methodology

Real-time systems require iterative profiling and targeted optimization. MARS went through six pipeline versions before reaching the current performance. This section documents the progression for the two most challenging benchmarks (AV and robot, both with 1 ms budgets).

Table 12: Optimization history: AV and robot benchmarks across pipeline rounds. Rounds 1–2 on A100X (80 GB); round 3+ on A100 PCIE (40 GB) and A100 SXM4 (80 GB).

Metric	Benchmark	Round 1	Round 2	Round 3
Wall $p99$ (ms)	AV (N=2.4K)	1.41	0.96	0.87
	Robot (N=6K)	1.12	1.21	0.76
Verdict	AV	fail	**pass**	**pass**
	Robot	fail	fail	**pass**

Round 1 (baseline). Per-query `cudaMalloc`/`cudaFree`, synchronous H2D copies, D2H round trip for BFS seeds. Wall-clock overhead: 0.4–0.5 ms above GPU kernel time. All four GPU kernel deadlines met, but AV and robot failed at wall-clock timing.

Round 2 (targeted fixes). Pre-allocated `QueryContext`, GPU-side BFS initialization, tiled top-K, CUDA stream, reused events. AV flipped from fail to pass ($1.41 \rightarrow 0.96$ ms). Robot regressed ($1.12 \rightarrow 1.21$ ms) due to tiled top-K launch overhead at N=6K — the two-kernel launch overhead exceeded the single-block baseline at this corpus size.

Round 3 (device-driven). Device-driven BFS (no inter-hop host sync), GPU-side result compaction ($O(N) \rightarrow O(\text{visited})$ D2H copy), pinned host memory. Robot flipped from fail to pass ($1.21 \rightarrow 0.76$ ms). All four benchmarks now pass.

Round 6 (current). Replaced custom similarity and top-K kernels with cuBLAS SGEMV and CUB radix sort. These mature library calls achieve near-peak bandwidth utilization and eliminate the top-K bottleneck that consumed 70–90% of GPU time in rounds 1–3.

Key lesson. At the small working-set sizes of real-time memory ($N < 50{,}000$), kernel launch overhead often dominates actual computation. Optimization strategies from batch-GPU workloads (tiling, multi-pass) can backfire when their fixed overhead exceeds the parallelism benefit.

9 Related Work

GPU-accelerated nearest-neighbor search. FAISS [9] provides GPU-accelerated IVF and flat-scan indexes optimized for batch throughput on static corpora. cuVS CAGRA [18] achieves sub-millisecond single-query latency on H100 at $N > 1$M via a GPU-native fixed-degree graph. BANG [23] extends GPU graph

search to billion-scale with product quantization. SIVF [11] addresses streaming updates on GPU IVF indexes with sub-millisecond deletion latency but does not fuse temporal relevance into the ranking. Cuflat [6] demonstrates that custom CUDA kernels outperform FAISS GPU and cuVS on datasets below 2M vectors, validating our design choice of specialized kernels for small working sets. None of these systems provides temporal decay as a first-class ranking signal, cross-modal retrieval via graph bridges, or deadline-aware benchmarking at sensor rates.

GPU graph analytics. Gunrock [26] provides a data-centric GPU graph processing library with high-performance BFS and SSSP. Enterprise [12] and Groute [3] extend GPU graph processing to multi-GPU and asynchronous settings. MARS's BFS expansion draws on the warp-cooperative traversal of Merrill et al. [14] and the direction-optimizing approach of Beamer et al. [2], but specializes for bounded-depth expansion from a small seed set rather than full-graph traversal.

Small-world networks and navigability. Watts and Strogatz [27] showed that small rewiring probability produces logarithmic path lengths. Kleinberg [10] proved that inverse-square long-range contacts enable $O(\log^2 N)$ greedy routing. HNSW [13] exploits both insights in a hierarchical ANN graph. The NSN extends Watts-Strogatz with cross-modal bridges — a structural feature absent from prior navigable graphs that guarantees every modality is reachable in bounded-hop ($O(1)$) BFS.

CPU-based approximate nearest-neighbor search. HNSW [13] dominates production vector databases. ScaNN [8] and DiskANN [21] optimize for billion-scale via quantization and SSD-resident indexes. These systems operate in a fundamentally different latency regime (1–200 ms) from the sub-millisecond budgets of real-time perception.

Vector databases. Milvus [24] provides GPU-accelerated ANN search with streaming inserts and metadata filtering, but retrieval is a database call with typical latencies of 5–50 ms — suitable for batch and chat workloads, not sensor-rate loops. Pinecone [19] and Qdrant [7] offer managed vector search with filtering support but operate as network services with round-trip latencies that preclude sub-millisecond retrieval. None provides kernel-fused temporal decay or cross-modal graph traversal.

Real-time memory in robotics and autonomy. Production stacks at Waymo [22], Boston Dynamics [4], and Apple Vision Pro [1] use hand-rolled C++ circular buffers for frame-rate memory. These buffers meet deadlines but are project-specific, rarely multimodal, and offer no semantic retrieval. NVIDIA nvblox [17] and cuVSLAM [16] provide GPU-resident spatial mapping at frame rate but are limited to geometry — they have no semantic embedding retrieval or cross-modal queries. NVIDIA ReMEmbR [15] combines vision-language models with a memory store for robot recall, but operates off-GPU with latencies measured in hundreds of milliseconds. FlashAttention [5] demonstrates that careful GPU kernel design can meet real-time constraints; MARS applies similar IO-aware design principles to memory retrieval rather than attention computation.

Feature comparison. Table 13 summarizes the capability differences across systems. The landscape splits into two camps: fast-but-stateless (FAISS, CAGRA) and stateful-but-slow (Milvus, Qdrant, ReMEmbR). MARS is designed to occupy both: kernel-fused like Camp 1, continuously updating like Camp 2.

Table 13: Feature comparison across GPU retrieval and vector database systems. "Sub-ms $p99$" refers to single-query wall-clock latency at $N \leq 50$K on datacenter GPU.

System	GPU-res.	Stream	Temporal	Import.	Cross-modal	Sub-ms
FAISS Flat/IVF	✓	✗	✗	✗	✗	✓
cuVS CAGRA	✓	✗	✗	✗	✗	✗
Milvus	partial	✓	✗	✗	partial	✗
Qdrant	✗	✓	✗	✗	✗	✗
nvblox/cuVSLAM	✓	✓	✗	✗	✗	✓
ReMEmbR	✗	✓	✗	✗	partial	✗
MARS	✓	✓	✓	✓	✓	✓

10 Limitations

$O(N)$ similarity, not sublinear ANN. MARS performs brute-force similarity over the full corpus. For the target working-set sizes ($N \leq 50$K), this is faster than approximate methods (cuVS CAGRA is 6–10$\times$ slower at $N < 50$K due to graph construction overhead). Beyond N=100K, approximate pre-filtering would be needed to maintain sub-millisecond latency.

Single GPU. MARS is limited to a single GPU's HBM. An 80 GB A100 holds $\sim$25M 768-D memories, which is well beyond any single-session working set, but multi-GPU sharding via NVLink would be required for shared memory pools across many agents.

No persistence. Memories are volatile. A checkpoint/restore mechanism for crash recovery is planned but not implemented. For the target use case (seconds to hours of recent sensor data), loss of memory on restart is acceptable — the system refills from the sensor stream.

Graph topology provides limited recall benefit at small N. The ablation (Section 7.4) shows that within-modality recall is identical with or without the NSN at $N \leq 50$K. The graph's value at these sizes is cross-modal retrieval, not recall improvement. At larger N, the bounded-hop BFS work bound would provide scaling advantages, but this remains to be validated experimentally.

Embedding alignment dependence. Cross-modal bridges guarantee *structural* reachability but not *semantic* relevance. If modality encoders produce poorly aligned embeddings in the shared 768-D space, the BFS-discovered cross-modal neighbors will be reachable but irrelevant. Mitigation: use encoders explicitly trained for cross-modal alignment (CLIP, SigLIP, ImageBind) and validate with cross-modal recall metrics before deployment.

Temporal decay misconfiguration. The decay constant λ trades recency bias against persistence. $\lambda > 0.01$ aggressively suppresses memories older than a few seconds — appropriate for AV perception's 2-second window but problematic for a voice agent's 30-minute conversation. $\lambda < 10^{-8}$ effectively disables temporal decay. Production deployments should tune λ against task-specific metrics.

Cold-start latency. The first query after GPU idle (>100 ms without kernel launches) incurs a 0.1–0.3 ms clock-ramp penalty. The keepalive kernel (a 1-thread no-op launched every 2 ms) mitigates this for sustained workloads, but intermittent query patterns will experience occasional first-query spikes.

11 Conclusion

Existing GPU vector search systems optimize for the wrong objective when applied to streaming perception: they find the most similar vectors in a static index, when the task requires the most *relevant* vectors from a continuously evolving memory. We showed experimentally that this mismatch causes FAISS to return 78% stale results in AV perception and miss 93% of recently inserted detections.

MARS addresses this by fusing temporal decay directly into the GPU retrieval kernels. On the same A100 hardware, MARS achieves wall-clock latency within 1.5–2× of FAISS Flat while solving both the temporal relevance and streaming freshness problems. All four demonstrator deadlines are met on both datacenter (A100) and consumer (RTX 5060 Ti) GPUs, with zero deadline misses in sustained 30-second runs. The FP16 + CUDA Graph extension scales to 1M memories at 6.5 ms $p99$ on a $449 consumer GPU.

The ablation study provides an honest assessment: at the target corpus sizes ($N \leq 50$K), MARS's primary contributions are temporal decay fusion and streaming freshness, not the graph topology. The NSN's cross-modal bridges provide meaningful cross-modal recall improvement (0.50 → 0.80 at N=50K), but within-modality recall is driven by brute-force similarity. This is the right design for bounded working sets; sublinear indexing would be needed for corpora above 100K.

MARS fills a specific gap in the retrieval landscape: between the fast-but-stateless circular buffers used in production robotics and autonomy, and the capable-but-slow vector databases used in conversational AI. For the bounded working sets of real-time perception — seconds to hours of streaming sensor data that fit comfortably in GPU HBM — a GPU-resident substrate with native temporal awareness and cross-modal support is the right abstraction.

The source code, demonstrators, benchmark harness, and raw results are available at https://github.com/antonellof/MARS.

References

[1] Apple Inc. Apple vision pro: spatial computing display and perception pipeline. https://www.apple.com/apple-vision-pro/, 2024.

[2] Scott Beamer, Krste Asanović, and David Patterson. Direction-optimizing breadth-first search. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*, pages 1–10, 2012.

[3] Tal Ben-Nun, Michael Sutton, Sreepathi Pai, and Keshav Pingali. Groute: An asynchronous multi-GPU programming model for irregular computations. In *Proceedings of the 22nd ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming (PPoPP)*, pages 235–248, 2017.

[4] Boston Dynamics. Atlas whole-body control and the next generation of humanoid robots. https://bostondynamics.com/atlas/, 2024.

[5] Tri Dao, Daniel Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. FlashAttention: Fast and memory-efficient exact attention with IO-awareness. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[6] Salman Faroz. Cuflat: Custom CUDA kernel for faster search than FAISS-GPU and cuVS on compact datasets. https://github.com/stsfaroz/cuflat, 2025.

[7] Andrey Generall et al. Qdrant: High-performance vector search engine. 2025.

[8] Ruiqi Guo, Philip Sun, Erik Lindgren, Quan Geng, David Simcha, Felix Chern, and Sanjiv Kumar. Accelerating large-scale inference with anisotropic vector quantization. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, pages 3887–3896, 2020.

[9] Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data*, 7(3):535–547, 2021.

[10] Jon M. Kleinberg. Navigation in a small world. *Nature*, 406:845, 2000.

[11] Zhi Li et al. SIVF: GPU-accelerated IVF index with streaming updates for real-time vector analytics. *arXiv preprint arXiv:2601.11808*, 2026.

[12] Hang Liu and H. Howie Huang. Enterprise: Breadth-first graph traversal on GPUs. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*, pages 1–12, 2015.

[13] Yury A. Malkov and Dmitry A. Yashunin. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(4):824–836, 2020.

[14] Duane Merrill, Michael Garland, and Andrew Grimshaw. Scalable GPU graph traversal. In *Proceedings of the 17th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming (PPoPP)*, pages 117–128, 2012.

[15] NVIDIA. ReMEmbR: Long-context robotic memory recall via vision-language models. https://nvidia-isaac-ros.github.io/repositories_and_packages/isaac_ros_remembr/, 2024.

[16] NVIDIA. cuVSLAM: CUDA-accelerated visual SLAM for Isaac ROS. https://github.com/NVIDIA-ISAAC-ROS/isaac_ros_visual_slam, 2025.

[17] NVIDIA. nvblox: GPU-accelerated 3D reconstruction and mapping for robotics. https://github.com/nvidia-isaac/nvblox, 2025.

[18] NVIDIA and RAPIDS AI. cuVS: GPU-accelerated vector search and clustering. https://github.com/rapidsai/cuvs, 2025. Includes CAGRA GPU-native graph ANN index; integrated into FAISS since v1.10.0.

[19] Pinecone Systems. Pinecone: Vector database for machine learning. https://www.pinecone.io/, 2025.

[20] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *Proceedings of the 38th International Conference on Machine Learning (ICML)*, pages 8748–8763, 2021.

[21] Suhas Jayaram Subramanya, Fnu Devvrit, Harsha Vardhan Simhadri, Ravishankar Krishnaswamy, and Rohan Kadekodi. DiskANN: Fast accurate billion-point nearest neighbor search on a single node. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[22] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, et al. Scalability in perception for autonomous driving: Waymo open dataset. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2446–2454, 2020.

[23] Karthik Venkatasubba, Saim Khan, et al. BANG: Billion-scale approximate nearest neighbour search using a single GPU. *IEEE Transactions on Big Data*, 11(6):3142–3157, 2025.

[24] Jianguo Wang, Xiaomeng Yi, Rentong Guo, Hai Jin, Peng Xu, Shengjun Li, Xiangyu Wang, Xiangzhou Guo, Chengming Li, Xiaohai Xu, et al. Milvus: A purpose-built vector data management system. *Proceedings of the ACM on Management of Data (SIGMOD)*, 1(2):1–22, 2021.

[25] Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. Text embeddings by weakly-supervised contrastive pre-training. *arXiv preprint arXiv:2212.03533*, 2022.

[26] Yangzihao Wang, Andrew Davidson, Yuechao Pan, Yuduo Wu, Andy Riffel, and John D. Owens. Gunrock: A high-performance graph processing library on the GPU. In *Proceedings of the 21st ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming (PPoPP)*, pages 1–12, 2016.

[27] Duncan J. Watts and Steven H. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393(6684):440–442, 1998.

[28] Yusong Wu, Ke Chen, Tianyu Zhang, Yuchen Hui, Taylor Berg-Kirkpatrick, and Shlomo Dubnov. Large-scale contrastive language-audio pretraining with feature fusion and keyword-to-caption augmentation. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2023.