

VISTA: An Efficient Late Interaction Engine for Multi-Vector Visual Document Retrieval

Haolin Jiang Jinwen Pan Abhishek Saini Weihang Li Hang Liu

Rutgers, The State University of New Jersey
{haolin.jiang, hang.liu}@rutgers.edu

Abstract

Visual document retrieval (VDR) bridges language-based systems with visually grounded knowledge. Recent VDR models adopt multi-vector late interaction, representing each document with multiple visual token embeddings and scoring relevance with MaxSim. However, existing late interaction engines struggle to support multi-vector VDR at scale due to the higher dimensionality and larger number of visual tokens per document.

We propose VISTA¹, a VDR-specific late-interaction engine that exploits the correlation among visual tokens within each document to optimize both compression and search. To the best of our knowledge, VISTA is the first VDR-specific late-interaction engine for scalable multi-vector visual document retrieval. Experiments on ViDoRe V2 and ViDoRe V3 show that VISTA achieves comparable or even better retrieval quality than compressed baselines, while consistently reducing latency by $1.2\text{--}6.7\times$ and index size by $1.4\text{--}6.1\times$.

1 Introduction

Visual document retrieval (VDR) bridges language-based systems with visually grounded knowledge (Yu et al., 2025; Faysse et al., 2024). Given a natural-language query, VDR aims to retrieve the most relevant document pages from visually rich document collections. In RAG (Lewis et al., 2021), a modality gap arises as many existing systems rely on text-only retrieval (Gutiérrez et al., 2024; Wang et al., 2026), while real-world knowledge is frequently stored in visually structured documents such as scanned reports, slides, forms, equations, tables, and figures (Macé et al., 2025; Loison et al., 2026). VDR therefore provides a key retrieval layer for shifting text-only RAG toward multimodal RAG, enabling applications in financial

report analysis (Zhao et al., 2025) and document question answering (Mathew et al., 2021).

Retrieving such visually structured pages requires matching queries to localized evidence from layouts, tables, figures, and text regions. To support this fine-grained matching, recent state-of-the-art visual document retrievers (Faysse et al., 2024; de Souza P. Moreira et al., 2026) adapt multi-vector late interaction from text retrieval (Khattab and Zaharia, 2020), representing each page with multiple token embeddings and computing query-document similarity through MaxSim scoring.

While accurate, this paradigm is costly at scale: it requires storing many token-level representations, and MaxSim scoring requires a dedicated search pipeline. Existing late-interaction engines (Bian et al., 2025; Mo et al., 2026; Nardini et al., 2024; Santhanam et al., 2022b; Scheerer et al., 2025) commonly rely on a centroid-based design to scale multi-vector retrieval. The idea is to cluster document tokens into centroids using k -means, so each token can be compactly represented by its nearest centroid and a quantized residual. These centroid assignments also form an inverted mapping from centroids to documents for retrieval. At query time, the top- n_{probe} nearest centroids are probed for each query token, and the associated documents are collected as candidates for subsequent reranking.

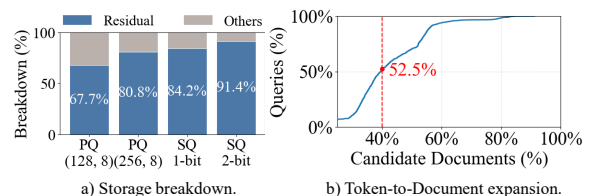


Figure 1: Profiling on ViDoRe V3 CS subset.

However, this design struggles to scale to VDR because visual documents differ from text passages in two key ways. (i) Visual embeddings are much higher-dimensional than text embeddings (2560–

¹Code is available at <https://github.com/Orrinn/VISTA>

4096 in Nemotron ColEmbed V2 (de Souza P. Moreira et al., 2026) versus 128 in ColBERTv2 (Santhanam et al., 2022b)), making residuals a substantial part of the index even after compression. Figure 1(a) shows that, even after scalar or product quantization, residuals still account for 70%–90% of the total index storage. (ii) Visual pages contain more patch tokens than text passages ($\sim 1\text{K}$ tokens per document (de Souza P. Moreira et al., 2026) versus ~ 180 tokens per passage (Santhanam et al., 2022b)), so each probed centroid tends to cover more documents, making even a small n_{probe} yield a large fraction of the corpus as candidates. Figure 1(b) shows that, even with the minimum budget ($n_{\text{probe}} = 1$), about half of the queries produce candidate sets covering more than 40% of the corpus, slowing down search.

Rather than treating VDR as a direct extension of text retrieval, we ground our design in prior evidence that visual tokens often exhibit strong token-wise similarity and shared global structure (Chen et al., 2022; Lu et al., 2026). This motivates us to exploit redundancy among visual tokens to guide both compression and retrieval, rather than treating them as independent units. We therefore introduce VISTA. To the best of our knowledge, VISTA is the first VDR-specific late-interaction engine that exploits redundancy among visual tokens to optimize both compression and retrieval.

In summary, our contributions are:

- We identify VDR-specific bottlenecks arising from high-dimensional visual embeddings and large token counts, and show that within-document token correlations offer opportunities to reduce both memory and retrieval costs. (Section 3)
- We propose VISTA, a VDR-specific late-interaction engine that uses low-rank decomposition for aggressive compression and document-level mean vectors for progressive retrieval. (Section 4)
- Experiments show that VISTA improves the accuracy–efficiency trade-off over compressed baselines, reducing latency by $1.2\text{--}6.7\times$ and index size by $1.4\text{--}6.1\times$. Under the most aggressive setting with Ops-4B, VISTA retains 98.0% of Oracle NDCG with only 1.22% of the uncompressed size. (Section 5)

VISTA differs from prior late-interaction systems (Bian et al., 2025; Faysse et al., 2024; Nardini

et al., 2024; Mo et al., 2026) in three key aspects. First, it is designed specifically for VDR, where high-dimensional visual embeddings and large per-document token counts make existing strategies inefficient. Second, existing frameworks largely ignore redundancy among visual tokens, leading to limited compression efficiency. Third, current systems rely on token-to-document expansion, which produces overly large candidate sets and results in suboptimal retrieval latency. VISTA addresses these challenges with low-rank decomposition for aggressive compression and document-level statistics for progressive retrieval.

2 Related Work & Preliminary

2.1 Related Work

Existing work related to scalable multi-vector retrieval can be organized into three directions: retrieval acceleration, visual token reduction, and general retrieval pipelines.

Retrieval acceleration improves search efficiency through specialized indexes or retrieval procedures for multi-vector search. Methods like COIL (Gao et al., 2021), CITADEL (Li et al., 2023), and SPLATE (Formal et al., 2024) exploit lexical structures in text retrieval to avoid exhaustive token-pair scoring. XTR (Lee et al., 2023) trains retrievers to score documents using retrieved document tokens, and WARP (Scheerer et al., 2025) further accelerates this XTR-style retrieval pipeline through system-level optimizations. Despite their efficiency, these methods do not directly transfer to VDR because visual tokens lack lexical structure and existing retrievers are not trained with XTR-style objectives.

Visual token reduction lowers the index size by reducing the number of stored visual tokens. DocPruner (Yan et al., 2025) prunes less important tokens using attention scores, while SAP (Liu et al., 2026) uses mid-layer activations to identify tokens with lower visual focus. Other variants (Kankanampati et al., 2026; Zong and Piwowarski, 2025) prune token comparisons with dominance relations or geometric bounds. Light-ColPali (Ma et al., 2025) applies hierarchical token merging with retraining. A recent work (Yan et al., 2026) further combines pruning with merging to reduce the visual token count. However, pruning or merging is inherently lossy and produces variable-size token sets, limiting the achievable compression ratio without harming retrieval accuracy and making it difficult to

design efficient search pipelines.

General retrieval pipelines provide end-to-end support for indexing and searching multi-vector representations. PLAID (Santhanam et al., 2022a) is a representative centroid-based engine with both CPU and GPU support. EMVB (Nardini et al., 2024) improves PLAID using product quantization and SIMD acceleration. IGP (Bian et al., 2025) improves candidate generation with a proximity graph index, producing smaller candidate sets with high recall. GEM (Tian et al., 2026) further develops native proximity-graph indexing for multi-vector retrieval. VecFlow-Chamfer (Mo et al., 2026) builds a GPU-based data management system for high-performance multi-vector search. These systems are general-purpose multi-vector engines and do not exploit the document-level structure of visual tokens in VDR, leading to higher index storage and retrieval latency in VDR settings.

Other directions in multi-vector retrieval. Beyond engine-level optimization, recent work also explores alternative formulations of multi-vector retrieval. MUVERA (Dhulipala et al., 2024) converts multi-vector retrieval into fixed-dimensional encodings, enabling standard single-vector retrieval. LEMUR (Jääsaari et al., 2026) learns multi-vector representations directly, while MetaEmbed (Xiao et al., 2025) studies flexible late interaction for multimodal retrieval at test time.

2.2 Preliminary

Late-interaction scoring. Given $Q \in \mathbb{R}^{N_Q \times d}$ and $D \in \mathbb{R}^{N_D \times d}$, late interaction computes the token-level similarity matrix $QD^\top$. The MaxSim score is then given by

$$s(Q, D) = \sum_{i=1}^{N_Q} \max_{j \in [1, N_D]} (QD^\top)_{ij}.$$

Random Hyperplane Projection for Angular Similarity. Random hyperplane projection offers a compact way to estimate angular similarity with binary sketches. Intuitively, two vectors with smaller angles are more likely to fall on the same side of a random hyperplane. Therefore, using multiple random hyperplane projections, one can estimate the original angular similarity from the similarity between their sketches. Representative examples include SimHash (Charikar, 2002) and QJL (Zandieh et al., 2025). We use QJL in VISTA, with details provided in Appendix A.

3 VISTA Overview

Our key observation is that visual tokens within a document are highly correlated, enabling a more compact and faster retrieval engine:

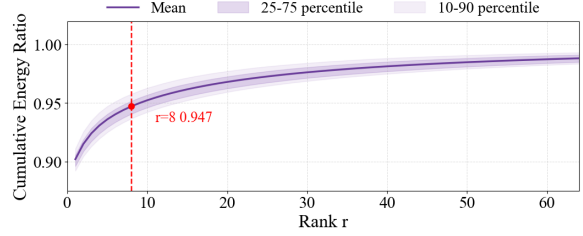


Figure 2: Low-rank structure of visual tokens.

Observation #1: Visual tokens have low-rank structure. Figure 2 reports low-rank profiling on the CS subset of ViDoRe V3 benchmark (Loison et al., 2026). Visual tokens within a document exhibit a strong low-rank structure, with a rank-8 approximation capturing 94.7% of the document information. By preserving such dominant low-rank information exactly, we can apply more aggressive compression to the remaining information to reduce the index footprint.

Design 1: Low-rank decomposition for compact representation. Concretely, we decompose the document matrix as $D = D_{\text{LowRank}} + D_{\text{remain}}$. Instead of storing the full matrix D_{LowRank} , we store it in product form: $D_{\text{LowRank}} = Z_D V_r^\top$, where V_r is the rank- r basis and $Z_D = D V_r$. For the remainder D_{remain} , we store its row-wise norm $\|D_{\text{remain}}\|$ exactly, and compress its unit direction U_D into a 1-bit sketch S_D using QJL.

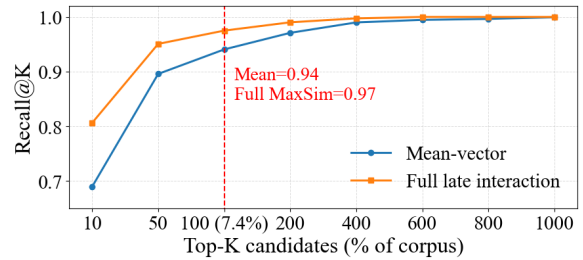


Figure 3: Recall@K profiling using mean vector.

Observation #2: Document mean vectors are effective for coarse retrieval. Figure 3 profiles the Recall@K using the mean vector D_{avg} in the ViDoRe V3 CS subset. The correlation among visual tokens makes D_{avg} effective for coarse retrieval: it achieves 0.94 Recall@100 using only 7.4% of the corpus as candidates and retains 97% of the late-interaction recall. We therefore use D_{avg} for candi-

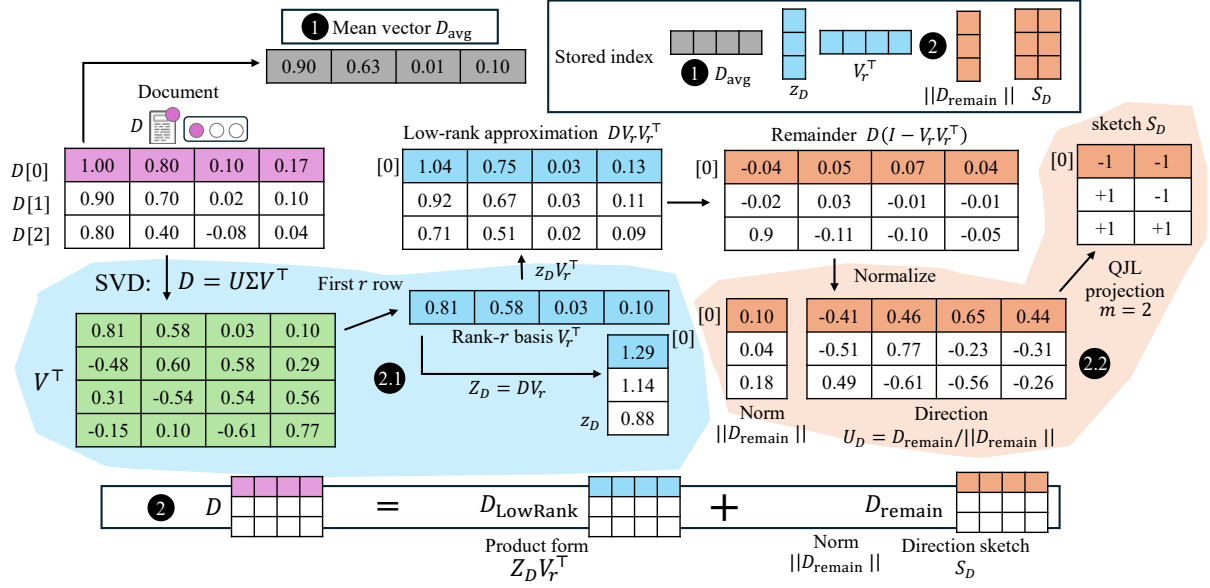


Figure 4: Offline index construction. Values are rounded for illustration.

date selection, followed by progressive reranking based on the decomposition proposed above.

Design 2: Progressive reranking for efficient retrieval. First, the document mean vector D_{avg} is used for coarse candidate retrieval. Following the decomposition $D = D_{LowRank} + D_{remain}$, we have $QD^T = QD_{LowRank}^T + QD_{remain}^T$. We therefore first rerank with $QD_{LowRank}^T$ and then refine the final ranking with QD_{remain}^T . For the first term, we have $QD_{LowRank}^T = (QV_r)Z_D^T$. The remaining term QD_{remain}^T is estimated from the stored norm $\|D_{remain}\|$ and sketch S_D using QJL. Since it can be noisy under aggressive compression, we further apply Bayesian shrinkage for calibration.

4 VISTA Engine

4.1 Index Construction

Figure 4 shows a toy example of compressing a document D with three tokens: $D[0]$, $D[1]$, $D[2]$. Index construction creates two index components: ① a mean vector D_{avg} for coarse retrieval (see Design 2), and ② the decomposed document representation: low-rank approximation $D_{LowRank}$ and remainder D_{remain} (see Design 1).

Mean-vector D_{avg} . We first build a document mean vector D_{avg} by averaging all token vectors: $D_{avg} = \frac{1}{N_D} \sum_{i=1}^{N_D} D[i]$, where N_D is the number of document tokens. In Figure 4 ①, this gives $D_{avg} = [0.90, 0.63, 0.01, 0.10]$.

Decomposed representation. In ②, we decompose the document D into (i) a low-rank approximation $D_{LowRank}$ and (ii) a remainder D_{remain} .

(i) **Low-rank approximation $D_{LowRank}$** preserves the dominant information of the document D . Mathematically, $D_{LowRank}$ can be written as the product of two smaller matrices, $D_{LowRank} = Z_D V_r^T$, so we store Z_D and V_r instead of the full matrix $D_{LowRank}$. As shown in 2.1 (blue region), to compute V_r^T , we apply SVD to the document D as $D = U\Sigma V^T$, obtaining the V^T (green matrix), and keeping the first r rows of V^T as V_r^T . Here we use $r = 1$, giving $V_r^T = [0.81, 0.58, 0.03, 0.10]$. By linear algebra, Z_D is computed as $Z_D = DV_r = [1.29, 1.14, 0.88]^T$. Therefore, we have $D_{LowRank} = Z_D V_r^T = DV_r V_r^T$.

(ii) **Remainder D_{remain}** is stored by keeping its norm $\|D_{remain}\|$ exactly and compressing its direction U_D into a sketch S_D . After deriving $D_{LowRank}$, we compute the remainder as $D_{remain} = D - D_{LowRank} = D - DV_r V_r^T = D(I - V_r V_r^T)$, giving the orange matrix with first row $[-0.04, 0.05, 0.07, 0.04]$. As shown in 2.2 (orange region), we then separate it into row-wise norms $\|D_{remain}\| = [0.10, 0.04, 0.18]^T$ and unit directions $U_D = D_{remain} / \|D_{remain}\|$, whose first row is $[-0.41, 0.46, 0.65, 0.44]$. To compress U_D more aggressively, we follow QJL (Zandieh et al., 2025) to project U_D into an m -dimensional 1-bit sketch S_D , giving the orange matrix with first row $[-1, -1]$. Since m can be much smaller than the original embedding dimension in practice, we apply calibration during retrieval to compensate for this aggressive compression.

Overall, after indexing, for each document D ,

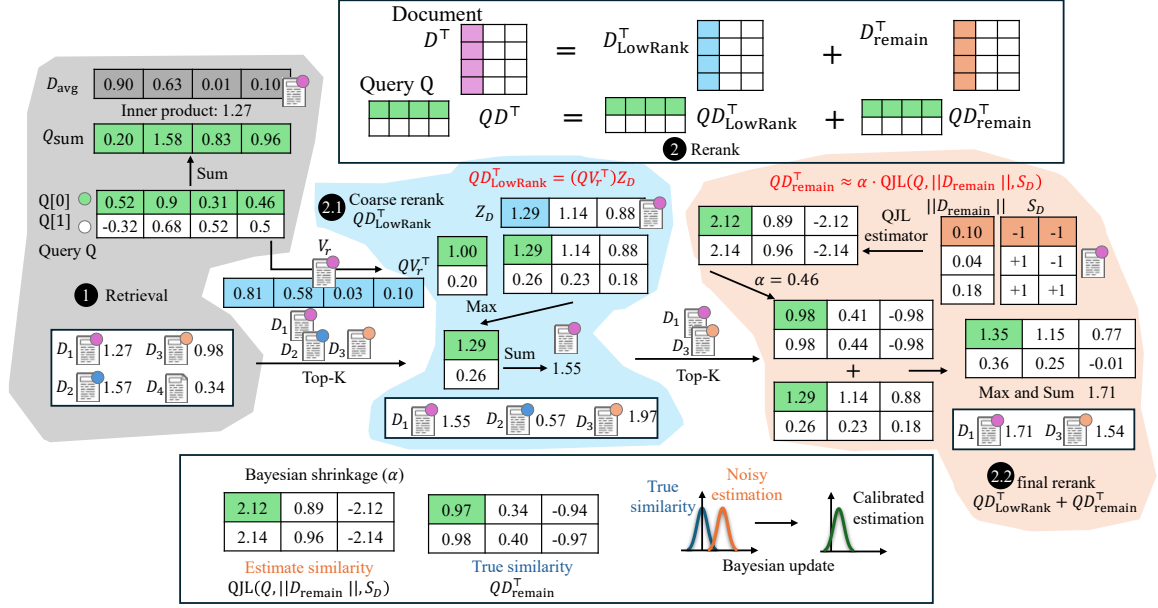


Figure 5: Online progressive retrieval. Values are rounded for illustration.

we store D_{avg} , V_r , Z_D , $\|D_{\text{remain}}\|$, and S_D .

4.2 Progressive retrieval

Figure 5 illustrates how the index from Section 4.1 is used for retrieval with four indexed documents D_1 – D_4 and a query Q with two tokens, $Q[0]$ and $Q[1]$. For each document, offline compression stores D_{avg} for candidate selection, (V_r, Z_D) for D_{LowRank} , and $(\|D_{\text{remain}}\|, S_D)$ for D_{remain} . At query time, VISTA uses these components progressively. It first scores documents with D_{avg} for candidate retrieval. It then follows $QD^T = QD_{\text{LowRank}}^T + QD_{\text{remain}}^T$: (i) (V_r, Z_D) are used to compute QD_{LowRank}^T for coarse reranking, and (ii) $(\|D_{\text{remain}}\|, S_D)$ are used to estimate QD_{remain}^T for final reranking.

Candidate retrieval. As shown in ❶ (gray region), for each document D , we compute the stage-1 score as $s_1(Q, D) = Q[0] \cdot D_{\text{avg}} + Q[1] \cdot D_{\text{avg}} = (Q[0] + Q[1]) \cdot D_{\text{avg}}$. For example, for D_1 , we have $Q[0] + Q[1] = [0.20, 1.58, 0.83, 0.96]$, and $D_{\text{avg}} = [0.90, 0.63, 0.01, 0.10]$, giving the $s_1(Q, D) = 1.27$. Similarly, for D_2 – D_4 , the stage-1 scores are 1.57, 0.98, and 0.34, respectively. We retain the top-3 documents as candidates, yielding the candidate set $\{D_1, D_2, D_3\}$.

Reranking. Since each document D has been decomposed as $D = D_{\text{LowRank}} + D_{\text{remain}}$, the query-document similarity matrix (Figure 5 ❷) can be naturally given by $QD^T = QD_{\text{LowRank}}^T + QD_{\text{remain}}^T$. Therefore, for (i) coarse reranking, we use the term QD_{LowRank}^T . For (ii) final reranking,

we add the estimated term QD_{remain}^T .

(i) Coarse reranking. For QD_{LowRank}^T , we do not explicitly reconstruct D_{LowRank} . Since $D_{\text{LowRank}} = Z_D V_r^T$, we have $QD_{\text{LowRank}}^T = Q(Z_D V_r^T)^T = (Q V_r) Z_D^T$ (highlighted in red). As shown in ❷.1 (blue region), for document D_1 , we use its V_r and query Q to obtain $Q V_r = [1.00, 0.20]^T$. We then multiply $Q V_r$ with the stored matrix Z_D^T , obtaining the coarse similarity matrix $\begin{bmatrix} 1.29 & 1.14 & 0.88 \\ 0.26 & 0.23 & 0.18 \end{bmatrix}$. Using MaxSim, we take the maximum value for each row and sum them, obtaining the coarse reranking score $s_2(Q, D_1) = 1.29 + 0.26 = 1.55$. Similarly, the low-rank similarity scores for D_2 and D_3 are 0.57 and 1.97, respectively. We keep the top-2 documents $\{D_1, D_3\}$ for final reranking.

(ii) Final reranking. For QD_{remain}^T , we use a calibrated QJL estimate: $QD_{\text{remain}}^T \approx \alpha \cdot \text{QJL}(Q, \|D_{\text{remain}}\|, S_D)$, where $\text{QJL}(\cdot)$ denotes the QJL estimator and α is a calibration factor for the noisy 1-bit estimate. As shown in ❷.2 (orange region), for document D_1 , we apply the QJL estimator using the query Q , the stored norm $\|D_{\text{remain}}\|$, and the direction sketch S_D from document D_1 . This gives the estimated matrix $\begin{bmatrix} 2.12 & 0.89 & -2.12 \\ 2.14 & 0.96 & -2.14 \end{bmatrix}$. However, as shown in the bottom-center of Figure 5, due to the aggressive compression in QJL, the raw estimate can deviate substantially from the

Model	Method	ViDoRe V2			ViDoRe V3		
		N@5	Latency (ms)	Size (MB)	N@10	Latency (ms)	Size (MB)
Nemo-4B	Oracle	64.3	–	4118.3	61.3	–	8796.6
	SAP	56.3	–	820.4	54.8	–	1432.1
	IGP (2-bit)	60.7	507.2	582.6	58.3	92.7	1186.6
	PLAID (2-bit)	64.1	45.7	582.6	60.4	48.4	1186.6
	PLAID _{PQ} (256,8)	64.9	8.8	275.0	60.8	12.3	528.1
	VISTA (16,512)	65.0	7.4	173.1	60.8	10.5	368.1
Ops-4B	Oracle	65.1	–	6137.7	60.6	–	14580.5
	SAP	60.8	–	1225.9	58.0	–	2915.3
	IGP (2-bit)	64.3	514.9	840.5	43.3	138.7	1944.1
	PLAID (2-bit)	65.1	65.9	840.5	59.9	82.0	1944.1
	PLAID _{PQ} (256,8)	65.2	13.8	381.4	59.7	22.2	851.8
	VISTA (16,512)	64.6	9.9	211.8	60.4	15.0	478.8

Table 1: Main results on ViDoRe V2 and V3. N@X denotes NDCG@X. All metrics are averaged across datasets. IGP/PLAID use 2-bit compression, PLAID_{PQ} uses (256, 8) PQ, and VISTA uses ($r = 16, m = 512$).

$QD_{\text{remain}}^{\top} = \begin{bmatrix} 0.97 & 0.34 & -0.94 \\ 0.98 & 0.40 & -0.97 \end{bmatrix}$. Therefore, we apply a shrinkage factor α to calibrate the QJL estimate. In this toy example, we use $\alpha = 0.46$, giving a calibrated estimate $QD_{\text{remain}}^{\top} \approx \begin{bmatrix} 0.98 & 0.41 & -0.98 \\ 0.98 & 0.44 & -0.98 \end{bmatrix}$. Intuitively, this calibration shrinks the raw QJL estimate (orange distribution) toward the expected range of the true remainder similarity (blue distribution). In practice, α is obtained from the Bayesian shrinkage estimate using statistics from a small calibration set. Details are provided in Appendix B. After estimating $QD_{\text{remain}}^{\top}$, we add it to the $QD_{\text{LowRank}}^{\top}$ from (2.1), obtaining the final query-document similarity matrix $\begin{bmatrix} 1.35 & 1.15 & 0.77 \\ 0.36 & 0.25 & -0.01 \end{bmatrix}$. Applying MaxSim gives the final score for D_1 : $s_3(Q, D_1) = 1.35 + 0.36 = 1.71$. Similarly, we obtain $s_3(Q, D_3) = 1.54$. Therefore, D_1 is ranked above D_3 .

5 Evaluation

5.1 Evaluation Setup

Visual document retrievers. We evaluate VISTA with two state-of-the-art visual document retrievers from the ViDoRe leaderboard²: Nemotron-ColEmbed-VL-4B-v2 (de Souza P. Moreira et al., 2026) and Ops-Colqwen3-4B (OpenSearch-AI, 2026). Details are provided in Appendix C.

²<https://huggingface.co/spaces/vidore/vidore-leaderboard>

Datasets and metrics. We evaluate the retrieval pipeline on both ViDoRe-V2 (Macé et al., 2025) and ViDoRe-V3 (Loison et al., 2026) benchmarks, covering diverse visually rich document retrieval tasks across 12 datasets. Dataset details are provided in Appendix D. Following prior work (Liu et al., 2026; Yan et al., 2025; OpenSearch-AI, 2026), we report NDCG@5 on ViDoRe-V2 and NDCG@10 on ViDoRe-V3. We also report index footprint and retrieval latency.

Baselines. We compare VISTA with Oracle, PLAID³ (Santhanam et al., 2022a), PLAID_{PQ}, IGP (Bian et al., 2025), and SAP (Liu et al., 2026). Oracle performs a brute-force MaxSim search over the original uncompressed embeddings. PLAID and PLAID_{PQ} are GPU-based multi-vector retrieval baselines, where PLAID_{PQ} is a PQ-based variant of PLAID. IGP is evaluated as a CPU-based multi-vector retrieval baseline with 24 threads. SAP is included as a visual token-reduction baseline. All GPU experiments use a single NVIDIA A100 GPU with 80GB memory.

5.2 Main results

Table 1 presents the main results on ViDoRe V2 and V3 benchmarks. We report NDCG@X, retrieval latency, and index footprint. Detailed configurations are provided in Appendix E.

VISTA achieves comparable or best retrieval quality while consistently reducing latency and in-

³<https://github.com/lightonai/fast-plaid>

dex footprint. For Nemo-4B, VISTA achieves the highest NDCG on ViDoRe V2 and matches the best NDCG on ViDoRe V3, while reducing GPU latency by $1.2\text{--}6.2\times$ and index size by $1.4\text{--}4.7\times$. For Ops-4B, VISTA achieves comparable NDCG on ViDoRe V2 and the best NDCG on ViDoRe V3, while reducing GPU latency by $1.4\text{--}6.7\times$ and index size by $1.8\text{--}6.1\times$. Notably, on Nemo-4B ViDoRe V3, VISTA reaches 60.8 vs. 61.3 Oracle NDCG@10 while using only 4.2% of the original index size. Additional results on different datasets and a different VDR architecture are provided in Appendices F and G.

Among baselines, SAP tends to suffer from lower retrieval quality due to dropped visual tokens. IGP generally preserves retrieval quality, except under Ops-4B on ViDoRe V3 where one subset (pharmaceuticals) shows a large drop. At the same time, its CPU-based search incurs much higher latency even with 24 threads. PLAID_{PQ} provides a strong performance–efficiency trade-off by reducing memory traffic with product quantization, but it still does not exploit within-document low-rank structure, leaving room for VISTA to achieve a better memory–accuracy trade-off.

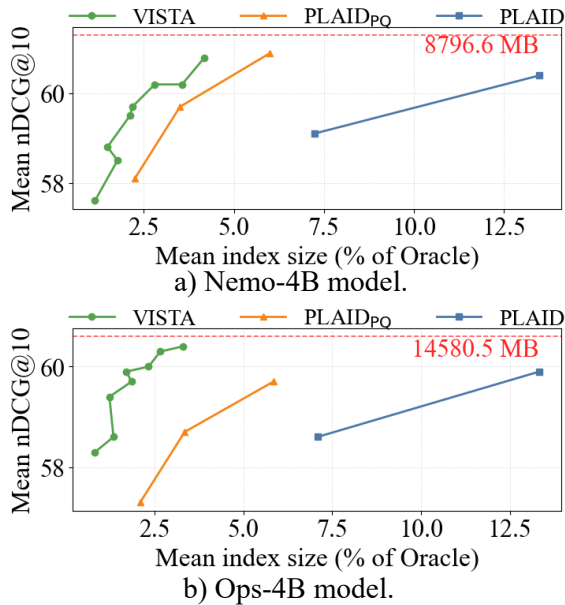


Figure 6: Memory–accuracy trade-off on ViDoRe V3.

Memory–accuracy trade-off. Figure 6 further compares the memory–accuracy trade-off of VISTA with PLAID and PLAID_{PQ} on ViDoRe V3 under different compression configurations. For VISTA, we vary the low-rank dimension $r \in \{2, 4, 8, 16\}$ and sketch dimension $m \in \{256, 512\}$. For

PLAID_{PQ}, we vary the number of PQ subquantizers $m \in \{64, 128, 256\}$ with 8-bit codebooks. For PLAID, we report both 1-bit and 2-bit.

Across both retrievers, VISTA consistently shifts the Pareto frontier toward smaller index sizes while maintaining competitive retrieval quality. For Ops-4B, the advantage is most pronounced: VISTA ($r = 4, m = 256$) reaches 59.4 NDCG@10 with only 178.4 MB, achieving similar accuracy to PLAID_{PQ} (256,8)’s 59.7 with 851.8 MB and PLAID (2-bit)’s 59.9 with 1944.1 MB, showing $4.8\times$ and $10.9\times$ index-size reductions. Relative to the Oracle, this setting retains 98.0% of NDCG@10 (59.4 vs. 60.6) with only 1.22% of the original index size ($81.7\times$ compression). A similar trend holds for Nemo-4B, where VISTA ($r = 8, m = 256$) reaches 59.7 NDCG@10 with only 191.7 MB, compared with PLAID_{PQ} (128,8)’s 59.7 with 308.2 MB and PLAID (1-bit)’s 59.1 with 636.8 MB, showing a $1.6\times$ and $3.3\times$ index-size reduction.

5.3 Analysis

Model	Stage	ViDoRe V2	ViDoRe V3
Nemo-4B	Retrieve	50.4	46.0
	+ Coarse	52.3	54.3
	+ Final	65.0	60.8
Ops-4B	Retrieve	48.2	46.3
	+ Coarse	56.5	55.1
	+ Final	64.6	60.4

Table 2: VISTA ablation on ViDoRe V2 and V3.

Ablation study. Table 2 presents the ablation results of VISTA using the same configuration as in Table 1. We compare three progressive stages: (i) RETRIEVE, which uses only mean vectors for coarse retrieval; (ii) +COARSE, which further adds coarse reranking with the stored low-rank approximation; and (iii) +FINAL, which denotes the full pipeline.

Removing later stages consistently reduces retrieval quality across both retrievers and benchmarks. Using only RETRIEVE gives limited accuracy, while +COARSE improves NDCG from 48.2/46.3 to 56.5/55.1 on Ops-4B and from 50.4/46.0 to 52.3/54.3 on Nemo-4B. The full +FINAL pipeline achieves the best performance, reaching 65.0/60.8 for Nemo-4B and 64.6/60.4 for Ops-4B. These results show that coarse stages effectively narrow the search space, while final reranking is necessary for high retrieval accuracy.

Model	Setting	$r = 2$			$r = 4$			$r = 8$			$r = 16$		
		128	256	512	128	256	512	128	256	512	128	256	512
Nemo-4B	w/o α	54.8	57.5	58.5	55.8	58.6	59.5	57.4	59.5	60.2	58.8	60.1	60.8
	w/ α	55.2	57.5	58.5	56.5	58.8	59.5	58.3	59.7	60.2	59.5	60.2	60.8
	Gain	+0.4	+0.0	+0.0	+0.7	+0.2	+0.0	+0.9	+0.2	+0.0	+0.7	+0.1	+0.0
Ops-4B	w/o α	56.5	57.9	58.4	57.4	58.8	59.4	58.4	59.4	59.7	59.0	59.8	60.2
	w/ α	57.2	58.3	58.6	58.4	59.4	59.7	59.2	59.9	60.0	59.7	60.3	60.4
	Gain	+0.7	+0.4	+0.2	+1.0	+0.6	+0.3	+0.8	+0.5	+0.3	+0.7	+0.5	+0.2

Table 3: Effect of shrinkage factor α for VISTA on ViDoRe V3. Values are average NDCG@10 across datasets, and gains reported as +0.0 indicate negligible changes after rounding.

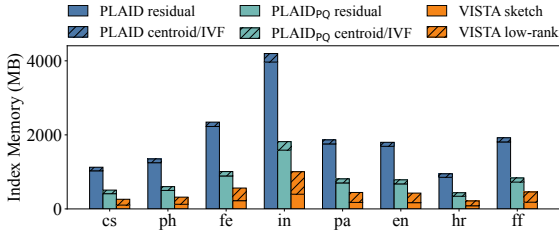


Figure 7: Index footprint breakdown on ViDoRe V3 using Ops-4B.

Index footprint breakdown. Figure 7 shows the index-footprint breakdown. All methods use the same configurations as in Table 1, and we report only the major components while omitting negligible components (<1%) of the index.

Overall, VISTA consistently achieves the smallest index footprint by substantially reducing residual-related storage. For example, the residual/sketch storage of PLAID, PLAID_{PQ}, and VISTA is 3966.6/1586.6/396.7 MB on the largest subset in, and 853.6/341.5/85.4 MB on the smallest subset hr. Across all eight subsets, residuals account for 89.9–94.9% of PLAID and 78.3–88.3% of PLAID_{PQ}, while VISTA uses only 39.3–39.7% of its index for remainder sketches. This confirms that separating the dominant low-rank component enables VISTA to aggressively compress residual storage, thereby reducing the footprint. We also report index-construction overhead in Appendix I.

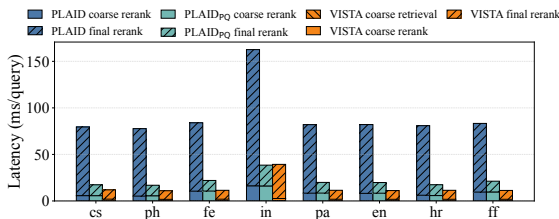


Figure 8: Latency breakdown on ViDoRe V3 using Ops-4B.

Latency breakdown. Figure 8 reports the latency breakdown. Overall, VISTA achieves the lowest end-to-end latency on most subsets, except on in, where its final reranking cost is higher due to a larger reranked candidate set. For VISTA, the initial coarse retrieval incurs negligible overhead, accounting for only 1.1–3.3% of total latency. It also produces a smaller candidate set, allowing VISTA to cut coarse reranking latency from 5.6–16.1 ms in PLAID_{PQ} to 1.5–2.2 ms. In the final reranking, both PLAID_{PQ} and VISTA are much faster than PLAID due to compact index and lower memory traffic: PLAID takes 72.5–146.3 ms, while PLAID_{PQ} takes 11.3–22.4 ms and VISTA takes 9.2–10.0 ms on most subsets, except in with 36.7 ms.

Effect of shrinkage factor α . Table 3 studies the effect of the shrinkage factor α across different ranks r and sketch dimensions m , showing that shrinkage mainly improves retrieval quality when m is small, while its advantage becomes less pronounced as m increases. For example, when $m = 128$, α improves NDCG@10 by up to 0.9 on Nemo-4B and 1.0 on Ops-4B, while the gains become small as m increases to 512. This matches our analysis in Appendix B: smaller m introduces larger estimation variance and therefore leads to stronger shrinkage, whereas as m increases, the Bayesian shrinkage factor naturally approaches 1, making the calibrated score closer to the estimate. We also conduct a sensitivity analysis on calibration-set size and query domain in Appendix H.

6 Conclusion

We present VISTA, an efficient late-interaction engine for visual document retrieval. By leveraging low-rank decomposition for aggressive compression and progressive retrieval, VISTA achieves both high effectiveness and efficiency. Experiments

on multiple visual retrievers and ViDoRe benchmarks show that VISTA consistently improves the efficiency–accuracy trade-off over existing late-interaction engines for VDR under various configurations.

Limitations

VISTA currently assumes a fixed document index during retrieval. It does not yet support efficient dynamic index updates, such as inserting new pages, modifying existing pages, or deleting documents without rebuilding the affected compressed representations. Supporting efficient and accurate streaming updates is non-trivial for multi-vector retrieval, and we leave it to future work (See Appendix J).

Acknowledgments

We thank the anonymous reviewers for their constructive feedback and valuable suggestions. This work was supported in part by NSF CAREER Award No. 2326141, NSF CyberTraining Award No. 2417750, NSF CSCI Award No. 2411294, NSF CICI Award No. 2319880, and NSF ExpandQISE Award No. 2328948, as well as by the Rutgers TechAdvance Award.

References

- Zheng Bian, Man Lung Yiu, and Bo Tang. 2025. Igp: Efficient multi-vector retrieval via proximity graph index. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2524–2533.
- Moses S Charikar. 2002. Similarity estimation techniques from rounding algorithms. In *Proceedings of the thirty-fourth annual ACM symposium on Theory of computing*, pages 380–388.
- Tianlong Chen, Zhenyu Zhang, Yu Cheng, Ahmed Awadallah, and Zhangyang Wang. 2022. The principle of diversity: Training stronger vision transformers calls for reducing all levels of redundancy. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12020–12030.
- Gabriel de Souza P. Moreira, Ronay Ak, Mengyao Xu, Oliver Holworthy, Benedikt Schifferer, Zhiding Yu, Yauhen Babakhin, Radek Osmulski, Jiarui Cai, Ryan Chesler, Bo Liu, and Even Oldridge. 2026. *Nemotron colembed v2: Top-performing late interaction embedding models for visual document retrieval*. Preprint, arXiv:2602.03992.
- Laxman Dhulipala, Majid Hadian, Rajesh Jayaram, Jason Lee, and Vahab Mirrokni. 2024. Muvera: Multi-vector retrieval via fixed dimensional encoding. *Advances in Neural Information Processing Systems*, 37:101042–101073.
- Manuel Faysse, Hugues Sibille, Tony Wu, Bilel Omrani, Gautier Viaud, Céline Hudelot, and Pierre Colombo. 2024. *Colpali: Efficient document retrieval with vision language models*. Preprint, arXiv:2407.01449.
- Thibault Formal, Stéphane Clinchant, Hervé Déjean, and Carlos Lassance. 2024. Splate: Sparse late interaction retrieval. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2635–2640.
- Luyu Gao, Zhuyun Dai, and Jamie Callan. 2021. Coil: Revisit exact lexical match in information retrieval with contextualized inverted list. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3030–3042.
- Bernal J Gutiérrez, Yiheng Shu, Yu Gu, Michihiro Yasunaga, and Yu Su. 2024. Hipporag: Neurobiologically inspired long-term memory for large language models. *Advances in neural information processing systems*, 37:59532–59569.
- Elias Jääsaari, Ville Hyvönen, and Teemu Roos. 2026. Lemur: Learned multi-vector retrieval. *arXiv preprint arXiv:2601.21853*.
- Yash Kankanampati, Yuxuan Zong, Nadi Tomeh, Benjamin Piwowarski, and Joseph Le Roux. 2026. *A voronoi cell formulation for principled token pruning in late-interaction retrieval models*. In *Proceedings of the 49th International ACM SIGIR Conference on Research and Development in Information Retrieval*, page 756–766. ACM.
- Omar Khattab and Matei Zaharia. 2020. Colbert: Efficient and effective passage search via contextualized late interaction over bert. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*, pages 39–48.
- Jinhyuk Lee, Zhuyun Dai, Sai Meher Karthik Duddu, Tao Lei, Iftekhar Naim, Ming-Wei Chang, and Vincent Zhao. 2023. Rethinking the role of token retrieval in multi-vector retrieval. *Advances in Neural Information Processing Systems*, 36:15384–15405.
- Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2021. *Retrieval-augmented generation for knowledge-intensive nlp tasks*. Preprint, arXiv:2005.11401.
- Minghan Li, Sheng-Chieh Lin, Barlas Oguz, Asish Ghoshal, Jimmy Lin, Yashar Mehdad, Wen-tau Yih, and Xilun Chen. 2023. Citadel: Conditional token

- interaction via dynamic lexical routing for efficient and effective multi-vector retrieval. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 11891–11907.
- Zhuchenyang Liu, Ziyu Hu, Yao Zhang, and Yu Xiao. 2026. Look in the middle: Structural anchor pruning for scalable visual rag indexing. *arXiv preprint arXiv:2601.20107*.
- António Loison, Quentin Macé, Antoine Edy, Victor Xing, Tom Balough, Gabriel Moreira, Bo Liu, Manuel Faysse, Céline Hudelot, and Gautier Viaud. 2026. *Vidore v3: A comprehensive evaluation of retrieval augmented generation in complex real-world scenarios*. Preprint, arXiv:2601.08620.
- Hongyu Lu, Feng Zhang, Wenwei Jin, Huanling Hu, Tianjun Shi, Shikai Jiang, Yao Hu, and Jiawei Li. 2026. Lrep: Low-rank compressibility guided visual token pruning for efficient lvlms. *arXiv preprint arXiv:2605.15621*.
- Yubo Ma, Jinsong Li, Yuhang Zang, Xiaobao Wu, Xiaoyi Dong, Pan Zhang, Yuhang Cao, Haodong Duan, Jiaqi Wang, Yixin Cao, and Aixin Sun. 2025. *Towards storage-efficient visual document retrieval: An empirical study on reducing patch-level embeddings*. Preprint, arXiv:2506.04997.
- Quentin Macé, António Loison, and Manuel Faysse. 2025. *Vidore benchmark v2: Raising the bar for visual retrieval*. Preprint, arXiv:2505.17166.
- Minesh Mathew, Dimosthenis Karatzas, and CV Jawahar. 2021. Docvqa: A dataset for vqa on document images. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pages 2200–2209.
- Hunter McCoy, Zikun Wang, and Prashant Pandey. 2026. *Jasper: Anns quantized for speed, built for change on gpu*. Preprint, arXiv:2601.07048.
- Chenghao Mo, Ben Karsin, Philip Adams, and Minjia Zhang. 2026. Vecflow-chamfer: A gpu-based data management system for high-performance multi-vector search on superchips. *Proceedings of the ACM on Management of Data*, 4(1 (SIGMOD)):1–26.
- Franco Maria Nardini, Cosimo Rulli, and Rossano Venturini. 2024. Efficient multi-vector dense retrieval using bit vectors. *arXiv preprint arXiv:2404.02805*.
- OpenSearch-AI. 2026. Ops-Colqwen3: State-of-the-Art Multimodal Embedding Model for Visual Document Retrieval. <https://huggingface.co/OpenSearch-AI/Ops-Colqwen3-4B>.
- Radek Osmulski, Gabriel de Souza P. Moreira, Ronay Ak, Mengyao Xu, Benedikt Schifferer, and Even Oldridge. 2025. *Miracl-vision: A large, multilingual, visual document retrieval benchmark*. Preprint, arXiv:2505.11651.
- Keshav Santhanam, Omar Khattab, Christopher Potts, and Matei Zaharia. 2022a. Plaid: an efficient engine for late interaction retrieval. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, pages 1747–1756.
- Keshav Santhanam, Omar Khattab, Jon Saad-Falcon, Christopher Potts, and Matei Zaharia. 2022b. Colbertv2: Effective and efficient retrieval via lightweight late interaction. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3715–3734.
- Jan Luca Scheerer, Matei Zaharia, Christopher Potts, Gustavo Alonso, and Omar Khattab. 2025. Warp: An efficient engine for multi-vector retrieval. In *Proceedings of the 48th international ACM SIGIR conference on research and development in information retrieval*, pages 2504–2512.
- Aditi Singh, Suhas Jayaram Subramanya, Ravishankar Krishnaswamy, and Harsha Vardhan Simhadri. 2021. FreshDiskANN: A fast and accurate graph-based ann index for streaming similarity search. *CoRR*, abs/2105.09613.
- Yao Tian, Zhoujin Tian, Xi Zhao, Ruiyuan Zhang, and Xiaofang Zhou. 2026. Gem: A native graph-based index for multi-vector retrieval. *arXiv preprint arXiv:2603.20336*.
- Juyuan Wang, Rongchen Zhao, Wei Wei, Yufeng Wang, Mo Yu, Jie Zhou, Jin Xu, and Liyan Xu. 2026. Comorag: A cognitive-inspired memory-organized rag for stateful long narrative reasoning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 33557–33565.
- Shixun Wu, Jinwen Pan, Jinyang Liu, Jiannan Tian, Ziwei Qiu, Jiajun Huang, Kai Zhao, Xin Liang, Sheng Di, Zizhong Chen, and Franck Cappello. 2025. *Boosting scientific error-bounded lossy compression through optimized synergistic lossy-lossless orchestration*. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, SC '25*, page 2024–2037. ACM.
- Zilin Xiao, Qi Ma, Mengting Gu, Chun-cheng Jason Chen, Xintao Chen, Vicente Ordonez, and Vijai Mohan. 2025. Metaembed: Scaling multimodal retrieval at test-time with flexible late interaction. *arXiv preprint arXiv:2509.18095*.
- Yuming Xu, Hengyu Liang, Jin Li, Shuotao Xu, Qi Chen, Qianxi Zhang, Cheng Li, Ziyue Yang, Fan Yang, Yuqing Yang, Peng Cheng, and Mao Yang. 2023. *SPFresh: Incremental in-place update for billion-scale vector search*. In *Proceedings of the 29th Symposium on Operating Systems Principles*, pages 545–561. Association for Computing Machinery.
- Yibo Yan, Mingdong Ou, Yi Cao, Xin Zou, Jiahao Huo, Shuliang Liu, James Kwok, and Xuming Hu.

2026. Sculpting the vector space: Towards efficient multi-vector visual document retrieval via prune-then-merge framework. *arXiv preprint arXiv:2602.19549*.
- Yibo Yan, Guangwei Xu, Xin Zou, Shuliang Liu, James Kwok, and Xuming Hu. 2025. Docpruner: A storage-efficient framework for multi-vector visual document retrieval via adaptive patch-level embedding pruning. *arXiv preprint arXiv:2509.23883*.
- Shi Yu, Chaoyue Tang, Bokai Xu, Junbo Cui, Junhao Ran, Yukun Yan, Zhenghao Liu, Shuo Wang, Xu Han, Zhiyuan Liu, and Maosong Sun. 2025. [Visrag: Vision-based retrieval-augmented generation on multi-modality documents](#). *Preprint*, arXiv:2410.10594.
- Amir Zandieh, Majid Daliri, and Insu Han. 2025. Qjl: 1-bit quantized jl transform for kv cache quantization with zero overhead. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 25805–25813.
- Suifeng Zhao, Zhuoran Jin, Sujian Li, and Jun Gao. 2025. [FinRAGBench-V: A benchmark for multi-modal RAG with visual citation in the financial domain](#). In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 4215–4249, Suzhou, China. Association for Computational Linguistics.
- Yuxuan Zong and Benjamin Piwowarski. 2025. Towards lossless token pruning in late-interaction retrieval models. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2407–2417.

A QJL estimator

We briefly review the asymmetric QJL estimator (Zandieh et al., 2025). Let $q, d \in \mathbb{R}^D$ be normalized vectors with $\|q\|_2 = \|d\|_2 = 1$. Given a random JL matrix $S \in \mathbb{R}^{m \times D}$ with i.i.d. Gaussian entries $S_{ij} \sim \mathcal{N}(0, 1)$, QJL stores only the sign sketch of the document vector:

$$H_S(d) = \text{sign}(Sd) \in \{-1, +1\}^m.$$

At query time, the query is projected but not quantized, and the inner product is estimated as

$$\widehat{\langle q, d \rangle}_{\text{QJL}} = \frac{\sqrt{\pi/2}}{m} \langle Sq, H_S(d) \rangle.$$

For non-normalized d , the estimator is multiplied by $\|d\|_2$; in our residual-direction setting, the directions are normalized and the residual norm is handled separately.

Zandieh et al. (Zandieh et al., 2025) show that this asymmetric estimator is unbiased:

$$\mathbb{E}_S \left[\widehat{\langle q, d \rangle}_{\text{QJL}} \right] = \langle q, d \rangle.$$

B Bayesian Shrinkage

This section derives the Bayesian shrinkage factor α used for residual score calibration. We present the derivation at the token level. Given a query token q and a document token d , let V_r be the low-rank basis of the document page and define the projection matrix $P_r = V_r V_r^\top$. After the low-rank score is computed exactly, the remaining score is

$$\langle q, d_{\text{remain}} \rangle = \langle q, (I - P_r)d \rangle.$$

Since d_{remain} lies in the orthogonal complement of V_r , only the remaining part of the query contributes:

$$\begin{aligned} \langle q, d_{\text{remain}} \rangle &= \langle (I - P_r)q, (I - P_r)d \rangle \\ &= \langle q_{\text{remain}}, d_{\text{remain}} \rangle. \end{aligned}$$

where

$$q_{\text{remain}} = (I - P_r)q, \quad d_{\text{remain}} = (I - P_r)d.$$

Thus, the remaining score can be written as

$$\langle q_{\text{remain}}, d_{\text{remain}} \rangle = \|q_{\text{remain}}\|_2 \|d_{\text{remain}}\|_2 \rho,$$

where

$$\rho = \langle u_q, u_d \rangle.$$

with

$$u_q = \frac{q_{\text{remain}}}{\|q_{\text{remain}}\|_2}, \quad u_d = \frac{d_{\text{remain}}}{\|d_{\text{remain}}\|_2}.$$

VISTA stores the residual norm $\|d_{\text{remain}}\|_2$ and the QJL sketch of its direction,

$$S_d = H_S(u_d) = \text{sign}(Su_d),$$

where $S \in \mathbb{R}^{m \times D}$ is a random Gaussian projection matrix. At query time, QJL estimates the directional similarity ρ by

$$\hat{\rho}_{\text{QJL}} = \frac{\sqrt{\pi/2}}{m} \langle Su_q, S_d \rangle.$$

For each row s_i of S , define

$$X_i = \sqrt{\frac{\pi}{2}} \langle s_i, u_q \rangle \text{sign}(\langle s_i, u_d \rangle)$$

And

$$\hat{\rho}_{\text{QJL}} = \frac{1}{m} \sum_{i=1}^m X_i.$$

Since $\langle s_i, u_q \rangle$ and $\langle s_i, u_d \rangle$ are jointly Gaussian with correlation ρ , the QJL estimator satisfies

$$\mathbb{E}[X_i] = \rho, \quad \mathbb{E}[X_i^2] = \frac{\pi}{2},$$

and therefore

$$\text{Var}(X_i) = \frac{\pi}{2} - \rho^2, \quad \text{Var}(\hat{\rho}_{\text{QJL}}) = \frac{1}{m} \left(\frac{\pi}{2} - \rho^2 \right).$$

Since the dominant information has been captured by the low-rank component, we model the remaining directional similarity as concentrated around zero:

$$\rho \sim \mathcal{N}(0, \sigma_0^2).$$

We view QJL as a noisy observation:

$$\hat{\rho}_{\text{QJL}} \mid \rho \sim \mathcal{N}(\rho, \sigma_{\text{QJL}}^2).$$

Since ρ is unknown during retrieval and the residual similarity is assumed to concentrate near zero after removing the low-rank component, we approximate the QJL variance by its conservative upper bound:

$$\sigma_{\text{QJL}}^2 = \frac{1}{m} \left(\frac{\pi}{2} - \rho^2 \right) \leq \frac{\pi}{2m}.$$

The posterior mean under the Gaussian prior and the Gaussian observation model is then

$$\mathbb{E}[\rho \mid \hat{\rho}_{\text{QJL}}] = \frac{\sigma_0^2}{\sigma_0^2 + \frac{\pi}{2m}} \hat{\rho}_{\text{QJL}}.$$

This gives the shrinkage update

$$\tilde{\rho} = \alpha \hat{\rho}_{\text{QJL}}, \quad \alpha = \frac{\sigma_0^2}{\sigma_0^2 + \frac{\pi}{2m}}.$$

Finally, the calibrated remaining score is

$$\langle \widehat{q_{\text{remain}}}, \widehat{d_{\text{remain}}} \rangle = \|q_{\text{remain}}\|_2 \|d_{\text{remain}}\|_2 \alpha \hat{\rho}_{\text{QJL}}.$$

This explains why shrinkage is more useful for smaller sketch dimensions m : the QJL noise term $\pi/(2m)$ is larger, so α applies stronger shrinkage. As m increases, the noise term decreases, and α approaches 1.

In practice, we use a variance-inflated prior by setting σ_0^2 to the empirical standard deviation s estimated from 32–64 sampled queries to avoid excessive shrinkage. We compute the directional similarities ρ on this sample to estimate the standard deviation s , and then plug it into the closed-form expression to obtain a separate α for each sketch dimension m .

C Visual Retrievers

We evaluate VISTA with two recent late-interaction visual document retrievers: Nemotron-ColEmbed-VL-4B-v2 (de Souza P. Moreira et al., 2026) and Ops-Colqwen3-4B (OpenSearch-AI, 2026). Both models encode text queries and page images into token-level multi-vector embeddings and use MaxSim-style scoring.

Retriever	Nemotron-ColEmbed-VL-4B-v2	Ops-Colqwen3-4B
Backbone	Qwen3-VL-4B-Instruct	Qwen3-VL-4B-Instruct
Retrieval type	Late interaction	Late interaction
Scoring	MaxSim	MaxSim
Model scale	4B variant	4B variant
Embedding dim.	2560	2560
Visual tokens/page	dynamic tiling	up to 1280
Context length	–	32K

Table 4: Visual retrievers used in our experiments.

For both retrievers, we extract page-level multi-vector embeddings offline and evaluate the serving-time retrieval engine after embedding extraction.

D Dataset details

We evaluate on ViDoRe-V2 (Macé et al., 2025) and ViDoRe-V3 (Loison et al., 2026). Table 5 summarizes the subsets used in our experiments. The query counts follow the official dataset cards and include translated queries.

Benchmark	Dataset	Abbr.	Corpus size	#Queries	#Qrels
ViDoRe V2	ESG reports	–	1,538	228	888
ViDoRe V2	Biomedical lectures	–	1,016	640	2,060
ViDoRe V2	Economics reports	–	452	232	3,628
ViDoRe V2	ESG reports human-labeled	–	1,538	52	128
ViDoRe V3	Computer Science	cs	1,360	1,290	6,294
ViDoRe V3	Physics	ph	1,674	1,812	13,068
ViDoRe V3	Finance-EN	fe	2,942	1,854	8,766
ViDoRe V3	Industrial	in	5,244	1,698	9,684
ViDoRe V3	Pharmaceuticals	pa	2,313	2,184	10,392
ViDoRe V3	Energy	en	2,229	1,848	6,618
ViDoRe V3	HR	hr	1,110	1,908	10,386
ViDoRe V3	Finance-FR	ff	2,384	1,920	8,808

Table 5: Dataset statistics for ViDoRe-V2 and ViDoRe-V3 used in our experiments. Corpus size denotes the number of page-level corpus items.

E System configuration

This section summarizes the representative configurations used in Table 1. For PLAID, we use 2-bit residual compression, IVF probing with $n_{\text{probe}} = 1$, and rerank the top-100 documents for most datasets. For the industrial subset, we use $n_{\text{probe}} = 2$ and rerank the top-200 documents due to its larger corpus size. For PLAID_{PQ}, we use PQ with 256 subquantizers and 8-bit codebooks, while keeping the same IVF and reranking configuration as PLAID.

For IGP, we use the same retrieval configuration as PLAID_{PQ} and run CPU search with 24 threads. We use 2-bit compression for IGP. For SAP, we set the pruning ratio to $\gamma = 0.2$. Following the original heuristic, we search over window sizes corresponding to roughly 60–80% token retention; for both Nemo-4B and Ops-4B, this corresponds to window sizes in $\{21, 22, 23, 24, 25, 26, 27, 28\}$.

For VISTA, we use $r = 16$ and $m = 512$ in Table 1. For most datasets, Stage 1 retrieves the top-400 candidates and Stage 2 reranks the top-100 candidates. For the industrial subset, Stage 1 retrieves the top-600 candidates and Stage 2 reranks the top-400 candidates due to its larger corpus size. The shrinkage factor α is computed using the procedure in Appendix B, with 64 sampled queries for estimating the prior variance σ_0 .

F Supplementary evaluation on multilingual MIRACL-Vision benchmark

We evaluate VISTA beyond the ViDoRe family on three larger MIRACL-Vision (Osmulski et al., 2025) datasets ($\sim 8\text{K}$ images each) spanning different languages. Specifically, we evaluate VISTA with Nemo-4B on Chinese (ZH), Hindi (HI), and Bengali (BN), using the same system configuration

Method	ZH			HI			BN		
	N@10	Lat. (ms)	Size (MB)	N@10	Lat. (ms)	Size (MB)	N@10	Lat. (ms)	Size (MB)
Oracle	70.4	–	25,177.7	52.9	–	23,118.6	51.3	–	24,503.0
PLAID	69.9	50.6	4,381.4	51.5	60.1	4,058.4	46.4	61.8	4,298.3
PLAID _{PQ}	69.8	10.9	1,892.8	52.4	17.8	1,761.6	47.6	20.8	1,860.6
VISTA	69.5	7.3	1,355.3	52.0	9.9	1,250.9	50.6	10.2	1,327.6

Table 6: Retrieval performance, latency, and index size on three MIRACL-Vision datasets.

Method	Energy			CS			Pharmaceuticals			Physics		
	N@10	Lat. (ms)	Size (MB)	N@10	Lat. (ms)	Size (MB)	N@10	Lat. (ms)	Size (MB)	N@10	Lat. (ms)	Size (MB)
Oracle	64.7	–	18,435.2	77.1	–	11,619.8	65.8	–	19,180.8	46.9	–	14,300.2
PLAID	64.5	133.2	3,053.3	76.3	131.4	1,902.4	65.6	132.6	3,168.9	46.4	131.1	2,322.2
PLAID _{PQ}	64.1	25.1	1,107.9	76.3	22.6	713.9	65.5	24.7	1,146.5	46.3	22.4	859.0
VISTA	64.0	14.6	594.3	76.4	16.1	363.3	65.8	15.3	617.8	47.1	15.3	447.1

Table 7: Retrieval performance, latency, and index size with Llama-Nemotron-ColEmbed-VL-3B-v2 on four ViDoRe V3 datasets.

as in Table 1.

As shown in Table 6, VISTA consistently achieves smaller index sizes and lower retrieval latency while maintaining competitive retrieval accuracy across all three languages. Specifically, VISTA reduces index size by $3.23\text{--}3.24\times$ compared with PLAID and $1.40\text{--}1.41\times$ compared with PLAID_{PQ}. It also accelerates retrieval by $6.1\text{--}6.9\times$ over PLAID and $1.5\text{--}2.0\times$ over PLAID_{PQ}. Meanwhile, VISTA achieves comparable or better NDCG@10, with differences of -0.4 to $+4.2$ points relative to PLAID and -0.4 to $+3.0$ points relative to PLAID_{PQ}. These results demonstrate that VISTA’s efficiency gains extend beyond the ViDoRe datasets and generalize across larger multilingual retrieval benchmarks.

G Supplementary evaluation on a different VDR architecture

We evaluate VISTA with a different VDR architecture, Llama-Nemotron-ColEmbed-VL-3B-v2 (de Souza P. Moreira et al., 2026), which uses a Llama 3.2 backbone and a SigLIP2 vision encoder, on four ViDoRe V3 datasets: Energy, Computer Science, Pharmaceuticals, and Physics.

As shown in Table 7, VISTA consistently reduces index size by $5.1\text{--}5.2\times$ compared with PLAID and $1.9\text{--}2.0\times$ compared with PLAID_{PQ} while accelerating retrieval by $8.2\text{--}9.1\times$ and $1.4\text{--}1.7\times$, respectively. At the same time, VISTA maintains comparable or better retrieval accuracy, with NDCG@10 differences ranging from -0.5 to $+0.7$ points relative to PLAID and from -0.1 to $+0.8$

points relative to PLAID_{PQ}. These results show that VISTA’s efficiency gains generalize to a different VDR architecture and vision encoder.

H Calibration sensitivity

We evaluate the sensitivity of VISTA’s shrinkage factor to the calibration-set size and query domain on four ViDoRe V3 datasets under an aggressive compression setting ($r = 4$, $m = 128$). We use 32, 64, or 128 calibration queries drawn either from the target dataset (in-domain) or from the `finance_en` dataset (out-of-domain).

Dataset	w/o α	In-domain (# samples)			Out-of-domain (# samples)		
		32	64	128	32	64	128
Energy	62.32	62.83	62.82	62.82	62.87	62.85	62.83
CS	74.25	75.65	75.61	75.61	75.67	75.69	75.69
Pharmaceuticals	62.79	62.92	62.92	62.92	62.95	62.95	62.96
Physics	47.17	47.99	47.98	47.99	48.08	48.07	48.08

Table 8: Sensitivity to calibration-set size and query domain under the aggressive compression setting ($r = 4$, $m = 128$). Results are reported in NDCG@10.

As shown in Table 8, the calibrated performance is highly stable across both calibration-set sizes and query domains. Across different sample sizes within each domain, NDCG@10 varies by at most 0.10 points, indicating that a small calibration set is sufficient to estimate the shrinkage factor reliably. Moreover, using out-of-domain Finance queries achieves performance comparable to, and in some cases slightly better than, in-domain calibration. In all settings, calibration consistently improves over the uncalibrated baseline (w/o α) by 0.13–1.44 points, demonstrating that the shrinkage factor

Method	Energy		CS		Pharmaceuticals		Physics	
	Time (s)	Throughput (Pages/s)	Time (s)	Throughput (Pages/s)	Time (s)	Throughput (Pages/s)	Time (s)	Throughput (Pages/s)
PLAID	41.8	53.2	13.0	105.0	26.0	89.0	17.7	94.5
PLAID _{PQ}	31.3	71.2	14.3	94.9	35.1	65.8	19.5	85.9
VISTA	33.4	66.6	16.5	82.5	28.3	81.7	19.1	87.8

Table 9: Index-construction time and throughput on four ViDoRe V3 datasets.

is insensitive to both calibration-set size and query domain.

I Indexing overhead

We compare VISTA’s index construction time and throughput with PLAID and PLAID_{PQ} across four ViDoRe V3 datasets, using the same configurations as in Table 1. As shown in Table 9, VISTA achieves comparable index-construction efficiency across all four datasets.

Running a full SVD on each page matrix, typically of size 744×2560 , would be costly at scale, since it computes the complete singular basis, whereas VISTA retains at most 16 components. We therefore adopt the Gram-matrix formulation of truncated SVD, using batched matrix multiplications to construct smaller Gram matrices and then performing eigen decomposition on them. This formulation is substantially more amenable to GPU parallelism. As shown in Table 9, VISTA achieves a throughput of 66–88 pages/s across the four datasets, comparable to PLAID and PLAID_{PQ}. At this throughput, indexing one million pages would take approximately 3.2–4.2 GPU-hours, which is an acceptable one-time indexing cost at the million-page scale.

J Potential support for dynamic index updates

VISTA does not fundamentally require a full index rebuild when documents change, since the SVD and compressed representation are constructed independently for each document rather than over the entire corpus. This per-document design provides a natural path toward supporting dynamic index updates.

Insertion. For a newly inserted document, only the new page needs to be processed and compressed. At the indexing throughput reported in Appendix I, this corresponds to approximately 11–15 ms per page. The resulting mean vector, low-rank components, residual norms, and sketches can then be appended to the existing index storage without re-computing representations for existing documents.

Deletion. Deletion can be supported via lazy deletion, where a deleted document or page is marked inactive with an $O(1)$ mask update, and inactive entries are filtered during retrieval and reranking. Following dynamic ANN systems that separate online updates from periodic index maintenance (Singh et al., 2021; Xu et al., 2023), a background compaction process can reclaim fragmented storage periodically.

Modification. Document modification can be handled via delete-then-insert: the old representation is marked inactive, while the modified document is recomputed and appended as a new representation.

These operations avoid rebuilding the full index. However, an efficient GPU implementation remains non-trivial (Wu et al., 2025). Recent GPU-native dynamic ANNS systems similarly require specialized designs to support efficient updates (McCoy et al., 2026). For VISTA, jointly managing its fixed- and variable-size index components would require careful design of GPU memory layout, allocation, and synchronization, which we leave to future work.