

Leveraging AV1 motion vectors for Fast and Dense Feature Matching

Julien Zouein, Hossein Javidnia, François Pitié, Anil Kokaram
Sigmedia Group,

Department of Electronic and Electrical Engineering, *Trinity College Dublin*, Dublin, Ireland
{zoueinj, hossein.javidnia, pitief, anil.kokaram}@tcd.ie

Abstract—We repurpose AV1 motion vectors to produce dense sub-pixel correspondences and short tracks filtered by cosine consistency. On short videos, this compressed-domain front end runs comparably to sequential SIFT while using far less CPU, and yields denser matches with competitive pairwise geometry. As a small SfM demo on a 117-frame clip, MV matches register all images and reconstruct 0.46–0.62M points at 0.51–0.53px reprojection error; BA time grows with match density. These results show compressed-domain correspondences are a practical, resource-efficient front end with clear paths to scaling in full pipelines.

Index Terms—AV1, Motion Vectors, Structure from Motion.

I. INTRODUCTION

Classical vision pipelines expend substantial computation on feature extraction and exhaustive matching, often consuming the majority of wall-clock time and CPU resources. Meanwhile, modern video codecs (e.g., AV1) embed motion information and block structure that can be repurposed for correspondence discovery at a fraction of the cost. Given that most digital video is stored and transmitted in a compressed format, with motion vectors required for decoding and readily available in the bitstream, re-using these pre-computed motion vectors is an inexpensive proxy for correspondence discovery. Prior work has explored this direction: in 2014, Kantorov et al. [1] showed that motion vectors from H.264 could be used to speed up action recognition. More recently, Richard N. C. Turner [2] explored a pipeline for Simultaneous Localization and Mapping (SLAM) from motion vectors extracted from the H.265 bitstream.

We present an efficient approach that converts AV1 motion vectors (MVs) into sub-pixel point-to-point correspondences and builds *multi-frame tracks*, unlocking long-baseline coverage. We evaluate the resulting correspondences with pairwise geometric tests and efficiency metrics, and include a small SfM demonstration; comprehensive 3D evaluation and BA scaling are left to a forthcoming longer paper.

The main contributions of this paper are:

- i) Sub-pixel correspondence extraction from AV1 MVs via precise target placement.
- ii) Track propagation and cosine filtering to extend coverage to non-adjacent pairs and prune outliers, producing a triangular image adjacency pattern.

- iii) Pairwise geometry and efficiency protocol reporting inlier ratio, median Sampson error, and pre-stage runtime / CPU load; BA only in a small demo.

II. METHOD

In this section, we provide an overview of Motion Vectors in AV1 and how to extract them. Next, the structure of our proposed framework is described in detail.

A. Converting Motion Vectors into Sub-pixel Point-to-Point Correspondences

The motion estimation process in modern hybrid video codecs like AV1 and H.265 relies on a shared block-based prediction architecture. In this paradigm, a frame is partitioned into blocks of various dimensions; AV1 supports a notably wide range from (4×4) to (128×128) . The prediction for each block is generated by referencing a motion vector (MV) which points to a source location in a designated Reference Frame. The AV1 specification accommodates up to seven such reference frames [3], [4] which can be configured for forward (predicting from future frames) or backward prediction. Our implementation focuses on a backward-predictive streaming configuration, where all motion vectors refer to locations in previously decoded frames.

To improve prediction accuracy, AV1 employs a high-precision motion model. The motion vectors are defined with a sub-pixel precision up to 1/8th of a pixel; in our experiments, we use 1/4th pixel precision. Pixel precision is achieved by generating an interpolated sub-pixel grid using separable filters [5]. This allows for a more finely-grained and accurate motion compensation. Within the bitstream itself, these motion vectors are encoded as integer values for compactness. To convert these raw integer values back to their true physical scale upon extraction, a division by a factor of 8 is required.

Some blocks are encoded using INTRA prediction or are designated as SKIPPED blocks. These blocks do not have any associated motion vectors, resulting in a $(0,0)$ motion vector during extraction. We do not consider blocks with a $(0,0)$ motion vector, as this value is ambiguous and can represent either a truly static block or a block with no motion information.

For each block (p,q) in a frame n , we emit a source keypoint at the center of the block and a *target point* displaced by the motion vector $v_{n,m}(p,q)$ in the reference frame m . The

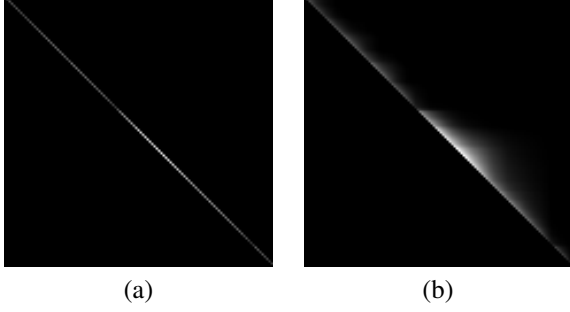


Fig. 1: Visualisation of the impact of using tracks on the adjacency matrix. (a): adjacency before using tracks, (b): adjacency after using tracks and cosine filtering. For a given row i and column j , the whiter a pixel is, the more matches there are.

generated target point is added to the source keypoints of frame m .

B. Track Propagation and Cosine Filtering

We build tracks by linking consistent correspondences across consecutive frames and discard short tracks (length < 3). Denote by $\mathbf{v}_{i,j}(k) \in \mathbb{R}^2$ the motion vector for keypoint k from frame i to frame j , and by $t_k = \{(n, \mathbf{x}_n), (m, \mathbf{x}_m), (\ell, \mathbf{x}_\ell), \dots\}$ the ordered set of detections (with $n < m < \ell < \dots$) that forms the track for k .

Codec motion vectors are block-prediction signals rather than true object motion, so some vectors are unreliable. We therefore enforce directional consistency within each track using a cosine test between adjacent MV segments. For any consecutive triple $(n, m, \ell) \in t_k$ we require

$$\frac{\langle \mathbf{v}_{n,m}(k), \mathbf{v}_{m,\ell}(k) \rangle}{\|\mathbf{v}_{n,m}(k)\| \|\mathbf{v}_{m,\ell}(k)\|} \geq 1 - \epsilon, \quad (1)$$

with a tolerance $\epsilon \in (0, 1)$. By default we use $\epsilon=0.1$ (i.e., $\cos \geq 0.9$); for sequences with large temporal gaps we disable the filter by setting $\epsilon=1$ (threshold 0). If either vector has magnitude below a small τ , we skip the test to avoid numerical instability.

Using cosine similarity in tracks allows us to filter out motion vectors with a bad direction. Motion vectors with bad cosine similarity are deleted and not considered for matches.

This processing step allows us to propagate good matches to non-adjacent frames, augmenting the number of matches. This transforms the diagonal adjacency (adjacent pair only for video and sequential matching) into a *triangular* pattern with broad-baseline coverage as shown on Figure 1.

Figure 2 is a representation of our current pipeline.

C. Pairwise Geometry Estimation and Scoring

For each image pair (i, j) we estimate a calibrated essential matrix $\mathbf{E}_{ij}$ or, for near-planar cases, a homography $\mathbf{H}_{ij}$ using (LO-)RANSAC with the fixed hyperparameters from §*Baseline and settings*. Points are expressed in normalized

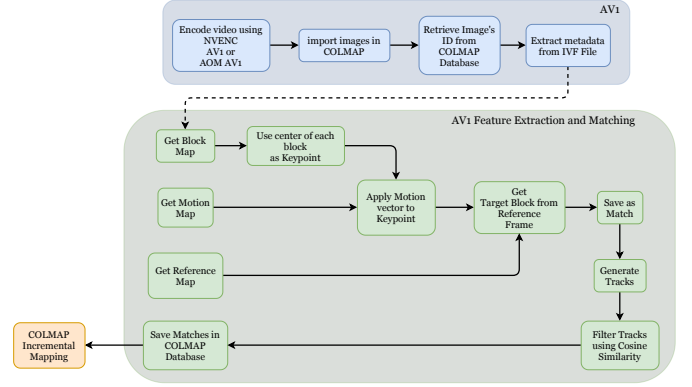


Fig. 2: Pipeline from AV1 bitstream to correspondences and tracks. We parse block/motion/reference maps, generate sub-pixel point-to-point matches, build and filter tracks and export correspondences. Downstream SfM/BA is not the focus; we include a small demo in §III-C.

camera coordinates $\tilde{\mathbf{x}} = K^{-1}\mathbf{x}$. For $\mathbf{E}$, the residual is the Sampson error [6]:

$$SE(x, x') = \frac{(x'^T \mathbf{E} x)^2}{(\mathbf{E} x)_1^2 + (\mathbf{E} x)_2^2 + (\mathbf{E}^T x')_1^2 + (\mathbf{E}^T x')_2^2},$$

which approximates the epipolar distance. We fit both $\mathbf{E}$ (five-point) and $\mathbf{H}$ (DLT) and select the model with the larger inlier set, breaking ties by lower median residual (favoring $\mathbf{E}$ unless $\mathbf{H}$ explains substantially more matches).

Scoring. We report (i) the RANSAC inlier ratio and (ii) the median Sampson error over inliers. All metrics use the same correspondences and identical RANSAC settings across methods.

III. EXPERIMENTS

We evaluate our method against traditional baselines using the datasets and settings described below. Performance is measured along three dimensions: pairwise geometry quality as defined in Section II-C, computational efficiency (runtime, CPU usage), and correspondence coverage (match count). We do not include the time for the final SfM reconstruction, as our goal is to evaluate the front-end pipeline.

A. Experimental Setup

We run experiments on a 12th Gen Intel(R) Core(TM) i7-12700K with 64GB RAM, running Ubuntu 22.04. The hardware encoding was performed using an NVIDIA RTX 6000 Ada.

B. Dataset

We evaluate our pipeline using outdoor sequences. Our test data includes *Gerrard Hall*, and *Person Hall* from COLMAP [7] collection of datasets. It is important to note that we use a subset of each dataset. Our technique requires images to have the same dimensions and to be temporally adjacent. These two image sets were converted into videos to meet our method's requirement for sequential inputs. Due to

TABLE I: SfM on a short video ($n=117$ frames). Times are seconds; *front-end* = features+matching+geometric verification, *BA* = mapping/BA. GPU methods marked * was performed using a NVIDIA T4.

Method	Reg. imgs	#3D pts	Reproj. err (px)	Time (front-end / BA)
MV (ours AOM)	117	460,152	0.51	122 / 1035
MV (ours, NVENC)	117	615,945	0.53	257 / 1262
SIFT-Seq	117	54,755	0.30	114 / 346
DISK*+LightGlue*	117	85,351	1.07	1067 / 1495
SP*+SuperGlue*	117	28,589	1.34	1320 / 412



Fig. 3: 3D reconstruction of Sequence Paris Seq 1. using MV (NVENC).

the large temporal gaps, we disable the cosine filter for these two sequences by setting $\epsilon = 1$ (threshold 0) to preserve track continuity.

We also recorded three custom sequences:

- Dublin Seq. 1: having 329 images and a resolution of 1080×1920 at 24 FPS, recorded at night.
- Paris Seq. 1: having 117 frames and a resolution of 1080×1920 at 10 FPS, recorded in daylight.
- Paris Seq. 2: having 122 frames and a resolution of 1920×1080 at 10 FPS, recorded in daylight.

All three sequences were recorded using an iPhone 15 Pro Max. We added the first 230 frames of the Sequence 0 from KITTI Odometry dataset [8].

C. SfM

Quantitative results. Table I reports SfM statistics on a 117-frame clip using sequential exhaustive pairing and identical SIMPLE_RADIAL intrinsics and mapper settings across all methods. Our motion-vector pipeline (MV) registers all 117 images and reconstructs substantially more 3D points (460k–616k) than feature-matching baselines, with competitive reprojection error (0.51–0.53 px). The front-end time of MV is moderate (122–257 s); the larger BA time is expected because BA scales with the number of tracks/points. SIFT-Seq attains the lowest reprojection error (0.30 px) but yields an order-of-magnitude fewer points. GPU deep matchers (DISK [9]+LightGlue [10] and SP [11]+SuperGlue [12]) run on GPU (marked *), but are either much slower or reconstruct very few points under identical mapping settings. Figure 3 shows the result of 3D reconstruction using our MV (NVENC).

D. Encoder Parameters

For AV1 encoding, we employed libaom-av1 [13] (v3.12.1) as the software encoder, and FFmpeg [14] (n.6.0-22) to access

TABLE II: Efficiency of pre-stages only (Median over all sequences). Times in seconds, CPU is average during the pre-stages. The underlined value is the best overall result.

Method	Pre-Processing (s)	Feature Matching (s)	CPU Usage (%)
NVENC AV1 (our)	13.06	201.03	4.14
AOM AV1 (our)	12.23	104.86	4.27
SIFT Sequential	17.25	<u>72.33</u>	46.65
SIFT Exhaustive	18.44	375.32	95.15

TABLE III: Median Sampson error per method per sequence. Lower is better. Our method using libaom av1 achieves the best overall.

Seq.	NVENC AV1	AOM AV1	SIFT Sequential	SIFT Exhaustive
Dublin Seq 1	0.0003	0.0004	0.015	0.015
Paris Seq 1	<u>9.62E-05</u>	<u>5.82E-05</u>	0.015	N/A
Paris Seq 2	1.44E-05	<u>9.8E-06</u>	0.083	N/A
KITTI Seq 0	0.002	<u>0.002</u>	0.111	N/A
Gerrard Hall	0.004	<u>0.003</u>	0.005	N/A
Person Hall	0.003	<u>0.001</u>	0.022	N/A

the NVIDIA AV1 hardware implementation (nvenc). Motion vectors from AV1-encoded IVF files were extracted using the inspect tool from AOM library.

Encoders are set up in a Streaming Configuration (S3-SCC-03 [15]) where all the reference frames are *in the past*, with only one intra-frame (first frame of the sequence) and all motion vectors pointing to the previous frame. We use preset 1 for NVENC-AV1 and cpu-used=6 for libaom.

E. Baseline and Settings

We use COLMAP SIFT [16] with exhaustive matching (all pairs) and also report sequential matching as an alternative.

For geometric verification and model estimation, we apply RANSAC with fixed hyperparameters across all evaluated methods to maintain consistency. We perform multiple independent runs with RANSAC and report median performance metrics over these repeated runs to reduce the impact of outliers. We estimate $\mathbf{E}$ with the five-point algorithm and RANSAC with default parameters (max_error = 4.0, min_inlier_ratio=0.25, max_num_trials=10000). Sampson error is computed on normalized coordinates.

Regarding cosine similarity filtering, we set $\epsilon = 0.1$ in equation 1, to ensure a cosine similarity above 0.9 except for *Person Hall* and *Gerrard Hall* sequences, where the high temporal difference between each frame is not compatible with this filtering.

F. Main Qualitative Results

As shown in Table II, our AOM AV1 method is over 3x faster than SIFT Exhaustive while using 95% less CPU power. While SIFT Sequential has a comparable runtime, its CPU usage is an order of magnitude higher. This efficiency gap is visualized in Figure 4a-4c.

In terms of geometric accuracy, our method consistently achieves the lowest median Sampson error (Table III) and superior or competitive inlier ratios (Table IV). Furthermore, our approach generates a denser set of matches than traditional methods (Figure 4d).

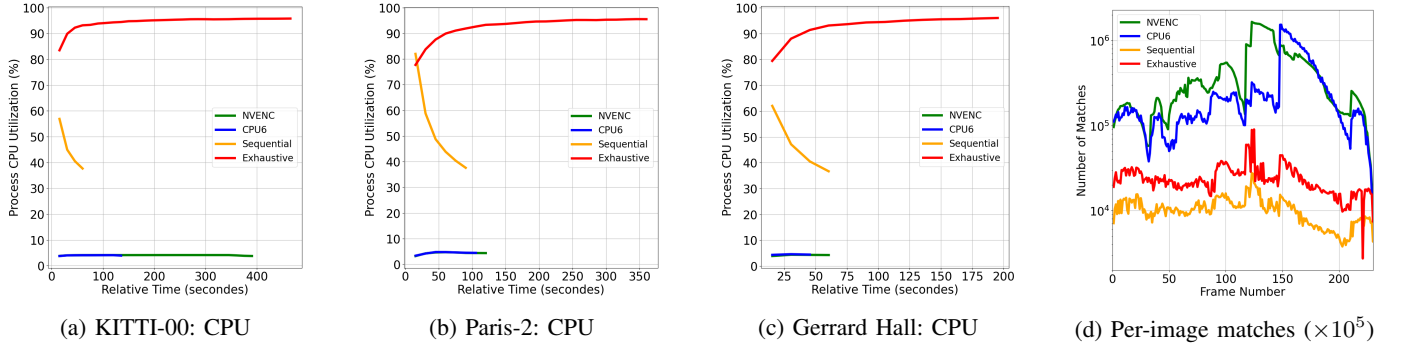


Fig. 4: Process CPU Utilization (%) for each method on (a) KITTI-00, (b) Paris-2, and (c) Gerrard Hall, and (d) the corresponding per-image match counts. Codec-based motion-vector pipelines (NVENC-AV1, AOM-AV1) produce higher match counts than classical feature pipelines under identical intrinsics and mapper settings.

TABLE IV: Median Inlier Ratio per method per sequence, higher is better. Our method using libaom on video sequences achieves highly competitive inlier ratios compared to classical method.

Seq.	NVENC AV1	AOM AV1	SIFT Sequential	SIFT Exhaustive
Dublin Seq 1	0.99	0.98	0.96	0.51
Paris Seq 1	0.96	0.99	0.94	0
Paris Seq 2	0.93	0.99	0.91	0
KITTI Seq 0	0.98	0.95	0.96	0
Gerrard Hall	0.47	0.96	0.95	0
Person Hall	0.43	0.96	0.98	0

IV. DISCUSSION AND LIMITATIONS

We observe that dense correspondence, while being beneficial for robustness and coverage, may significantly increase the computational cost of downstream bundle adjustment (BA) and reconstruction pipelines. This motivates future investigation into strategies to reduce the number of keypoints, therefore balancing geometric fidelity with efficiency.

The current implementation is also not optimized. Work will be done to reduce reading and writing to disk, reduce the number of loops, and parallelize the processing. In particular, the use of DAVID decoder to extract metadata from AV1 bitstream might be explored.

A comprehensive Structure from Motion evaluation will be conducted. Reconstruction completeness (number of registered camera and triangulated points), Reprojection accuracy, 3D point cloud quality (using the Chamfer and Hausdorff distances to ground truth) and bundle adjustment scalability will be presented in a separate submission.

V. CONCLUSION

We have introduced a method leveraging AV1 motion vectors for dense, sub-pixel correspondence matching in Structure from Motion. Our approach achieves substantial speed and efficiency gains over SIFT-based baselines, while delivering competitive geometric accuracy and match density. The results highlight the potential of compressed-domain features for scalable, resource-efficient 3D vision. Future work will assess the impact on full SfM reconstruction and Bundle Adjustment performance.

REFERENCES

- [1] V. Kantorov and I. Laptev, “Efficient feature extraction, encoding, and classification for action recognition,” in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 2593–2600.
- [2] R. N. Turner, N. K. Banerjee, and S. Banerjee, “Mov-slam: Using motion vectors for real-time single-cpu visual slam,” in *2023 Seventh IEEE International Conference on Robotic Computing (IRC)*, 2023, pp. 51–58.
- [3] Z. Liu, D. Mukherjee, W.-T. Lin, P. Wilkins, J. Han, and Y. Xu, “Adaptive multi-reference prediction using a symmetric framework,” *Electronic Imaging*, vol. 2017, no. 2, 2017.
- [4] X. Zhao, S. Liu, A. Grange, and A. Norkin, “Tool description for av1 and libaom,” *Alliance for Open Media, Codec Working Group, Document: CWG-B0780*, 2021.
- [5] J. Han, B. Li, D. Mukherjee, C.-H. Chiang, A. Grange, C. Chen, H. Su, S. Parker, S. Deng, U. Joshi, Y. Chen, Y. Wang, P. Wilkins, Y. Xu, and J. Bankoski, “A technical overview of av1,” 2021. [Online]. Available: <https://arxiv.org/abs/2008.06091>
- [6] P. Sampson, “Sampson, p.d.: Fitting conic sections to “very scattered” data: An iterative refinement of the bookstein algorithm. *comput. graphics image process.* 18, 97-108,” *Computer Graphics and Image Processing*, vol. 18, pp. 97–108, 01 1982.
- [7] J. L. Schönberger and J.-M. Frahm, “Structure-from-motion revisited,” in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.
- [8] A. Geiger, P. Lenz, and R. Urtasun, “Are we ready for autonomous driving? the kitti vision benchmark suite,” in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.
- [9] M. J. Tyszkiewicz, P. Fua, and E. Trulls, “Disk: Learning local features with policy gradient,” 2020. [Online]. Available: <https://arxiv.org/abs/2006.13566>
- [10] P. Lindenberger, P.-E. Sarlin, and M. Pollefeys, “Lightglue: Local feature matching at light speed,” 2023. [Online]. Available: <https://arxiv.org/abs/2306.13643>
- [11] D. DeTone, T. Malisiewicz, and A. Rabinovich, “Superpoint: Self-supervised interest point detection and description,” 2018. [Online]. Available: <https://arxiv.org/abs/1712.07629>
- [12] P.-E. Sarlin, D. DeTone, T. Malisiewicz, and A. Rabinovich, “Superglue: Learning feature matching with graph neural networks,” 2020. [Online]. Available: <https://arxiv.org/abs/1911.11763>
- [13] “libaom-av1,” <https://aomedia.googlesource.com/aom/>.
- [14] “Ffmpeg,” <https://ffmpeg.org/>.
- [15] “3rd generation partnership project; technical specification group services and system aspects; 5g video codec characteristics,” <https://www.3gpp.org/specifications-technologies/specifications-by-series>.
- [16] D. G. Lowe, “Distinctive image features from scale-invariant keypoints,” *Int. J. Comput. Vision*, vol. 60, no. 2, p. 91–110, Nov. 2004. [Online]. Available: <https://doi.org/10.1023/B:VISI.0000029664.99615.94>