Computational VBSME Optimization: Motion-Only Partitioning with Progressive Early Termination

Vinutha Mallikarjunappa
Department of Electronics Engineering (VLSI Design and Technology), NMAM Institute of Technology (NMAMIT), Nitte (Deemed to be University), Nitte, Udupi, Karnataka, India | Visvesvaraya Technological University, Belagavi, Karnataka, India
m.vinutha24@gmail.com (corresponding author)

T. M. Manu
Department of Electronics and Communication Engineering, KLE Institute of Technology (KLEIT), Hubballi, Karnataka, India | Visvesvaraya Technological University, Belagavi, Karnataka, India
manutmece@kleit.ac.in

Received: 26 November 2025 | Revised: 1 January 2026 | Accepted: 6 January 2026

 DOI: https://doi.org/10.48084/etasr.16435

ABSTRACT

This study presents two motion-only Variable Block Size Motion Estimation (VBSME) algorithms: Hierarchical Motion-Only VBSME (HMO-VBSME) and Integrated Bit-Planar Motion-Only Block Matching (IBPMO-BM). Both algorithms utilize pure motion vector characteristics for block size decisions while extending traditional motion estimation to support 32×32 blocks through a four-level structure (32→16→8→4). Both algorithms integrate directional search pattern optimization that adapts search positions based on previous motion vectors, reducing search complexity by 44% to 89%. HMO-VBSME employs hierarchical planar Sum of Absolute Differences (SAD) computation with multi-level early termination. IBPMO-BM introduces bit-SAD computation with early termination, using motion-driven block size selection. Experimental results on standard video sequences demonstrate that both algorithms maintain PSNR values comparable to Full Search Block Matching Estimation (FSBME), while achieving computational savings of 77% to 97%. A significant proportion of 32×32 blocks are selected (28.4% to 97.4% across sequences), reducing motion vector overhead and improving compression efficiency. These characteristics make both algorithms ideal for ultra-low-complexity real-time video coding applications.

Keywords-motion estimation; variable block size; hierarchical SAD; bit-planar processing; motion-only analysis; video compression

I. INTRODUCTION

Motion Estimation (ME) is a fundamental component of video encoding and often dominates overall computational complexity. Although Full Search Block Matching (FSBME) provides accurate motion vectors, its excessive computational cost limits real-time and hardware-efficient implementations. Variable Block Size Motion Estimation (VBSME) improves efficiency by adapting block sizes to motion activity, typically using metrics such as Sum of Absolute Differences (SAD) or Sum of Squared Differences (SSD) along with search strategies such as full, spiral, or diamond search. Reconstruction quality is commonly evaluated using PSNR or MSE. Several fast motion estimation techniques have been proposed to reduce computational complexity, including Three-Step Search (TSS), New Three-Step Search (NTSS) [1], Diamond Search (DS) [2], hierarchical and one-dimensional searches [3-5], as well as early termination and content-adaptive methods [6-8]. Modern standards such as H.264/AVC support variable block sizes from 16×16 to 4×4 [9-11]; however, larger block sizes remain underutilized despite their effectiveness in low-motion regions. More recently, learning-based approaches using convolutional neural networks [12-13], reinforcement learning [14], and attention mechanisms [15] have shown promising performance, but their reliance on extensive training and specialized hardware limits their suitability for low-complexity or real-time systems [16, 17].

This study presents two motion-only ME schemes based solely on algorithmic optimization, eliminating both rate-distortion dependency and learning overhead. The first method, Hierarchical Motion-Only VBSME (HMO-VBSME), combines hierarchical SAD computation with motion-driven block size selection, while the second, Integrated Bit-Planar Motion-Only Block Matching (IBPMO-BM), uses bit-plane decomposition and progressive XOR-based matching with

early termination to reject poor candidates. Both employ a 32×32 hierarchical block structure recursively decomposed into 16×16, 8×8, and 4×4 sub-blocks, with 4×4 as the basic processing unit. In IBPMO-BM, bit-plane processing enables coarse-to-fine motion evaluation, significantly reducing computation while maintaining accuracy (Figure 1).

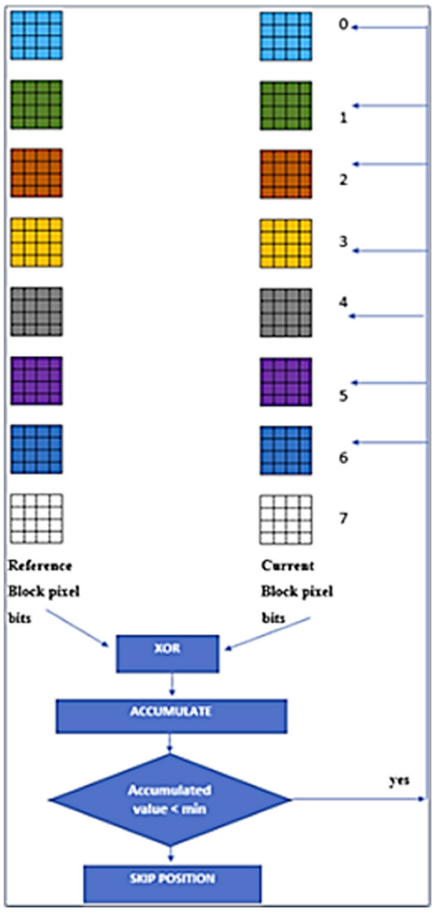

Fig. 1. Bit-Plane-based motion estimation flow using early termination.

Overall, this work presents a hardware-aware hierarchical motion estimation framework that combines multi-level cost approximation, directional search, and early termination across multiple hierarchy levels. By explicitly modelling computational operations and applying threshold-based pruning before full SAD evaluation, the proposed methods achieve significant reductions in computational complexity without degrading motion estimation performance. Major contributions include:

- Develops two motion-only motion estimation algorithms, HMO-VBSME and IBPMO-BM, that avoid rate-distortion optimization and learning-based inference, enabling low-complexity and hardware-friendly implementation.
- Introduces a 32×32 hierarchical block framework with recursive decomposition into 16×16, 8×8, and 4×4 blocks, allowing efficient variable block size motion estimation with SAD reuse.
- Proposes a bit-planar early-rejection mechanism using progressive XOR-based evaluation to prune poor motion candidates before full SAD computation.
- A comprehensive computational complexity and block-size distribution analysis demonstrates substantial reductions in arithmetic operations while maintaining reconstruction quality.

II. METHODOLOGY

This section presents the proposed algorithms, HMO-VBSME and IBPMO-BM. Both algorithms aim to reduce computational complexity for motion estimation. The number of computations due to comparisons, additions, absolute calculations, subtractions, and XOR calculations is determined at each level.

A. Hierarchical Motion-Only VBSME (HMO-VBSME)

HMO-VBSME is a motion-sensitive, adaptive search algorithm that consciously incorporates early termination at multiple resolution levels, thereby significantly reducing computational overhead. The algorithm operates by progressively refining the search process based on motion activity and follows the steps outlined.

1. Extract the frames from the video set.
2. Consider 2 consecutive frames, the reference frame and the current frame.
3. Divide the current frame into non-overlapping macroblocks of size 32×32.
4. Extract the 32×32 macroblock for which the motion vector has to be found.
5. Generate an adaptive search pattern [17] based on the previous MV for the corresponding block.
   - For the first frame: Use the full spiral search pattern [18] from (−4,−4) to (+4,+4), generating 81 positions in a spiral outward from the center.
   - For subsequent frames: Use directional spiral search based on previous motion vectors.
6. For each search position:
   a. Initialize accumulated SAD = 0.
   b. For each 4×4 sub-block (64 total):
      i. Compute Hierarchical SAD with early termination.
         - Level 1: Sample every 4th pixel.
            1. Calculate SAD and scale to a full block estimate.
            2. If Level1_SAD ≥ min_sad_threshold → Return infinity and terminate early.

- Level 2: Sample every 2nd pixel.
  1. Calculate SAD and scale to a full block estimate.
  2. If Level2_SAD ≥ min_sad_threshold → Return infinity and terminate early.
- Level 3: Calculate full SAD on all 16 pixels. This is the final SAD value (no more early termination possible).

ii. Add subblock SAD to the accumulated SAD.

- If accumulated SAD ⩾ min_sad_threshold → Skip remaining sub-blocks (early termination).

c. Aggregate 4×4 SADs to compute 8×8, 16×16, and 32×32 SADs.

7. Find the best motion vector for the 32×32 block.
8. Calculate motion-only block size using MV magnitude.
   - $If\ |mv\_x| + |mv\_y| = 0$: select 32×32
   - $Else\ if\ |mv\_x| + |mv\_y| < 2$: select 16×16
   - $Else\ if\ |mv\_x| + |mv\_y| = 2$: select 8×8
   - $Else$: select 4×4
9. Based on the selected block size, find optimal motion vectors.
   - If 32×32: use one MV for the entire block.
   - If 16×16: find the best MV for each of the 4 sub-blocks.
   - If 8×8: find the best MV for each of the 16 sub-blocks.
   - If 4×4: find the best MV for each of the 64 sub-blocks.
10. Reconstruct the macroblock and entire frame using the selected block size and motion vectors.
11. Find the PSNR using original and reconstructed frames.
12. Find the total number of computations at all levels of the algorithm.

The proposed algorithm achieves an average PSNR of 35.52 dB, which is comparable to the 35.25 dB obtained with FSBME, while providing average computational savings of 91.06% compared to FSBME across 10 CIF video sequences.

B. Integrated Bit-Planar Motion-Only Block Matching (IBPMO-BM)

IBPMO-BM is a motion-sensitive adaptive search algorithm that introduces early termination at the bit-plane level to reduce computational complexity, while operating at a single spatial resolution. The algorithm refines the search based on motion activity and proceeds as described below.

1. Extract the frames from the video set.
2. Consider two consecutive frames, the reference frame and the current frame.
3. Divide the current frame into non-overlapping macroblocks of size 32×32. Extract the 32×32 macroblock for which the motion vector must be found.
4. Generate an adaptive search pattern for a search size of 4, based on the previous MV.
5. For each search position:
   a. Initialize accumulated SAD = 0.
   b. For each 4×4 sub-block (64 total):
      i. Compute Bit_Planar_SAD as shown in Figure 1.
      ii. Add bit-level SAD to the accumulated SAD.
      iii. If accumulated SAD ⩾ min_sad_threshold:
         - Skip remaining sub-blocks (early termination).
         - Aggregate 4×4 SADs to compute higher-level SADs.
6. Find the best motion vector for the 32×32 block.
7. Calculate motion-only block size using MV magnitude.
8. Execute block size decision, reconstruction, and quality analysis using PSNR.
9. Find the total number of computations at all levels of the algorithm.

The proposed algorithm achieves an average PSNR of 35.19 dB and an average computational savings of 91.39% compared to FSBME across 10 CIF video sequences.

III. MATHEMATICAL FORMULATIONS AND COMPUTATIONAL ANALYSIS

This section describes the mathematical and computational aspects of the proposed motion-only VBSME algorithms.

A. HMO-VBSME: Hierarchical SAD Computation Operations

The hierarchical SAD is computed in three progressive levels, with early termination at each stage to reduce unnecessary calculations:

1) Level 1 (1/16 Sampling)

Sample every 4th pixel in both directions; each 4×4 block has 1 sampled pixel.

$$Operations = 2 \times 1(sub + abs) + 1(mult) + 1(compare) = 4 \quad (1)$$

2) Level 2 (1/4 Sampling)

Sample every 2nd pixel; 4 sampled pixels per 4×4 block.

$$Operations = (2 \times 4) + 1 + 1 = 4 \quad (2)$$

3) Level 3 (Full SAD): Use All 16 Pixels in the Block

$$Operations = 2 \times 16 = 32 \quad (3)$$

B. IBPMO-BM: Bit-Planar SAD Computation Operations

In the Bit-Planar SAD method, each 4×4 block is processed across 8-bit planes (from MSB to LSB):

- For each bit plane:
  - 16 XOR operations (1 per pixel).
  - 15 additions to sum XOR results.
  - 1 multiplication (or bit-shift) for bit significance.
  - 1 addition to accumulate weighted SAD.
  - 1 comparison for early termination.
- Bit extraction is assumed hardware-efficient and excluded from the operation count.
- Early termination skips the remaining bit planes when partial.

SAD exceeds the best match found so far.

C. Search Pattern Operations

- If previous MV = (0, 0):
  - 2 comparisons + 1 logical AND.
  - Selects 9 search positions.
- Else:
  - Requires up to 4 comparisons to determine direction.
  - Selects 45 directional search positions.
- Total pattern selection operations per macroblock: 3 to 7.

D. Best Motion Vector Finding Operations

Finding the best MV requires comparing SAD for N search positions:

$$Comparisons = N - 1 \quad (4)$$

- Full search (81): 80 comparisons.
- Directional search (45): 44 comparisons
- Minimal pattern (9): 8 comparisons

E. Motion Vector Magnitude Calculation

The L1 norm is used:

$$mv\ magnitude = |mv\_x| + |m\_vy| \quad (5)$$

- 2 Absolute Value Operations.
- 1 Addition.

F. Motion-Only Block Size Decision Rules

Block size is decided based on the MV magnitude:

- If magnitude = 0: select 32×32 (1 comparison)
- If magnitude < 2: select 16×16 (2 comparisons)
- If magnitude = 2: select 8×8 (4 comparisons + 1 AND)
- Else: select 4×4 (default after all checks, 5 operations).

G. SAD Aggregation Operations

Hierarchical aggregation is used for computing SADs for larger blocks:

- 8×8 from 4 adjacent 4×4: 3 additions per block ×16 blocks = 48 additions.
- 16×16 from 4 adjacent 8×8: 3 additions per block ×4 blocks = 12 additions.
- 32×32 from 4 adjacent 16×16: 3 additions.

H. Early Termination Operations

Early termination is applied at both the block and search position level:

- In IBPMO-BM: Terminates bit-plane processing in a 4×4 block when partial weighted SAD exceeds best match so far.
- In HMO-VBSME: Multi-resolution termination from Level 1 to Level 3.
- Both: Terminate entire 4×4 processing at a search position once accumulated SAD exceeds minimum SAD. Dynamic SAD minimum tracking ensures an increasing likelihood of early termination.

IV. EXPERIMENTAL RESULTS

Experiments were carried out in MATLAB R2024a using standard CIF video sequences [19] with a resolution of 352×288 pixels. The sequences were chosen to cover a variety of motion characteristics: low-motion sequences such as akiyo and mother-daughter, medium-motion sequences such as foreman, news, and hall, and high-motion sequences including waterfall, tempete, highway, and container. This diverse selection ensures that the proposed motion estimation algorithms are tested under conditions ranging from nearly static scenes to highly dynamic content.

For motion estimation, a search range (S) of ±4 pixels was employed. For each block in the current frame, the algorithm examines a 9×9 window $(2S + 1, 2S + 1)$ in the reference frame centered on the same row and column as the block in the current frame to find the best match [17]. The performance of the proposed IBPMO-BM and HMO-VBSME algorithms was compared with the conventional FSBME approach in terms of PSNR, computational savings, and block-size adaptation.

A. PSNR Performance

Table I presents the PSNR comparison for 32×32 blocks across 10 standard video sequences. The proposed methods maintain PSNR performance comparable to FSBME while substantially reducing computational cost. The average PSNR values achieved are 35.19 dB for IBPMO-BM and 35.52 dB for HMO-VBSME, very close to 35.25 dB of FSBME. Minor variations (≤0.35 dB) between methods are perceptually negligible. For low-motion videos, such as akiyo and mother-daughter, both algorithms slightly outperform FSBME due to reduced motion vector noise. In high-motion sequences (highway, container), the difference remains within ±0.3 dB, confirming robustness across varying motion levels.

TABLE I. PSNR (DB) COMPARISON FOR 32×32 BLOCK SIZE ACROSS VIDEO SEQUENCES

Dataset	Algorithms		
	FSBME	**IBPMO-BM**	**HMO-VBSME**
waterfall	35.18	34.93	35.24
tempete	25.64	25.51	25.64
paris	30.19	30.49	30.59
news	37.56	37.36	37.86
mother-daughter	41.72	41.68	41.93
highway	34.23	34.08	34.53
hall	34.92	34.72	34.94
foreman	31.53	31.38	32.63
container	38.37	38.37	38.37
akiyo	43.14	43.42	43.48

B. Computational Savings

The computational savings percentage was calculated as:

$$CS(\%) = \frac{(FSBME_{ops} - Proposed_{ops})}{FSBME_{ops}} \times 100 \qquad (5)$$

Table II summarizes the reduction in operations compared to FSBME. The proposed algorithms achieve significant efficiency gains—91.39% for IBPMO-BM and 91.06% for HMO-VBSME on average. Savings exceed 95% in static sequences such as akiyo and container, while even in complex, high-motion videos (highway, foreman), the savings remain above 77%. These results confirm high-quality estimation with drastically fewer computations, making them well-suited for real-time and hardware-constrained systems.

TABLE II. PERCENTAGE SAVINGS IN OPERATIONS (32×32 BLOCK)

Dataset	Algorithms	
	IBPMO-BM	**HMO-VBSME**
waterfall	94.29	94.46
tempete	92.30	92.37
paris	94.01	93.82
news	95.75	95.20
mother-daughter	91.27	91.16
highway	79.23	77.96
hall	91.99	92.50
foreman	81.92	80.19
container	95.65	95.68
akiyo	97.53	97.27

C. Block Size Distribution

The block size distribution provides insight into the adaptive nature of the proposed algorithms. Tables III and IV show that both methods predominantly favor larger blocks (32×32) in static or low-motion regions, minimizing unnecessary subdivisions and computation. In akiyo and container, over 97% of blocks are 32×32, indicating strong temporal stability. In contrast, foreman and highway exhibit more 4×4 and 8×8 blocks (~40%), reflecting localized motion adaptation. HMO-VBSME shows a slightly stronger preference for larger blocks in smooth regions, aligning with its marginally higher computational efficiency. The results confirm that IBPMO-BM and HMO-VBSME have comparable PSNR to FSBME, with up to 97% reduction in computation. Both algorithms adaptively select block sizes based on motion activity, yielding perceptually identical reconstruction quality.

TABLE III. BLOCK SIZE DISTRIBUTION FOR IBPMO-BM (DATASETS: 32×32 BLOCKS)

Dataset	4×4	8×8	16×16	32×32
waterfall	0.0	1.3	8.5	90.2
tempete	0.6	1.3	9.6	88.5
paris	0.9	3.0	11.9	84.2
news	0.5	0.6	5.3	93.6
mother-daughter	7.5	4.1	10.0	78.4
highway	18.8	14.6	31.2	35.4
hall	2.2	1.6	8.4	87.8
foreman	23.5	17.2	30.9	28.4
container	1.1	0.9	1.9	96.1
akiyo	0.1	0.1	2.6	97.2

TABLE IV. BLOCK SIZE DISTRIBUTION FOR HMO-VBSME (DATASETS: 32×32 BLOCKS)

Dataset	4×4	8×8	16×16	32×32
waterfall	0.0	1.3	5.9	92.8
tempete	0.6	1.1	6.8	91.5
paris	0.5	2.6	12	85.0
news	0.3	0.7	6.6	92.4
mother-daughter	4.2	2.6	9.6	83.5
highway	13.0	11.4	30.2	45.4
hall	0.7	0.7	4.2	94.4
foreman	21.6	17.5	30.3	30.7
container	0.1	0.2	1.5	98.2
akiyo	0.1	0.1	2.4	97.4

V. CONCLUSION

This paper presented two motion-only variable block size motion estimation algorithms, HMO-VBSME and IBPMO-BM, which achieve computational savings of 79% to 97% compared to full search block motion estimation (FSBME) while maintaining comparable PSNR performance. Both algorithms effectively utilize 32×32 blocks, with selection rates ranging from 28.4% to 97.4%, based on motion-vector-magnitude-driven decisions, thereby reducing motion vector overhead and improving compression efficiency.

HMO-VBSME exploits a hierarchical block structure with motion-sensitive block splitting, while IBPMO-BM further reduces complexity through bit-planar evaluation, aggregation of 4×4 sub-block errors, and early termination during motion matching. By restricting the full SAD evaluation to promising motion candidates, the proposed methods significantly reduce the number of SAD computations without degrading reconstruction quality.

Due to their low computational complexity, regular processing structure, and motion-only design, the proposed algorithms are well-suited for real-time and hardware-oriented video coding applications. The proposed framework is inherently scalable by design and supports efficient processing of larger block structures, making it suitable for hardware implementation.

VI. FUTURE WORK

Future research directions include:

- Adaptive motion thresholds based on global motion characteristics.

- Integration with modern video codecs (H.266/VVC, AV1).
- Hardware-specific optimizations for parallel processing architectures.
- Extension to non-square blocks (32×16, 16×32, 64×32).
- Multi-reference frame support for enhanced compression efficiency.

DECLARATION OF COMPETING INTERESTS

The authors declare that there are no competing interests that could have influenced the work reported in this paper.

ACKNOWLEGDEMENT

Not applicable in this article

DATA AVAILABILITY

The data used in this study are publicly available online and can be accessed at [19].

REFERENCES

[1] R. Li, B. Zeng, and M. L. Liou, "A new three-step search algorithm for block motion estimation," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 4, no. 4, pp. 438–442, Dec. 1994, https://doi.org/10.1109/76.313138.

[2] S. Zhu and K. K. Ma, "A new diamond search algorithm for fast block-matching motion estimation," *IEEE Transactions on Image Processing*, vol. 9, no. 2, pp. 287–290, Oct. 2000, https://doi.org/10.1109/83.821744.

[3] M. Bierling, "Displacement Estimation By Hierarchical Blockmatching," in *Visual Communications and Image Processing '88: Third in a Series*, Oct. 1988, vol. 1001, pp. 942–953, https://doi.org/10.1117/12.969046.

[4] W. Li and E. Salari, "Successive elimination algorithm for motion estimation," *IEEE Transactions on Image Processing*, vol. 4, no. 1, pp. 105–107, Jan. 1995, https://doi.org/10.1109/83.350809.

[5] M. J. Chen, L. G. Chen, and T. D. Chiueh, "One-dimensional full search motion estimation algorithm for video coding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 4, no. 5, pp. 504–509, July 1994, https://doi.org/10.1109/76.322998.

[6] Y. W. Huang, B. Y. Hsieh, T. C. Chen, and L. G. Chen, "Analysis, fast algorithm, and VLSI architecture design for H.264/AVC intra frame coder," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 15, no. 3, pp. 378–401, Mar. 2005, https://doi.org/10.1109/TCSVT.2004.842620.

[7] Y. W. Chen, M. H. Hsiao, H. T. Chen, C. Y. Liu, and S. Y. Lee, "Content-Aware Fast Motion Estimation Algorithm," *Journal of Visual Communication and Image Representation*, vol. 19, no. 4, pp. 256–269, May 2008, https://doi.org/10.1016/j.jvcir.2008.01.002.

[8] Y. G. Lee, "Early search termination for fast motion estimation," *EURASIP Journal on Image and Video Processing*, vol. 2015, no. 1, Dec. 2015, Art. no. 29, https://doi.org/10.1186/s13640-015-0083-4.

[9] T. Wiegand, G. J. Sullivan, G. Bjontegaard, and A. Luthra, "Overview of the H.264/AVC video coding standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 7, pp. 560–576, July 2003, https://doi.org/10.1109/TCSVT.2003.815165.

[10] G. Thapa, K. Sharma, and M. K. Ghose, "Multi Resolution Motion Estimation Techniques for Video Compression: A Survey," *International Journal of Computer Engineering & Technology (IJCET)*, vol. 3, no. 2, pp. 399–406, 2012.

[11] X. Liyin, S. Xiuqin, and Z. Shun, "A review of motion estimation algorithms for video compression," in *2010 International Conference on Computer Application and System Modeling (ICCASM 2010)*, Oct. 2010, pp. 446-450, https://doi.org/10.1109/ICCASM.2010.5620542.

[12] A. Dosovitskiy *et al.*, "FlowNet: Learning Optical Flow with Convolutional Networks," in *2015 IEEE International Conference on Computer Vision (ICCV)*, Dec. 2015, pp. 2758–2766, https://doi.org/10.1109/ICCV.2015.316.

[13] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox, "FlowNet 2.0: Evolution of Optical Flow Estimation with Deep Networks," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017, pp. 1647–1655, https://doi.org/10.1109/CVPR.2017.179.

[14] Y. H. Ho, C. Y. Cho, and W. H. Peng, "Deep Reinforcement Learning for Video Prediction," in *2019 IEEE International Conference on Image Processing (ICIP)*, Sept. 2019, pp. 604–608, https://doi.org/10.1109/ICIP.2019.8803825.

[15] Z. Teed and J. Deng, "RAFT: Recurrent All-Pairs Field Transforms for Optical Flow," in *Computer Vision – ECCV 2020*, vol. 12347, A. Vedaldi, H. Bischof, T. Brox, and J. M. Frahm, Eds. Springer International Publishing, 2020, pp. 402–419.

[16] M. Atif, Z. H. Khand, S. Khan, F. Akhtar, and A. Rajput, "Storage Optimization using Adaptive Thresholding Motion Detection," *Engineering, Technology & Applied Science Research*, vol. 11, no. 2, pp. 6869–6872, Apr. 2021, https://doi.org/10.48084/etasr.3951.

[17] M. Vinutha and T. M. Manu, "Interframe Prediction Based Stationary Block Revalidation VBSME Using Full Search and Vector-Driven Techniques," *Journal of Mobile Multimedia*, vol. 21, no. 6, pp. 1195–1220, Nov. 2025, https://doi.org/10.13052/jmm1550-4646.2168.

[18] M. Vinutha and T. M. Manu, "A novel Direction-based Spiral-Skip Search Block_Matching Motion Estimation," in *2024 Global Conference on Communications and Information Technologies (GCCIT)*, Oct. 2024, pp. 1–6, https://doi.org/10.1109/GCCIT63234.2024.10862604.

[19] "Derf's Test Media Collection," *Xiph.org*. https://media.xiph.org/video/derf/.