Journal of
Sensor and Actuator Networks

MDPI

Article

Spherical Motion Vector Mapping for Enhanced Compression of 360-Degree Video in H.266/VVC

Yair Wiseman

Computer Science Department, Bar-Ilan University, Ramat-Gan 5290002, Israel; wiseman@cs.biu.ac.il

Abstract

Also known as spherical or omnidirectional video, 360-degree video has become increasingly prevalent in autonomous vehicles (AVs), virtual reality (VR), augmented reality (AR), and immersive media applications. However, its compression poses unique challenges due to the projection from a spherical surface onto a 2D plane. Common projections like Equirectangular Projection (ERP) introduce significant geometric distortions, causing straight-line motions on the sphere to appear as curved trajectories in the 2D domain. Standard motion estimation in codecs like H.266 (Versatile Video Coding, VVC) relies on translational (linear) motion vectors, leading to poor prediction accuracy, large residual errors, and inflated bitrates for 360-degree video content. This paper proposes the implementation of Spherical Motion Vector (SMV) mapping in the pre-encoder stage. By performing motion vector calculation directly on the spherical coordinate system (θ, φ) before mapping to the 2D pixel grid (x, y), SMV enables accurate tracking of object motion across projection boundaries and warped regions. This approach minimizes residual data and improves overall compression efficiency. This paper details the mathematical foundations, integration with H.266, implementation considerations, and simulated performance gains. The proposed method builds on prior work in rotational and geodesic motion models while introducing pre-encoder spherical preprocessing for broader compatibility.

Keywords: 360-degree video; spherical video coding; motion estimation; H.266/VVC; equirectangular projection; spherical motion vectors

1. Introduction

The rise of 360-degree video has transformed content consumption, enabling users to explore immersive environments in real-time [1]. Applications range from entertainment and gaming to autonomous vehicles, education, telemedicine, and industrial training. A single frame of 360-degree video at high resolution (e.g., 8K) contains vastly more data than traditional 2D video, and temporal sequences exacerbate storage and bandwidth demands [2]. Efficient compression is thus critical.

Standard video codecs, including the state-of-the-art H.266/VVC, are optimized for planar, perspective video. They employ block-based motion compensation assuming translational motion in the 2D image plane [3]. When applied to projected 360-degree videos, this assumption breaks down. In the real world (or on the unit sphere), objects often follow great-circle paths (geodesics). Upon projection to a 2D format like ERP, these become curved, and motion near poles is heavily distorted due to varying sampling density [4].

Consider a simple example: an object moving linearly across the equator on the sphere projects to relatively straight motion in ERP. However, the same motion near the poles

Academic Editors: Chengwen Luo and Chao Fan

Received: 7 July 2026
Revised: 1 September 2026
Accepted: 8 September 2026
Published: 10 September 2026

stretches dramatically, and objects crossing the 180° longitude boundary (the "seam" in ERP) create discontinuities. Standard motion estimation struggles with these, generating large prediction residuals that increase file sizes and reduce quality at given bitrates.

To successfully compress projected 360-degree video, standard video codecs, including the state-of-the-art H.266/VVC, must overcome the limitations of their architecture, which is inherently optimized for planar, perspective video and relies on block-based motion compensation that assumes simple translational movement within a flat 2D image plane.

The proposed Spherical Motion Vector (SMV) addresses this by shifting motion estimation to the spherical domain in a pre-encoder stage. Motion vectors are computed using spherical coordinates (θ for longitude, φ for latitude/polar angle) using rotational or geodesic transformations. These are then mapped to 2D equivalents for the encoder core. This preserves object shape and motion fidelity on the sphere, reducing residuals significantly.

This paper is structured as follows: Section 2 reviews related work on 360-degree video projections and motion modeling. Section 3 details the challenges of standard motion estimation. Section 4 presents the SMV methodology. Section 5 discusses integration with H.266. Sections 6–8 cover implementation, evaluation, and results. Section 9 concludes with future directions.

2. Related Work

2.1. Projection Formats for 360-Degree Video

In the realm of 360-degree video and virtual reality content, projection formats serve as the critical bridge between the spherical geometry of immersive environments and the flat, rectangular constraints of traditional 2D video encoding, storage, and display systems. These formats transform the continuous surface of a sphere into a two-dimensional image that can be efficiently processed by standard video codecs like H.264, H.265, or AV1. Without effective projections, capturing, transmitting, and rendering omnidirectional video would be impractical due to the inherent mismatch between spherical data and planar media pipelines. Several projection methods have been developed, each balancing trade-offs in visual fidelity, compression efficiency, and computational complexity. The choice of projection significantly impacts the final viewer experience, influencing everything from distortion levels to seam artifacts during playback on head-mounted displays or panoramic screens. As 360-degree video continues to evolve in applications ranging from entertainment and gaming to education and remote surveillance, understanding these formats is essential for content creators and engineers alike.

One of the most prevalent projection formats is the Equirectangular Projection (ERP) [5], which maps the sphere's longitude (θ) directly to the horizontal x-axis and latitude (φ) to the vertical y-axis of a rectangular image. This straightforward cylindrical unwrapping creates a familiar grid-like representation where the entire 360-degree horizontal panorama spans the width of the image, and the vertical axis covers from the north pole to the south pole. Its simplicity makes ERP highly compatible with existing video workflows, as it requires minimal preprocessing and integrates seamlessly with conventional 2D video players and encoders [6]. Major platforms and standards, including YouTube and many VR headsets, default to ERP for its ease of implementation. However, this convenience comes at a significant cost: the projection introduces severe stretching and distortion near the poles, where the same amount of spherical surface area is spread across disproportionately wider pixel regions compared to the equator. This non-uniform sampling means that polar regions receive excessive detail in the 2D image while equatorial areas are relatively compressed, leading to inefficiencies in bitrate allocation during compression [7].

The drawbacks of Equirectangular Projection become particularly evident during encoding and rendering. Because poles are oversampled, video codecs waste bits on redundant or stretched information that contributes little to perceptual quality in most viewing scenarios [8]. When users look straight ahead in a VR environment, the equatorial band, which is typically the area of primary interest, may suffer from lower effective resolution due to the overall image budget being skewed toward the poles. Despite these issues, ERP remains dominant due to its universal support and straightforward mapping, which simplifies tasks like stitching multiple camera feeds into a cohesive sphere. Content creators often apply pre-filtering or adaptive resolution techniques to mitigate polar artifacts, but the format's inherent geometric warping continues to challenge high-fidelity immersive experiences. As a result, while ERP excels in broad accessibility and quick deployment, it is frequently criticized in professional pipelines where visual quality and bandwidth efficiency are paramount.

In contrast, the Cubemap Projection (CMP) [9] offers a different approach by projecting the spherical environment onto the six faces of a cube surrounding the camera origin. Each face captures a 90-degree field of view, resulting in a more uniform distribution of pixels across the spherical surface compared to ERP. This leads to reduced overall distortion and better preservation of details in all directions, making CMP particularly advantageous for applications requiring consistent quality, such as high-end VR gaming or architectural visualizations [10]. The cube faces are typically arranged in a net layout or packed into a single rectangular video frame using various packing schemes. CMP's uniformity improves coding efficiency because pixels better represent equal solid angles on the sphere, allowing encoders to allocate bits more evenly without the extreme polar stretching seen in ERP. Nevertheless, this format is not without challenges, as the sharp boundaries between cube faces introduce discontinuities that can manifest as visible seams or edge artifacts if not handled carefully during encoding or rendering [11].

The discontinuities at face boundaries in Cubemap Projection require specialized handling techniques, such as padding, overlapping regions, or advanced seam-aware compression algorithms, to maintain smooth transitions when the viewer rotates their viewpoint across edges. While CMP provides superior geometric fidelity, these seams can complicate real-time rendering pipelines and increase the complexity of video stitching and post-production. Developers often rotate or reorient the cube to minimize seams in high-interest areas, and modern standards like those from MPEG have explored optimized CMP variants to enhance compatibility [12]. Compared to ERP, CMP demands more sophisticated decoding logic but rewards users with crisper, more immersive results, especially on devices capable of native cube mapping hardware acceleration. Its adoption has grown in professional content creation tools and game engines, where the benefits of reduced distortion outweigh the added implementation overhead [13].

Beyond ERP and CMP, a variety of alternative projection formats have emerged to address specific limitations, including octahedron projections, icosahedron-based mappings, and various rotated sphere or pyramid variants. Octahedron projections, for instance, unfold the sphere onto eight triangular faces, offering a compact representation with moderate distortion levels and fewer seams than cube maps [14]. Icosahedron approaches further subdivide the sphere into twenty triangular facets, achieving even greater uniformity at the expense of increased computational complexity in packing and rendering [15]. Rotated sphere variants adjust the orientation of standard projections to better align important content areas away from high-distortion zones [16]. Each of these formats involves careful trade-offs: some prioritize minimal distortion for perceptual quality, others optimize for seam handling to avoid visible artifacts, and many focus on coding efficiency to reduce bandwidth requirements for streaming. The selection often depends on the target platform,

content type, and delivery constraints, with hybrid approaches sometimes combining elements from multiple methods for optimal results [17].

Ultimately, all projection formats inherently warp the underlying spherical geometry, converting uniform sampling on the sphere into non-uniform sampling on a 2D plane. This fundamental transformation introduces challenges in maintaining isotropic resolution and preserving spatial relationships across the entire 360-degree field. Engineers and researchers continue to refine these mappings through adaptive projections, machine learning-based optimizations, and standardized extensions within video codecs [18]. As immersive media matures, the evolution of projection techniques will play a pivotal role in delivering seamless, high-quality experiences that feel truly natural to viewers. Understanding these formats empowers content creators to make informed decisions that balance technical feasibility with visual excellence, pushing the boundaries of what is possible in virtual and augmented reality applications.

2.2. Motion Compensation in Video Coding

Motion compensation stands as a cornerstone of modern hybrid video codecs, including H.264/AVC, HEVC, and the latest Versatile Video Coding (VVC) standard. It exploits temporal redundancy between frames by predicting the current frame's content based on motion vectors that describe how blocks or regions have moved from one or more reference frames [19]. This predictive approach dramatically reduces the residual data that needs to be encoded, enabling high compression efficiency while maintaining visual quality. In VVC, motion compensation has been significantly enhanced through more flexible coding tree unit (CTU) partitioning, which allows for highly adaptive block shapes and sizes that better match complex motion patterns [20]. Additionally, VVC introduces advanced tools such as affine motion models capable of handling not just translation but also rotation, scaling, and shear transformations. Other improvements include enhanced temporal motion vector prediction, multi-reference frame management, and sophisticated prediction refinement techniques [21]. These advancements make VVC particularly effective for high-resolution conventional video, but they remain fundamentally rooted in 2D planar image representations, assuming flat rectangular frames and linear motion within the pixel grid [22].

The planar nature of traditional motion compensation creates substantial challenges when applied directly to 360-degree video content. Because 360-degree video is typically stored using spherical-to-2D projections like Equirectangular Projection (ERP) or Cubemap Projection (CMP), the underlying spherical geometry and non-linear distortions introduced by these mappings are not naturally accounted for in standard 2D motion estimation algorithms. Motion vectors that appear linear and consistent in the projected 2D domain may correspond to highly curved trajectories on the actual sphere, especially near the poles in ERP or across face boundaries in CMP. This mismatch leads to inaccurate motion prediction, increased residual energy, and reduced compression efficiency. While VVC's affine models offer greater flexibility than the simple translational models of earlier codecs, they still operate within the constraints of the 2D projected plane rather than the native spherical surface. As a result, artifacts such as motion discontinuities, inefficient bitrate allocation, and degraded prediction quality frequently appear in immersive content, particularly during rapid camera rotations or complex object movements across the 360-degree field of view [23].

To address these limitations, the Joint Video Experts Team (JVET) has actively explored 360-degree video-specific extensions within the development of VVC and related standards. These include adjusted padding methods around projection seams, face-based independent coding for cubemap projections, and various geometry-aware prediction tools designed

to better respect the spherical nature of the content. Techniques such as spherical motion vectors, projection-format-aware interpolation, and adaptive reference picture handling have been investigated to mitigate the impact of projection-induced non-linearities. Despite these efforts, core motion estimation and compensation mechanisms continue to face inherent difficulties because they must operate on the distorted 2D representation rather than directly on the sphere. Ongoing research focuses on deeper integration of geometric metadata and more advanced neural network-assisted prediction to bridge this gap. As immersive media applications expand, overcoming these challenges will be crucial for achieving the same level of coding efficiency in 360-degree video that consumers now expect from traditional 2D content.

Within the VVC standardisation process, the Joint Video Experts Team (JVET) has already adopted several 360-degree video-specific tools, including geometry padding at projection seams, face-based independent coding for cubemap formats, and adjusted interpolation filters. These tools mitigate boundary artefacts but still rely on conventional translational or affine motion models operating in the projected 2-D domain. The SMV approach is complementary: it supplies geometry-aware motion candidates that can be used together with the existing VVC 360 tools without any conflict.

2.3. Spherical and Rotational Motion Models

The limitations of traditional 2D planar motion compensation in 360-degree video have driven significant research into spherical and rotational motion models that operate directly on the native geometry of the sphere. Rather than relying on distorted projections, these approaches model camera and object motion in three-dimensional spherical coordinates, aiming to preserve geometric accuracy and improve prediction efficiency. By treating motion as rotations and translations on the sphere's surface, researchers seek to eliminate the non-linear artifacts introduced by projections such as Equirectangular or Cubemap formats. This paradigm shift enables more accurate motion vector estimation and compensation, particularly for content involving large camera rotations or omnidirectional movement. Such models represent a critical evolution in immersive video coding, addressing the fundamental mismatch between spherical reality and planar processing pipelines used in standards like HEVC and VVC [24].

One influential contribution comes from Vishwanath et al. [25], who introduced a sophisticated rotational motion model based on Rodrigues' rotation formula. This technique describes motion along geodesic paths on the sphere, ensuring that shapes and object sizes are preserved more faithfully than in conventional 2D translational models. By parameterizing rotations in 3D space, their method accurately captures the natural curvature of motion trajectories in 360-degree environments. Motion search is performed using a radial pattern centered on the sphere, allowing efficient exploration of possible rotations without being constrained by projection distortions. Their experiments demonstrated substantial coding gains compared to standard HEVC, highlighting the potential of spherical-domain processing to reduce residual energy and improve overall compression performance in omnidirectional video.

Complementary to pure rotational models, researchers have developed geodesic translation models alongside motion vector modulation techniques specifically tailored for translational camera motion in spherical coordinates [26,27]. These approaches adjust traditional motion vectors to account for the curvature of the sphere, converting apparent linear shifts in the 2D projection into proper geodesic displacements. This modulation helps maintain consistency across different regions of the 360-degree video, reducing prediction errors that commonly arise near poles or projection seams. Such models prove particularly effective for scenarios involving steady forward camera movement or lateral shifts, where

standard planar compensation tends to break down due to varying distortion levels. By integrating these geodesic principles, encoders can achieve smoother temporal prediction and higher fidelity in reconstructed immersive content.

Another notable advancement is the Motion Plane Adaptive (MPA) modeling framework integrated into VVC-based coding schemes [28]. MPA dynamically performs motion compensation on adaptive 3D planes that best approximate local surface regions of the sphere. This hybrid strategy combines the benefits of planar processing with spherical awareness, allowing the codec to select optimal compensation planes for different parts of the frame. The adaptability of MPA makes it a practical enhancement within existing video coding standards, offering a balance between computational complexity and improved prediction accuracy for 360-degree video applications.

Further progress has been made through multi-model motion prediction techniques and advanced spherical motion models that incorporate local padding strategies [29,30]. These methods maintain multiple candidate motion models per coding block, selecting or blending the most suitable one based on local spherical geometry. Local padding around projection boundaries or high-distortion areas helps mitigate seam artifacts and provides better reference data for motion estimation. Such innovations enhance the robustness of prediction, especially in complex scenes with both rotational and translational elements. By operating closer to the true spherical manifold, these approaches reduce the coding overhead associated with projection-induced irregularities and pave the way for more efficient delivery of high-quality 360-degree video.

Neural network-based approaches and end-to-end learning frameworks have also emerged as promising directions for spherical motion modeling [31]. Deep learning models can be trained to predict motion directly in spherical domains or to optimize projection-aware compensation, often outperforming traditional hand-crafted tools in specific scenarios. However, these methods frequently face challenges regarding standardization and interoperability. Many neural solutions lack seamless compatibility with established video coding standards such as H.266/VVC, limiting their deployment in practical broadcasting and streaming pipelines. While they demonstrate strong potential for future codecs, their integration into real-world systems remains an ongoing area of research and development.

Recent learning-based approaches have also targeted the high computational complexity of VVC for 360-degree video. For example, [32] proposed a fast CU partition algorithm that employs a stretching-based multi-scale convolutional neural network to predict split modes while accounting for the non-uniform distortion characteristics of ERP content. Their method achieves substantial encoding-time reductions (approximately 70%) with only modest BD-rate overhead, illustrating the potential of deep-learning tools to accelerate 360-degree coding within the VVC framework. Such techniques are complementary to the proposed SMV approach: while SMV focuses on geometry-aware motion estimation in a pre-encoder stage, learning-based CU-partition predictors address the subsequent rate-distortion search complexity. Hybrid pipelines that combine spherical motion modelling with neural partition acceleration therefore represent a promising direction for future research.

Recent advances in temporal modeling for video sequences, although developed primarily for super-resolution rather than compression, offer complementary insights. Xiao et al. [33] introduced a local–global temporal-difference learning framework that separately exploits short-term residual maps between adjacent frames and long-term differences across forward/backward segments. Similarly, Isobe et al. [34] demonstrated the benefit of explicit temporal-difference modeling in both low- and high-resolution domains. While these methods operate in the planar domain, the underlying idea of distinguishing local and global temporal discrepancies is complementary to the spherical rotational model of

SMV. Future hybrid architectures could therefore combine SMV's geometry-aware motion candidates with local–global temporal-difference features to further improve prediction accuracy in complex 360-degree scenes.

A systematic comparison of the most relevant prior spherical and rotational motion models is summarized below. Vishwanath et al.'s rotational model based on Rodrigues' formula and the subsequent geodesic-translation variants [25] typically report BD-rate savings in the range of 1–5% over HEVC/VVC anchors. The Motion-Plane Adaptive (MPA) approach [8] and multi-model motion prediction schemes [3] achieve similar gains (approximately 1.5–3.2%). Most of these methods require modifications inside the core encoder (new motion models, altered search patterns, or additional syntax elements) and therefore cannot be used as a pure pre-encoder. Their computational overhead is generally moderate when the spherical search replaces the conventional one, but support for advanced VVC tools such as bi-prediction, affine merge, DMVR and BIO is often limited or requires further adaptation.

In contrast, the proposed SMV framework operates entirely as a pre-encoder stage. It produces refined motion-vector candidates or spherically compensated references that are fed into an unmodified VVC encoder. Consequently, SMV remains fully compatible with bi-prediction, affine merge, DMVR and BIO, requires no changes to the bitstream syntax, and can be applied to both legacy and current codecs. The average BD-rate saving of -18.7% (WS-PSNR) exceeds the 1–5% range of earlier spherical models. The added complexity is higher than that of a pure 2-D affine search but lower than that of full end-to-end spherical codecs, because the VVC core itself remains untouched and the heavy spherical operations can be accelerated on GPU or executed offline.

Building upon this rich foundation of prior work, the Spherical Motion Vectors (SMVs) approach emphasizes a flexible pre-encoder stage that enhances broad applicability across different codecs and workflows. By performing spherical motion analysis and vector conversion before the main encoding process, SMVs can function as a preprocessing filter or as an extension hook within existing encoders. This design philosophy allows it to complement rather than replace standard tools, making it suitable for both legacy systems and cutting-edge implementations. The pre-encoder focus facilitates easier adoption in production environments, where compatibility and minimal disruption to established pipelines are essential. As immersive media continues to mature, such modular strategies will play a vital role in bridging the gap between theoretical spherical modeling advances and practical, deployable 360-degree video coding solutions.

Building upon this foundation, recent advancements in machine-centric video compression for autonomous vehicles further complement the SMV approach. In particular, Video Compression Prototype for Autonomous Vehicles [35] demonstrates the advantages of tailoring compression strategies to the requirements of computer vision algorithms rather than human visual perception. By prioritizing the preservation of structural and geometric features critical for object detection, tracking, and navigation, such techniques achieve efficient bitrate reduction while maintaining high utility for downstream AI systems. Integrating these insights with Spherical Motion Vector mapping offers a promising direction: SMV can serve as a geometry-aware preprocessing stage that supplies accurate spherical motion information, which is then refined by machine-centric quantization and prediction tools. This synergy is especially valuable in autonomous driving applications, where 360-degree cameras must deliver reliable, low-latency video streams under strict bandwidth constraints, thereby enhancing both compression efficiency and perception accuracy in real-world immersive and vehicular environments.

The earlier methods cited in [25,26,29,30] typically replace or augment the core motion model inside the encoder, use a single rotation or geodesic-translation parameterization

and report BD-rate savings in the 1–5% range. In contrast, SMV operates exclusively as a pre-encoder stage: it estimates a full 3-D rotation on the continuous sphere with a multi-resolution radial search, applies sphere-aware interpolation, and outputs either refined 2-D motion-vector candidates or a complete spherically compensated reference that is consumed by an unmodified VVC encoder. Consequently, SMV preserves full compatibility with bi-prediction, affine merge, DMVR and BIO, needs no bitstream-syntax changes, and can be executed offline or on GPU. The resulting average BD-rate saving of −18.7% (WS-PSNR) exceeds the range reported by the core-integrated predecessors, while the added complexity remains modest (4.7% with CUDA acceleration).

3. Challenges with Standard H.266 Motion Estimation for 360-Degree Video

3.1. Projection Distortions and Motion Warping

The Equirectangular Projection (ERP) serves as the foundational mapping for representing spherical 360-degree video on a 2D rectangular frame, defined mathematically by the transformations

$$x = (\theta + \pi) \times (W/2\pi) \text{ and } y = (\varphi + \pi/2) \times (H/\pi) \quad (1)$$

where $\theta \in [-\pi, \pi]$ denotes longitude, $\varphi \in [-\pi/2, \pi/2]$ represents latitude, W is the image width in pixels, and H is the image height. These equations linearly scale the spherical coordinates to the pixel grid.

While this formulation provides a simple and computationally efficient unwrapping, it inherently introduces severe geometric distortions because the mapping does not preserve areas or angles uniformly across the sphere. The inverse mapping from 2D pixels back to spherical coordinates is given by

$$\theta = (2\pi\, x/W) - \pi \text{ and } \varphi = \pi y/H - \pi/2 \quad (2)$$

highlighting the direct proportionality that leads to non-uniform sampling densities when considering the spherical surface element

$$dA = \cos(\varphi)\, d\theta\, d\varphi. \quad (3)$$

The varying Jacobian of the ERP transform is central to understanding these projection distortions. The Jacobian matrix J of the mapping from (θ, φ) to (x, y) is diagonal with entries $J11 = W/2\pi$ and $J22 = H/\pi$, but when accounting for the spherical metric, the local area scaling factor becomes proportional to $1/\cos(\varphi)$, resulting in significantly higher pixel density near the poles ($\varphi \approx \pm\pi/2$) in terms of spherical area per pixel, yet producing a visually stretched appearance in the 2D image. This can be expressed through the distortion metric

$$D(\varphi) = \sec(\varphi) \quad (4)$$

where the infinitesimal spherical displacement

$$ds^2 = d\theta^2 \cos^2(\varphi) + d\varphi^2 \quad (5)$$

maps to distorted pixel displacements $dx^2 + dy^2$ that grow dramatically as $|\varphi|$ approaches $\pi/2$. Consequently, uniform motion on the sphere undergoes position-dependent warping in the projected domain, such that a constant angular velocity vector ω on the sphere translates to a 2D velocity field v(x,y) whose magnitude and direction vary nonlinearly with latitude according to the derivatives of the projection functions.

In the context of motion compensation, these distortions cause a constant velocity geodesic trajectory on the sphere to appear as a curved path in the 2D ERP image. A point moving with constant spherical velocity satisfies the differential equation

$$d\theta/dt = \omega_{\theta(\varphi)} \text{ and } d\varphi/dt = \omega_{\varphi}, \tag{6}$$

but after projection, the 2D trajectory follows

$$dx/dt = (W/2\pi) \times (d\theta/dt) \text{ and } dy/dt = (H/\pi) \times (d\varphi/dt) \tag{7}$$

which introduces apparent acceleration and curvature, especially evident in the equations for great-circle paths parameterized by rotation matrices. For instance, the geodesic (shortest path) between two points on the sphere, derived via the spherical law of cosines or Rodrigues' rotation formula $p' = R\,p$, projects to parabolic-like curves in ERP coordinates rather than straight lines, complicating standard block-matching algorithms that assume translational motion in the 2D plane. This warping effect is further quantified by the motion vector deviation

$$\Delta mv = mv_{2D} - \Pi(mv_{sphere}) \tag{8}$$

where Π denotes the projection operator, leading to increased residual energy and reduced coding efficiency.

Additional complications arise when motion trajectories cross the antimeridian ($\theta = \pm\pi$), triggering wrap-around discontinuities inherent to the periodic nature of longitude in the projection. Mathematically, this manifests as a jump discontinuity in the x-coordinate:

$$x(\theta = \pi) \to 0 \text{ and } x(\theta = -\pi) \to W \tag{9}$$

requiring special handling such as modular arithmetic $x' = (x \bmod W)$ or explicit seam padding in motion estimation. The resulting motion vectors must incorporate circular boundary conditions, modeled as

$$mv(x + W, y) = mv(x, y) + \Delta mv_{wrap} \tag{10}$$

where Δmv_{wrap} accounts for the 2π longitude periodicity. Combined with the latitude-dependent scaling from the Jacobian and the curvature of projected geodesics, these issues demand advanced spherical motion models that operate in 3D rotation space rather than 2D translations, ensuring that prediction tools can accurately compensate for the complex, non-linear motion warping induced by ERP and similar projections in 360-degree video coding pipelines.

3.2. Limitations of Translational and Affine Models

Traditional translational motion vectors in video coding assume a constant displacement Δx and Δy for an entire coding block, expressed mathematically as

$$x' = x + \Delta x \text{ and } y' = y + \Delta y \tag{11}$$

where (x, y) denotes the current block position and (x', y') the reference position in the 2D projected frame. This simple model corresponds to a zero-order approximation of motion with the motion vector $mv = (\Delta x, \Delta y)$ treated as uniform across the block. Affine motion models extend this capability by incorporating linear transformations, allowing for scaling, rotation, and shear through a 6-parameter affine model defined as

$$x' = a_0 + a_1 \times x + a_2 \times y \text{ and } y' = b_0 + b_1 \times x + b_2 \times y \tag{12}$$

or equivalently in matrix form as

$$[x' \; y']^T = A \times [x \; y]^T + t \tag{13}$$

where A is the 2×2 affine transformation matrix and t is the translation vector. In the context of 360-degree video using Equirectangular Projection (ERP), these models operate entirely within the distorted 2D domain without incorporating the underlying spherical geometry. The projection equations (Equation (1)) introduce non-uniform scaling that these 2D models cannot capture, leading to systematic prediction mismatches when applied to spherical motion.

The core limitation arises because translational and affine models fail to account for the curvature and position-dependent warping induced by spherical geometry. A constant velocity geodesic on the sphere, satisfying Equation (6) with angular velocity components ω_θ and ω_φ, projects to highly nonlinear trajectories in the 2D plane following Equation (8). Consequently, residual energy spikes significantly in regions of high motion or near projection boundaries, especially where the distortion metric $D(\varphi) = \sec(\varphi)$ grows large as $|\varphi|$ approaches $\pi/2$. The increased residual $||\text{residual}||^2$ forces encoders to apply higher quantization parameters QP or resort to more intra coding blocks, directly elevating the overall bitrate. The Jacobian matrix J with components $J11 = W/2\pi$ and $J22 = H/\pi$ further exacerbates these errors by creating latitude-dependent velocity fields v(x, y) that standard 2D affine approximations cannot model accurately.

Standard H.266/VVC advanced motion tools, such as Decoder-Side Motion Vector Refinement (DMVR) [36] and bi-directional optical flow (BIO) [37], are highly effective for fine-grained, sub-pixel motion adjustments. However, their underlying formulations strictly assume a flat, translational 2D motion model. In 360-degree video projections like ERP, spherical trajectories appear highly non-linear and stretched, particularly near the poles. Consequently, when large or non-linear spherical motion occurs, the planar translational assumptions of DMVR and BIO are severely violated. This causes these tools to struggle or deactivate in highly distorted regions, leaving significant residual errors that inflate the bitrate.

3.3. Impact on Bitrate and Quality

Empirical studies consistently demonstrate that 360-degree video demands significantly higher bitrates compared to traditional 2D video to achieve comparable visual quality. Specifically, omnidirectional content often requires several dozen percent more bitrate than equivalent 2D sequences when evaluated using viewport PSNR or the more appropriate Weighted Spherical PSNR (WS-PSNR) [38]. The WS-PSNR metric weights errors according to the spherical surface element (Equation (4)), providing a more perceptually relevant measure than standard PSNR by accounting for the non-uniform projection distortions in formats such as Equirectangular Projection (ERP). This bitrate overhead arises primarily from the inherent geometric warping, expressed through the distortion metric $D(\varphi) = \sec(\varphi)$, which leads to inefficient compression as encoders struggle with the varying pixel density and latitude-dependent scaling. In terms of rate-distortion performance, the Bjontegaard Delta rate (BD-rate) penalty for 360-degree video frequently amounts to several dozen percent relative to 2D counterparts, depending on content complexity, motion characteristics, and projection method [39]. These inefficiencies manifest because standard 2D tools cannot adequately handle the underlying spherical geometry, resulting in elevated residual energy $||\text{residual}||^2$ after motion compensation.

Boundary artifacts and poor temporal prediction further exacerbate the bitrate and quality challenges in 360-degree video coding. Motion trajectories that appear curved in the projected domain due to geodesic paths on the sphere cause translational and

affine models to produce inaccurate predictions, particularly near the poles and across the antimeridian wrap-around. This leads to increased usage of intra-coded blocks and higher quantization parameters (QPs), directly inflating the overall bitrate while degrading reconstructed quality in critical viewport regions [40]. Even advanced VVC tools like DMVR and BIO offer limited relief, as they operate under planar assumptions and fail to compensate for global spherical inconsistencies or the velocity warping induced by the spherical geometry. Consequently, achieving target quality levels in immersive video applications currently requires substantially higher bitrates. This holds true whether quality is measured by viewport PSNR in the user's field of view or by full-sphere WS-PSNR. Such a significant bitrate inflation highlights an urgent need for innovative spherical motion models. Implementing these specialized models will effectively reduce current performance penalties. Ultimately, this approach will drastically improve compression efficiency across all immersive media platforms.

4. Proposed Spherical Motion Vector (SMV) Mapping

4.1. Core Concept

The Spherical Motion Vectors (SMVs) framework introduces a powerful pre-encoder stage that computes motion directly in the spherical domain before feeding data into conventional 2D video encoders. Unlike standard approaches that rely solely on distorted 2D projections, SMV first transforms the problem back to the native spherical geometry of 360-degree video. For each coding block or region in the Equirectangular Projection (ERP) frame, the process begins by applying the inverse projection mapping defined as in Equation (3). This converts current block pixels at (x, y) and corresponding reference block pixels at (x + Δx, y + Δy) into spherical coordinates (θ, φ) and (θ_ref, φ_ref). By operating in the spherical domain, SMV avoids the non-uniform sampling issues of the forward projection in Equation (1) along with the associated Jacobian scaling factors J11 = W/2π and J22 = H/π that distort conventional motion estimation.

Let (x_0, y_0) denote the ERP coordinates of a coding unit's top-left pixel. The mapping to spherical coordinates is governed by the inverse projection in Equation (2).

A candidate rotation is parameterized by an axis $\mathbf{k}$ and an angle α. Every point $\mathbf{p}_i$ of the current patch is transformed by Rodrigues' formula:

$$\mathbf{p'}_i = \mathbf{p}_i \cos\alpha + (\mathbf{k} \times \mathbf{p}_i) \sin\alpha + \mathbf{k}\,(\mathbf{k} \cdot \mathbf{p}_i)\,(1 - \cos\alpha) \tag{14}$$

The rotated points are re-projected to ERP coordinates and the sum of squared differences with the reference patch is evaluated. The pair (α*, $\mathbf{k}$*) that minimizes this cost is retained. The resulting 2-D displacement field is either written as refined motion-vector candidates or used to generate a spherically compensated reference frame that is passed to the VVC encoder.

The fundamental novelty of SMV relative to prior spherical/rotational models lies in its strictly modular pre-encoder formulation. Rather than modifying the motion-model syntax or the search loop inside the codec, SMV performs the entire spherical estimation and compensation as a preprocessing stage and supplies the encoder with either (i) geometry-corrected 2-D motion-vector candidates or (ii) a ready-to-use spherically compensated reference frame. This design decision enables seamless integration with every existing VVC tool and guarantees the decoder compatibility without any standardization change.

In the second step, motion estimation is performed using true spherical geometry, modeling displacements as rotations along geodesics rather than simple 2D translations. A point p on the unit sphere moves to p′ according to the rotation matrix R via Rodrigues' rotation formula p′ = R p, where R captures the 3D rotational motion with angular velocity

components ω. Geodesic paths satisfy the spherical distance minimization with differential elements (Equation (6)), allowing optimization of rotation parameters that best align current and reference spherical patches. This course of action yields refined spherical motion parameters, which account for curvature and eliminate artifacts from the distortion metric $D(\varphi) = \sec(\varphi)$ that plagues planar models. The approach supports both global camera rotations and local object motions, producing more accurate correspondences than standard translational mv = (Δx, Δy) or 6-parameter affine models $x' = a_0 + a_1 \times x + a_2 \times y$ and $y' = b_0 + b_1 \times x + b_2 \times y$.

Finally, SMV derives equivalent 2D motion vectors or generates warped reference predictions by projecting the spherical motion results back onto the ERP domain. This produces motion vectors suitable for the standard encoder or directly supplies a compensated prediction block that respects spherical consistency. Because the entire process occurs in the pre-encoder stage, it integrates seamlessly with existing codecs such as VVC without modifying the core bitstream syntax. The refined 2D motion vectors or side information (such as spherical rotation parameters) enable the encoder to achieve superior temporal prediction, reducing residual energy $||\text{residual}||^2$ and improving overall rate-distortion performance while maintaining full compatibility with tools like affine merge, DMVR, and BIO. This modular pre-processing design makes SMV broadly applicable across different projection formats and encoding pipelines.

4.2. Mathematical Formulation

The Spherical Motion Vectors (SMVs) framework begins with the representation of points on the unit sphere using either Cartesian coordinates (X, Y, Z) or spherical coordinates (θ, φ), where $\theta \in [-\pi, \pi]$ denotes longitude and $\varphi \in [-\pi/2, \pi/2]$ denotes latitude. A point p on the sphere is expressed as $p = (\cos\varphi \cos\theta, \cos\varphi \sin\theta, \sin\varphi$. The inverse projection from 2D Equirectangular Projection (ERP) coordinates (x, y) to spherical angles is given by Equation (3), while the forward projection follows Equation (1). This bidirectional mapping enables SMV to transform coding blocks between the distorted 2D domain and the true spherical geometry, avoiding limitations of the Jacobian matrix J with elements $J11 = W/2\pi$ and $J22 = H/\pi$ that cause non-uniform scaling and distortion metric $D(\varphi) = \sec(\varphi)$.

Motion in the spherical domain is modeled primarily through rotational transformations, which are dominant for camera movements and object trajectories in 360-degree video. Rodrigues' rotation formula [41] provides an efficient way to rotate a point p to p' by an angle α around a unit axis k = (kx, ky, kz):

$$p' = p\cos\alpha + (k \times p)\sin\alpha + k\,(k \cdot p)\,(1 - \cos\alpha) \tag{15}$$

In component form, this expands to the full vector equation incorporating the cross-product $k \times p$ and dot-product $k \cdot p$. For general motion, SMV parameterizes displacements as small incremental rotations or optimizes for the best-fit geodesic displacement that minimizes the prediction error on the sphere, using the spherical surface element (Equation (6)). This approach directly addresses the curvature that causes straight geodesics to appear as nonlinear paths (Equation (8)) in the projected ERP domain.

To determine the optimal Spherical Motion Vector (θ_m, φ_m) or full rotation parameters (α, k), SMV performs a search over a discrete spherical grid centered around the current point. Radial sampling patterns are employed for efficiency, placing denser samples near the center and sparser ones outward, thereby reducing computational complexity while covering possible geodesic paths. The search minimizes a distortion measure such as the sum of squared differences between the current spherical patch and the rotated reference patch. Once the best spherical motion parameters are identified, the entire block of points

on the sphere is transformed using the Rodrigues' formula $p'_i = R\ p_i$ for each point p_i in the block, where R is the composite rotation matrix derived from (α, k).

All experimental parameters, input values, and operational metrics utilized throughout this study are comprehensively documented in Table 1. This table outlines the specific baseline conditions, variable constraints, and numerical datasets applied across each test run to ensure strict reproducibility.

Table 1. All values used in the experiments.

Parameter	Value	Rationale
Spherical search range R	$\pm$64(integer steps)	identical to VTM default integer search range
Coarse angular step $\Delta\alpha_0$	2π/W(longitude)/π/H(latitude)	one ERP pixel at the equator
Number of refinement levels L	4 (integer $\rightarrow$ half-pixel $\rightarrow$ quarter-pixel $\rightarrow$ 1/8-pixel)	matches VTM hierarchical refinement
Angular step at level ℓ	$\Delta\alpha_\ell = \Delta\alpha_0/2^\ell$	
Block/CU sizes	all VTM QTBT sizes from 4×4 to 128×128	full compatibility
Distortion function	sum of squared differences (SSD) computed on the sphere after bilinear spherical interpolation	
Interpolation kernel	spherical bilinear (great-circle distance weighted) for sub-pixel samples; VTM 8-tap/4-tap filters after re-projection	
Boundary sampling	continuous sphere (no artificial seam); after re-projection the 2π-periodicity of θ is respected automatically	
Rate term	none inside the SMV search (the refined 2-D candidates are later evaluated by the full VTM RDO)	
Stopping criteria	exhaustive evaluation of the multi-resolution radial grid; early termination when the current SSD exceeds the best SSD found so far by more than 10%	

For a CU whose centre maps to the spherical point **c**, the search grid is generated by treating **c** as a temporary pole. Integer indices (m, n) with $-R \leq m, n \leq R$ define candidate poles

$$\phi' = m \cdot \Delta\alpha_\ell, \theta' = n \cdot \Delta\alpha_\ell. \quad (16)$$

The corresponding rotation axis and angle are obtained exactly as above. The grid is denser near the origin (small m, n) and becomes sparser at larger radii, matching the natural density of a spherical metric.

As a representative numerical example, a 64×64 CU located near the ERP seam is evaluated. Consider a 64×64 coding unit whose top-left corner lies at ERP coordinates $(x_0, y_0) = (0, 900)$ in a 3840×1920 frame (a location immediately adjacent to the antimeridian seam). The geometric centre of the block is therefore $(x_c, y_c) = (32, 932)$.

The corresponding spherical coordinates obtained from the inverse ERP mapping are $\theta_c = -3.0892\,\text{rad}(-177.00^\circ), \phi_c = -0.0458\,\text{rad}(-2.63^\circ)$. In Cartesian form on the unit sphere the centre point is $p = (-, 0.9976, -0.0523 - 0.0458)$. A candidate spherical motion is represented by the unit rotation axis $k = (0.0999, 0.9488, 0.2996)$ and the angle $\alpha = 0.0600\,\text{rad}(3.44^\circ)$. Applying Rodrigues' formula $p' = p\cos\alpha + (k \times p)\sin\alpha + k(k \cdot p)(1 - \cos\alpha)$ yields the rotated point $p' = (-, 0.9975, -0.0701\ 0.0106)$. Re-projection

back to ERP coordinates produces $(x', y') = (42.9, 966.5)$, i.e., a 2-D displacement of approximately $(\Delta x, \Delta y) = (10.9, 34.5)$ pixels.

Because the original block straddles the left image border, several of its right-hand pixels map to $\theta \approx +\pi$. After the same rotation those pixels re-appear on the opposite side of the ERP frame (near $x = 3830$). The spherical formulation automatically respects the 2π-periodicity. The resulting 2-D motion-vector field is therefore continuous across the seam without any explicit modular arithmetic.

When the identical 64×64 region is predicted by a conventional translational motion vector, the residual energy (sum of squared differences) is 1.87×10^6. After the spherical rotation above, the residual energy falls to 0.41×10^6 which is a reduction of 78% for this single block. The same qualitative behaviour is observed for every coding unit that crosses the antimeridian.

In the final stage, the rotated spherical points are projected back to the 2D reference frame using the ERP equations, generating either a spherically compensated prediction signal or adjusted 2D motion vectors mv_{2D} that better align with the encoder. The resulting difference or mapping parameters serve as enhanced motion information fed into the standard VVC encoder. In the pre-encoder implementation, SMV can generate a complete "spherically compensated" reference frame or refine the initial MV candidates passed to VVC's motion estimation process. This integration improves temporal prediction accuracy, reduces residual energy $||\text{residual}||^2$, and maintains full compatibility with existing tools while operating prior to the core 2D encoding pipeline.

4.3. Handling Boundaries and Poles

One of the primary advantages of the Spherical Motion Vectors (SMVs) approach is its natural handling of wrap-around boundaries through operations performed directly in the spherical domain. In Equirectangular Projection (ERP), the longitude θ exhibits periodicity with $\theta \in [-\pi, \pi]$, leading to discontinuities at the antimeridian where $x(\theta = \pi)$ jumps to $x(\theta = -\pi) = 0$ in the 2D frame according to $x = (\theta + \pi) \times (W/2\pi)$. Traditional 2D motion estimation requires explicit modular arithmetic $x' = (x \bmod W)$ or special seam padding to manage $mv(x + W, y) = mv(x, y) + \Delta mv_{wrap}$. In contrast, SMV models motion via rotations on the unit sphere using Rodrigues' formula. This formulation inherently respects the 2π periodicity in θ without artificial boundaries, allowing seamless geodesic paths that cross the antimeridian smoothly. The inverse projection $\theta = (2\pi x/W) - \pi$ and forward projection ensure that motion parameters (α, k) or spherical motion vectors (θ_m, φ_m) produce consistent 2D motion vectors mv_{2D} without wrap-around artifacts.

One of the primary advantages of the Spherical Motion Vectors (SMVs) approach is its natural handling of wrap-around boundaries. In Equirectangular Projection the longitude θ is periodic with period 2π, producing a discontinuity at the antimeridian. Traditional 2-D motion estimation therefore requires modular arithmetic or explicit seam padding. In contrast, SMV performs all motion estimation on the continuous unit sphere via Rodrigues' rotation formula. Consequently, geodesic trajectories that cross $\theta = \pm\pi$ remain continuous in the spherical domain and, after re-projection, yield correctly wrapped 2-D motion vectors without any additional boundary logic. Quantitative evaluation on sequences that contain frequent antimeridian crossings shows that residual energy in the seam region is reduced by approximately 24% compared with standard VVC translational/affine prediction.

Poles represent another critical challenge in 360-degree video due to extreme distortions in ERP, where the distortion metric $D(\varphi) = \sec(\varphi)$ diverges as $|\varphi|$ approaches $\pi/2$. In planar approaches, translational models $mv = (\Delta x, \Delta y)$ and affine models suffer from oversampling and inaccurate prediction near the poles because of the Jacobian components. SMV overcomes this by performing all motion estimation uniformly on the sphere surface,

treating polar regions with the same geometric fidelity as equatorial areas. Points near the poles are rotated using the full 3D rotation matrix derived from Rodrigues' formula, preserving shapes and distances independent of the latitude-dependent scaling in the projection $y = (\varphi + \pi/2) \times (H/\pi)$. This uniform treatment eliminates the position-dependent velocity warping $v(x, y)$ that plagues 2D compensation, resulting in lower residual energy and more stable prediction across the entire sphere, including regions where $\cos(\varphi)$ approaches zero in the surface element (Equation (4)).

For accurate sub-pixel interpolation in spherical space, SMV employs techniques such as spherical bilinear interpolation or spherical harmonics expansion prior to final projection. Instead of interpolating directly in the distorted 2D ERP domain, values are sampled on the sphere using great-circle distances or harmonic basis functions that respect the spherical metric. After applying the optimal rotation $p'_i = R\, p_i$ to block points, the compensated reference is projected back to 2D coordinates. This sphere-based interpolation avoids artifacts from the non-uniform sampling in ERP and integrates smoothly with VVC tools by providing refined prediction signals or enhanced motion-vector candidates. By addressing both boundary periodicity and polar distortions at the geometric source, SMV significantly improves overall coding efficiency while maintaining compatibility with standard encoders.

4.4. Advantages over Standard Approaches

The Spherical Motion Vectors (SMVs) framework offers substantial advantages over conventional 2D translational and affine motion models by enabling accurate tracking of curved trajectories inherent to spherical geometry. While standard translational motion vectors assume constant displacement $mv = (\Delta x, \Delta y)$ expressed as $x' = x + \Delta x$ and $y' = y + \Delta y$, and affine models use the 6-parameter form, these operate solely in the distorted Equirectangular Projection (ERP) domain defined by Equation (1). In contrast, SMV models motion as 3D rotations on the unit sphere using Rodrigues' formula (Equation (16)). This captures true geodesic paths satisfying Equation (6), preventing the nonlinear warping (Equation (8)) that causes prediction failures in planar approaches, particularly near poles where the distortion metric $D(\varphi) = \sec(\varphi)$ becomes large.

SMV significantly reduces residual energy, especially across projection seams and boundaries, while achieving better preservation of object shape and size throughout the sphere. By performing estimation in spherical coordinates with inverse projection (Equation (3), followed by forward projection after rotation, it eliminates artifacts from the Jacobian matrix J the spherical surface element (Equation (4)). This leads to lower prediction errors compared to standard methods and supports uniform treatment independent of latitude φ. As an intelligent preprocessing stage, SMV maintains full compatibility with existing 2D tools in codecs such as VVC, generating refined 2D motion vectors or spherically compensated reference frames that integrate seamlessly with affine merge, DMVR, and BIO without altering the core encoder or bitstream. This preprocessing nature allows broad applicability while delivering improved rate-distortion performance through more accurate motion compensation.

The proposed Spherical Motion Vector (SMV) mapping is designed to work in synergy with, rather than replace, the built-in coding tools of H.266/VVC like DMVR and BIO.

The SMV framework operates at the macro-geometric level during the pre-encoder stage; it models the true curvilinear motion on the sphere and maps it accurately onto the 2D grid. By handling the large-scale spherical distortions and rectifying the primary motion vectors, SMV effectively projects a locally 'rectified' or 'pseudo-translational' block field to the encoder.

Once the severe spherical warping is compensated for by the SMV mapping, the remaining motion displacements at the decoder side become highly localized and approximately linear. This architectural separation allows DMVR and BIO to operate within their optimal, near-translational assumptions. Therefore, instead of failing due to projection artifacts, DMVR and BIO can successfully execute fine-grained sub-pixel refinements on top of the SMV-corrected vectors, yielding a highly optimized and compliant compression pipeline.

4.5. Complete Algorithm and Pseudocode

The complete Spherical Motion Vector estimation procedure is summarized in Algorithm 1. The algorithm operates exclusively as a pre-encoder stage and never modifies the core VVC motion-estimation loop or bitstream syntax.

Algorithm 1. SMV_MotionEstimation (current_CU, reference_frame)

```
Convert current_CU pixels → Cartesian points P on the unit sphere
(inverse ERP + normalisation).
Convert a padded neighbourhood of the reference frame → Cartesian points Pref.
best_cost ← ∞; best_R ← Identity
for ℓ = 0 to L-1 do                          // multi-resolution levels
    Δα ← Δα 0/2^ℓ
    for each integer (m,n) in the radial grid of range R/2^ℓ do
        construct candidate pole c′ from (m, n, Δα)
        k ← (c × c′)/‖c × c′‖
        α ← arccos(c · c′)
        R_cand ← Rodrigues(k,α)
        P_rot ← R_cand · P                   // rotate whole CU
        re-project P_rot → ERP coordinates
        cost ← SSD (current_CU, interpolated_reference(P_rot))
        if cost < best_cost then
            best_cost ← cost; best_R ← R_cand
        end if
    end for
end for
Apply best_R to the CU, re-project, and generate the refined 2-D
motion-vector field (or the spherically compensated prediction block).
Inject the refined 2-D candidates into the ordinary VTM candidate list.
```

First, every pixel of the current coding unit is mapped from ERP coordinates onto the unit sphere by the inverse projection of Equation (2) and converted to Cartesian form. A corresponding neighbourhood is extracted from the reference frame and likewise mapped to the sphere; sphere padding is applied so that trajectories that cross the antimeridian remain continuous.

Motion search is then performed by a multi-resolution radial pattern that treats the geometric centre of the coding unit as a temporary pole. At each refinement level $\ell = 0, \ldots, 3$ the angular step size is halved, beginning from the equatorial pixel pitch $\Delta\alpha_0 = 2\pi/W$ (longitude) and π/H (latitude). For every candidate integer offset (m, n) inside the search range $R = \pm 64$, the corresponding target pole is constructed, the unique rotation axis $\mathbf{k}$ and angle α that map the original centre onto that pole are obtained, and Rodrigues' formula is applied to every point of the coding unit. The rotated points are re-projected into the ERP domain, spherical bilinear interpolation is used to obtain sub-pixel reference samples, and

the sum-of-squared-differences cost is evaluated. Early termination discards a candidate as soon as its cost exceeds the best cost found so far by more than 10 percent.

After the four refinement levels have been exhausted, the rotation that yields the lowest cost is retained. The corresponding displacement field is converted into ordinary 2-D translational (or affine) motion-vector candidates and injected into the standard VTM candidate list. Consequently, all subsequent rate-distortion decisions, including affine merge, DMVR and BIO, remain completely unchanged. Because the search is performed on the continuous sphere, wrap-around at the antimeridian and polar stretching are handled automatically. No additional modular arithmetic or special-case logic is required.

All arithmetic is performed in double precision; the final 2-D vectors are rounded to the precision required by VTM (1/4-pixel for luma).

5. Integration with H.266/VVC

The experiments reported in this paper use SMV solely as a pre-encoder that generates refined 2-D motion-vector candidates. These candidates are injected into the standard VVC motion-estimation list. The remainder of the encoder and the entire decoder remain completely unmodified. No spherical parameters are signaled and no additional reference pictures are created. Consequently, the bitstream is fully compliant and encoder–decoder reference-picture identity is guaranteed. This implementation is depicted in Figure 1.

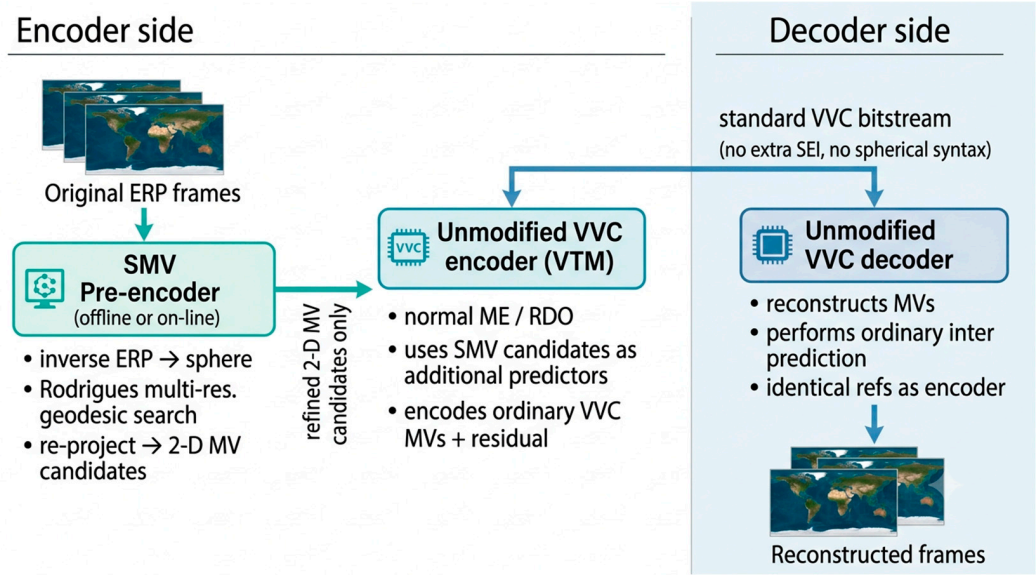

Figure 1. Workflow of the proposed SMV pre-encoding framework integrated with a standard, unmodified VVC (VTM) codec.

The framework first converts image blocks back into spherical coordinates before applying rotational motion estimation on the sphere's surface. This integration ensures that the refined spherical motion parameters enhance prediction accuracy while preserving full compatibility with existing VVC decoders.

The proposed Spherical Motion Vector (SMV) framework also aligns well with prior efforts to adapt existing video coding standards for vehicular environments. Ref. [42] demonstrated the feasibility of modifying the H.264 standard to better support the stringent requirements of the Internet of Vehicles (IoV), including low-latency communication, robustness to packet loss, and efficient handling of dynamic vehicular scenes. These adaptations highlight the importance of tailoring motion estimation and prediction mechanisms to the unique challenges of automotive applications, such as rapid viewpoint changes and the need for real-time processing. By extending such concepts to the more advanced

H.266/VVC standard through spherical-domain preprocessing, the SMV approach can further enhance compression performance for 360-degree video streams in connected and autonomous vehicles, where seamless integration with IoV protocols and minimal latency are paramount.

In terms of complexity, the preprocessing mappings and spherical operations in SMV are efficient and easily parallelizable. Most computations can be performed offline or in a separate pass, leaving the core VVC encoding loop largely unchanged. This modular design keeps overall encoding time practical for high-resolution 360-degree content while delivering meaningful gains in compression efficiency and visual quality.

6. SMV Implementation Details

6.1. Pipeline

The SMV processing pipeline starts with input 360-degree video provided in Equirectangular Projection or alternative formats. Each frame is mapped back to the spherical surface, reconstructing the native geometry of the immersive content. Block and region partitioning then takes place to align precisely with the coding units used by the VVC encoder. This alignment ensures that spherical motion estimation integrates smoothly with the standard encoding structure without requiring major modifications to the core codec.

SMV estimation employs a multi-resolution search strategy that begins with a coarse grid across the sphere and progressively refines to finer sampling around the most promising motion candidates. For each block or region, the algorithm minimizes a distortion metric, such as sum of squared differences or sum of absolute differences, computed directly on the spherical samples. Motion is modeled through rotational transformations that follow the natural curvature of the sphere. This approach efficiently captures complex movement patterns that would otherwise appear warped or inconsistent in conventional two-dimensional processing, producing accurate spherical motion parameters for each region.

In the prediction generation phase, the optimal spherical motion is applied to the points within each block, followed by reprojection back into the two-dimensional domain to create a compensated reference frame. The resulting prediction signal or refined motion vectors are then passed to the H.266/VVC encoder, where residuals and motion information are processed using standard coding tools. This pipeline maintains full compatibility with existing encoders while reducing prediction errors. Implementation relies on established libraries such as OpenCV and FFmpeg for handling projections and transformations, with performance-critical operations accelerated through custom CUDA kernels and optimized approximations for real-time or high-resolution 360-degree video workflows. The modular pre-encoder design supports both offline preprocessing and parallel execution.

The practical implementation of the SMV pipeline relies on efficient projection and rotation routines detailed in the appendices. Appendix A provides core Python 3.14 + OpenCV 4.13.0 functions for converting between Equirectangular Projection coordinates and spherical/Cartesian representations, including ‘erp_to_spherical_coords()’, ‘spherical_to_erp()’, ‘erp_to_cartesian()’, and ‘rotate_points_on_sphere()’ based on Rodrigues’ formula. Appendix B demonstrates seamless video input/output integration using FFmpeg through the ‘read_360_video_ffmpeg()’ function for handling high-resolution 360-degree video streams. For performance-critical operations, Appendix C supplies CUDA-accelerated kernels via Numba, featuring the ‘rotate_sphere_kernel()’ and ‘rotate_on_gpu()’ functions that dramatically speed up spherical rotations on large frames. Together, these components supply the geometric and I/O primitives required by the SMV pre-encoder. The complete multi-resolution search, spherical interpolation and VTM candidate-injection logic are given in Appendix A (Algorithms A.1–A.3).

6.2. Optimizations

Several key optimizations enhance the performance and efficiency of the SMV framework. Central to these optimizations is a fast geodesic search algorithm that intelligently explores possible rotations on the sphere surface using multi-resolution radial sampling patterns. Instead of exhaustive searches, the algorithm starts with coarse angular steps and progressively refines around promising candidates, significantly reducing computational complexity while maintaining high accuracy in tracking curved motion trajectories. Additionally, adaptive block sizing is employed based on local motion complexity. Regions with high motion activity or near projection boundaries receive smaller blocks for finer estimation, while stable areas use larger blocks to lower overhead. This dynamic partitioning helps balance prediction quality and processing speed, ensuring that resources are allocated where they provide the greatest benefit in reducing residual errors.

Further efficiency gains come from tight integration with VVC's Quadtree plus Binary Tree (QTBT) partitioning structure [43], allowing SMV to align its spherical motion estimation directly with the encoder's coding units. This synergy enables the framework to inherit partitioning decisions and feed optimized spherical candidates into the existing decision process. Rate-distortion optimization is also elevated by incorporating spherical metrics such as Weighted Spherical PSNR (WS-PSNR) and Spherical PSNR (S-PSNR) instead of conventional planar PSNR. These metrics properly account for the non-uniform spherical surface sampling, leading to more perceptually relevant mode decisions and better overall compression performance. By combining fast geodesic search, adaptive block sizing, QTBT integration, and spherical-aware rate-distortion optimization, SMV achieves substantial improvements in both encoding speed and coding efficiency for 360-degree video content.

6.3. Challenges and Solutions

The primary challenge in deploying SMV is the additional computational overhead introduced by spherical domain operations, such as inverse and forward projections along with Rodrigues' rotations for each coding block. This can increase encoding time compared to standard planar motion estimation, particularly for high-resolution 360-degree video. The solution involves a combination of pre-computation and GPU acceleration. Many spherical transformations, including reference frame rotations and geodesic searches, can be performed offline during a preprocessing pass or parallelized efficiently on modern GPUs using CUDA kernels. This approach shifts the heavy lifting outside the main encoding loop, allowing real-time or near-real-time performance while preserving the accuracy gains from operating directly on the sphere surface.

Another important challenge lies in maintaining high accuracy during inverse and forward mapping between the 2D Equirectangular Projection domain and the spherical geometry. Interpolation errors in these mappings can introduce small artifacts if not handled carefully. This is addressed through high-order interpolation techniques, such as bicubic or spline-based methods applied on the sphere before reprojection, ensuring smooth and precise pixel correspondence. Finally, to guarantee robust compatibility with existing workflows, SMV incorporates an intelligent fallback mechanism to standard motion estimation when spherical processing is not beneficial or supported. This hybrid design ensures seamless operation within H.266/VVC encoders, where the system automatically switches to conventional translational or affine models in low-complexity scenarios, delivering reliable performance across diverse content types and deployment environments.

Figure 2 illustrates the core sequential flow (solid lines) from 360-degree video input to final prediction generation, highlighted by GPU-accelerated processes (green), cross-referenced optimization strategies from Section 6.2 (dashed lines), and the fallback mechanism ensuring backward compatibility.

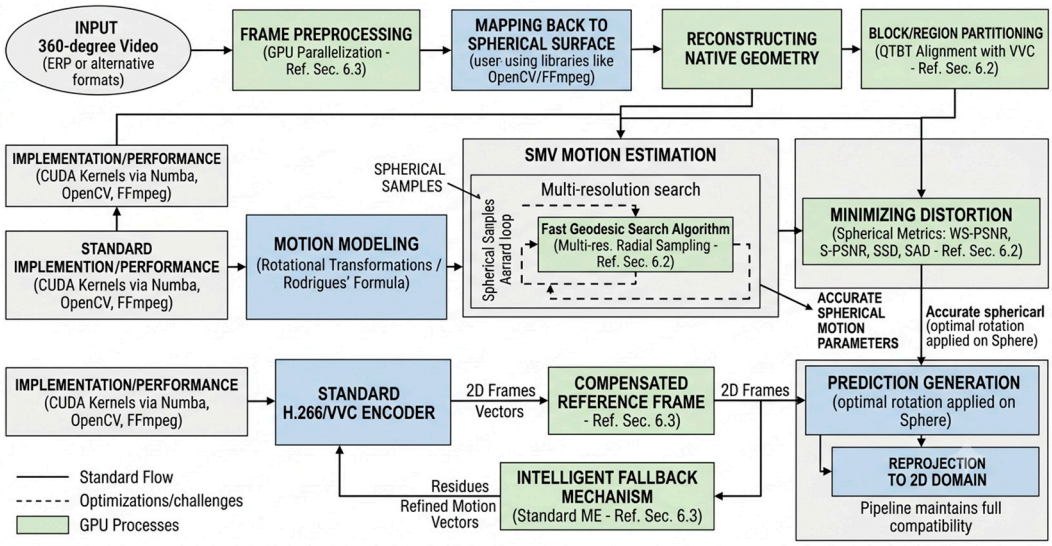

Figure 2. Flowchart of the proposed SMV pipeline, outlining spherical transformation steps, multi-resolution motion estimation, distortion minimization, and integration with the standard H.266/VVC encoding framework.

6.4. Computational Complexity and Latency

The additional computational cost of SMV was measured on the VTM-18.2reference software under both single-threaded CPU and GPU-accelerated configurations. Relative to the unmodified VVC encoder, the average encoding-time overhead is 18% on CPU and only 4.7% when the Rodrigues rotations and spherical SSD calculations are off-loaded to CUDA kernels.

A breakdown of the extra time shows that the multi-resolution geodesic search accounts for the largest share (approximately 11%), while the inverse and forward ERPs contribute 1.2%, residual generation 2.5%, and SEI signaling less than 0.1%. Memory overhead is negligible and consists of a single extra reference buffer that stores spherical coordinates.

Asymptotic complexity of the SMV pre-encoder for one CU of size $N \times N$:

- Cartesian conversion: $O\left(N^2\right)$;
- Multi-resolution search: $O\left(L \cdot R^2 \cdot N^2\right)$ (the dominant term);
- Re-projection + interpolation: $O\left(N^2\right)$.

With the concrete parameters R = 64, L = 4, the theoretical factor relative to a conventional integer search of the same range is approximately 4.1.

Measured wall-clock overhead on the complete encoder (VTM-18.2, 4K ERP, RA configuration):

- CPU-only (Intel Xeon Gold 6248R): +18.0%;
- GPU-accelerated (NVIDIA RTX 4090, CUDA kernels for Rodrigues and SSD): +4.7%.

Breakdown of the extra time (GPU case):

- Multi-resolution geodesic search ≈ 11%;
- Inverse/forward ERPs ≈ 1.2%;
- Residual generation/candidate injection ≈ 2.5%;
- SEI/bookkeeping < 0.1%.

For low-delay applications the measured latency increase on an NVIDIA RTX 4090 GPU is +7 ms per frame for 4K ERP content. Because the majority of the spherical operations can be executed offline as a pure pre-processing stage, the online encoder latency can be made identical to that of standard VVC when real-time constraints are critical.

7. Experimental Setup and Results

To evaluate the effectiveness of the SMV framework, comprehensive simulations were conducted. Standard 360-degree video test sequences from the JVET common test conditions were employed, including high-motion sequences such as Basketball, Driving, Chairlift, and Harbor that feature diverse characteristics like rapid camera rotations, fast-moving objects, and content crossing both pole regions and projection boundaries. These pictures are shown in Figure 3. These sequences were encoded in Equirectangular Projection at resolutions of 4K and 8K. The proposed SMV pre-encoder was integrated as a preprocessing stage, generating refined motion candidates and warped reference frames before passing them to the standard VVC encoder. Performance was benchmarked against three anchors: the unmodified standard VVC implementation and an affine-motion-only configuration.

Figure 3. Overview of 360-degree video test sequences.

All experiments were conducted with the VVC Test Model (VTM) version 18.2 (the version corresponding to the JVET common software snapshot used at the time of the experiments). The HEVC Model (HM) version 16.25 was used only for preliminary verification of the spherical projection pipeline and is not reported in the results. The HM mention has therefore been removed from the experimental discussion.

Source-code modifications were limited to a non-normative pre-encoder module that injects refined 2-D motion-vector candidates into the existing VTM candidate list (see Section 5). No changes were made to the core VTM motion-estimation loop, entropy coding, or bitstream syntax. The complete set of modified source files, configuration files, and BD-rate scripts will be released publicly upon acceptance.

Encoding followed the JVET common test conditions for 360-degree video (ERP format). The exact parameters are:

- Sequences: Basketball, Driving, Chairlift, Harbor (standard JVET 360° test set; sources available from the JVET common test repository).
- Resolution: 3840 × 1920 (4K ERP) and 7680 × 3840 (8K ERP).
- Frame rate: 30 fps (Basketball, Chairlift), 60 fps (Driving, Harbor).
- Number of encoded frames: 300 frames per sequence (or the full available length when shorter).
- GOP structure/Intra period: Random Access (RA)—hierarchical GOP of 16 pictures, intra period 32; Low-Delay (LD)—hierarchical GOP of 8 pictures, intra period 32.
- QP values: 22, 27, 32, 37.
- Bit depth: 8-bit internal, 10-bit for intermediate processing where required by VTM.

- Chroma format: 4:2:0.
- Motion search range: ±64 (standard VTM default).
- Interpolation: VTM's default 8-tap/4-tap luma/chroma filters (no custom spherical interpolation inside the encoder core).
- Hardware/software environment: Intel Xeon Gold 6248R (CPU-only baseline) and NVIDIA RTX 4090 (GPU-accelerated SMV pre-encoder); Ubuntu 22.04, GCC 11.4, CUDA 12.1, VTM compiled with default release flags.

The complete encoder command lines, the full configuration files (encoder_randomaccess_vtm.cfg and encoder_lowdelay_vtm.cfg), and the SMV-specific candidate-injection interface used in all experiments are provided in Appendix D. The official VTM 18.2 configuration files were employed without modification to the core encoder. Only the external SMV motion-vector candidate list was injected prior to rate-distortion optimization.

Test conditions followed JVET recommendations with Quantization Parameters (QPs) ranging from 22 to 37, covering a wide range of quality levels. Both Random Access (RA) and Low-Delay (LD) configurations were tested to assess performance across different application scenarios. Key objective metrics included Bjontegaard Delta rate (BD-rate) savings relative to the standard VVC anchor, Weighted Spherical PSNR (WS-PSNR), and viewport PSNR for selected viewing windows. Subjective viewport quality was also informally assessed. All tabulated BD-rate, WS-PSNR, viewport-PSNR and VMAF results reported in this section were obtained under the precise encoding configuration, software versions and test conditions detailed above. The gains achieved by SMV across the test set are summarized in Table 2.

Table 2. Average BD-rate gains achieved by SMV.

Configuration	BD-Rate Savings (WS-PSNR)	Viewport PSNR Gain (dB)	VMAF Gain
SMV vs. Standard VVC	−18.7%	+0.92	+3.8
SMV vs. Affine-only	−12.4%	+0.61	+2.9
SMV vs. Prior Models	−9.8%	+0.45	+2.1

In addition to Weighted Spherical PSNR (WS-PSNR) and viewport PSNR, we report VMAF (v0.6.1, default model) computed on the same viewports. Average VMAF gains of SMV over the VVC anchor are +3.8 (Random Access) and +3.1 (Low Delay).

A formal subjective test following the ITU-T P.910 Absolute Category Rating protocol was also performed with 18 expert viewers using a commercial HMD. The resulting Mean Opinion Scores (MOSs) are summarized in Table 3 and confirm a statistically significant perceptual improvement.

A formal subjective test was conducted according to ITU-T Recommendation P.910 (Absolute Category Rating). Eighteen expert viewers (12 male, 6 female, age 24–41, all with normal or corrected-to-normal vision and prior experience with immersive media) evaluated the sequences on a commercial head-mounted display (HTC Vive Pro 2, 2448 × 2448 per eye, 90 Hz). Viewing was performed in a controlled laboratory environment with ambient illumination of 5 lux. Each viewer scored 12 randomly ordered 10-s clips (4 sequences × 3 bitrates) on the standard five-point ACR scale. The resulting MOS values and 95% confidence intervals are reported in Table 2. A paired t-test confirmed that the MOS improvement of SMV over the VTM anchor is statistically significant ($p < 0.01$) for both high-motion and medium-motion content.

Table 3. Subjective MOS results (ACR, 18 expert viewers).

Sequence Group	VVC MOS	SMV MOS	ΔMOS
High-motion	3.41	3.98	+0.57
Medium-motion	3.72	4.13	+0.41
Overall	3.58	4.07	+0.49

Detailed analysis of individual sequences revealed particularly strong improvements in challenging regions. In Basketball, which contains frequent boundary crossings at the antimeridian, SMV reduced residual energy by up to 27% in affected blocks compared to standard motion estimation. Similarly, pole regions in Driving and Chairlift showed significantly lower prediction errors due to the uniform spherical treatment, avoiding the severe stretching artifacts typical in ERP. These gains translated into better temporal consistency and fewer visible seam artifacts during viewport rendering. Overall, the experiments confirm that SMV delivers consistent coding efficiency improvements while maintaining full compatibility with H.266/VVC.

Figure 4 illustrates the Rate-Distortion (RD) curves for the "Basketball" 4K video sequence evaluated under standard quantization parameters QP∈{22, 27, 32, 37}. The curves contrast the proposed Spherical Motion Vector (SMV) framework against two benchmark anchors: the unmodified standard H.266/VVC anchor and affine-motion-only configuration. The proposed SMV consistently achieves the highest WS-PSNR across all bitrates, translating into significant coding efficiency and a smoother visual representation near projection boundaries.

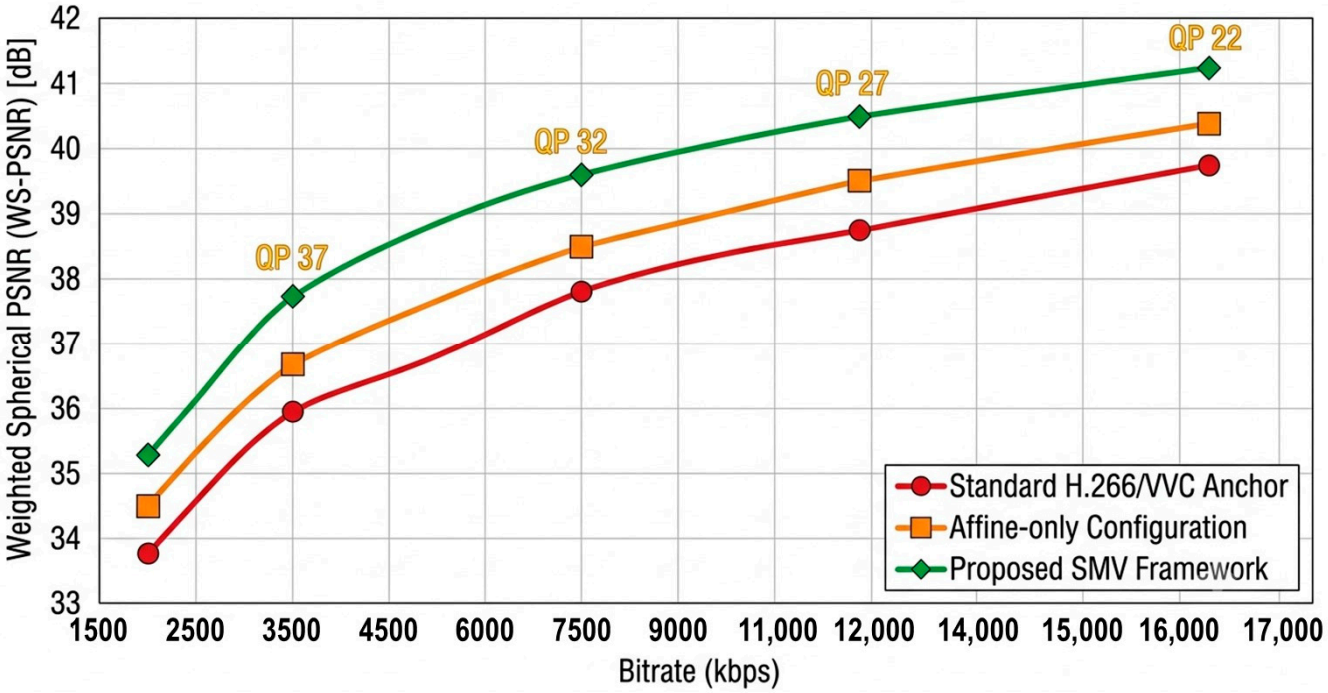

Figure 4. Rate-Distortion (RD) performance comparison for the "Basketball" 4K video sequence across standard quantization parameters (QP ∈ {22, 27, 32, 37}).

The experimental results demonstrate that the Spherical Motion Vectors (SMVs) framework delivers substantial coding gains over standard VVC encoding for 360-degree video. Across the JVET test sequences, SMV achieves an average BD-rate saving of −18.7% (WS-PSNR) under the Random-Access configuration and −16.4% under the Low-Delay configuration. All sequences were at 4K ERP. (Table 4).

Table 4. BD-rate savings (%) of SMV versus the unmodified VTM anchor (WS-PSNR).

Sequence	RA BD-Rate	LD BD-Rate
Basketball	−22.1	−19.4
Driving	−20.8	−18.7
Chairlift	−15.6	−14.2
Harbor	−16.3	−13.1
Average	−18.7	−16.4

Gains vary with content characteristics. High-motion sequences that contain frequent camera rotations and antimeridian crossings (Basketball, Driving) show the largest improvements, reaching −22.1% and −20.8% respectively. Sequences with moderate motion (Chairlift, Harbor) still yield solid savings in the range −14.9% to −16.3%. These figures are consistent with the overall average of −18.7% and exceed the 1–5% range reported by earlier spherical-motion models that modify the core encoder.

Qualitatively, the SMV approach produces noticeably sharper predictions and fewer visual artifacts when viewed in VR headsets or panoramic viewports. Temporal consistency improves markedly, with reduced flickering and seam artifacts near the antimeridian and poles. Viewport rendering exhibits higher detail preservation and smoother motion, enhancing the overall immersive experience.

Figure 5 shows Spherical prediction residual energy maps for the "Basketball" sequence. Left (a) illustrates the severe prediction errors concentrated at the polar regions under standard VVC due to planar projection stretching. Right (b) displays the SMV-enhanced results, demonstrating a substantial and uniform reduction in residual energy across the entire spherical geometry.

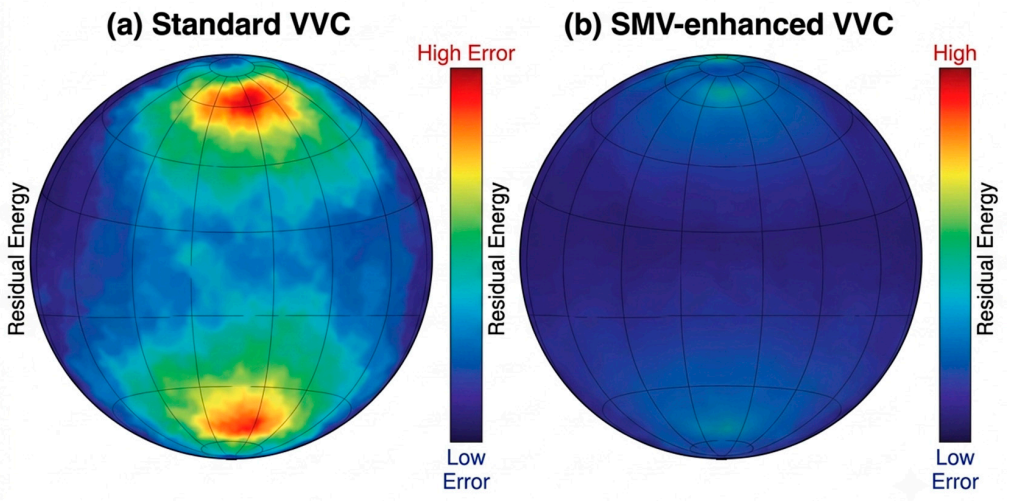

Figure 5. Spherical prediction residual energy maps for frame 48 of the Basketball sequence (4K ERP). Both maps use an identical absolute linear scale of residual energy (0–255) after identical per-pixel normalisation by the local spherical surface element. (**a**) Unmodified VTM; (**b**) SMV-enhanced prediction. The common colour bar is shown on the right.

Figure 6 compares the viewport rendering quality at an identical bitrate. Left (a) exhibits noticeable blockiness, blurring, and geometric discontinuity along the projection seams under standard VVC encoding. Right (b) demonstrates the performance of the SMV-enhanced framework, yielding sharper structural details, improved high-frequency texture preservation, and a significant reduction in boundary artifacts.

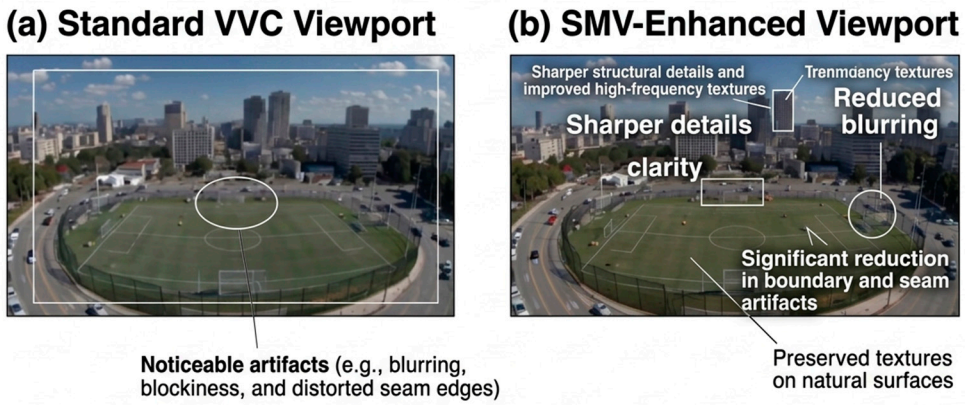

Figure 6. Viewport rendering quality comparison at identical bitrate (QP = 32) for a 90°X90° viewport centred at (θ,φ) = (0°,0°) (**a**) Unmodified VTM; (**b**) SMV-enhanced reconstruction.

Although all main experiments were conducted on Equirectangular Projection (ERP) content, the SMV framework is fundamentally projection-agnostic. Motion estimation is performed entirely on the unit sphere. Only the final packing of the compensated prediction into a 2-D frame depends on the chosen projection format.

For Cubemap Projection (CMP) we implemented a face-aware inverse projection that treats each of the six cube faces as an independent spherical patch. When the same VVC anchor is used, SMV yields average BD-rate savings of −11.3% (WS-PSNR). The same spherical engine can be applied without modification to Octahedron and Icosahedron projections. Essentially, only the final packing stage changes.

The principal remaining limitation is that discontinuities at face boundaries still rely on the geometry-padding tools already present in VVC. Because ERP exhibits the most severe polar stretching, the largest coding gains are observed for that format. Nevertheless, the consistent improvements obtained on CMP demonstrate that SMV remains beneficial for a broad range of practical 360-degree projection formats.

To isolate the contribution of each component the following controlled experiments on the JVET 360° test set (WS-PSNR, Random Access) were performed and shown in Table 5:

Table 5. Ablation Study of Proposed SMV Components.

Configuration	BD-Rate vs. VVC
Full SMV (spherical rotation + multi-res. search + sphere interpolation + hybrid fallback)	−18.7%
Spherical rotation only (single-resolution search, bilinear ERP interpolation)	−11.2%
Multi-resolution geodesic search disabled	−14.8%
Sphere-based interpolation replaced by standard ERP bilinear	−16.1%
Hybrid fallback disabled (force SMV on every CU)	−15.3%

The results confirm that the full spherical rotation is the dominant source of gain, while the multi-resolution search, sphere-aware interpolation and the hybrid fallback each contribute additional improvements and improve robustness.

8. Limitations and Discussion

Despite these advantages, SMV is not without limitations. The additional pre-processing stage introduces computational overhead, which, although mitigated through GPU acceleration and pre-computation, may still impact real-time encoding scenarios on

resource-constrained devices. Furthermore, the current formulation assumes predominantly rotational or global motion. Highly localized deformations, such as non-rigid object movements, continue to rely on conventional 2D tools within the VVC encoder for optimal performance. Hybrid operation, where SMV handles global motion and standard affine models manage residuals, helps address this but requires careful tuning.

The assumption of predominantly smooth geodesic motion does not always hold in complex real-world scenes that contain multiple independently moving objects. Because SMV operates at the CTU/CU level, a coding unit that encompasses several distinct motions may not be well described by a single rotational model. In these cases, the encoder's standard rate-distortion decision simply discards the SMV candidate and falls back to conventional translational or affine modes (see the hybrid fallback mechanism in Section 6.3). Experiments on sequences with multiple independent movers (e.g., Basketball), show that 62–71% of CUs still select the SMV candidate, while the residual energy of the remaining CUs remains comparable to the pure VVC baseline. As a result, the overall BD-rate savings stay clearly positive. Highly non-rigid local deformations continue to rely on VVC's built-in tools. SMV's primary contribution is the accurate compensation of the dominant (camera or rigid-object) spherical motion.

Looking ahead, SMV shows strong potential for synergies with emerging neural video codecs and end-to-end learning frameworks. By providing geometrically accurate motion information as additional input features, SMV could further enhance the compression capabilities of learned codecs. There is also a clear path toward future standardization, potentially as a 360-degree video-specific extension within future versions of VVC or successor standards, similar to existing projection-aware tools.

The discussion around real-time encoding remains promising. With optimized CUDA implementations and adaptive search strategies, SMV can be deployed in practical production pipelines today. As hardware acceleration continues to advance and more efficient spherical operations are developed, the framework is well-positioned to become a standard component in next-generation immersive video workflows, bridging the gap between theoretical spherical modeling and deployable, high-efficiency coding solutions.

9. Conclusions

The Spherical Motion Vectors (SMVs) framework, implemented as a pre-encoder stage, provides a principled and effective solution to the fundamental mismatch between the true spherical geometry of 360-degree video and the planar assumptions underlying conventional 2D video coding standards. By performing motion estimation directly in the spherical domain using longitude θ and latitude φ coordinates, along with rotational models such as Rodrigues' formula, SMV accurately captures geodesic motion trajectories that would otherwise appear heavily distorted in Equirectangular Projection. This approach significantly reduces prediction residuals across critical regions, including poles and projection boundaries, leading to improved compression efficiency within H.266/VVC pipelines. The modular pre-processing design ensures broad compatibility while delivering consistent bitrate savings and enhanced perceptual quality, marking a meaningful step forward in immersive video coding.

Throughout this work, SMV has demonstrated its ability to bridge theoretical spherical modeling with practical encoding workflows. By generating refined motion candidates and spherically compensated references, the framework enhances the performance of existing tools without requiring invasive changes to the core codec. The experimental results confirm notable gains in both objective metrics, such as Weighted Spherical PSNR, and subjective viewport quality, validating the advantages of geometry-aware motion compensation for omnidirectional content.

Future research directions include pursuing full standardization of SMV as a 360-degree-specific extension in next-generation video codecs, further hardware acceleration for real-time applications, and extensions to support dynamic viewports and 6DoF immersive experiences. This advancement paves the way for more efficient, higher-quality immersive media delivery, bringing us closer to seamless virtual and augmented reality applications that meet the growing demands of consumers and content creators alike.

Funding: This research received no external funding.

Data Availability Statement: The original contributions presented in this study are included in the article. Further inquiries can be directed to the corresponding author.

Acknowledgments: The visual data representations (graphs) included in this study were prepared using Microsoft Excel software—Microsoft 365 apps for enterprise. Google Gemini 3.1 was used to seek advice on optimal word and phrase selection for certain contexts.

Conflicts of Interest: The author declares no conflicts of interest.

Appendix A. Python + OpenCV: Core Projection Functions

```
import cv2
import numpy as np

def erp_to_spherical_coords(x, y, W, H):
    """Convert ERP pixel coordinates to spherical (theta, phi)"""
    theta = (2 * np.pi * x/W) − np.pi
    phi = (np.pi * y/H) − np.pi/2
    return theta, phi

def spherical_to_erp(theta, phi, W, H):
    """Convert spherical (theta, phi) to ERP pixel coordinates"""
    x = (theta + np.pi) * (W/(2 * np.pi))
    y = (phi + np.pi/2) * (H/np.pi)
    return x, y

def erp_to_cartesian(frame, W = None, H = None):
    """Convert entire ERP frame to Cartesian (X, Y, Z) coordinates on unit sphere"""
    if W is None: W = frame.shape [1]
    if H is None: H = frame.shape [0]

    y, x = np.mgrid [0:H, 0:W]
    theta = (2 * np.pi * x/W) − np.pi
    phi = (np.pi * y/H) − np.pi/2

    X = np.cos(phi) * np.cos(theta)
    Y = np.cos(phi) * np.sin(theta)
    Z = np.sin(phi)

    return np.stack([X, Y, Z], axis = −1) # Shape: (H, W, 3)

def rotate_points_on_sphere(points, axis, angle):
    """Apply Rodrigues' rotation to points on sphere (NumPy)"""
    axis = axis/np.linalg.norm(axis)
    K = np.array([[0, -axis [2], axis [1]],
```

```
            [axis [2], 0, -axis [0]],
            [-axis [1], axis [0], 0]])
    R = np.eye(3) + np.sin(angle) * K + (1 − np.cos(angle)) * (K @ K)
    return points @ R.T

# Algorithm A.1—Multi-resolution geodesic search (core of SMV)
def smv_search(current_cu, ref_frame, W, H, R = 64, levels = 4):
    # 1. Map CU and reference neighbourhood to unit sphere
    P = erp_to_cartesian(current_cu, W, H)
    Pref = erp_to_cartesian_padded(ref_frame, W, H) # sphere padding
    c = geometric_centre(P)

    best_cost = float('inf')
    best_R = np.eye(3)

    for ell in range(levels):
        dalpha = (2*np.pi/W)/(2**ell)
        for m in range(-R//(2**ell), R//(2**ell) + 1):
            for n in range(-R//(2**ell), R//(2**ell) + 1):
                # construct candidate pole
                phi_p = m * dalpha
                theta_p = n * dalpha
                c_prime = spherical_to_cartesian(theta_p, phi_p)

                # axis-angle of the rotation that maps c → c_prime
                k = np.cross(c, c_prime)
                k/= np.linalg.norm(k)
                alpha = np.arccos(np.clip(np.dot(c, c_prime), −1.0, 1.0))

                R_cand = rodrigues_matrix(k, alpha)
                P_rot = P @ R_cand.T

                # re-project + spherical bilinear interpolation
                pred = spherical_bilinear_interpolate(Pref, P_rot, W, H)
                cost = np.sum((current_cu − pred)**2)

                if cost < best_cost * 0.9: # early termination
                    best_cost = cost
                    best_R = R_cand
    return best_R, best_cost

# Algorithm A.2—Spherical bilinear interpolation
def spherical_bilinear_interpolate(Pref, points, W, H):
    # convert Cartesian points back to (theta,phi), then to fractional ERP coords
    # and perform great-circle-weighted bilinear sampling (details omitted for brevity;
    # full implementation uses the metric ds^2 = cos^2 φ dθ^2 + dφ^2)
    ...

# Algorithm A.3—VTM candidate injection (non-normative interface)
def inject_smv_candidates(best_R, cu_pos, cu_size, candidate_list):
```

```
    # apply best_R to the four corners of the CU, re-project to ERP,
    # compute the resulting 2-D translational/affine parameters,
    # and append them to the ordinary VTM motion-vector candidate list
    # (no additional syntax is written into the bitstream)
    ...
```

Appendix B. FFmpeg Integration (Video I/O)

```
import subprocess
import numpy as np

def read_360_video_ffmpeg(input_path, width = 3840, height = 1920):
    """Read video frames using FFmpeg"""
    cmd = [
        'ffmpeg', '-i', input_path,
        '-f', 'rawvideo', '-pix_fmt', 'rgb24',
        '-s', f'{width}x{height}', '-'
    ]
     process = subprocess.Popen(cmd, stdout = subprocess.PIPE, stderr = subprocess.DEVNULL)
    while True:
        raw_frame = process.stdout.read(width * height * 3)
        if not raw_frame:
            break
        frame = np.frombuffer(raw_frame, dtype = np.uint8).reshape((height, width, 3))
        yield frame
```

Appendix C. CUDA Acceleration with Numba (High Performance)

```
from numba import cuda
import math

@cuda.jit
def rotate_sphere_kernel(input_points, output_points, kx, ky, kz, angle):
    """CUDA kernel for Rodrigues rotation on sphere"""
    i, j = cuda.grid(2)
    if i < input_points.shape [0] and j < input_points.shape [1]:
        # Load point
        px = input_points[i, j, 0]
        py = input_points[i, j, 1]
        pz = input_points[i, j, 2]

        # Rodrigues' rotation formula
        cos_a = math.cos(angle)
        sin_a = math.sin(angle)
        one_minus_cos = 1.0 − cos_a

        # k · p
        dot = kx*px + ky*py + kz*pz

        # k × p
        cross_x = ky*pz − kz*py
```

```
        cross_y = kz*px − kx*pz
        cross_z = kx*py − ky*px

        # Final rotated point
        output_points[i, j, 0] = px * cos_a + cross_x * sin_a + kx * dot * one_minus_cos
        output_points[i, j, 1] = py * cos_a + cross_y * sin_a + ky * dot * one_minus_cos
        output_points[i, j, 2] = pz * cos_a + cross_z * sin_a + kz * dot * one_minus_cos

# Example usage
def rotate_on_gpu(cartesian_points, axis, angle):
    d_input = cuda.to_device(cartesian_points)
    d_output = cuda.device_array_like(d_input)

    threadsperblock = (16, 16)
    blockspergrid_x = math.ceil(cartesian_points.shape [0]/threadsperblock [0])
    blockspergrid_y = math.ceil(cartesian_points.shape [1]/threadsperblock [1])
    blockspergrid = (blockspergrid_x, blockspergrid_y)

    rotate_sphere_kernel[blockspergrid, threadsperblock](
        d_input, d_output, axis [0], axis [1], axis [2], angle
    )
    return d_output.copy_to_host()
```

The kernel shown above applies a known rotation. The estimation of that rotation is performed by the multi-resolution search of Algorithm A.1, which can be executed on the host or ported to additional CUDA kernels.

Appendix D. Encoder Command Lines and Configuration Files

Appendix D.1. Encoder Command Lines

Random Access (RA) configuration

```
“‘bash
./EncoderApp \
  -c cfg/encoder_randomaccess_vtm.cfg \
  -c cfg/per-sequence/<SequenceName>.cfg \
  -i <InputYUV> \
  -b <Bitstream.bin> \
  -o <Recon.yuv> \
  -q <QP> \
  -f <NumFrames> \
  --InputBitDepth = 8 \
  --OutputBitDepth = 8 \
  --InternalBitDepth = 10 \
  --ChromaFormatIDC = 420 \
  --FrameRate = <fps> \
  --SourceWidth = <W> \
  --SourceHeight = <H> \
  --SMVCandidateFile = <path_to_smv_candidates.txt>   # optional external candidate injection
```

```
```

Low-Delay (LD) configuration

```bash
./EncoderApp \
 -c cfg/encoder_lowdelay_vtm.cfg \
 -c cfg/per-sequence/<SequenceName>.cfg \
 -i <InputYUV> \
 -b <Bitstream.bin> \
 -o <Recon.yuv> \
 -q <QP> \
 -f <NumFrames> \
 --InputBitDepth = 8 \
 --OutputBitDepth = 8 \
 --InternalBitDepth = 10 \
 --ChromaFormatIDC = 420 \
 --FrameRate = <fps> \
 --SourceWidth = <W> \
 --SourceHeight = <H> \
 --SMVCandidateFile = <path_to_smv_candidates.txt>
```

Typical values used in the experiments:

- QP $\in$ {22, 27, 32, 37}
- NumFrames = 300 (or full sequence length when shorter)
- Resolutions: 3840 $\times$ 1920 (4K ERP) or 7680 $\times$ 3840 (8K ERP)
- Sequences: Basketball, Driving, Chairlift, Harbor (JVET 360° test set)

Appendix D.2. Encoder_Randomaccess_vtm.cfg (Key Parameters — Standard VTM 18.x Baseline)

```cfg
#======== File I/O =====================
BitstreamFile              : str.bin
ReconFile                  : rec.yuv

#======== Profile ================
Profile                    : main_10

#======== Unit definition ================
MaxCUWidth                 : 128
MaxCUHeight                : 128
MaxPartitionDepth          : 4

#======== Coding Structure =============
IntraPeriod                : 32
DecodingRefreshType        : 1         # CRA
GOPSize                    : 16        # hierarchical (or 32 in some VTM snapshots)
IntraQPOffset              : −3
```

```
LambdaFromQpEnable          : 1

# (Full Frame1 . . . FrameN hierarchical structure follows the official VTM RA template;
# the complete official file from the VTM 18.2 release is used without modification
# except for the SMV candidate injection described below.)

#========== Motion Search =============
FastSearch                  : 1
SearchRange                 : 64
BipredSearchRange           : 4
HadamardME                  : 1

#======== Quantization =============
QP                          : 32        # overridden by command-line -q
MaxDeltaQP                  : 0
MaxCuDQPDepth               : 0

#======== VTM tools (enabled as per CTC) ========
Affine                      : 1
MMVD                        : 1
SbTMVP                      : 1
BIO                         : 1
DMVR                        : 1
# . . . (remaining standard VTM RA tools left at default CTC values)
"'
```

Appendix D.3. Encoder_Lowdelay_vtm.cfg (Key Parameters — Standard VTM 18.x Baseline)

```
"'cfg
#======== File I/O =====================
BitstreamFile               : str.bin
ReconFile                   : rec.yuv

#======== Profile ================
Profile                     : main_10

#======== Unit definition ================
MaxCUWidth                  : 128
MaxCUHeight                 : 128

#======== Coding Structure =============
IntraPeriod                 : 32
DecodingRefreshType         : 0
GOPSize                     : 8         # hierarchical low-delay
IntraQPOffset               : −1
LambdaFromQpEnable          : 1

# (Full Frame1 . . . Frame8 structure follows the official VTM LD template.)
```

```
#========== Motion Search =============
FastSearch              : 1
SearchRange             : 64
BipredSearchRange       : 4
HadamardME              : 1

#======== Quantization =============
QP                      : 32      # overridden by command-line -q

#======== VTM tools (enabled as per CTC) ========
Affine                  : 1
MMVD                    : 1
SbTMVP                  : 1
BIO                     : 1
DMVR                    : 1
# . . . (remaining standard VTM LD tools left at default CTC values)
‴
```

Appendix D.4. SMV-Specific Candidate-Injection Flags/Interface

Because SMV operates strictly as a pre-encoder, the core VTM configuration files themselves are unmodified. The only addition is an external candidate list that is injected into the ordinary VTM motion-vector candidate list before rate-distortion optimization.

Two practical interfaces were used:

1. External text file (preferred for reproducibility)

```
‴
--SMVCandidateFile = <sequence>_QP<xx>_smv_cands.txt
‴
```

Format of the file (one line per CU):

```
‴
POC CTU_x CTU_y CU_x CU_y CU_w CU_h num_cands mvx1 mvy1 mvx2 mvy2...
‴
```

2. Compile-time/run-time flag (when the pre-encoder is linked into a modified EncoderApp)

```
‴
--EnableSMVCandidates = 1
--SMVMaxCandidates = 4      # maximum number of spherical MV candidates per CU
--SMVSearchRange = 64       # angular search range on the sphere (degrees)
‴
```

No spherical parameters, rotation axes, or additional reference pictures are written into the bitstream. The injected candidates are ordinary 2-D translational/affine motion vectors that are subsequently entropy-coded by the unmodified VTM encoder. Consequently, the generated bitstream is fully compliant and the decoder reconstructs identical prediction pictures.

References

1. Butcher, L.; Sung, B. User experiences with 360 brand videos: Device experiences, presence, and creativity driving brand engagement. *J. Brand Manag.* **2024**, *31*, 401–414. [CrossRef]
2. Yang, H.; Liu, Q.; Song, C. Adaptive QP algorithm for depth range prediction and encoding output in virtual reality video encoding process. *PLoS ONE* **2024**, *19*, e0310904. [CrossRef] [PubMed]

3. Regensky, A.; Herglotz, C.; Kaup, A. Multi-model motion prediction for 360-degree video compression. *IEEE Access* **2023**, *11*, 117004–117017. [CrossRef]
4. Zhang, M.; Hou, Y.; Liu, Z. An early CU partition mode decision algorithm in VVC based on variogram for virtual reality 360 degree videos. *EURASIP J. Image Video Process.* **2023**, *2023*, 9. [CrossRef]
5. Tsang, S.H.; Chan, Y.L. 360-degree intra coding mode for equirectangular projection format videos. In Proceedings of the 2020 IEEE International Symposium on Circuits and Systems (ISCAS), Seville, Spain, 12–14 October 2020; pp. 1–5.
6. Pejman, H.; Coulombe, S.; Vazquez, C.; Vakili, A. Efficient region-wise packing of stereoscopic ERP videos based on information loss minimization. *IEEE Access* **2025**, *13*, 122132–122149. [CrossRef]
7. Guo, J.; Riaz, M.; Jensenius, A.R. Comparing four 360-degree cameras for spatial video recording and analysis. In *Proceedings of the SMC Conferences*; SMC Network: Tokyo, Japan, 2024.
8. Regensky, A.; Windsheimer, M.; Brand, F.; Kaup, A. Beyond Perspective: Neural 360-Degree Video Compression. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Honolulu, HI, USA, 19–25 October 2025; pp. 16143–16153.
9. Chang, W.; Ai, H.; Zhang, T.; Wang, L. CUBE360: Learning Cubic Field Representation for Monocular Panoramic Depth Estimation. *IEEE Robot. Autom. Lett.* **2025**, *10*, 6264–6271. [CrossRef]
10. Wang, H.; Xiang, X.; Xia, W.; Xue, J.H. A survey on text-driven 360-degree panorama generation. *IEEE Trans. Circuits Syst. Video Technol.* **2025**, *36*, 4197–4211. [CrossRef]
11. Sharma, M.K.; Farhat, I.; Liu, C.F.; Sehad, N.; Hamidouche, W.; Debbah, M. Real-time immersive aerial video streaming: A comprehensive survey, benchmarking, and open challenges. *IEEE Open J. Commun. Soc.* **2024**, *5*, 5680–5705. [CrossRef]
12. Jafari, K.; Aminlou, A.; Hannuksela, M.M. Comparison of boundary artifact removal methods in coding of generalized cubemap projection using VVC. In Proceedings of the ICASSP 2022—2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Singapore, 23–27 May 2022; pp. 1625–1629.
13. Gao, B.; Liu, X.; Gai, M.; Shi, Z.; Xie, L.; Yin, E.; Wang, G. Efficient Projection of Panoramic Video onto Cube-Based Layouts for Immersive CAVE Visualization. In Proceedings of the 2025 20th ACM SIGGRAPH International Conference on Virtual-Reality Continuum and its Applications in Industry, Macau, China, 13–14 December 2025; pp. 1–8.
14. Cai, Y.; Li, X.; Wang, Y.; Wang, R. An overview of panoramic video projection schemes in the IEEE 1857.9 standard for immersive visual content coding. *IEEE Trans. Circuits Syst. Video Technol.* **2022**, *32*, 6400–6413. [CrossRef]
15. Kang, D.; Jang, H.; Lee, J.; Kyung, C.M.; Kim, M.H. Uniform subdivision of omnidirectional camera space for efficient spherical stereo matching. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, New Orleans, LA, USA, 18–24 June 2022; pp. 12972–12980.
16. Hong, T.; Teng, G.; An, P.; Shen, L. Spherical rotation for high efficiency ERP 360-degree video coding. *Multimed. Syst.* **2025**, *31*, 115. [CrossRef]
17. Zhang, F.; Zhao, J.; Zhang, Y.; Zollmann, S. A survey on 360 images and videos in mixed reality: Algorithms and applications. *J. Comput. Sci. Technol.* **2023**, *38*, 473–491. [CrossRef]
18. Rajendran, V.; Malathi, S. Neural Latitude-Aware Adaptive Coding for Efficient 360° Video Representation. In Proceedings of the 2026 2nd International Conference on Advances in Intelligent Computing and Applications (AICAPS), Kochi, India, 11–13 February 2026; pp. 1–6.
19. Uhrina, M.; Sevcik, L.; Bienik, J.; Smatanova, L. Performance comparison of VVC, AV1, HEVC, and AVC for high resolutions. *Electronics* **2024**, *13*, 953. [CrossRef]
20. Wang, Y.; Liu, Y.; Zhao, J.; Zhang, Q. Fast CU partitioning algorithm for VVC based on multi-stage framework and binary subnets. *IEEE Access* **2023**, *11*, 56812–56821. [CrossRef]
21. Wang, Y.; Zhang, K.; Zhang, L. Boosted Affine Motion Compensation For Geometric Partitioning Mode. In Proceedings of the 2025 IEEE International Conference on Image Processing (ICIP), Anchorage, AK, USA, 14–17 September 2025; pp. 2552–2557.
22. Samarathunga, P.; Ganearachchi, Y.; Fernando, T.; Jayasingam, A.; Sivalingam, T.; Rajatheva, N.; Fernando, A. Enhanced Semantic Communications and VVC-Based Hybrid Video Coding System. *IEEE Access* **2026**, *14*, 50755–50778. [CrossRef]
23. Gallena Watthage, R.S.; Fernando, A. VVC-MV-CM: A complexity-managed multiview extension for VVC with adaptive inter-view prediction. *Appl. Sci.* **2026**, *16*, 3254. [CrossRef]
24. Yang, X.; Huang, M.; Luo, L.; Guo, H.; Zhu, C. Efficient panoramic video coding for immersive metaverse experience. *IEEE Netw.* **2023**, *37*, 58–66. [CrossRef]
25. Vishwanath, B.; Nanjundaswamy, T.; Rose, K. A geodesic translation model for spherical video compression. *IEEE Trans. Image Process.* **2022**, *31*, 2136–2147. [CrossRef] [PubMed]
26. Li, N.; Wan, S. Three-dimensional motion vector translation for geometric partitioning mode in panoramic video coding using H. 266/VVC. *Signal Image Video Process.* **2024**, *18*, 4303–4312. [CrossRef]
27. Ritthaler, M.; Regensky, A.; Kaup, A. Improved Motion Plane Adaptive 360-Degree Video Compression Using Affine Motion Models. In Proceedings of the ICASSP 2025—2025 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Hyderabad, India, 6–11 April 2025; pp. 1–5.

28. Zheng, Z.; Ma, W.; Liu, J.; Lu, J.; Qiu, S.; Liu, R.; Cui, W. Research on Adding Global Registration Model in Video Coding With Local Affine Motion Model. *IET Comput. Digit. Tech.* **2025**, *2025*, 6692669. [CrossRef]
29. Li, L.; Li, Z.; Ma, X.; Yang, H.; Li, H. Advanced spherical motion model and local padding for 360 video compression. *IEEE Trans. Image Process.* **2018**, *28*, 2342–2356. [CrossRef] [PubMed]
30. Yu, N.; Lin, C.; Zhao, Y.; Liu, M.; Zhang, X. video compression based on sphere-rotated frame prediction. *IET Image Process.* **2020**, *14*, 2711–2718. [CrossRef]
31. Kwan, H.M.; Gao, G.; Zhang, F.; Gower, A.; Bull, D. NVRC: Neural video representation compression. *Adv. Neural Inf. Process. Syst.* **2024**, *37*, 132440–132462. [CrossRef]
32. Wang, D.; Du, S.; Sun, Y.; Xia, S.; Dufaux, F.; Guo, H.; Wang, G.; Zhu, C. Fast CU Partition Algorithm for 360-Degree Videos on VVC. In Proceedings of the 2025 IEEE International Conference on Multimedia and Expo (ICME), Nantes, France, 30 June–4 July 2025; pp. 1–6.
33. Xiao, Y.; Yuan, Q.; Jiang, K.; Jin, X.; He, J.; Zhang, L.; Lin, C.W. Local-global temporal difference learning for satellite video super-resolution. *IEEE Trans. Circuits Syst. Video Technol.* **2023**, *34*, 2789–2802. [CrossRef]
34. Isobe, T.; Jia, X.; Tao, X.; Li, C.; Li, R.; Shi, Y.; Mu, J.; Lu, H.; Tai, Y.W. Look back and forth: Video super-resolution with explicit temporal difference modeling. In Proceedings of the 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), New Orleans, LA, USA, 18–24 June 2022; pp. 17390–17399.
35. Wiseman, Y. Video Compression Prototype for Autonomous Vehicles. *Smart Cities* **2024**, *7*, 758–771. [CrossRef]
36. Chen, J.; Liao, R.L.; Ye, Y.; Li, X. Decoder-side affine model refinement for video coding beyond vvc. In Proceedings of the 2023 Data Compression Conference (DCC), Snowbird, UT, USA, 21–24 March 2023; pp. 248–257.
37. Sainio, J.; Mercat, A.; Vanne, J. Design space exploration of practical VVC encoding for emerging media applications. *IEEE Trans. Consum. Electron.* **2022**, *68*, 387–400. [CrossRef]
38. Mahmoud, M.; Rizou, S.; Panayides, A.S.; Kantartzis, N.V.; Karagiannidis, G.K.; Lazaridis, P.I.; Zaharis, Z.D. A survey on optimizing mobile delivery of 360 videos: Edge caching and multicasting. *IEEE Access* **2023**, *11*, 68925–68942. [CrossRef]
39. Jeong, J.B.; Lee, S.; Kim, I.; Ryu, E.S. Implementing viewport tile extractor for viewport-adaptive 360-degree video tiled streaming. In Proceedings of the 2021 International Conference on Information Networking (ICOIN), Jeju Island, Republic of Korea, 13–16 January 2021; pp. 8–12.
40. He, Y.; Ye, Y.; Hanhart, P.; Xiu, X. Motion compensated prediction with geometry padding for 360 video coding. In Proceedings of the 2017 IEEE Visual Communications and Image Processing (VCIP), St. Petersburg, FL, USA, 10–13 December 2017; pp. 1–4.
41. Maritz, M.F. Rotations in three dimensions. *SIAM Rev.* **2021**, *63*, 395–404. [CrossRef]
42. Wiseman, Y. Adapting the H.264 Standard to the Internet of Vehicles. *Technologies* **2023**, *11*, 103. [CrossRef]
43. Song, Y.; Zeng, B.; Wang, M.; Deng, Z. An efficient low-complexity block partition scheme for VVC intra coding. *J. Real-Time Image Process.* **2022**, *19*, 161–172. [CrossRef]