





# 360DVO: Deep Visual Odometry for Monocular 360-Degree Camera

Xiaopeng Guo , Yinzhe Xu , Huajian Huang , *Member, IEEE*, and Sai-Kit Yeung , *Senior Member, IEEE*

**Abstract**—Monocular omnidirectional visual odometry (OVO) systems leverage 360-degree cameras to overcome field-of-view limitations of perspective VO systems. However, existing methods, reliant on handcrafted features or photometric objectives, often lack robustness in challenging scenarios, such as aggressive motion and varying illumination. To address this, we present 360DVO, the first deep learning-based OVO framework. Our approach introduces a distortion-aware spherical feature extractor (DAS-Feat) that adaptively learns distortion-resistant features from 360-degree images. These sparse feature patches are then used to establish constraints for effective pose estimation within a novel omnidirectional differentiable bundle adjustment (ODBA) module. To facilitate evaluation in realistic settings, we also contribute a new real-world OVO benchmark. Extensive experiments on this benchmark and public synthetic datasets (TartanAir V2 and 360VO) demonstrate that 360DVO surpasses state-of-the-art baselines (including 360VO and OpenVSLAM), improving robustness by 50% and accuracy by 37.5%. Home-page: <https://360dvo.hkustvtd.com>

**Index Terms**—visual odometry, omnidirectional vision

## I. INTRODUCTION

VISUAL odometry (VO) and simultaneous localization and mapping (VSLAM) estimate agent’s ego-motion from image sequences which enable various applications, including autonomous navigation and augmented reality. 360-degree cameras capture omnidirectional field-of-view (FoV) information and produce full-sphere images in the widely used equirectangular projection, thereby providing substantially richer scene coverage than perspective sensors. Usage of 360-degree cameras in VO system has emerged as a practical and effective solution to enhance system performance. Early works, such as OpenVSLAM [1] and 360VO [2], extend typical VO systems [3], [4] to exploit the merit of the 360-degree camera. However, without revisiting feature representations and overall pipeline, existing omnidirectional visual odometry (OVO) systems remain suboptimal in real-world challenges characterized by significant lighting variations, low frame rates, motion blur, and so on.

This work was supported by the HKUST Marine Robotics and Blue Economy Technology Grant (internal ref number FS148), the Marine Conservation Enhancement Fund (MCEF20107 and MCEF22112), and the Hong Kong Center for Construction Robotics (Hong Kong ITC-sponsored InnoHK Center). (Corresponding authors: Huajian Huang.)

Xiaopeng Guo and Yinzhe Xu are with the Division of Integrative Systems and Design, Hong Kong University of Science and Technology (e-mail: xguoay@connect.ust.hk; yxuck@connect.ust.hk).

Huajian Huang is with the Department of Computer Science and Engineering, Hong Kong University of Science and Technology (e-mail: hhuangbg@connect.ust.hk).

Sai-Kit Yeung is with the Division of Integrative Systems and Design, the Department of Computer Science and Engineering, and the Department of Ocean Science, Hong Kong University of Science and Technology (e-mail: saikit@ust.hk).

Digital Object Identifier (DOI): 10.1109/LRA.2026.3655280

©2026 IEEE

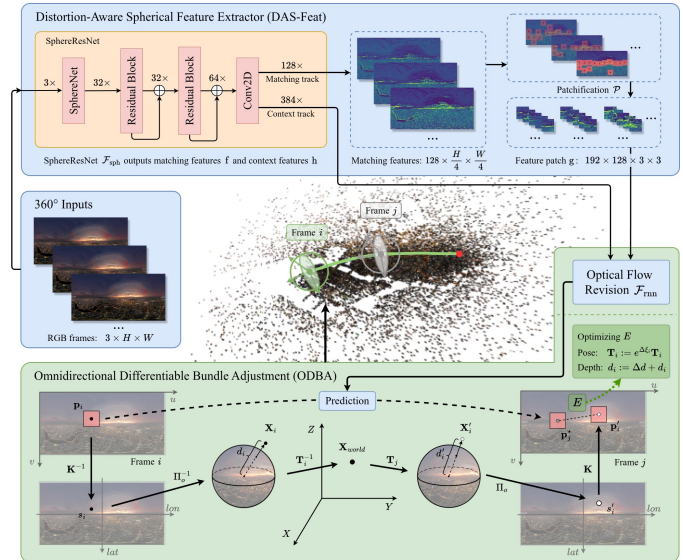


Fig. 1: Framework overview of 360DVO. Our method takes sequential 360-degree RGB frames as input and extracts matching features and context features using our proposed DAS-Feat module (Sec. III-A) on each of them. In DAS-Feat, the key component SphereResNet extracts distortion-resistant features, allowing patches to be cropped without deformation. After patchifying (Sec. III-A) the matching features around their gradient maxima, we compute the correlation of patch features and context features and estimate optical flow through a recurrent network. In the ODBA module, the pose  $T_i$  and depth  $d_i$  of frame  $i$  are jointly optimized by minimizing the distance between predicted patch  $p_j^*$  (from optical flow) and reprojected patch  $p_i'$  on an adjacent frame  $j$  (Sec. III-B).

In this work, we aim to enhance OVO performance in challenging scenarios by leveraging deep learning features. However, one of the prominent properties of 360-degree images is the strong nonlinear distortion caused by projection. Most deep feature extractors [5], [6], [7], [8], [9], [10] trained on large-scale perspective images implicitly assume linear sampling, which produces unreliable features from omnidirectional images, degrading the stability and accuracy of pose estimation. Additionally, computational power would be wasted in the non-linear region, reducing the running speed. Therefore, feature extraction with distortion perception capabilities and efficient omnidirectional pose optimization constraints are the key elements for building robust and precise deep learning-based OVO systems.

To narrow the gap, we propose 360DVO, a novel deep omnidirectional visual odometry system using a monocular 360-degree camera. We develop a distortion-aware spherical feature extractor (DAS-Feat) with a novel spherical residual network

SphereResNet to extract robust features from omnidirectional images. The SphereResNet integrates SphereNet [11] with residual blocks [12], which allows learning and accommodating projection-induced distortions effectively. Instead of using dense feature maps, we formulate the pose-estimation problem over sparse feature patches, similar to DPVO [8]. Furthermore, we derive an omnidirectional differentiable bundle adjustment (ODBA) component suitable to jointly optimize omnidirectional camera poses and depth of 3D points. These components exploit the unique properties of spherical imagery and improve accuracy and efficiency of our system. The overview of the 360DVO framework is illustrated in Fig. 1.

Recognizing the critical need for real-world evaluation, we present a new benchmark dataset comprising 360 videos filmed in 20 different real-world scenes. Finally, we conduct experiments on both real-world and synthetic benchmark datasets to comprehensively verify the efficacy of our approach. Our 360DVO achieves state-of-the-art performance than other baselines even in challenging scenarios.

In summary, the contributions of this work include:

- We propose 360DVO, the first deep omnidirectional visual odometry framework that learns ego-motion from monocular 360-degree videos.
- We develop a distortion-aware spherical feature extractor (DAS-Feat) built on a novel network SphereResNet to obtain reliable omnidirectional image features.
- We derive deep spherical feature constraints and establish omnidirectional differentiable bundle adjustment (ODBA), enabling efficient optimization of camera poses.
- We present a new challenging benchmark dataset dedicated to comprehensively evaluate OVO methodologies performance in real-world scenarios.

## II. RELATED WORK

**Omnidirectional Visual Odometry.** Omnidirectional cameras can obtain rich image information and sufficient environmental texture details [13], which makes it possible to collect the necessary data and bridge the research gaps. Prior works [14], [15], [16] expand the field of view by employing fisheye cameras and adapting indirect and direct VO/SLAM pipelines [17], [4], [18] with appropriate projection models [19], [20]. Although these methods broaden the field of view, they remain extensions tailored to fisheye cameras, not fully accommodating omnidirectional scenarios. To achieve a 360-degree field-of-view, ROVO [21] builds a wide-baseline multi-fisheye camera system. In contrast to multi-camera setups, OpenVSLAM [1] supports omnidirectional input captured by a portable 360-degree camera, offering great flexibility. 360VO [2] proposes a direct OVO and is able to recover semi-dense point clouds while having a higher requirement of the input image quality. Some works, such as 360-VIO [22] and LF-VISLAM [22], integrate IMU into the OVO framework for robustness improvement.

**Spherical Feature Extraction.** Traditional Convolutional Neural Networks (CNNs) [23] typically process 2D images using a rectangular grid structure. For 360-degree imagery, the most common representation is the equirectangular projection, which suffers from significant distortions. Spheri-

cal CNNs [24] encode rotation equivariance into the CNNs for classification. Flat2Sphere [24] adjusts the kernel size based on spherical coordinates to approximate distortion. EquiConvs [25] and SphereNet [11] both perform sampling on the sphere and project the convolution kernels onto the equirectangular image. EquiConvs [25] samples on the sphere and defines the kernel by its angular size and resolution, while SphereNet [11] samples and defines the square kernel on the tangent plane. SphereNet [11] facilitates fast spherical feature extraction and, due to the structure similarity to traditional CNNs, can be readily integrated into other network architectures. We adapt SphereNet [11] with residual blocks and propose SphereResNet as the feature extraction network of 360DVO, which can effectively learn and extract robust, distortion-resistant features.

**Bundle Adjustment.** In traditional VO and SLAM systems, bundle adjustment (BA) plays a crucial role in refining camera poses and 3D structure by minimizing reprojection error [16] or photometric error [4] across image sequences. Classical BA frameworks like g2o [26] and Ceres Solver [27] optimize nonlinear cost functions using second-order, gradient-based methods, typically Gauss–Newton or Levenberg–Marquardt. Some methods have combined traditional BA with deep learning techniques. DeepSfM [28] uses two cost volumes to iteratively optimize both camera pose and depth. DROID-SLAM [7] embeds a differentiable dense bundle adjustment layer and integrates it into the network architecture, enabling end-to-end training, while DPVO [8] implements a sparse version of it. With the recent advances in differentiable rendering, SC-OmniGS [29] has explored a new framework that bundle-adjusting omnidirectional camera poses along with radiance field represented by 3D Gaussians [30]. However, SC-OmniGS is primarily designed for scene reconstruction rather than online VO, and it faces challenges in large-scale settings when initial poses are not provided by an SfM or VO frontend. In contrast, we focus specifically on the VO problem and propose an ODBA module that efficiently estimates camera trajectories without reliance on external initialization.

## III. METHODOLOGY: 360DVO

360DVO is composed of two core components, including distortion-aware spherical feature extractor (DAS-Feat) and omnidirectional differentiable bundle adjustment (ODBA), as shown in Fig. 1. In DAS-Feat, we employ SphereResNet to extract distortion-resistant features from each input omnidirectional image  $I_i$  of size  $H \times W$ . The convolution kernel size is dynamically adjusted to accommodate nonlinear image projection distortion. Benefiting from the distortion-aware features, we can sample unwarped square patches from the extracted feature maps. Following DPVO [8], a sparse patch graph is maintained to encode the patch-to-frame relationships. After predicting and revising sparse patch motions, ODBA incorporates spherical reprojection constraints and jointly estimates accurate camera poses and 3D points.

**Spherical Camera Model Basic.** To handle omnidirectional imagery, we adopt a unified spherical camera model [2], which maps each image pixel to a point on the unit sphere. As illustrated in Fig. 1, the equirectangular image coordinate system

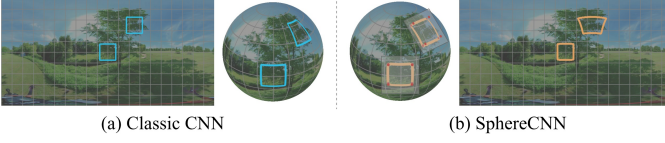


Fig. 2: Comparison of feature extraction between the classic CNN and the SphereCNN [11]. The distortion-aware convolution kernels differ in their pixel sampling manners along the height in an omnidirectional image, which is guided by the tangent projection across latitudes on the image’s corresponding sphere.

is parameterized by longitude angles  $\theta \in [-\pi, \pi]$  and latitude angles  $\phi \in [-\pi/2, \pi/2]$ . Given  $\mathbf{X} = (x, y, z, 1)^T$  representing the 3D point, the projection  $\Pi$  and inverse projection  $\Pi^{-1}$  of the spherical camera model are formulated as:

$$\Pi(\mathbf{X}) = \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \mathbf{K} \begin{bmatrix} \theta \\ \phi \\ 1 \end{bmatrix} = \mathbf{K} \begin{bmatrix} \arctan(x/z) \\ -\arcsin(d \cdot y) \\ 1 \end{bmatrix}, \quad (1)$$

$$\Pi^{-1}(\mathbf{p}, \mathbf{d}) = \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \frac{1}{\mathbf{d}} \begin{bmatrix} \cos(\phi) \sin(\theta) \\ -\sin(\phi) \\ \cos(\phi) \cos(\theta) \\ \mathbf{d} \end{bmatrix}, \quad (2)$$

$$\mathbf{K} = \begin{bmatrix} W/2\pi & 0 & W/2 \\ 0 & -H/\pi & H/2 \\ 0 & 0 & 1 \end{bmatrix}, \quad \begin{bmatrix} \theta \\ \phi \\ 1 \end{bmatrix} = \mathbf{K}^{-1} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}. \quad (3)$$

where  $d = 1/\sqrt{x^2 + y^2 + z^2}$  is the inverse distance from  $\mathbf{X}$  to the sphere center, and  $\mathbf{d}$  denotes the depth variable associated with the 2D patch. The intrinsic matrix  $\mathbf{K}$  of the spherical model depends solely on the omnidirectional image resolution  $(H, W)$  and converts between spherical coordinates  $(\theta, \phi)$  and pixel coordinates  $(u, v)$ .

#### A. Distortion-Aware Spherical Feature Extractor

**SphereResNet.** To address distortion in omnidirectional images, we introduce a network structure coupling a spherical convolution module [11] with residual blocks [12], referred to SphereResNet. Each input  $I_i$  is fed into the spherical convolution module that inversely projects the omnidirectional image onto a unit sphere. A  $7 \times 7$  spherical convolution kernel samples pixels on the sphere’s tangent plane and then projects those samples back onto the omnidirectional image. Fig. 2 presents the difference in the pixel sampling between the classic CNN and the spherical convolution module (SphereCNN) [11]. Following the spherical convolution module, two pairs of residual blocks with dimensions 32 and 64 are employed to mitigate vanishing gradients and enhance overall performance. The feature map channels are increased to 128 (for matching track) and 384 (for context track) via  $1 \times 1$  convolutions. As shown in Fig. 1, the SphereResNet  $\mathcal{F}_{\text{sph}}$  ultimately produces matching features  $f_i$  and context features  $h_i$  from the matching and context tracks, respectively:

$$f_i, h_i \leftarrow \mathcal{F}_{\text{sph}}(I_i). \quad (4)$$

Distortion-aware features are obtained by deforming standard convolution kernels, which correct spherical distortions across latitudes and enable more accurate trajectory prediction.

**Patchification.** Leveraging distortion-resistant features from SphereResNet, square patches are extracted directly from the matching features  $f_i$  without warping via the patchification module  $\mathcal{P}$ . Assuming constant depth within each patch, patch  $k$  in frame  $i$  is represented by the homogeneous coordinate  $\mathbf{p}_{ki} = (u, v, 1)^T$  of its center pixel with depth  $\mathbf{d}_k$ , where  $\mathbf{d}_k$  is initialized to the mean depth across patches in last keyframe. For every patch,  $\mathcal{P}$  outputs patchified matching features that are indicated by  $g$ , along with coordinates  $\mathbf{p}$  and depth  $\mathbf{d}$ :

$$\{(\mathbf{p}_k, \mathbf{d}_k, g_k)\}_{k=1}^N = \mathcal{P}(f_i), \quad (5)$$

where  $N$  is the number of patches extracted per frame. Specifically, per-channel spatial gradients are computed on the matching feature maps and aggregated across channels to obtain a single gradient-magnitude map. Patch centers are selected as the pixels with maximal values on this map, and a fixed  $3 \times 3$  patch is extracted around each center.

We maintain a sparse patch graph  $\mathcal{E}$  whose edges connect patches and frames. For a patch  $k$  extracted from frame  $i$ , add edges  $\{(k, j) \in \mathcal{E} \mid |j - i| < r\}$ , where  $j$  indexes adjacent frames and  $r$  is a temporal radius.

#### B. Omnidirectional Differentiable Bundle Adjustment

**Spherical Reprojection Constraints.** Based on the spherical camera model defined at the beginning of Sec. III, we formulate the reprojection relationships used in ODBA. Let  $\mathbf{T}_n \in \mathbf{SE}(3)$  denote the camera pose at frame  $n$ . For a patch  $\mathbf{p}_k$  observed in frame  $i$  with estimated depth  $\mathbf{d}_k$ , its reprojection into another frame  $j$  is computed as:

$$\mathbf{p}'_{kj} = \Pi(\mathbf{T}_{ij} \cdot \Pi^{-1}(\mathbf{p}_{ki}, \mathbf{d}_k)), \quad (6)$$

where  $\mathbf{T}_{ij} = \mathbf{T}_j \circ \mathbf{T}_i^{-1}$  represents the camera pose transformation from frame  $i$  to  $j$ . This operation constrains the reprojected spherical coordinates across frames, allowing ODBA to jointly refine the camera poses and geometry.

**Optical Flow Revision.** For each edge  $(k, j) \in \mathcal{E}$ , the correlation volume  $\langle g_k, h_j \rangle$  is computed between the patchified matching features  $g_k$  from  $\mathcal{P}$  and context features  $h_j$  from  $\mathcal{F}_{\text{sph}}$ . Specifically,  $\mathbf{p}_k$  is reprojected from frame  $i$  to  $j$  by Eq. 6. A square feature map centered at the reprojection is cropped from  $h_j$ , and inner products with  $g_k$  are computed. Then the update operator  $\mathcal{F}_{\text{mn}}$  [8], a recurrent network, takes  $\langle g_k, h_j \rangle$  as input and estimates the 2D motion (optical flow) of patch  $k$  from frame  $i$  to frame  $j$ :

$$\mathbf{p}^*_{kj} = \mathbf{p}_{ki} + \mathcal{F}_{\text{mn}}(\langle g_k, h_j \rangle), \quad (7)$$

where  $\mathbf{p}^*_{kj}$  is the predicted center point of patch  $k$  in frame  $j$ .

**Nonlinear Optimization.** ODBA jointly optimizes camera poses and 3D point depths by minimizing the coordinate error between the predicted and reprojected patches. The resulting nonlinear problem is solved using the Gauss–Newton algorithm by minimizing the following cost function  $E$ :

$$\arg \min_{\mathbf{T}_i, \mathbf{T}_j, \mathbf{d}} E = \sum_{(k, j) \in \mathcal{E}} \|\mathbf{p}^*_{kj} - \mathbf{p}'_{kj}\|_{\sum_{k, j}}^2. \quad (8)$$

Using Lie algebra and the chain rule, the Jacobians of the reprojection patch  $\mathbf{p}'$  with respect to  $\mathbf{T}_i$ ,  $\mathbf{T}_j$ , and  $\mathbf{d}$  are

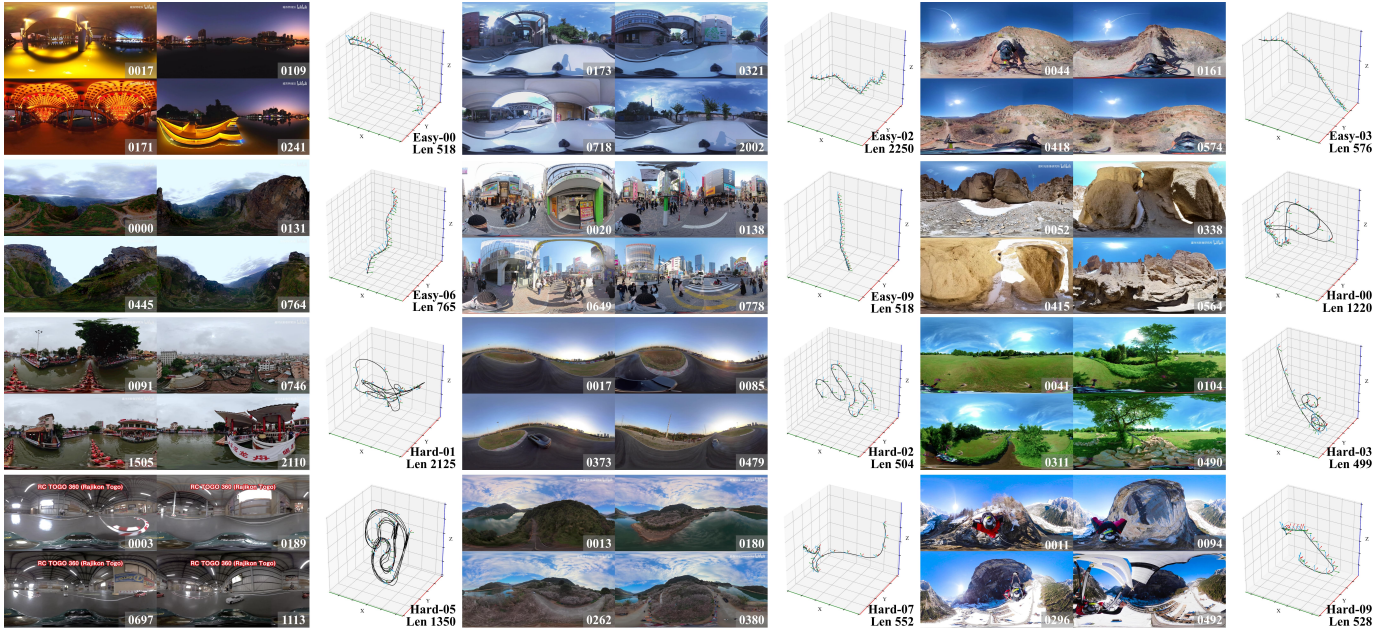


Fig. 3: Sample sequences of 360DVO dataset, demonstrating representative frames, 3D trajectory, and length for each sequence.

computed, namely  $J_i, J_j \in \mathbb{R}^{2 \times 6}$  and  $J_d \in \mathbb{R}^{2 \times 1}$ . Detailed derivations are provided in the supplementary material. The corresponding Hessian matrix is then constructed from these Jacobians, and Eq. 8 is solved by:

$$\begin{bmatrix} \mathbf{w}J_i^T J_i & \mathbf{w}J_i^T J_j & \mathbf{w}J_i^T J_d \\ \mathbf{w}J_j^T J_i & \mathbf{w}J_j^T J_j & \mathbf{w}J_j^T J_d \\ \mathbf{w}J_d^T J_i & \mathbf{w}J_d^T J_j & \mathbf{w}J_d^T J_d \end{bmatrix} \begin{bmatrix} \Delta \xi_i \\ \Delta \xi_j \\ \Delta d \end{bmatrix} = \mathbf{e}^* \begin{bmatrix} \mathbf{w}J_i^T \\ \mathbf{w}J_j^T \\ \mathbf{w}J_d^T \end{bmatrix}, \quad (9)$$

where  $\xi_i, \xi_j \in \mathfrak{se}(3)$  represent the Lie algebra coordinates of  $\mathbf{T}_i$  and  $\mathbf{T}_j$ ,  $\mathbf{e}^* = |\mathbf{p}^* - \mathbf{p}'|$  represents the reprojection error and  $\mathbf{w}$  is the matrix of weights, obtained from  $\mathcal{F}_{\text{rnn}}$ .

The Hessian matrix can be blocked and solved for the updates to camera poses  $\Delta \xi$  and depth  $\Delta d$  by Schur complement:

$$\begin{bmatrix} \mathbf{A}_{[12 \times 12]} & \mathbf{B}_{[12 \times 1]} \\ \mathbf{B}_{[1 \times 12]}^T & \mathbf{C}_{[1 \times 1]} \end{bmatrix} \begin{bmatrix} \Delta \xi_{[12 \times 1]} \\ \Delta d_{[1 \times 1]} \end{bmatrix} = \begin{bmatrix} \mathbf{r}_{ij} \\ \mathbf{r}_d \end{bmatrix}. \quad (10)$$

The obtained updates are finally applied to optimize the camera poses by  $\mathbf{T} := \exp^{\Delta \xi} \mathbf{T}$  and depth by  $\mathbf{d} := \Delta d + \mathbf{d}$ .

#### IV. 360DVO DATASET

Existing datasets for evaluating OVO methods remain limited in scope and realism. 360VO [2] offers 10 synthetic urban sequences of omnidirectional images with smooth trajectories. Recently, TartanAirV2 [31] further expands a large-scale simulation SLAM dataset TartanAir [32] by introducing additional environments and modalities, such as fisheye and panoramic views. In addition, it provides multi-modal sensor data and precise ground truth labels, including stereo RGB images, depth maps, segmentation, optical flow, and camera poses. Though pioneering in OVO evaluation, both datasets lack real-world complexity, limiting their utility for benchmarking robustness in practical applications.

To address this gap, we present a large-scale real-world OVO benchmark dataset, partially illustrated in Fig. 3. Our dataset prioritizes real-world challenges, including various

environments (e.g., wild, indoor, urban, aerial view), intense camera motions (e.g., motion blur, frequent rotations, complex trajectories), and dynamic lighting conditions (e.g., underexposure or overexposure). The collected sequences are sourced from three streams: (1) 3 sequences from 360VOTS [33] meeting strict VO suitability criteria (stable ego-motion, appropriate motion blur), (2) 15 internet-sourced videos covering extreme scenarios (e.g., snow, nighttime, crowded markets), and (3) 2 long sequences recorded on a Hong Kong tramway. All raw videos are processed at 10 FPS and standardized to  $3840 \times 1920$  resolution, while the sequences of sufficient length (i.e.,  $\geq 500$  frames) and continuous ego-motion are chosen with higher priorities for better VO tracking applicability. The collected in-the-wild videos lack true motion trajectories; therefore, we reconstruct pseudo ground truths via SfM software, such as COLMAP [34], [35] and Agisoft Metashape [36]. We assess the accuracy of the pseudo ground truth by comparing COLMAP and Metashape on TartanAirV2 [31]. The results show that Agisoft Metashape consistently produces trajectories with negligible ATE-RMSE (i.e., 0.027), which closely matches the provided ground truth, whereas COLMAP yields larger errors and drift (i.e., 2.535). Therefore, we adopt Agisoft Metashape as the pseudo ground-truth generation medium for our dataset.

Our final 360DVO dataset comprises 20 sequences with an average length of approximately 1k frames, which are partitioned into *Easy* and *Hard* subsets (10 each) based on trajectory complexity and environmental dynamics. *Easy* sequences feature linear motions in static scenes (e.g., straight streets, open roads), while *Hard* sequences incorporate aggressive rotations, rapid illumination changes (e.g., entering tunnels), and dynamic occlusions (e.g., crowds, vegetation). Representative frames are shown in Fig. 3. The dataset's multi-domain coverage and explicit challenge stratification are designed to rigorously evaluate VO systems' robustness against real-world

TABLE I: Comparison of ATE and RPE (translation/rotation) on TartanAirV2 evaluation set. Best result per sequence is marked in red.

Methods	CountryHouse		House		OldTownNight		VictorianStreet		Avg.	Succ.
	Easy	Hard	Easy	Hard	Easy	Hard	Easy	Hard		
ORB-SLAM3 [16]	1.759/0.269/5.336	1.421/0.311/10.96	3.901/0.251/6.744	- / - / -	7.828/0.470/5.613	- / - / -	1.317/0.351/2.995	7.745/0.692/12.62	- / - / -	65%
Droid-SLAM [7]	0.006/0.096/5.000	0.014/0.107/5.978	<b>0.007</b> /0.047/2.846	1.392/0.302/4.186	2.792/0.102/2.754	11.18/0.365/12.34	0.019/0.049/2.134	0.202/0.345/5.985	1.951/0.177/5.152	100%
DPVO [8]	0.027/0.097/5.003	0.359/0.102/5.693	0.604/0.062/2.858	2.749/0.208/4.378	0.196/0.107/2.940	8.138/0.347/12.17	0.049/0.051/2.147	1.219/0.241/5.683	1.668/0.152/5.110	100%
DPV-SLAM [37]	0.008/0.097/5.001	0.149/0.193/12.30	0.409/0.053/2.581	0.056/0.191/12.80	0.493/0.106/2.915	<b>0.104</b> /0.191/4.905	0.063/0.050/2.143	0.065/0.178/8.706	0.168/0.132/6.420	100%
OpenVSLAM [1]	0.283/0.007/0.376	- / - / -	0.020/0.007/0.090	- / - / -	<b>0.033</b> /0.019/0.355	- / - / -	<b>0.016</b> /0.010/0.060	- / - / -	- / - / -	53%
360VO [2]	2.277/0.184/3.869	1.977/0.238/9.066	- / - / -	- / - / -	- / - / -	- / - / -	- / - / -	- / - / -	- / - / -	27%
360DVO (ours)	<b>0.004/0.001/0.010</b>	<b>0.005/0.001/0.020</b>	0.025/ <b>0.002/0.016</b>	<b>0.038/0.005/0.037</b>	0.039/ <b>0.003/0.009</b>	0.145/ <b>0.031/0.038</b>	0.025/ <b>0.002/0.010</b>	<b>0.020/0.002/0.013</b>	<b>0.038/0.006/0.019</b>	100%

perturbations, complementing existing synthetic benchmarks.

## V. EXPERIMENTS

### A. Implementation Details

360DVO is implemented in PyTorch with CUDA. With 62 scenes from TartanAirV2 [31] for training, the remaining 4 scenes are held out for evaluation. From each training scene, one easy and two hard sequences are selected, resulting in a training set of over 220k equirectangular images with depth maps. The system is trained end-to-end, supervised by relative pose loss and optical flow loss computed from ground-truth depth. Following the training setup of DPVO [8], training runs for 100k iterations on a single RTX-4090D GPU and takes approximately two days.

### B. Evaluation

We compare 360DVO with state-of-the-art pinhole methods and OVO baselines, OpenVSLAM [1] and 360VO [2], on 360VO dataset [2], TartanAirV2 [31] evaluation set, and our new proposed 360DVO dataset. Following Sim(3) Umeyama alignment [38] of the predicted trajectories to the ground truth, the root mean squared error (RMSE) is reported for three metrics: absolute trajectory error (ATE, in meters), translational relative pose error (RPE(t), in m/frame), and rotational relative pose error (RPE(r), in deg/frame). Results are averaged over five runs. For timestamp mismatches, evaluation is restricted to overlapping trajectory segments. In each run, if tracking is lost in more than half of the frames in a sequence, the run is counted as a failure for that sequence. Cases that fail in more than half of the five runs are indicated as “-”.

**360VO Dataset.** The 360VO dataset [2] is a relatively easy synthetic dataset with smooth camera motion and static illumination. As shown in Tab. II, 360DVO achieves better performance on all metrics than both OpenVSLAM and 360VO, reducing the ATE by 10% compared with 360VO.

**TartanAirV2 Evaluation Set.** As noted above, we construct an evaluation set from 4 scenes in TartanAirV2 [31], spanning day/night and indoor/outdoor settings. We select one easy and one hard sequence per scene, with hard sequences exhibiting stronger rotations. We also evaluate pinhole baselines on the corresponding perspective images provided by TartanAirV2. As Tab. I shows, our 360DVO with default settings attains the highest success rate and accuracy, whereas OpenVSLAM and 360VO nearly fail on all hard sequences. Learning-based pinhole methods [7], [8], [37], trained on TartanAir [32], perform reasonably well but struggle on the Hard sequences, highlighting the benefit of omnidirectional FOVs for visual

TABLE II: Comparison of average ATE and RPE (translation/rotation) on 360VO synthetic dataset.

Methods	ATE	RPE(t)	RPE(r)
OpenVSLAM [1]	2.25	0.312	0.452
360VO [2]	1.24	0.291	0.455
360DVO (ours)	<b>1.11</b>	<b>0.235</b>	<b>0.440</b>

odometry. These results highlight 360DVO’s robustness to challenging scenarios involving intense camera rotations.

**360DVO Dataset.** To decouple the benefits of a wide field of view (FOV) from algorithmic performance, we evaluate existing pinhole methods alongside OVO methods on our 360DVO dataset. Virtual monocular sequences are derived by extracting 90-degree FOV perspective images of size  $640 \times 640$  from the omnidirectional images. SOTA pinhole methods are benchmarked on this branch, including ORB-SLAM3 [16], Droid-SLAM [7], DPVO [8], and DPV-SLAM [37]. For the pinhole methods, the camera intrinsics are uniformly set to the ideal values (e.g.,  $f_x = f_y = c_x = c_y = 320$ ). For OVO methods constrained by the spherical camera model, the intrinsics are computed from the input resolution as in Eq. 3.

Narrow FOVs make challenging scenes harder by increasing feature leave-view events and shortening long-term co-visibility. Overall, OVO methods demonstrate more stable and higher performance than pinhole methods shown in Tab. III, revealing the inherent value of 360-degree coverage. 360DVO reduces ATE over DPV-SLAM by 56.2% on *Easy* and outperforms DPVO with 37.1% improvement on *Hard*. Notably, the pinhole methods outperform the OVO methods on Hard-06 captured on a crowded bridge. The pinhole crops predominantly contain static bridge structures, whereas the omnidirectional images include many dynamic objects, which degrades feature matching performance.

Additionally, learning-based methods outperform classical ones on average. Among pinhole methods, learning-based DPV-SLAM is best on *Easy* and DPVO outperforms others on *Hard*, while Droid-SLAM is competitive but fails on Easy-07. On the omnidirectional side, 360DVO surpasses OpenVSLAM by 27.6% on *Easy* and 43.4% on *Hard*. OpenVSLAM is competitive on *Easy* (i.e., 4.57) but underperforms on *Hard* (i.e., 7.69). Direct method 360VO fails on many sequences, with an average success rate of 43%, far below 360DVO’s 100%. Qualitatively, our 360DVO system demonstrates the capabilities for rapid 6-DoF motions, large rotations, and dynamic illumination from aerial views (e.g., Hard-02, Hard-07, Hard-09) compared with OpenVSLAM, as shown in Fig. 4. In the indoor cases (e.g., Hard-05), 360DVO remains

TABLE III: Quantitative comparison of trajectory accuracy (ATE/RPE(t)/RPE(r)) and tracking success rate (%) on the 360DVO dataset (*Easy* and *Hard*). “-” indicates failure. Best result per sequence is in **red**, second-best result in **blue**. 360DVO (fast) denotes the modification of using lower resolution images as input while maintaining sparser sampling patches. All results are shown with 2-decimal precision for clarity.

Methods	Easy Sequences									Avg.	Succ.	
	00	01	02	03	04	05	06	07	08			09
ORB-SLAM3 [16]	24.2/0.86/4.43	0.33/0.44/0.11	- / - / -	- / - / -	3.75/0.23/0.11	- / - / -	0.47/0.29/0.53	- / - / -	- / - / -	- / - / -	- / - / -	46%
Droid-SLAM [7]	2.28/0.40/0.35	0.61/0.13/0.04	44.2/1.33/0.19	22.7/1.08/0.69	0.57/0.02/0.03	5.27/0.15/0.07	0.37/0.22/0.41	- / - / -	23.7/0.72/1.58	2.70/0.04/0.09	- / - / -	90%
DPVO [8]	1.37/0.32/0.11	0.13/0.44/0.11	37.5/1.36/0.13	6.91/1.33/0.46	0.57/0.23/0.03	4.89/0.24/0.07	0.35/0.21/0.52	16.4/1.68/0.15	7.93/0.88/0.44	2.23/0.41/0.10	7.83/0.71/0.21	100%
DPV-SLAM [37]	1.19/0.32/0.11	0.23/0.44/0.10	33.7/1.38/0.13	6.74/1.34/0.46	0.50/0.23/0.03	4.97/0.25/0.07	0.30/0.21/0.52	18.3/1.68/0.13	7.31/1.02/0.60	2.36/0.40/0.10	7.56/0.73/0.22	100%
OpenVSLAM [1]	1.73/0.33/0.11	0.38/0.44/0.08	18.5/1.36/0.11	2.77/1.44/0.45	4.36/0.23/0.04	0.71/0.25/0.03	0.49/0.21/0.49	11.9/1.68/0.13	3.11/0.89/0.27	1.66/0.41/0.11	4.57/0.72/0.18	94%
360VO [2]	15.1/0.37/0.38	14.3/1.00/1.02	- / - / -	- / - / -	- / - / -	- / - / -	15.0/0.28/0.98	91.8/3.28/2.12	84.1/1.84/2.41	24.1/0.53/1.17	- / - / -	42%
360DVO (default)	1.06/0.32/0.09	0.22/0.43/0.07	22.5/1.36/0.07	3.26/1.32/0.44	0.45/0.22/0.05	0.50/0.24/0.02	0.28/0.20/0.51	1.08/1.67/0.10	1.82/0.86/0.06	2.00/0.40/0.09	3.31/0.70/0.15	100%
360DVO (fast)	1.90/0.32/0.09	0.66/0.44/0.09	23.4/1.37/0.07	3.53/1.33/0.45	5.02/0.23/0.12	0.62/0.24/0.05	0.45/0.21/0.51	19.6/1.67/0.19	4.25/0.86/0.07	2.52/0.40/0.09	6.20/0.71/0.17	100%

Methods	Hard Sequences									Avg.	Succ.	
	00	01	02	03	04	05	06	07	08			09
ORB-SLAM3 [16]	0.40/0.71/0.13	12.9/0.56/6.05	3.49/0.94/0.73	7.84/0.32/9.83	6.84/0.79/3.69	3.11/1.18/6.50	- / - / -	- / - / -	- / - / -	37.0/6.60/1.53	- / - / -	68%
Droid-SLAM [7]	0.20/0.02/0.04	2.93/0.60/2.81	2.27/0.20/0.65	2.42/0.33/0.43	10.2/0.77/0.60	2.72/0.88/1.41	12.9/0.61/0.64	3.50/0.94/0.66	5.48/1.25/0.13	29.4/1.27/1.54	7.20/0.69/0.89	100%
DPVO [8]	0.72/0.71/0.03	10.4/0.37/1.96	1.34/0.91/0.61	0.71/0.38/0.10	0.61/0.59/0.06	4.03/1.03/1.33	19.3/0.53/0.26	1.66/1.19/0.01	7.97/1.69/0.12	22.5/1.06/1.63	6.92/0.85/0.71	100%
DPV-SLAM [37]	0.64/0.71/0.03	9.14/0.32/2.51	0.95/0.99/0.61	0.70/0.39/0.14	0.58/0.60/0.06	3.54/1.04/1.32	17.7/0.54/0.22	2.09/1.20/1.02	6.80/1.68/0.12	28.4/0.98/1.74	7.05/0.84/0.78	100%
OpenVSLAM [1]	0.20/0.71/0.04	1.68/0.27/0.24	4.41/0.87/0.65	0.11/0.39/0.07	0.28/0.60/0.06	10.3/0.98/5.35	21.4/0.64/0.18	18.6/1.23/1.00	5.17/1.74/0.16	14.8/1.14/1.54	7.69/0.86/0.93	92%
360VO [2]	- / - / -	- / - / -	19.1/0.93/1.13	- / - / -	3.43/0.38/0.50	10.3/0.57/0.17	- / - / -	34.3/1.73/1.65	105/3.73/1.30	- / - / -	- / - / -	44%
360DVO (default)	0.45/0.70/0.02	4.97/0.23/1.01	0.61/0.90/0.60	1.18/0.31/0.31	5.10/0.41/0.32	2.82/0.86/4.39	23.1/0.49/0.15	1.50/1.18/1.06	3.33/1.69/0.11	0.36/1.03/1.55	4.35/0.78/0.95	100%
360DVO (fast)	0.82/0.70/0.02	0.81/0.22/0.87	0.61/0.90/0.60	2.67/0.31/0.15	0.19/0.42/0.04	2.71/1.05/1.30	22.7/0.49/0.16	2.46/1.17/1.08	3.08/1.69/0.11	0.69/1.03/1.55	3.68/0.80/0.59	100%

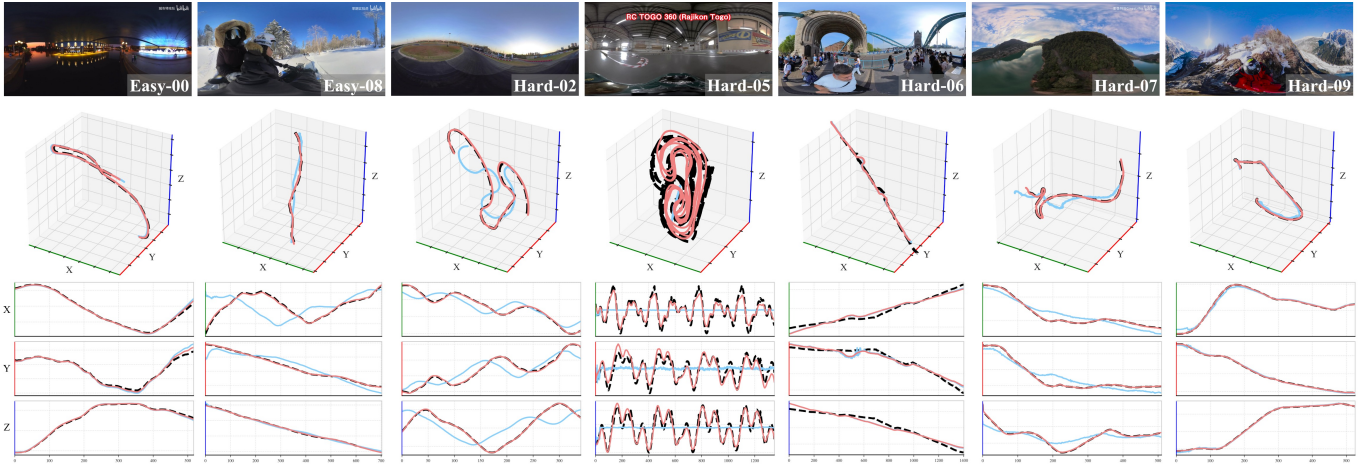


Fig. 4: Trajectories Comparison on the 360DVO dataset in 3D space, with position variations along the X, Y, and Z axes plotted over all frames. The ground truth is shown in **black dashed lines**, 360DVO results in **red solid lines**, and OpenVSLAM results in **blue solid lines**.

competitive under the challenges of substantial motion blur. These advancements highlight the advantages of the DAS-Feat module in extracting deep features from 360-degree context, and the stability provided by ODBA in optimization.

To further evaluate the efficiency of our proposed 360DVO, we establish two baselines with default and fast settings. 360DVO (default) executes under the configuration of inputs in  $3840 \times 1920$  size, 192 patches per image (i.e.,  $N$  in Eq. 5), and a gradient-based patch selection strategy. 360DVO (fast) is deployed by downsampling inputs to  $1920 \times 960$  and halving  $N$  to 96, meeting real-time constraints. Detailed in Tab. III, 360DVO (default) achieves the best overall performance on the 360DVO dataset, ranking first on *Easy* (i.e., 3.31) and second on *Hard* (i.e., 4.35), while 360DVO (fast) performs best on *Hard* (i.e., 3.68). We also observe a characteristic failure mode for 360DVO (default) with the high-resolution setting. On sequence Hard-04, 360DVO (default) underperforms both classical and learning-based methods (e.g., OpenVSLAM [1], DPVO [8]), while 360DVO (fast) achieves the best result (i.e., 0.190). This sequence is dominated by repetitive textures (e.g., sky, grass, foliage). High-resolution gradient-based selection

TABLE IV: Ablation study of our 360DVO system to verify the effectiveness of proposal components.

ID	Variants			ATE		
	Input	Feature	BA	Easy	Hard	Avg.
DPVO [8]	Pinhole	ResNet[12]	DBA	7.83	6.92	7.37
#1	360°	ResNet[12]	ODBA	7.33	12.7	9.99
#2	360°	SphereNet [11]	ODBA	-	-	-
360DVO	360°	SphereResNet	ODBA	3.31	4.35	3.83

admits many ambiguous patches, yielding unstable correspondences in new keyframes. The fast configuration implicitly regularizes the problem by reducing per-frame patches and feature scales, thereby avoiding low-entropy, look-alike regions.

### C. Ablation Studies

We perform ablation studies of components and hyperparameters on our dataset, reported in Tab. IV and Tab. V.

**From Pinhole to Omnidirection.** Replacing the BA module of DPVO [8] with the novel omnidirectional differentiable bundle adjustment (ODBA) module enables 360-degree camera pose optimization. However, although the modification

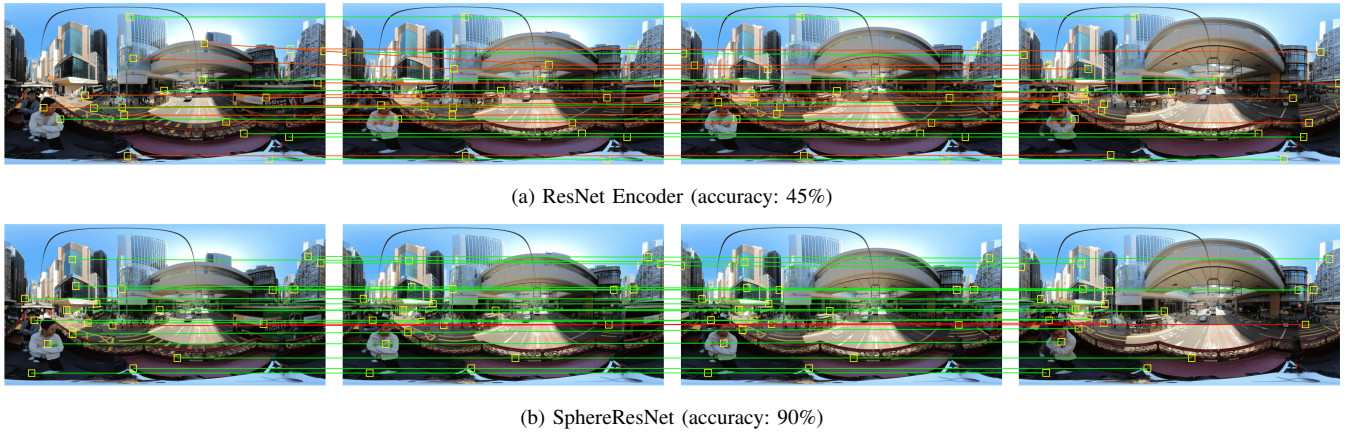


Fig. 5: Two samples of predicted patch trajectories (ResNet Encoder vs. SphereResNet). Correct in green, incorrect in red. Patches are highlighted by yellow squares. The proposed SphereResNet yields higher tracking accuracy.

TABLE V: The influence of 360DVO hyperparameters in terms of trajectory accuracy. #D denotes default setting, #F denotes fast version. #R, #P, and #S denote resolution, patch count ( $N$ ), and patch selection, respectively.

ID	Settings			ATE			FPS
	Resolution	$N$	Selection	Easy	Hard	Avg.	
#D	3840×1920	192	Gradient	3.31	4.35	3.83	8
#R0	1920×960	192	Gradient	5.58	3.75	4.66	17
#R1	960×480	192	Gradient	6.95	3.86	5.41	22
#P0	3840×1920	384	Gradient	3.86	6.82	5.34	5
#P1	3840×1920	96	Gradient	3.90	6.51	5.20	10
#P2	3840×1920	48	Gradient	3.90	5.95	4.93	14
#S	3840×1920	192	Random	4.40	7.29	5.85	8
#F	1920×960	96	Gradient	6.20	3.68	4.94	27

allows 360-degree image processing, the method relying on classic CNN features has degraded performance as reported in Tab. IV (#1). It highlights the necessity of a tailored network for the 360-degree image feature extraction.

**Distortion-Aware Spherical Features.** However, when replacing the classic CNN with a spherical feature extractor, SphereNet [11], the model suffers gradient explosions despite clipping gradient and tuning learning-rate during training. It fails to estimate any camera poses consequently, shown in Tab. IV (#2). By contrast, the proposed SphereResNet is able to stably learn distortion-resistant features and yield the best accuracy, proving its efficacy. The efficacy of SphereResNet has also been verified in Fig. 5.

**Hyperparameters.** Tab. V reveals clear trade-offs between accuracy and runtime efficiency by various hyperparameters. Downscaling the default resolution to  $1920 \times 960$  (#R0) and  $960 \times 480$  (#R1) raises FPS from 8 to 17 and 22, while worsening average ATE by 21.6% and 41.0%. Low-resolution settings reduce apparent pixel displacement and blur, stabilizing aggressive motion but sacrificing details in general scenarios. The default setting (#D in Tab. V) with patch count  $N = 192$  reaches an optimal result. Increased to 384 per-frame patches (#P0), the BA module is overconstrained to ambiguous correspondences. The computational cost increases while

TABLE VI: Runtime (fps) and Performance (ATE, m) comparisons on Jetson Orin.

Methods	Easy	Hard	Avg.	Fps
OpenVSLAM [1]	6.01	6.75	6.38	3-7
360DVO (default)	3.46	4.49	3.98	2
360DVO (fast)	6.19	3.95	5.07	5

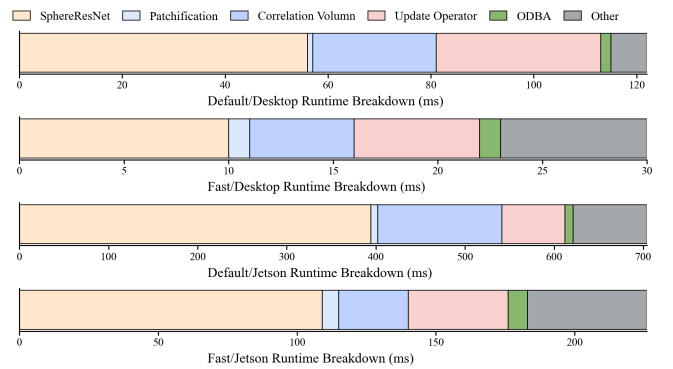


Fig. 6: Per-module runtime decomposition across settings and deployment environments.

the estimated trajectory accuracy decreases. When cutting down  $N$  to 96 (#P1) and 48 (#P2), the extracted features are underrepresented across the entire image, affecting accuracy as well. In addition, the gradient-based strategy outperforms the random-based #S, concentrating patches on high-entropy regions. The fast variant of #F in Tab. V corresponding to 360DVO (fast) in Tab. III, achieves real-time performance at 27 FPS with a competitive accuracy (i.e., 4.94).

#### D. Edge Deployment and Limitation

To assess the applicability of 360DVO in robotics, we compare accuracy (ATE-RMSE) and throughput (FPS) for 360DVO and OpenVSLAM [1] on a Jetson Orin Developer Kit. As shown in Tab. VI, 360DVO (default) is the most accurate but also the slowest. 360DVO (fast) recovers efficiency ( $\approx 5$  FPS) while remaining competitive in accuracy, outperforming OpenVSLAM (5.07 vs. 6.38). Although computationally slightly more expensive than conventional OVO methods,

360DVO achieves noticeably higher accuracy and more robust tracking. We believe this substantial performance gain offsets the increased computational cost. A quantitative breakdown of per-module runtime for 360DVO under different settings and environments is shown in Fig. 6. Modules operating on high-resolution feature maps constitute the principal bottleneck. While reducing input resolution or increasing hardware compute capacity can improve inference speed, developing a more lightweight yet effective feature extractor is necessary for enabling real-time operation on embedded platforms, which is a promising direction for future work.

## VI. CONCLUSION

We present 360DVO, the first deep learning-based omnidirectional visual odometry (OVO) system. 360DVO extracts sparse, informative patch features via a distortion-aware spherical feature extractor, while jointly optimizing camera poses and 3D points through an omnidirectional differentiable bundle adjustment module. To enable comprehensive OVO evaluation, we also introduce a new real-world benchmark dataset. Extensive experiments demonstrate that 360DVO achieves state-of-the-art performance, substantially outperforming prior methods across accuracy and robustness metrics.

**Acknowledgement.** The authors would like to express their sincere gratitude to the “Sustainable Smart Campus as a Living Lab” (SSC) program at HKUST, ePropulsion, and the Hong Kong Center for Construction Robotics (InnoHK center supported by the Hong Kong ITC, InnoHK-HKCR) for their support.

## REFERENCES

- [1] S. Sumikura, M. Shibuya, and K. Sakurada, “Openslam: A versatile visual slam framework,” in *Proceedings of the 27th ACM International Conference on Multimedia*, 2019, p. 2292–2295. 1, 2, 5, 6, 7
- [2] H. Huang and S.-K. Yeung, “360vo: Visual odometry using a single 360 camera,” in *ICRA*, 2022. 1, 2, 4, 5, 6
- [3] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, “Orb: An efficient alternative to sift or surf,” in *ICCV*, 2011. 1
- [4] J. Engel, V. Koltun, and D. Cremers, “Direct sparse odometry,” *IEEE transactions on pattern analysis and machine intelligence*, 2017. 1, 2
- [5] D. DeTone, T. Malisiewicz, and A. Rabinovich, “Superpoint: Self-supervised interest point detection and description,” *CoRR*, 2017. 1
- [6] M. Caron, H. Touvron, I. Misra, H. Jégou, J. Mairal, P. Bojanowski, and A. Joulin, “Emerging properties in self-supervised vision transformers,” in *ICCV*, 2021. 1
- [7] Z. Teed and J. Deng, “DROID-SLAM: Deep Visual SLAM for Monocular, Stereo, and RGB-D Cameras,” *Advances in neural information processing systems*, 2021. 1, 2, 5, 6
- [8] Z. Teed, L. Lipson, and J. Deng, “Deep patch visual odometry,” *Advances in Neural Information Processing Systems*, 2023. 1, 2, 3, 5, 6
- [9] M. Oquab, T. Darcet, T. Moutakanni, H. V. Vo, M. Szafraniec, V. Khalidov, P. Fernandez, D. Haziza, F. Massa, A. El-Nouby, R. Howes, P.-Y. Huang, H. Xu, V. Sharma, S.-W. Li, W. Galuba, M. Rabbat, M. Assran, N. Ballas, G. Synnaeve, I. Misra, H. Jégou, J. Mairal, P. Labatut, A. Joulin, and P. Bojanowski, “Dinov2: Learning robust visual features without supervision,” *arXiv:2304.07193*, 2023. 1
- [10] X. Shen, Z. Cai, W. Yin, M. Müller, Z. Li, K. Wang, X. Chen, and C. Wang, “Gim: Learning generalizable image matcher from internet videos,” in *ICLR*, 2024. 1
- [11] B. Coors, A. P. Condurache, and A. Geiger, “Spherenet: Learning spherical representations for detection and classification in omnidirectional images,” in *ECCV*, 2018. 2, 3, 6, 7
- [12] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *CVPR*, 2016, pp. 770–778. 2, 3, 6
- [13] Z. Zhang, H. Rebecq, C. Forster, and D. Scaramuzza, “Benefit of large field-of-view cameras for visual odometry,” in *ICRA*, 2016. 2
- [14] D. Caruso, J. Engel, and D. Cremers, “Large-scale direct slam for omnidirectional cameras,” in *IROS*, 2015, pp. 141–148. 2
- [15] H. Matsuki, L. von Stumberg, V. Usenko, J. Stückler, and D. Cremers, “Omnidirectional dso: Direct sparse odometry with fisheye cameras,” *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 3693–3700, 2018. 2
- [16] C. Campos, R. Elvira, J. J. Gomez, J. M. M. Montiel, and J. D. Tardós, “ORB-SLAM3: An accurate open-source library for visual, visual-inertial and multi-map SLAM,” *IEEE Transactions on Robotics*, vol. 37, no. 6, pp. 1874–1890, 2021. 2, 5, 6
- [17] D. Caruso, J. Engel, and D. Cremers, “Large-scale direct slam for omnidirectional cameras,” in *IROS*, 2015, pp. 141–148. 2
- [18] R. Mur-Artal and J. D. Tardós, “ORB-SLAM2: an open-source SLAM system for monocular, stereo and RGB-D cameras,” *IEEE Transactions on Robotics*, vol. 33, no. 5, pp. 1255–1262, 2017. 2
- [19] C. Geyer and K. Daniilidis, “A unifying theory for central panoramic systems and practical implications,” in *ECCV*, 2000. 2
- [20] J. Kannala and S. Brandt, “A generic camera model and calibration method for conventional, wide-angle, and fish-eye lenses,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 8, pp. 1335–1340, 2006. 2
- [21] H. Seok and J. Lim, “Rovo: Robust omnidirectional visual odometry for wide-baseline wide-fov camera systems,” in *ICRA*, 2019. 2
- [22] Q. Wu, X. Xu, X. Chen, L. Pei, C. Long, J. Deng, G. Liu, S. Yang, S. Wen, and W. Yu, “360-vio: A robust visual-inertial odometry using a 360° camera,” *IEEE Transactions on Industrial Electronics*, vol. 71, no. 9, pp. 11 136–11 145, 2024. 2
- [23] Z. Li, F. Liu, W. Yang, S. Peng, and J. Zhou, “A survey of convolutional neural networks: Analysis, applications, and prospects,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, no. 12, pp. 6999–7019, 2022. 2
- [24] T. S. Cohen, M. Geiger, J. Köhler, and M. Welling, “Spherical CNNs,” in *ICLR*, 2018. 2
- [25] C. Fernandez-Labrador, J. M. Facil, A. Perez-Yus, C. Demonceaux, J. Civera, and J. J. Guerrero, “Corners for layout: End-to-end layout recovery from 360 images,” *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 1255–1262, 2020. 2
- [26] R. Kummerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, “g2o: A general framework for graph optimization,” in *ICRA*, 2011. 2
- [27] S. Agarwal, K. Mierle, and T. C. S. Team, “Ceres Solver,” 10 2023. [Online]. Available: <https://github.com/ceres-solver/ceres-solver> 2
- [28] X. Wei, Y. Zhang, Z. Li, Y. Fu, and X. Xue, “DeepSfm: Structure from motion via deep bundle adjustment,” in *ECCV*, 2020. 2
- [29] H. Huang, Y. Chen, L. Li, H. Cheng, T. Braud, Y. Zhao, and S.-K. Yeung, “SC-omniGS: Self-calibrating omnidirectional gaussian splatting,” in *ICLR*, 2025. 2
- [30] B. Kerbl, G. Kopanas, T. Leimkuehler, and G. Drettakis, “3d gaussian splatting for real-time radiance field rendering,” *ACM Trans. Graph.*, vol. 42, no. 4, July 2023. 2
- [31] M. Patel, F. Yang, Y. Qiu, C. Cadena, S. Scherer, M. Hutter, and W. Wang, “Tartanground: A large-scale dataset for ground robot perception and navigation,” in *IROS*, 2025, pp. 20 524–20 531. 4, 5
- [32] W. Wang, D. Zhu, X. Wang, Y. Hu, Y. Qiu, C. Wang, Y. Hu, A. Kapoor, and S. Scherer, “Tartanair: A dataset to push the limits of visual slam,” in *IROS*, 2020. 4, 5
- [33] Y. Xu, H. Huang, Y. Chen, and S.-K. Yeung, “360vots: Visual object tracking and segmentation in omnidirectional videos,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 47, no. 11, pp. 9785–9797, 2025. 4
- [34] J. L. Schönberger and J.-M. Frahm, “Structure-from-motion revisited,” in *CVPR*, 2016. 4
- [35] J. L. Schönberger, E. Zheng, M. Pollefeys, and J.-M. Frahm, “Pixelwise view selection for unstructured multi-view stereo,” in *ECCV*, 2016. 4
- [36] “AgiSoft Metashape Professional,” 9 2024. [Online]. Available: <http://www.agisoft.com/downloads/installer> 4
- [37] L. Lipson, Z. Teed, and J. Deng, “Deep patch visual slam,” in *ECCV*. Springer, 2024, pp. 424–440. 5, 6
- [38] S. Umeyama, “Least-squares estimation of transformation parameters between two point patterns,” *IEEE Transactions on pattern analysis and machine intelligence*, vol. 13, no. 4, pp. 376–380, 2002. 5