HELVIPAD: A Real-World Dataset for Omnidirectional Stereo Depth Estimation

Supplementary Material

A. Dataset Specifications

This section complements the overview of the dataset by detailing the video sequences and providing additional histograms for depth and disparity.

A.1. Sequences

The HELVIPAD dataset includes 26 video sequences captured between December 2023 and February 2024., recorded at a frame rate of 10 Hz.

Each sequence is synchronized with its corresponding LiDAR point clouds, which are projected on frames to obtain depth maps, and subsequently disparity maps. Top images, bottom images, depth maps, and disparity maps are then cropped to remove unnecessary borders and downsized from a width of 1024 pixels to 512 to enable more efficient training.

The sequences span a diverse array of settings and conditions, as detailed in Tab. 4. The dataset includes recordings taken at various times of the day, from early morning to night, under a variety of weather conditions, including cloudy and sunny skies. These recordings were made in multiple indoor and outdoor locations, from pedestrian squares and footpaths to corridors and parking areas, offering a wide spectrum of environmental contexts. The table also provides information on the duration of each sequence, with an average of 2 minutes and 41 seconds. Furthermore, the dynamic nature of the recorded scenes is emphasized by the presence of pedestrians, with an average of 13.33 pedestrians in indoor sequences and 17.65 pedestrians in outdoor sequences.

A.2. Additional Histograms of Depth and Disparity Labels

In addition to the histograms provided in the main paper, we include more detailed histograms of depth labels (Fig. 9) and disparity labels (Fig. 13) across the train and test splits. Additionally, we provide histograms for the augmented train set, *i.e.,* after depth completion, including depth (Fig. 10) and disparity Fig. 14.

The analysis shows that train and test distributions maintain consistent patterns across each environment — indoor, outdoor, and night outdoor settings —, suggesting a well-aligned partition. Notably, a distinct concentration of short-range distances within the 0–10 meter range in all settings is attributed to the LiDAR laser capturing the ground surface in each environment. Furthermore, the data exhibits an exponential decay in pixel percentage with increasing depth, correlating with the decrease in LiDAR precision over longer distances. A comparison between indoor and outdoor scenes reveals that indoor sequences, both in training and testing, exhibit a more rapid decline in pixel percentage beyond 10 meters. A similar pattern is observed for disparity values, where train and test distributions align closely. As anticipated, we identify higher disparity values for indoor sequences compared to outdoor scenes due to the greater angular difference between the two cameras for points projected closer to the recording system and lower disparity for further away objects.

When comparing the depth distributions in the training set before and after depth completion, we observe that depth completion increases the density of low-range depth values, particularly within the 0–10 meter range. This shift is likely a result of the algorithm interpolating missing values and removing out-of-distribution samples. This process tends to enrich the representation of closer objects in the dataset. Similar conclusions can be drawn when comparing disparity distributions before and after depth completion.

B. Data Collection

This section provides details about the acquisition device, the synchronization between sensors and dataset quality.

B.1. Data Capture Setup

The hardware setup consists of a custom-designed support system that aligns all devices horizontally and stacks them vertically. The system integrates two Ricoh Theta V cameras, which capture images in 4K/UHD equirectangular format at 30 fps with an initial resolution of 3840 × 1920 pixels. It also includes an Ouster OS1-64 LiDAR sensor, operating at 10 fp with 64 beams and a vertical field of view of 42.4°. The LiDAR is mounted at the bottom, with the first camera at the top ("*top camera*") and the second camera in the middle ("*bottom camera*"). The vertical distances between the devices are precisely configured, with 19.1 cm separating the two camera lenses and 45 cm between the LiDAR and the bottom camera, as shown in Fig. 8.

The cameras function as external modules, while the LiDAR operates via Robot Operating System (ROS) on an embedded NVIDIA Jetson Xavier. This central processor manages data capture and ensures synchronization across all devices. The entire setup is mounted on a custom-built, remotely controlled robot chassis, offering mobility and a fully integrated, portable acquisition solution.

Sequence	Split	Date	Setting	Dur.	Time of day	Area type	Weather	# Fr.	# Peds
20231206_REC_01_OUT	train	2023-12-06	outdoor	01:46	afternoon	ped. sq.	cloudy	1001	37
20231206_REC_02_OUT	test	2023-12-06	outdoor	03:45	afternoon	ped. sq.	cloudy	1911	45
20240120_REC_02_OUT	train	2024-01-20	outdoor	02:37	afternoon	road	sunny	1445	7
20240120_REC_03_OUT	train	2024-01-20	outdoor	02:34	afternoon	road	sunny	1379	9
20240120_REC_04_OUT	train	2024-01-20	outdoor	02:54	afternoon	footpath	sunny	1569	11
20240120_REC_05_IN	train	2024-01-20	indoor	02:43	end of day	corridor	n.a.	1530	4
20240120_REC_06_IN	test	2024-01-20	indoor	02:11	end of day	corridor	n.a.	998	29
20240120_REC_07_IN	train	2024-01-20	indoor	01:54	end of day	corridor	n.a.	998	5
20240121_REC_01_OUT	train	2024-01-21	outdoor	02:47	afternoon	footpath	sunny	1650	6
20240121_REC_02_OUT	train	2024-01-21	outdoor	02:26	afternoon	footpath	sunny	1425	2
20240121_REC_03_OUT	train	2024-01-21	outdoor	02:21	afternoon	road	sunny	1375	4
20240121_REC_04_OUT	train	2024-01-21	outdoor	02:54	afternoon	road	sunny	1780	7
20240121_REC_05_OUT	train	2024-01-21	outdoor	02:03	end of day	road	sunny	1237	6
20240124_REC_01_OUT	train	2024-01-24	outdoor	02:48	morning	road	cloudy	1549	10
20240124_REC_02_OUT	val	2024-01-24	outdoor	03:21	morning	footpath	cloudy	1675	10
20240124_REC_03_OUT	test	2024-01-24	outdoor	02:54	morning	ped. sq.	cloudy	1681	9
20240124_REC_04_OUT	train	2024-01-24	outdoor	03:51	morning	road	cloudy	2180	6
20240124_REC_05_OUT	train	2024-01-24	outdoor	02:46	morning	road	cloudy	1500	4
20240124_REC_06_IN	train	2024-01-24	indoor	02:32	afternoon	corridor	n.a.	1429	44
20240124_REC_07_NOUT	train	2024-01-24	outdoor	02:51	night	footpath	night	1700	22
20240124_REC_08_NOUT	test	2024-01-24	outdoor	03:51	night	ped. sq.	night	2925	54
20240124_REC_09_NOUT	train	2024-01-24	outdoor	02:50	night	footpath	night	1800	58
20240124_REC_11_IN	train	2024-01-24	indoor	01:39	end of day	hall	n.a.	1000	11
20240124_REC_12_IN	val	2024-01-24	indoor	02:13	end of day	hall	n.a.	1320	13
20240127_REC_01_IN	test	2024-01-27	indoor	02:01	morning	corridor	n.a.	1201	2
20240127_REC_02_OUT	test	2024-01-27	indoor	02:20	morning	parking	n.a.	1430	2

Table 4. **Overview of the collected sequences.** Abbreviations: Dur. = Duration, displayed in minutes:seconds; ped. sq. = pedestrian square; # Fr. = number of frames; # Peds = number of pedestrians.

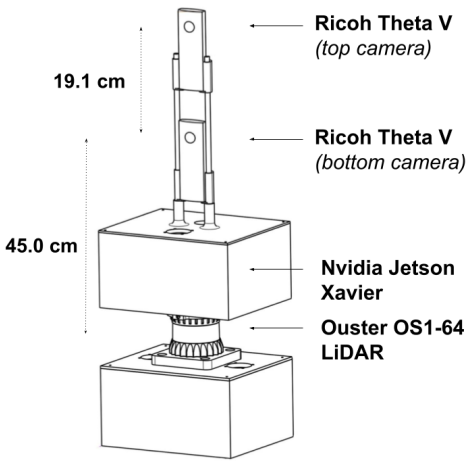


Figure 8. **HELVIPAD data acquisition setup**: dual Ricoh Theta V cameras in a top-bottom configuration above an Ouster OS1-64 LiDAR Sensor, and integrated with a NVIDIA Jetson Xavier.

B.2. Synchronization between sensors device

To ensure accurate synchronization between the sensors, we use a hardware-triggered synchronization method. At the start of each recording, an external flash lasting 33 ms is activated in front of the cameras, creating a visible synchronization marker in the video streams. Simultaneously, the precise ROS timestamp of the flash event is recorded in the LiDAR data, which provides a precise timestamp. During post-processing, we identify the blinded frames and corresponding ROS timestamps, re-aligning all data streams to start from this synchronized reference, as shown in Fig. 11

To match the LiDAR's frame rate (10 fps), we retain one out of every three camera frames. The 33ms flash duration ensures it is captured in at least one camera frame, with a maximum potential de-synchronization of half a frame interval (16.67 ms) if the flash occurs just after a frame is captured. This reasoning extends to the LiDAR-camera synchronization, resulting in a maximum de-synchronization of 16.67 ms across all sensors.

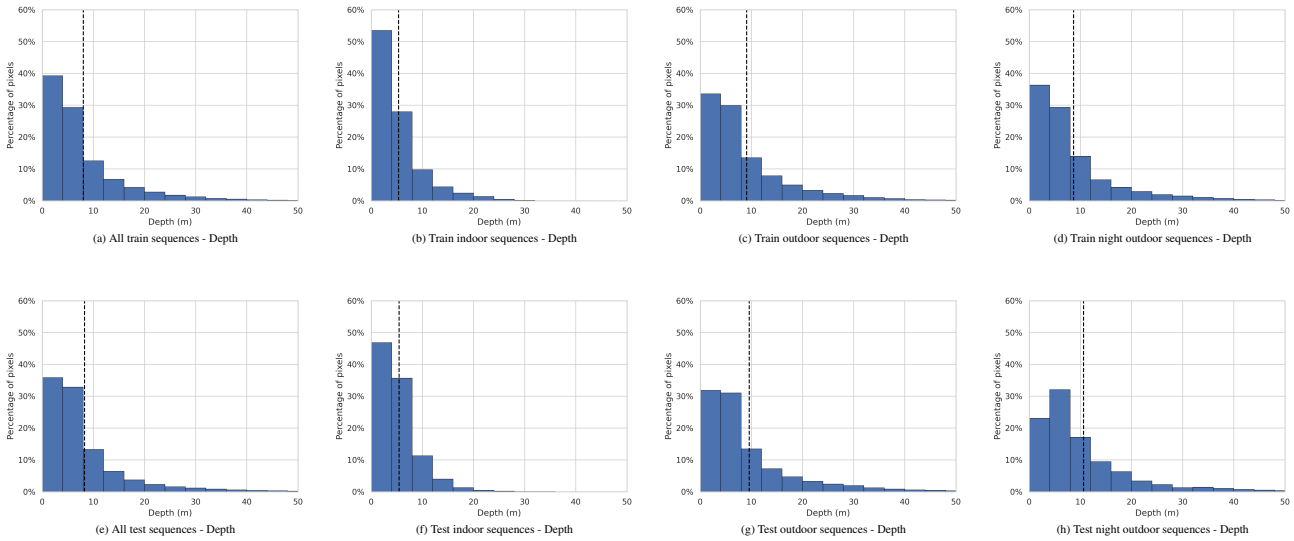


Figure 9. **Histograms of depth labels across train** (first row) **and test splits** (second row). Each plot's vertical dotted line denotes the average depth for the respective setting.

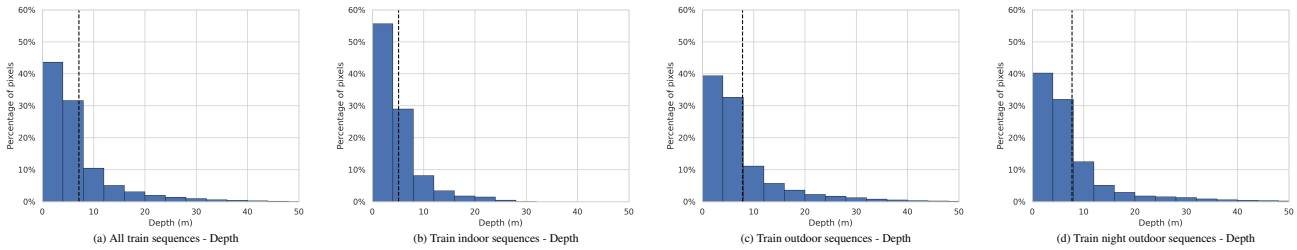


Figure 10. **Histograms of depth labels across across train splits after depth completion**. Each plot's vertical dotted line denotes the considered setting average.

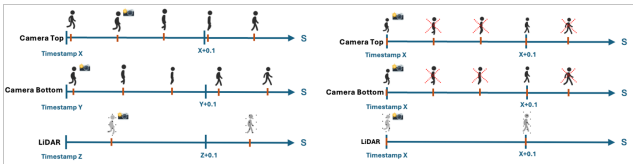


Figure 11. **Synchronization of LiDAR and cameras using a flash trigger.** The illustration shows data alignment before (**left**) and after (**right**) synchronization.

B.3. LiDAR-Image Projection Quality Assessment

In the absence of a reliable, standardized method to evaluate the accuracy of LiDAR point projections onto equirectangular image planes, manual validation serves as a practical and precise alternative. Visual inspection and manual selection of corresponding points have been highlighted in studies such as [10], where the authors emphasized the role of manual evaluation in aligning data when automated methods are insufficient. Similarly, in [11], the challenges of achieving accurate projections without ground truth were addressed, underscoring the importance of visual assessment for high-precision tasks.

Therefore, in this work, we adopt a manual point selection approach to evaluate the projection of LiDAR points onto 2D equirectangular image planes. This process involves manually selecting corresponding points, such as object edges, on both the LiDAR-projected data and the images. These selected points are then used to compute the pixel-wise error, which quantifies the projection's accuracy.

For each selected point pair, we calculate the Euclidean distance between the projection point $(x_{\text{proj}}, y_{\text{proj}})$ and the corresponding real point $(x_{\text{real}}, y_{\text{real}})$:

$$\text{Error} = \sqrt{(x_{\text{proj}} - x_{\text{real}})^2 + (y_{\text{proj}} - y_{\text{real}})^2}. \quad (5)$$

This error is averaged across multiple images to provide a comprehensive assessment of the projection's accuracy. Additionally, a relative error metric is computed by normalizing the pixel error by the image diagonal, enabling a con-

Figure 12. **Illustration of LiDAR point projection onto an equirectangular image**. The red dots represent the projected points from LiDAR data, while the green dots indicate the expected projection points on the image. The red arrows show the pixel-wise error between the projected points and the expected points, which is used to quantify the projection accuracy. This error metric aids in evaluating the fidelity of LiDAR-to-image projection in the HELVIPAD dataset.

sistent evaluation across different resolutions.

Tab. 5 provides a summary of the pixel-wise errors measured accross sequences in the dataset. Each sequence was evaluated by manually selecting corresponding points between the projected LiDAR points and the equirectangular images, followed by the calculation of Euclidean distances as described earlier, and average errors for each sequence are reported along with the overall dataset average.

Sequence Name	**Avg. Pixel Error** (px)	**Relative Error** (%)
20231206_REC_01_OUT	6.4	0.32
20240120_REC_02_OUT	9.5	0.48
20240120_REC_03_OUT	8.7	0.44
20240120_REC_04_OUT	7.1	0.36
20240120_REC_05_IN	10.2	0.51
20240120_REC_07_IN	6.8	0.34
20240121_REC_01_OUT	9.1	0.46
20240121_REC_02_OUT	7.5	0.38
20240121_REC_03_OUT	8.3	0.42
20240121_REC_04_OUT	7.9	0.40
20240121_REC_05_OUT	8.0	0.40
20240124_REC_01_OUT	10.4	0.52
20240124_REC_02_OUT	6.9	0.35
20240124_REC_04_OUT	8.1	0.41
20240124_REC_05_OUT	7.3	0.37
20240124_REC_06_IN	9.6	0.48
20240124_REC_07_NOUT	7.4	0.37
20240124_REC_09_NOUT	8.9	0.45
20240124_REC_11_IN	6.7	0.33
20240124_REC_12_IN	8.8	0.44
Overall	**8.0**	**0.40**

Table 5. **Pixel-wise projection errors of LiDAR points onto equirectangular images**, per sequence. Relative error is expressed as a percentage of the image diagonal for context clarity.

The overall average pixel error across the dataset is 8.0 pixels, corresponding to a relative error of 0.40% of the image diagonal. This level of precision validates the high-quality LiDAR-to-image projection tasks in the HELVIPAD dataset.

C. Depth Completion

This section provides an in-depth evaluation of our depth completion pipeline, detailing the evaluation methods, hyperparameter selection, and comparison of temporal aggregation techniques. Quantitative and qualitative results are also presented to demonstrate the effectiveness of our approach.

C.1. Evaluation Method

The evaluation of our depth completion method follows a structure akin to standard machine learning. The dataset is split into training and test sets, then performance metrics are computed on the test set within the 3D space.

Creation of training and test set. To identify the optimal hyperparameters for depth completion, the points of each measured point cloud are divided into a training and test set using a classical 80-20-split. Points are sampled from a uniform distribution over all input points without replacement.

This approach primarily evaluates metrics over points with low distances to its neighbors, which are easier to estimate. When the pipeline generates points on a uniform grid (e.g., an image), the metrics reflect lower bounds on actual errors due to this distribution shift. While this bias limits direct comparison to image-based errors, it is acceptable for hyperparameter optimization and data augmentation purposes.

Evaluation metrics. To evaluate the merits of different options for the depth completion pipeline, we calculate the following metrics: Mean Absolute Error (*MAE*), Root Mean Squared Error (*RMSE*), mean absolute relative error (*MARE*), Inlier Ratio (*IR*) and Actual Ratio of Interpolated Points (*ARIP*). The calculation of the first three mentioned metrics is the same as for depth estimation methods and can be found in Appendix D.1. The *IR* corresponds to the ratio of estimated depth labels that have an absolute error of less than $t_{\text{inlier}} = 1\%$. Given the number N of estimated depths among points of all point clouds and sequences, the *IR* can be calculated in the following way:

$$\text{IR} = \frac{1}{N} \sum_{i=1}^{N} \mathbb{I}_{|r_{\text{est, i}} - r_{\text{true, i}}| < t_{\text{inlier}}}. \tag{6}$$

In Sec. 3.3, the *RIP* is defined as the ratio of interpolated points after filtering. As the uncertainty estimates of all

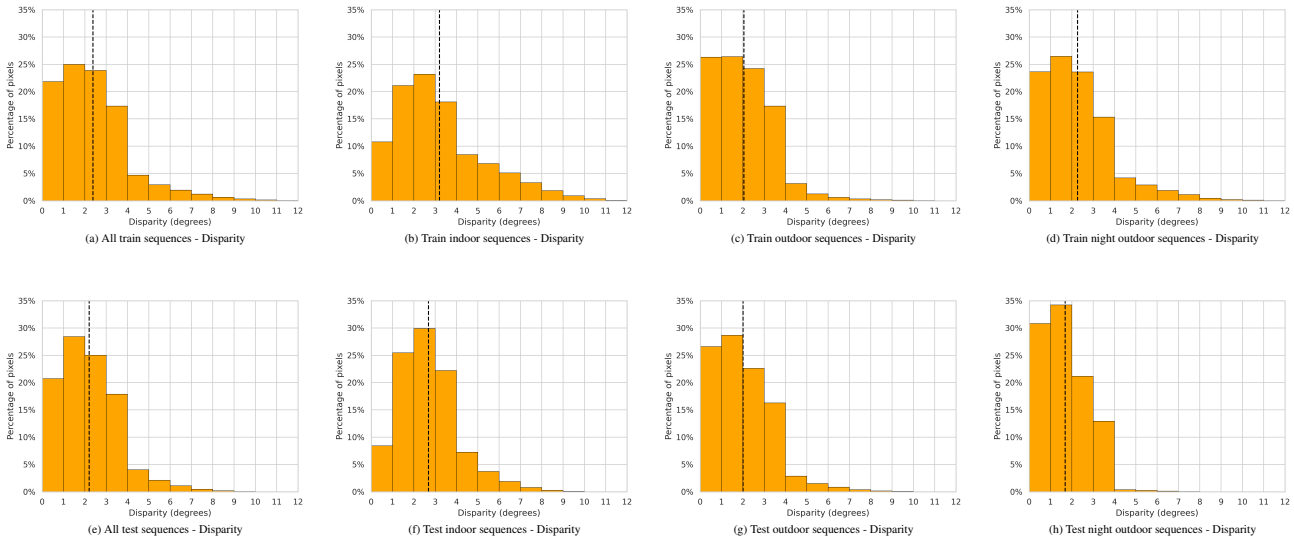


Figure 13. **Histograms of disparity labels across train** (first row) **and test splits** (second row). Each plot's vertical dotted line denotes the average disparity for the respective setting.

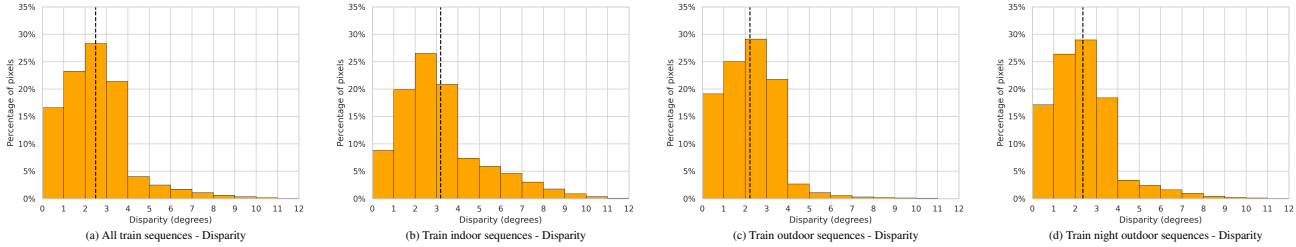


Figure 14. **Histograms of disparity labels across train splits after depth completion**. Each plot's vertical dotted line denotes the considered setting average.

query points within one sequence do not fit into memory, thresholds for the uncertainty-based filtering are calculated for each point cloud and then averaged per sequence. This leads to an *ARIP* which differs slightly from the desired *RIP*.

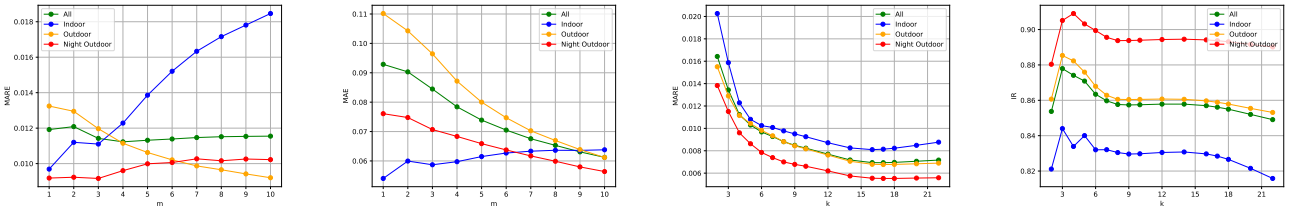
Ratio of labeled pixels. To calculate the ratio of labeled pixels the region within an image that contains potential labeled pixels must be identified. While the whole image width W can be labeled, the potential labeled region along the height is restricted. Specifically, the minimum (H_{min}) and maximum (H_{max}) height with potential labels depend on the distance r of the points at the minimum (θ_{min}) and maximum (θ_{max}) of the LiDAR's vertical field of view. For each image, we individually determine H_{min} and H_{max} based on the smallest and largest row index containing at least one label in the original depth map. Any labels in the completed depth map that fall outside this height range are filtered out. With this, the Ratio of Labeled Pixels (*RLP*) can be calculated as a function of the total number of labeled pixels n_{label} in the entire image:

$$\text{RLP}(n_{\text{label}}) = \frac{n_{\text{label}}}{W \times (H_{\text{max}} - H_{\text{min}})} \tag{7}$$

C.2. Choice of Hyperparameters

This section describes how the hyperparameters for the depth completion pipeline, introduced in Sec. 3.3, have been chosen.

Number of aggregated point clouds. For temporal aggregation, we fuse the m previous and m following point clouds. The minimum *MARE* for *all* sequences is observed for $m = 4$ (Fig. 15a). However, the *MAE* continues to decrease for all sequences except *indoor*, where it increases for higher m values (Fig. 15b). As the total number of points for all *indoor* sequences is lower than for all *outdoor* and *night outdoor* train sequences, the error for *all* sequences is less influenced by *indoor* sequences. We set



(a) Mean Absolute Relative Error (*MARE*). (b) Mean Absolute Error (*MAE*). (c) Mean Absolute Relative Error (*MARE*). (d) Inlier Ratio (*IR*).

Figure 15. Selected metrics with a variable number of aggregated point clouds (Fig. 15a and Fig. 15b) or $m = 4$ previous / next point clouds (Fig. 15c and Fig. 15d) on *all*, *indoor*, *outdoor* and *night outdoor* train sequences. The ratio of interpolated points is set to RIP $= 80\%$ and $k = 4$ is chosen as the numbers of neighbors (Fig. 15a and Fig. 15b) or a variable number of neighbors k (Fig. 15c and Fig. 15d). We report all depth metrics in m

$m = 4$ to provide high quality depth labels also for *indoor* sequences and low depth values.

Number of neighbors. The number of neighbors k in interpolation can be observed to reduce the *MARE* until $k = 17$ is reached (Fig. 15c). Then, the *MARE* then becomes greater again. However, the *IR* starts to decrease from $k = 3$ already (Fig. 15d). We select $k = 17$ because the positive influence on the *MARE* is greater than the negative influence on the *IR*.

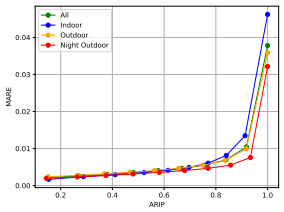


Figure 16. *MARE* for $m = 4$ on *all*, *indoor*, *outdoor* and *night outdoor* sequences. A variable *ARIP* is interpolated with $k = 17$ neighbors.

Ratio of interpolated points. As expected, the evaluation metrics become worse for a higher *ARIP*. By observing the *MARE* in dependence of the *ARIP* in Fig. 16, it can be seen that setting this ratio slightly below 1 reduces the *MARE* significantly. Setting RIP $= 0.8$ for ARIP ≈ 0.8 appears to represent a good balance between the provision of labels for a large number of points and the simultaneous minimization of the induced errors.

Out-of-distribution threshold. The out-of-distribution threshold t_{OOD} to filter query points with insufficient neighbors cannot be determined based on the train-test split. Instead, a heuristic has to be derived theoretically. Each LiDAR scan provides depth labels on a spherical, regular grid with a vertical, angular resolution of

$$\Delta\theta = \frac{\text{FOV}_\text{v}}{n_\text{beams}} = \frac{42.4^\circ}{64} \approx 0.66^\circ \tag{8}$$

and a horizontal, angular resolution of

$$\Delta\varphi = \frac{\text{FOV}_\text{h}}{n_\text{channels, h}} = \frac{360^\circ}{1024} \approx 0.35^\circ. \tag{9}$$

Given the number of accumulated point clouds $2m + 1$ and the number of neighbors for interpolation k and the assumption that all accumulated scans provide measurements in the neighborhood of a query position, the number of neighbors, chosen per LiDAR scan, $n_\text{neighbors, grid}$ can be approximated to:

$$n_\text{neighbors, grid} = \frac{k}{2m+1} = \frac{17}{9} \approx 2. \tag{10}$$

A position in the regular LiDAR depth grid has the maximum distances to its 2 nearest neighbors, if its position is exactly in the middle of two rows and two columns in the grid. As a result, the average distance d_q of a query position (θ_q, φ_q) to its neighbors is below the following threshold, if all accumulated point clouds provide labels in the region of the query:

$$t_\text{OOD} = \sqrt{\left(\frac{\Delta\theta}{2}\right)^2 + \left(\frac{\Delta\varphi}{2}\right)^2} \approx 0.37^\circ. \tag{11}$$

Number of spherical grid points. To map the points from the 3D space to the image we create a spherical grid with approximately uniformly distributed points. The number of spherical grid points is set to $n_\text{grid} = 20,000,000$. This value represents a compromise between high computational loads for high n_grid values and missing depth information for the subsequent projection for low n_grid. Missing depth information leads to less labeled pixels.

Method	ARIP ↑	MAE ↓	RMSE ↓	MARE ↓	IR ↑
No Aggregation	0.811	0.096	0.864	**0.011**	**0.757**
No Movement	**0.841**	**0.093**	**0.801**	0.012	0.749
KISS-ICP	0.828	0.103	0.828	0.017	0.618

Table 6. Evaluation metrics for different temporal aggregation methods with $m = 1$, RIP $= 0.8$ and $k = 4$ on *all* test sequences. We report all depth metrics in m. The best results are highlighted in **bold**.

Range of spherical grid points. As the vertical field of view of the LiDAR sensor is limited, we filter all grid points that are out of its view. The value for the threshold limiting the polar angle can be determined in the following way:

$$t_\theta = \frac{180° - \text{FOV}_\text{v}}{2} = \frac{180° - 42.4°}{2} = 68.8°. \quad (12)$$

All query grid points whose polar angle θ_q is not in the range $\theta_q \in [t_\theta, 180° - t_\theta]$ are not mapped to the image.

To summarize, we set the hyperparameters in the following way: $m = 4$, $k = 17$, RIP $= 0.8$, $t_\text{OOD} \approx 0.37°$, $n_\text{grid} = 20,000,000$ and $t_\theta = 68.8°$.

C.3. Temporal Aggregation Comparison

To aggregate multiple point clouds, previous and following scans can be aggregated directly (*no movement*) or transformed based on odometry information of the robot. The HELVIPAD dataset provides only omnidirectional stereo images and LiDAR point clouds but no odometry information. KISS-ICP [33] is one of the state-of-the-art approaches for LiDAR odometry. It is based on the ICP algorithm and creates a local map of the environment. As it is not possible to evaluate the quality of estimated odometry data with the dataset, using a robust method that does not need hyperparameter optimization, such as KISS-ICP, is a suitable choice to obtain odometry information for the dataset.

We compare no temporal aggregation (*no aggregation*), temporal aggregation without transforming point clouds (*no movement*) and temporal aggregation based on odometry information obtained with the KISS-ICP (*KISS-ICP*) in Tab. 6. The *KISS-ICP* approach yields the least favorable results in most of the metrics. This may be attributed to the presence of moving people in the scene, coupled with the employed interpolation method. *No movement* is clearly better than *no aggregation* in *RMSE* and slightly better in *MAE*. *No aggregation* is the best approach in terms of *MARE* and *IR*. However, it must be noted that the *ARIP* is also the lowest for this method. Consequently, the *MARE* and *IR* for the same ratio may be lower than the ones of the *no movement* method. Overall, the results of the *no movement* temporal aggregation method are the most favorable. Thus, we use it for the temporal aggregation.

C.4. Results

Quantitative results. The evaluation metrics, described in Appendix C.1, for the final hyperparameter configuration, specified in Appendix C.2, are summarized in Tab. 7. MAE, RMSE and MARE are significantly lower than the depth results of the depth estimation baselines of Tab. 2. Thus, the induced errors by the depth completion are acceptable as a data augmentation technique.

Sequences	ARIP ↑	MAE ↓	RMSE ↓	MARE ↓	IR ↑
All	0.839	0.054	0.398	0.007	0.856
Indoor	0.840	0.046	0.264	0.008	0.828
Outdoor	0.836	0.058	0.440	0.007	0.859
Night outdoor	0.857	0.048	0.371	0.006	0.894

Table 7. Evaluation metrics for the final hyperparameters of the depth completion on *all*, *indoor*, *outdoor*, and *night outdoor* train sequences. We report all depth metrics in m.

Tab. 8 exhibits that the *RLP*, as defined in Appendix C.1, is increased approximately by a factor of 5 following the application of depth completion across all sequence types. The maximum number of labelable pixels $\text{n}_\text{lab, max}$ corresponds to the denominator in Eq. (7).

Sequences	$\text{n}_\text{lab, max}$	$\text{n}_\text{lab, ori}$	RLP_ori	$\text{n}_\text{lab, aug}$	RLP_aug
All	14.4B	1.7B	11.9%	9.6B	60.7%
Indoor	3.1B	0.4B	12.8%	1.9B	62.1%
Outdoor	9.6B	1.1B	11.5%	5.7B	59.9%
Night outdoor	1.7B	0.2B	11.8%	1.1B	62.0%

Table 8. Maximum number of labelable pixels $\text{n}_\text{lab, max}$, labeled pixels n_lab and ratio of labeled points RLP for *all*, *indoor*, *outdoor*, and *night outdoor* train sequences.

Qualitative results. Fig. 17 depicts a completed depth map and an original depth map together with the corresponding image of a patch from an *outdoor* sequence. In areas of homogeneous depth within the LiDAR's field of view, the completed depth map provides dense depth labels. It is evident that the depth completion method does not provide labels for pixels at object boundaries. For instance, this can be observed at the street lamp located in the upper half of the image, around column 165, as well as at the boundaries of the tree in the upper half of the image, between columns 190 and 270. This is an understandable limitation, as it is challenging to ascertain the precise locations of such boundaries based on depth data alone. Even in the original depth map, it is evident that the boundaries of the tree are not clearly defined, and also some measurements at its boundaries ap-

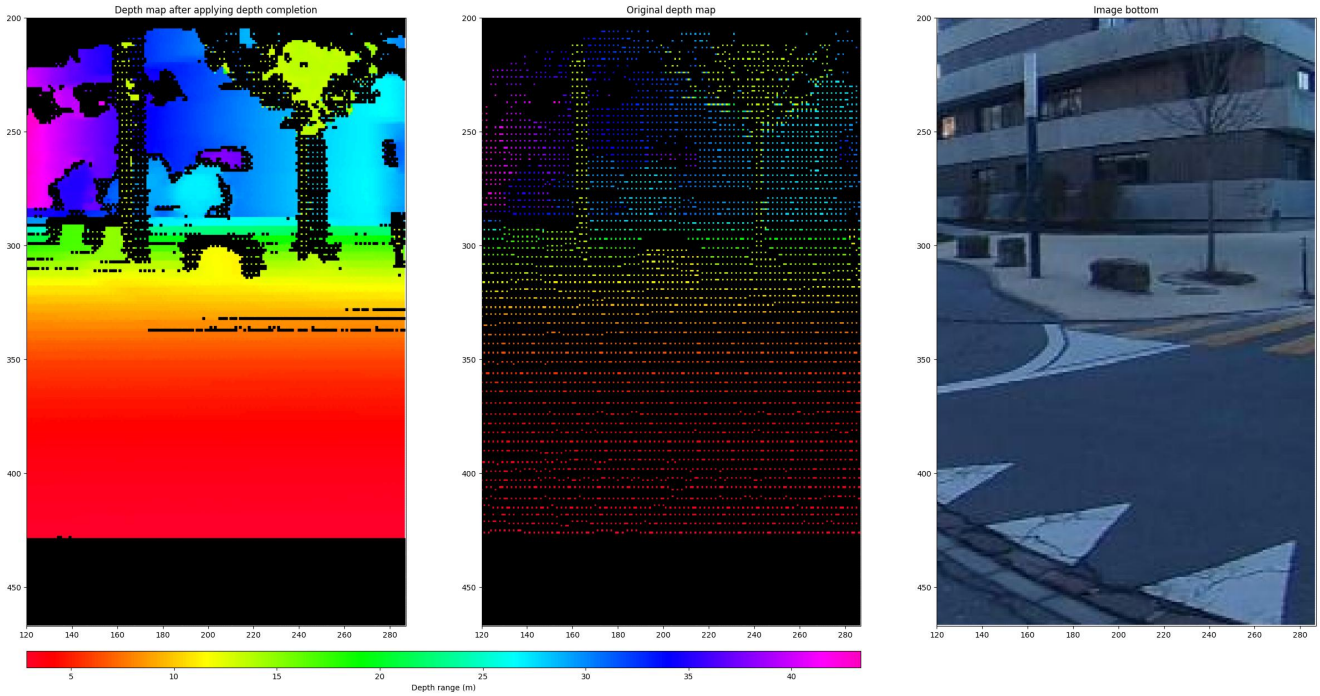


Figure 17. Depth completed depth map, original depth map and bottom image in detail view from an *outdoor* sequence.

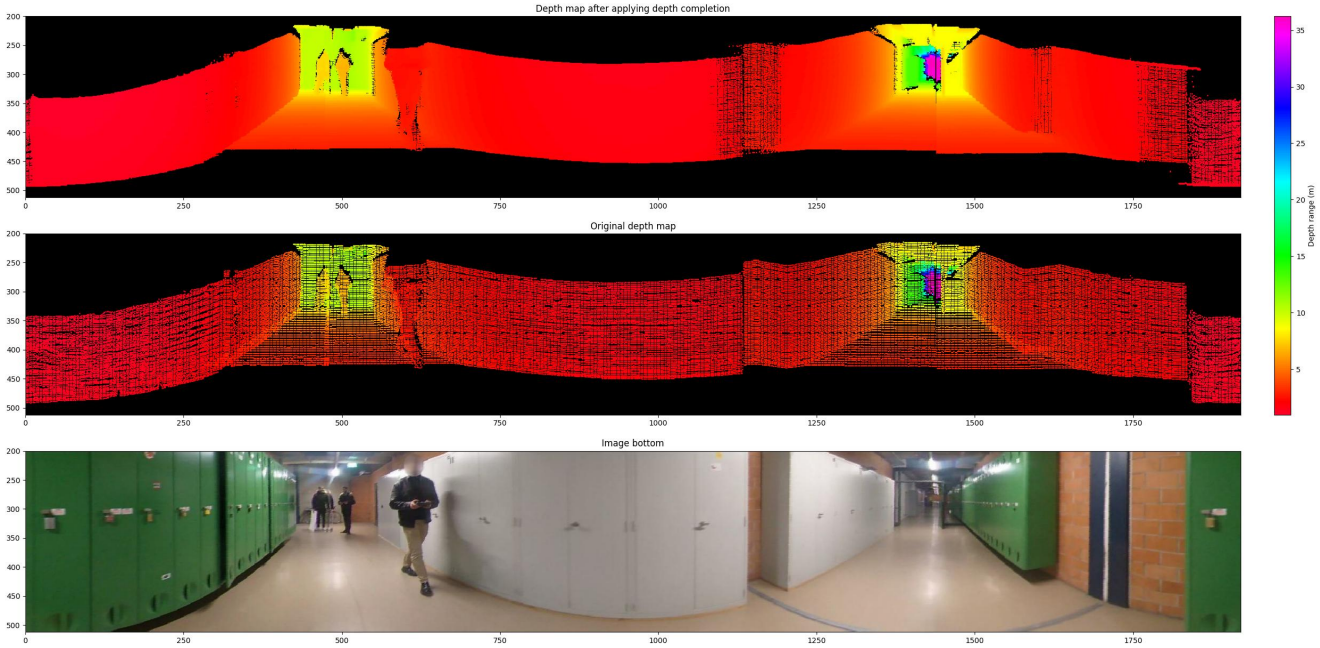


Figure 18. Depth completed depth map, original depth map and bottom image in complete view from an *indoor* sequence.

pear to be erroneous. Consequently, minor discrepancies may also be present in the measured point clouds.

The comprehensive representation of the *indoor* scene in Fig. 18 substantiates the favorable visual impression of the depth completion when the original depth map exhibits no abrupt depth transitions. Instead of providing labels with high errors in ambiguous regions, the original labels are retained due to the filtering of regions with high uncertainty.

Overall, the majority of pixels are labeled when applying depth completion, and no substantial errors are discernible

visually.

D. Benchmark Specifications

This section outlines the evaluation framework and benchmark specifications for assessing model performance on the HELVIPAD dataset. Furthermore, it details the architecture of 360-IGEV-Stereo adaptations.

D.1. Evaluation Metrics

To assess the performance of models, we rely on several metrics, each providing insights into different aspects of the model's disparity and depth estimation accuracy. Given the sparse nature of our ground-truth data for disparity and depth, we apply a masking technique to evaluate models' predictions only in areas with available ground truth values. The metrics are computed by summing over all pixels for which ground truth is available.

More formally, let us define $\mathcal{I}$ as the set of test set images and p_{ij} a pixel j within. We denote the depth and disparity ground truth values for a pixel j in image $i \in \mathcal{I}$ as r_{ij} and d_{ij} respectively. Similarly, the corresponding values of this pixel j in image i predicted by the model are denoted respectively as $\hat{r}_{ij}$ and $\hat{d}_{ij}$. Among all pixels of the image i, we denote $\mathcal{A}_i$ the subset of pixels with available ground truth values in the image.

- **Mean Absolute Error (MAE):** MAE measure the average magnitude of errors between the predicted and actual disparity in degrees (and depth in meters), offering a direct assessment of overall error. For disparity and depth respectively, the MAE is defined as:

$$\text{MAE} = \frac{1}{|\mathcal{I}|} \sum_{i \in \mathcal{I}} \frac{1}{|\mathcal{A}_i|} \sum_{j \in \mathcal{A}_i} |y_{ij} - \hat{y}_{ij}|, \tag{13}$$

  where y can represent either the depth (r) or disparity (d) values.

- **Root Mean Square Error (RMSE):** RMSE measures the square root of the average squared differences, emphasizing larger errors. It is defined as:

$$\text{RMSE} = \frac{1}{|\mathcal{I}|} \sum_{i \in \mathcal{I}} \sqrt{\frac{1}{|\mathcal{A}_i|} \sum_{j \in \mathcal{A}_i} ||y_{ij} - \hat{y}_{ij}||^2}, \tag{14}$$

  where y can represent either the depth (r) or disparity (d) values.

- **Mean Absolute Relative Error (MARE):** Considering the varying range of disparity (and depth) values, MARE is crucial. The metrics normalizes the error against the actual depth values, offering a nuanced measure of accuracy. The MARE is defined as:

$$\text{MARE} = \frac{1}{|\mathcal{I}|} \sum_{i \in \mathcal{I}} \frac{1}{|\mathcal{A}_i|} \sum_{j \in \mathcal{A}_i} \left| \frac{y_{ij} - \hat{y}_{ij}}{y_{ij}} \right|, \tag{15}$$

  where y can represent either the depth (r) or disparity (d) values.

- **Left-Right Consistency Error (LRCE):** This metrics evaluates the consistency at the left and right boundaries of 360° images by measuring the discrepancy between predicted depth values across the image edges. In the original work introducing LRCE [30], the metrics also accounts for left-right discrepancies in ground-truth data to address extreme cases where object edges align exactly with the image boundaries. Due to the sparsity of the LiDAR depth maps in the test set, there are very few valid points simultaneously at both image edges for computing LRCE metric (3 pixel pairs per image in average). As an alternative, we use the depth-completed tests maps (136 pixels in average) and compute LRCE with this augmented ground-truth. Given $\mathcal{B}_i$ the subset of valid pixel pairs in image i where ground-truth labels exist for both the leftmost and rightmost columns ($\mathcal{B}_i \subset \mathcal{A}_i$), LRCE is defined as the sum of absolute differences between left and right edges for both predicted and ground-truth disparity values:

$$\text{LRCE} = \frac{1}{|\mathcal{I}|} \sum_{i \in \mathcal{I}} \frac{1}{|\mathcal{B}i|} \sum_{j \in \mathcal{B}i} \left| e^{\text{gt}}_{i,j} - e^{\text{pred}}_{i,j} \right|, \tag{16}$$

  where $e_{i,j} = |d_{\text{left},i,j} - d_{\text{right},i,j}|$ is the left-right discrepancy term in image i for pixel pair j, computed for predicted disparity error ($e^{\text{pred}}_{i,j}$) and ground-truth disparity error ($e^{\text{gt}}_{i,j}$).

D.2. Implementation Details

In the following, we elaborate on the implementation and training details of each model included in the benchmark. All our experiments are conducted using Nvidia A100-SXM4-80GB GPUs.

PSMNet. Despite its age, PSMNet is a robust and popular method for conventional stereo depth estimation that we included in our benchmark. Our implementation is based on the code provided by the authors[2]. We initialize our model with weights from a SceneFlow-pretrained model and finetune it on our dataset for 24 epochs. The model is trained with a batch size of 20 images, with an Adam optimizer, an initial learning rate of 0.0001, and no weight decay.

[2] https://github.com/JiaRenChang/PSMNet

360SD-Net. The model is trained from scratch for 40 epochs using an Adam optimizer with an initial learning rate of 0.001, no weight decay, and a batch size of 16. It undergoes further fine-tuning for 10 epochs at a reduced learning rate of 0.0001 to enhance performance. Our implementation is based on the code provided by the authors[3].

IGEV-Stereo. Again, our implementation is based on the code provided by the authors[4]. We initialize the model from SceneFlow-pretrained weights and fine-tune it on our dataset employing an AdamW optimizer and a one-cycle learning rate schedule with a maximum learning rate of $3e^{-5}$, alongside a weight decay of $1e^{-5}$. The training spans 200k steps with a batch size of 16, equivalent to approximately 92 epochs.

360-IGEV-Stereo. We adapt the IGEV-Stereo code to include our architecture modification stated in the paper and further detailed in Appendix D.3. The implementation details of 360-IGEV-Stereo are similar to IGEV-Stereo, with some small modifications.

We convert the disparity, given in degree, to pixels to be able to warp the top image appropriately for constructing the cost volume:

$$d_{\text{pix}} = \frac{960 \text{ px} \times d_{\text{deg}}}{180^\circ}. \tag{17}$$

Hereby, 960 px is the height of the downsampled image before cropping.

During training 360-IGEV-Stereo had some instabilities. To mitigate this issue, we clamp the disparity d_{deg} in each step to $d_{\text{deg}} \in [d_{\text{deg, min}}, d_{\text{deg, max}}]$. According to the statistics of the dataset, the minimum and maximum disparity are set to $d_{\text{deg, min}} = 0.048^\circ$ and $d_{\text{deg, max}} = 23^\circ$.

To enable a better understanding of the context, we use the the full image size of 512 x 1920 for training. Except of a common photometric data augmentation, we do not apply any data augmentations.

All weights that have not been modified are initialized with the original IGEV-Stereo weights created with pretraining on Sceneflow by the authors. The model is trained with a batch size of 4 for 20 epochs which corresponds to around 130k steps. Furthermore, the maximum disparity for constructing the cost volumes is set to 128 which is the smallest number that is divisible by 32 and larger than the maximum disparity in pixels.

D.3. 360-IGEV-Stereo Architecture

To construct 360-IGEV-Stereo, we introduce three key enhancements to the IGEV-Stereo architecture.

[3] https://github.com/albert100121/360SD-Net

[4] https://github.com/gangweiX/IGEV/tree/main/IGEV-Stereo

Layer	Channels in	Channels out	Scaling	Input
1. Image feature extractor				
img_conv	3	32	1/2	top / bottom image
img_bottleneck1	32	16	1	img_conv
img_bottleneck2	16	24	1/2	img_bottleneck1
img_bottleneck3	24	32	1/2	img_bottleneck2
img_bottleneck4	32	96	1/2	img_bottleneck3
img_bottleneck5	96	160	1/2	img_bottleneck4
2. Feature concatenation				
concat_img_pm	(160+32)	192	1	(img_bottleneck4, pm_bottleneck4)
concat_conv	192	160	1	concat_img_pm
3. Upsampling layers				
up_bottleneck6	160	192	2	(concat_conv, img_bottleneck4)
up_bottleneck7	192	64	2	(up_bottleneck6, img_bottleneck3)
up_bottleneck8	64	48	2	(up_bottleneck7, img_bottleneck2)
final_conv3x3	48	48	1	up_bottleneck8
4. Stem part				
stem2pre	3	32	1/2	top / bottom image
stemconcat	(32+32)	64	1	(stem2pre, pm_coord2)
stem2post	64	32	1	stemconcat
stem4	32	48	1/2	stem2post

Table 9. **Architecture of 360-IGEV-Stereo's feature network**. The steps 1 to 3 are part of the main feature network whose features at the scales 1/4, 1/8, 1/16, and 1/32 are used to build the CGEV. Its features are concatenated with the encoded polar map at its bottleneck in 1/32 of the original image size. The orange part in the bottom of the feature network in Fig. 4 is called stem. At scale 1/4 it is used for the construction of the CGEV and at scale 1/2 the feature map obtained for the bottom image is used for spatial upsampling. The encoded polar map is concatenated with stem at 1/2 of the original image size.

Layer	Channels in	Channels out	Scaling	Input
1. Image feature extractor				
img_conv7x7	3	64	1	bottom image
img_resblock1	64	64	1	img_conv7x7
img_resblock2	64	96	1/2	img_resblock1
img_resblock3	96	128	1/2	img_resblock2
2. Feature concatenation				
concat_img_pm	(128+32)	160	1	(img_resblock3, pm_coord4)
concat_conv	160	128	1	concat_img_pm
3. Multi-scale outputs				
output04_conv	128	128	1	concat_conv
output04_resblock4	128	128	1/2	concat_conv
output08_conv	128	128	1	output04_resblock4
output08_resblock5	128	128	1/2	output04_resblock4
output16_conv	128	128	1	output04_resblock5

Table 10. **Architecture of 360-IGEV-Stereo's context network**. The features of the image are concatenated with the encoded polar map at 1/4 of the original image size. Context features at the scales 1/4, 1/8, and 1/16 are used by the ConvGRU block.

Firstly, the polar map is added as an additional input to the network. It has the same size as the input image and consists of repeated columns within a range of $[48^\circ, 144^\circ]$, corresponding to the vertical field of view of the input image. Its encoder is shared between feature and context network. The encoder contains convolutional layers with a stride of 2 to decrease the polar map size gradually. For fusing the encoded polar map with the feature we concatenate the features at the lowest possible resolution before

Layer	Channels		Scaling	Input
	in	out		
pm_coord2	1	32	1/2	polar map
pm_coord4	32	32	1/2	pm_coord2
pm_coord8	32	32	1/2	pm_coord4
pm_coord16	32	32	1/2	pm_coord8
pm_coord32	32	32	1/2	pm_coord16

Table 11. **Architecture of 360-IGEV-Stereo's polar encoder**. The polar encoder's at 1/2, 1/4, and 1/32 of the original polar map size are used. All layers are convolutional with a kernel size of 3, a stride of 2. After each layer batch normalization and the Leaky ReLU activation function are applied.

producing multi-scale outputs, followed by a convolution that recreates the number of channels. The overall architecture is outlined more in detail in Tab. 9, Tab. 10 and Tab. 11.

Secondly, we build the cost volume based on vertical instead of horizontal warping. This means that in the construction of the geometry encoding volume the group-wise correlation volume is calculated by shifting the top image about the corresponding disparity index vertically. Similarly, for building the all-pairs correlation volume the top image is warped downwards according to the disparity index.

Lastly, we apply circular padding at evaluation time. Assuming the original image I has height H and width W, the value of the pixel with the row index i and the column index j of the circular padded image I^{cp} can be calculated in dependence of the amount of padding P with the following formula:

$$I^{\text{cp}}_{i,j} = \begin{cases} I_{i,j+W-P} & \text{if } 0 \leq j < P \\ I_{i,j-P} & \text{if } P \leq j < W+P \\ I_{i,j-W-P} & \text{if } W+P \leq j < W+2P \end{cases} \tag{18}$$

Circular padding is omitted during training to reduce computations and enable larger batch sizes.

E. Additional Results

In this section, we further study the impact of pretrained weight initialization and cross-dataset generalization.

E.1. Effect of Pretrained Weights

In addition to the ablative studies detailed in the main paper, we report in Tab. 13 a detailed comparison of each model performances using randomly initialization versus fine-tuning from pretrained weights.

Both 360-IGEV-Stereo and IGEV-Stereo show significant improvements when initialized with Scene Flow pretrained weights, outperforming their randomly initialized counterparts across all depth and disparity metrics. For example, 360-IGEV-Stereo achieves a reduced depth MAE of 1.77m and RMSE of 4.36m, demonstrating the effectiveness of leveraging models trained on standard images like Scene Flow for omnidirectional image training. Surprisingly, this pattern does not hold for PSMNet, where the model initialized randomly achieves better performance than using Scene Flow pretrained weights. In contrast, 360SD-Net shows minimal improvement with Stereo-MP3D pretraining, despite the dataset being omnidirectional. This discrepancy could be attributed to Stereo-MP3D's limitation to indoor scenes, whereas HELVIPAD has a broader range of scenes.

E.2. Left-Right Consistency

In addition to the main results available in Tab. 2, we provide a more detailed analysis of Left-Right Consistency Error (LRCE) across different scene types in Appendix E.2.

Model	All	Indoor	Outdoor	Night Outdoor
PSMNet	1.80	0.93	1.31	1.16
360SD-Net	0.90	0.52	1.02	1.01
IGEV-Stereo	1.20	0.79	1.21	1.55
360-IGEV-Stereo	**0.38**	**0.17**	**0.38**	**0.46**

Table 12. **Depth-LRCE with augmented ground-truth across different scene types.** Results are reported in meters.

The results indicate that left-right consistency is more challenging to maintain in outdoor scenes, with the highest errors observed in night outdoor conditions due to low-light environments and reduced texture details. In contrast, indoor scenes achieve the lowest LRCE, likely due to more structured environments with well-defined depth boundaries and fewer extreme lighting variations.

E.3. Cross-Dataset Generalization

We study the cross-dataset generalization of 360SD-Net by first training the model on the HELVIPAD dataset and then fine-tuning it on Stereo-MP3D and Stereo-SF3D datasets [34] , which share a similar top-bottom camera configuration. Results are available in Tab. 14. Compared to a baseline trained from random initialization on Stereo-MP3D, fine-tuning on HELVIPAD significantly improved all depth and disparity metrics. On Stereo-MP3D, fine-tuning on HELVIPAD reduces depth MAE from 0.087m (random initialization) to 0.072m, along with consistent improvements in all disparity metrics. A similar trend is observed on Stereo-SF3D, where HELVIPAD pretraining improves depth MAE from 0.029m to 0.027m and disparity MAE from 0.105° to 0.099°, outperforming models pretrained on Stereo-MP3D. Note that although we used the authors' provided code[5] and the reported hyperparameters, our repro-

[5] https://github.com/albert100121/360SD-Net

duced results differ from those reported in the original paper. We present our outcomes for a fair comparison.

Fine-tuning on HELVIPAD offers a broader diversity of scenes, particularly outdoor environments. These results suggest that HELVIPAD not only captures a wider range of scenarios but also provides robust features that enhance generalization to datasets with overlapping characteristics.

Real-world representation. While collected on a university campus, the dataset captures many common urban environments, such as parking lots, roads, underpasses, pedestrian squares, footpaths, corridors and crowded halls. To further demonstrate transferability, we present below the qualitative result of 360-IGEV-Stereo on a real-world image without labels from the 360SD-Net paper:

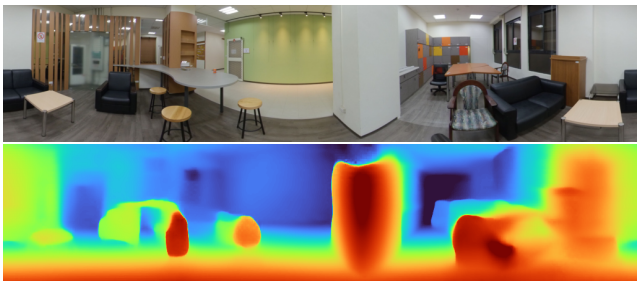


Figure 19. **Qualitative result of 360-IGEV-Stereo on a real-world image.** The top image shows the bottom image of the input, while the bottom image displays the predicted disparity map.

Trained solely on HELVIPAD, the model demonstrates zero-shot capabilities in this environment.

Method	**Initialization**	**Disparity** (°)			**Depth** (m)		
		MAE ↓	RMSE ↓	MARE ↓	MAE ↓	RMSE ↓	MARE ↓
PSMNet [6]	random	0.29	0.50	0.25	2.51	5.67	0.18
	Scene Flow	0.33	0.54	0.29	2.78	6.17	0.19
360SD-Net [34]	random	0.22	0.42	0.19	2.12	5.08	0.18
	Stereo-MP3D	0.23	0.44	0.20	2.31	5.41	0.16
IGEV-Stereo [38]	random	0.23	0.44	0.18	2.10	5.30	0.17
	Scene Flow	0.23	0.42	0.17	1.86	4.47	0.15
360-IGEV-Stereo	random	0.20	0.40	0.16	1.91	4.60	0.14
	Scene Flow	0.19	0.40	0.15	1.72	4.30	0.13

Table 13. Comparison of model performance with random initialization vs. fine-tuned from pretrained weights.

Dataset	**Initialization**	**Disparity** (°)			**Depth** (m)		
		MAE ↓	RMSE ↓	MARE ↓	MAE ↓	RMSE ↓	MARE ↓
Stereo-MP3D	reported [34]	0.145	0.693	–	0.059	0.218	–
	random	0.148	0.994	0.074	0.087	0.294	0.050
	HELVIPAD	**0.129**	**0.930**	**0.063**	**0.072**	**0.252**	**0.039**
Stereo-SF3D	reported [34]	0.103	0.369	–	0.003	0.091	–
	random	0.105	**0.468**	0.020	0.029	0.071	0.016
	Stereo-MP3D	0.121	0.505	0.023	0.035	0.079	0.019
	HELVIPAD	**0.099**	0.469	**0.018**	**0.027**	**0.069**	**0.015**

Table 14. Cross-dataset generalization results by fine-tuning 360SD-Net [34].