Online Video Depth Anything: Temporally-Consistent Depth Prediction With Low Memory Consumption

Johann-Friedrich Feiden Tim Küchler Denis Zavadski Bogdan Savchynskyy
Carsten Rother
Heidelberg University
{firstname.lastname}@iwr.uni-heidelberg.de

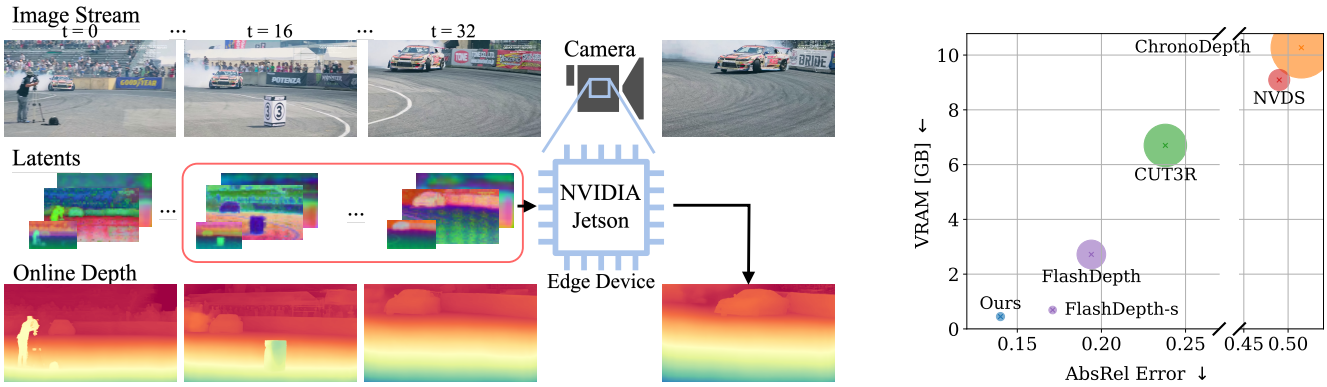

Figure 1. (Left) Our *online Video Depth Anything* approach (oVDA) produces high-quality depth predictions at up to 20 FPS on an edge device (NVIDIA Jetson Orin NX) by caching a sliding window of past latent features (red box). (Right) Performance comparison of various *online* video depth estimation methods for the KITTI [12] dataset. Our oVDA approach outperforms all competitors in both AbsRel error and VRAM usage. Note that low VRAM usage is crucial for many real-world applications, such as deployment on edge devices. The size of each circle represents the number of parameters.

Abstract

Depth estimation from monocular video has become a key component of many real-world computer vision systems. Recently, Video Depth Anything (VDA) has demonstrated strong performance on long video sequences. However, it relies on batch-processing which prohibits its use in an online setting. In this work, we overcome this limitation and introduce online VDA (oVDA). The key innovation is to employ techniques from Large Language Models (LLMs), namely, caching latent features during inference and masking frames at training. Our oVDA method outperforms all competing online video depth estimation methods in both accuracy and VRAM usage. Low VRAM usage is particularly important for deployment on edge devices. We demonstrate that oVDA runs at 42 FPS on an NVIDIA A100 and at 20 FPS on an NVIDIA Jetson edge device. We will release both, code and compilation scripts, making oVDA easy to deploy on low-power hardware.

1. Introduction

Online monocular depth prediction has become an increasingly relevant topic for real-time, industrial applications, such as robotics, drone technology or augmented reality. This is primarily due to its potential to replace expensive sensor systems such as LiDAR with low-cost monocular cameras [18]. We aim at developing a monocular video depth estimation method that can be applied to any input (zero-shot capability) and is fast and accurate while running on low-powered, low-memory hardware (see Fig. 1).

The field of *single-image* depth prediction has made tremendous strides forward in recent years [3, 10, 16, 34, 46, 47, 50]. Today, we have powerful foundation models for non-metric, i.e. scale- and shift-invariant, depth estimation working well across diverse domains. Among these, Depth Anything v2 [47] stands out for its impressive performance and manageable computational cost, particularly in its smaller variant. However, applying such a non-metric, single-image method to each frame of a video sequence, naturally produces temporally flickering depth maps, neces-

sitating frame-by-frame alignment or post-processing.

For *video* depth estimation, the temporal dimension can be used to improve the performance with respect to single-image methods. But, at the same time, for many applications such as augmented reality, the temporal dimension poses the additional challenge of producing *temporally consistent* depth maps. Earlier approaches to this challenge often relied on test-time training [20, 27, 53]. While this strategy is effective, it is computationally expensive and too slow for real-time settings [6]. Modern research instead focuses on feed-forward methods, which can be broadly grouped into three categories (see Sec. 2). However, methods from two categories do not have the potential to be deployed on low-powered hardware due to high computational cost and memory consumption, i.e. diffusion-based approaches [15, 36, 45] as well as methods that maintain an underlying 3D representation [19, 21, 23, 37, 40, 41, 52]. In contrast, methods of the third category that extend existing single-image models [6, 7, 9, 17, 43] do have this potential.

Our work falls into this third category. We build upon Video Depth Anything (VDA) [6], a recent method that achieves state-of-the-art results for non-metric depth prediction of long videos. However, VDA is designed for *offline* video processing and lacks support for online inference, as it can only process entire *batches of frames* at a time. To address this limitation, we introduce oVDA, which transforms VDA to an online setting for predicting temporally consistent video depth maps. We draw inspiration from the inference strategies employed in Large Language Models (LLMs). In LLMs, it is common to cache previously computed latent features (i.e. a context window), allowing for efficient prediction of new tokens without recomputing the entire context [24, 33, 44]. Analogously, oVDA stores latent features of past video frames and uses them as temporal context for predicting future depth maps. To efficiently train our model, we again draw inspiration from LLMs. We adapt masked attention [38], which enables the training process to selectively attend to past frames while preventing the model from using future frames. Finally, to further enhance temporal consistency we introduce a new scale-and-shift consistency loss. Our contributions are:

- We introduce oVDA which transforms VDA to an online depth prediction method by employing LLM-inspired inference and training techniques, as well as a new scale-and-shift consistency loss.
- oVDA outperforms competing online video depth estimation methods, both in terms of accuracy and VRAM usage. It runs at 42 FPS on an NVIDIA A100 and at 20 FPS on an NVIDIA Jetson edge device.
- We will release both, code and the compilation script for oVDA, making it easy to deploy on low-power edge device hardware such as NVIDIA Jetson.

2. Related Work

Early learning-based, single-image depth estimation methods were limited to one datasets and constrained scenarios [10, 25, 28]. The introduction of the scale- and shift-invariant loss by MiDaS [34] redirected the focus towards multi-dataset training and zero-shot evaluation. With powerful backbones such as Stable Diffusion [35] or DINOv2 [29] and access to vast amounts of training data, numerous zero-shot single-image depth estimation approaches have emerged [11, 16, 46–48, 50], capable of predicting scale- and shift-invariant depth for almost any image. While these methods achieve outstanding results for single-images, they produce flickering and temporally inconsistent depth maps when applied frame-by-frame to videos [43]. Note that *metric*, single-image depth prediction has been less popular due to limited amount of ground truth data.

Monocular Video Depth Estimation aims at exploiting temporal information in order to improve on single-image methods and, at the same time, reduce flickering artefacts. Early works relied on test-time training [20, 27, 53], for instance by using photometric or geometric constraints over multiple frames. However, due to their high computational cost, such methods are rarely suitable for real-time inference and often have to operate at low resolutions [6, 27]. Using LSTMs [14] for temporal information flow has also been explored [22, 31, 51], but since these methods are typically trained on relatively small datasets, their zero-shot capability is limited. Moreover, despite recurrent blocks, they still tend to produce flickering results [43].

Modern approaches follow one of three strategies. The first leverages video diffusion models [1, 2], which possess a strong understanding of temporal dynamics. Methods such as [15, 36, 45] adapt these models to predict depth, resulting in highly consistent temporal and spatial predictions. This, however, comes at the cost of high computational demand, or, in some methods, batch-wise processing [15, 45], which prevents their use in online settings.

The second strategy stores a continuous 3D representation for a video. DUSt3R [41] estimates a point cloud from an unconstrained image collection, from which camera parameters and depth can be derived. Building upon this idea, several works [19, 21, 23, 37, 40, 52] predict depth for multiple frames and subsequently perform global alignment, which again hinders online usage. Addressing this, other methods [40, 55] construct a persistent state that is updated with each incoming frame, resulting in long, consistent video predictions.

In the third strategy, single-frame methods are extended with temporal modules, thereby preserving the strong single-image prior. For example, NVDS [43] utilises an off-the-shelf single-image depth predictor, followed by a smoothing network that temporally aligns the predictions. While this allows for online predictions, best results still

require backward refinement which is not available in an online setting. Another approach is to perform temporal alignment directly within the decoder. Video Depth Anything (VDA) [6], for instance, uses Depth Anything v2 (DAv2) [47] as its image backbone, achieving results surpassing DAv2. Due to training and architectural constraints, VDA can only process videos in batches, limiting its applicability to online depth estimation. FlashDepth [7] also builds upon DAv2 and extends it with the state-space model MambaV2 [8, 13] for incorporating temporal information more efficiently, while at the same time enabling online prediction. Other learning-based approaches, such as MAMo [49], introduce a temporal memory which is updated during inference. However, such an approach cannot modify the backbone predictions, leading to flickering depth for longer videos [43].

Our work follows the third strategy. We transform VDA to an online method, which we denote as oVDA, that runs at 42 FPS on an NVIDIA A100. We compare against ChronoDepth [36], NVDS [43], FlashDepth [7] and CUT3R [40] as recent online methods, and show that we outperform them in terms of accuracy and VRAM usage.

3. Method

In Sec. 3.1 we present the architecture of the Video Depth Anything (VDA) model, on which we build. For our online Video Depth Anything (oVDA) method, we improve on three aspects of VDA. Firstly, Sec. 3.2 describes our online inference strategy. Secondly, Sec. 3.3 explains our new training procedure, and finally, Sec. 3.4 introduces the new loss function of oVDA.

3.1. VDA Architecture

VDA has demonstrated strong performance for offline, non-metric depth prediction of very long video sequences [6]. Our oVDA method uses the same basic architecture as VDA, as well as its pre-trained weights. For computational efficiency we build upon VDA-s(mall). The key difference between our oVDA method and VDA is that VDA processes frames in parallel, i.e., in batches, whereas oVDA processes only a single (the current) frame at a time. This change requires to redesign the VDA Motion Modules (see Sec. 3.2), since these are the only blocks that perform temporal reasoning, while we keep all other VDA blocks unchanged. Our full oVDA architecture is shown in Fig. 2 (left).

VDA uses a DINOv2 encoder for all frames of a batch, inspired by single-image Depth Anything v2 (DAv2) [47]. Given input images $X \in \mathbb{R}^{N\times C\times H\times W}$, where N is the number of frames, C the number of channels, and H, W the height and width, the encoder extracts feature maps $F_i \in \mathbb{R}^{N\times \frac{H}{p}\times \frac{W}{p}\times C_i}$. Here, p denotes the patch size, and i is the index of the transformer block from which the feature map originates. For clarity, we ignore the batch dimension in our notation, as it can be incorporated into the number of frames. Since DINOv2 has no temporal information flow, the extracted features are independent for each frame.

The extracted DINOv2 features are first resized by the Spatiotemporal Head according to their layer within the encoder, with factors $4, 2, 1$ and $1/2$ respectively. These latent features L_j are then transformed and iteratively fused by Motion Modules, Residual Blocks, and Resize Blocks (see Fig. 2 (left)). Here j is the block number (in total 12 blocks) within the Spatiotemporal Head. To enable temporally consistent predictions, VDA uses temporal self-attention within the Motion Modules. For this, the temporal and spatial dimensions are reordered, allowing attentions to operate across time (frames): $Att(L'_j)$ with $L'_j \in \mathbb{R}^{(\frac{H}{p}\times\frac{W}{p})\times N\times C_i}$. At the end of a Motion Module, the original tensor shape is restored, and the head continues processing in a frame-wise manner. As shown in VDA [6], the Motion Modules are powerful enough to maintain temporal consistency while even improving depth prediction quality compared to the single-frame DAv2 model.

To transform the batch-wise VDA method to an online setting, where only the depth of the current frame is predicted, a self-attention mechanism is no longer applicable. The reason is that future frames are not yet available, and, secondly, depth predictions of past frames are fixed.

3.2. oVDA Inference Strategy

To enable online depth prediction for VDA, we draw inspiration from Large Language Models (LLMs). Just as LLMs predict the next token given a sequence of previous tokens [38, 44], our goal is to predict the depth of the current frame using a moving window of the past c frames as context. Analogously to LLMs that cache previously hidden states, such that predicting a new token does not require recomputation of the whole context window, we also cache the hidden representations of past frames and reuse them during inference to predict the next frame. Instead of performing self-attention across all frames in the context window (e.g. 32 for VDA), our approach reduces to cross-attention at inference time, using the cache as attention context (i.e. keys K and values V) for predicting the current frame. This change allows for a more efficient forward pass, leveraging past information at real-time instead of batch-processing. Although this approach does work without retraining the model, it results in a noticeable drop in performance compared to VDA (see Sec. 5.1). We overcome this performance drop with our LLM-inspired training technique (see Sec. 3.3).

As shown in Fig. 2 (middle), we cache hidden features $L'_{j,t-c}, \dots, L'_{j,t}$ before every attention within each Motion Module, where t is the index of the current frame and c the maximum context length. At the beginning of a sequence, when no prior context is yet available, we apply

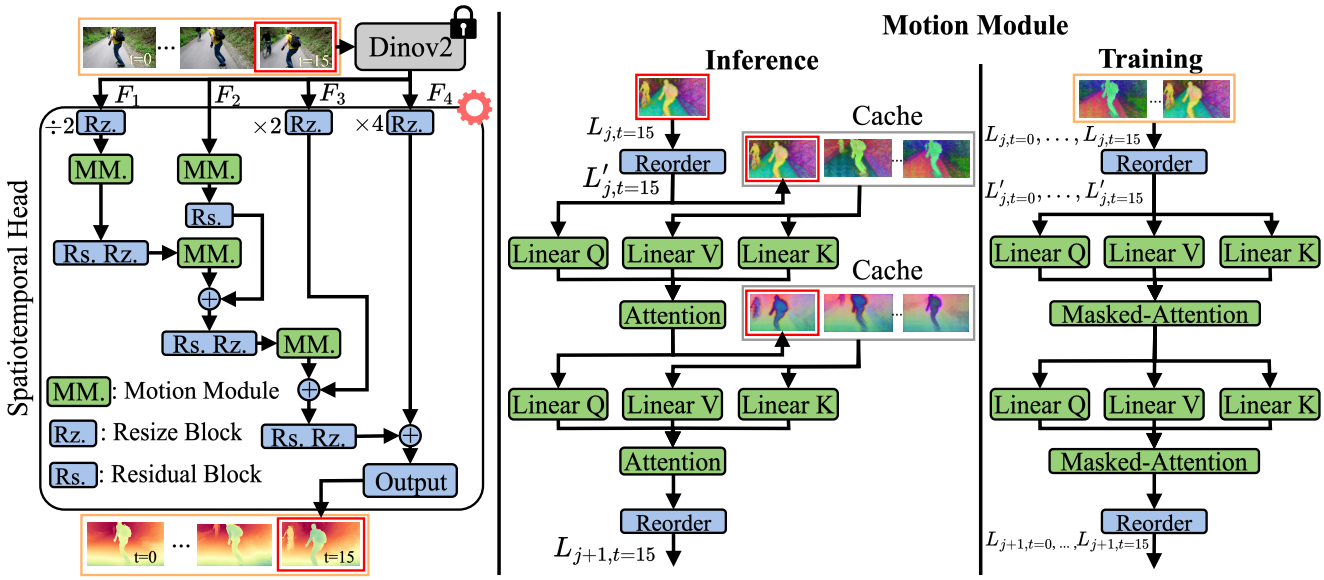

Figure 2. (Left) Our oVDA Architecture operating on the current frame $t = 15$. In contrast to offline VDA, which runs in batches of frames (orange box), our online oVDA approach processes only the current frame (red box). The blue blocks process only spatial dimensions, while the Motion Modules (green) perform temporal reasoning and differ from VDA. During training, only the Spatiotemporal Head is fine-tuned, while we keep the DINOv2 backbone frozen. (Middle and Right) Illustration of the inference and training procedure of our new Motion Module with number j. Firstly, the hidden features are re-ordered to perform temporal reasoning. During inference, the cache is updated by adding the current latent feature ($L'_{j,t=15}$) and removing the latent feature of the temporally last frame. Afterwards, cross-attention is applied between the current latent feature and the cached latent features. At training time, similar to VDA, we process a batch (here $t = 0, ..., 15$) of frames but apply masked self-attention, ensuring that the current frame can only attend to past frames.

standard self-attention. As subsequent frames arrive, the context window is incrementally populated until it reaches its maximum context length c. Beyond this point, the context operates as a sliding window over the video sequence, while always discarding the temporally last cached frame to maintain a fixed-size context window.

Unlike LLMs, which typically employ context windows spanning tens of thousands of tokens, we limit our context length to $c = 16$ for efficiency (see ablation in supplement).

3.3. oVDA Fine-Tuning

We propose to use an LLM-inspired training approach called *masked attention* (see Fig. 2 (right)). In LLMs, masked attention forces the model to predict the next token as a function of previous tokens by masking the attention weights as a lower-triangular matrix, while still allowing for batch-wise propagation through the network during training [38]. Analogously, we apply attention masking within the Motion Modules of the Spatiotemporal Head, ensuring that past frames influence future frames, but not vice versa. This design enables batch-wise training to preserve a training procedure consistent with that of VDA.

Since we utilise pre-trained VDA weights, which were already optimised for video depth prediction, we only need to fine-tune the model for the online setting. Taking this into account, we train with a comparably small learning rate of 10^{-6} (10^{-4} for VDA) and a batch size of 16 for 130,000 steps, using a cosine learning rate scheduler.

To enable dense supervision and given that VDA has already been trained on real-world data, we fine-tune exclusively on synthetic datasets. Specifically, we use IRS [39], PointOdyssey [54], where we filter PointOdyssey scenes that lack background depth, TartanAir [42] and VKITTI [5]. This results in a total of 956 scenes of varying lengths, and overall approximately 864,000 frames. Furthermore, to prevent overfitting to larger datasets, we apply uniform sampling across datasets. We simulate different frame rates by randomly sampling every first up to every fourth frame in a sequence. Following VDA, we apply random cropping to a resolution of 512×512.

On top of VDA's training procedure, we utilise a simple frame augmentation technique to increase information exchange in the temporal dimension. For this, we select random rectangular-shaped regions in every frame and set the respective RGB values to 0. The number of affected pixels varies uniformly between 0 % and 40 % for every frame. Experiments show that this strategy improves accuracy (see Sec. 5.1).

3.4. oVDA Loss Function

VDA uses a linear combination of the scale- and shift-invariant loss (SSI) $\mathcal{L}_{\text{SSI}}^{\text{Image}}$, initially introduced in

MiDaS [34], and a newly proposed Temporal Gradient Matching loss $\mathcal{L}_{\text{TGM}}$ [6]. Note that the SSI loss is modified by calculating the scale and shift parameters over sequences rather than over individual images, to encourage consistent depth within each sequence [6], which we denote as $\mathcal{L}_{\text{SSI}}^{\text{Scene}}$. The total VDA loss is then defined as

$$\mathcal{L}_{\text{VDA}} = \alpha \cdot \mathcal{L}_{\text{SSI}}^{\text{Scene}} + \beta \cdot \mathcal{L}_{\text{TGM}}, \tag{1}$$

where α and β are weighting parameters.

While this loss works well in the offline setting, it is suboptimal for our online setting. In VDA, the SSI loss is applied across the entire sequence, which aligns all frames jointly. As a result, each frame contributes equally to the scale, regardless of its temporal position in the sequence, pushing the model to predict scale-consistent batches. In an online setting, this is suboptimal since the model should predict the current frame at the same scale as preceding frames, resulting in a considerably more constrained task.

We address this issue by proposing the Scale-and-Shift Consistency Loss (SaSCon). First, we calculate the optimal scale and shift for the first frame in the sequence. This alignment (scale and shift) is then applied to the entire sequence. Simultaneously, we compute the optimal per-frame scale and shift to align each frame individually. Then, we calculate the L_1 loss between two depth maps:

$$\mathcal{L}_{\text{SaSCon}} = \sum_{i=1}^{c} \left| \hat{D}_i^{\text{first}} - \hat{D}_i^{\text{indi.}} \right|_1, \tag{2}$$

where $\hat{D}_i^{\text{first}}$ is the predicted depth of the i-th frame aligned using the scale and shift derived from the first frame, and $\hat{D}_i^{\text{indi.}}$ is the predicted depth of the same frame aligned with its own per-frame scale and shift. While the $\mathcal{L}_{\text{SSI}}^{\text{Scene}}$ learning objective aims to generate a globally consistent depth sequence, $\mathcal{L}_{\text{SaSCon}}$ focuses solely on temporal consistency, i.e. predicting a scale- and shift-consistent next frame. This loss is advantageous in our sliding-window-based online setting. Our final training loss is defined as

$$\mathcal{L}_{\text{oVDA}} = \alpha \cdot \mathcal{L}_{\text{SSI}}^{\text{Scene}} + \beta \cdot \mathcal{L}_{\text{TGM}} + \gamma \cdot \mathcal{L}_{\text{SaSCon}}, \tag{3}$$

where $\alpha = \beta = \gamma$ are chosen as 1.

4. Experiments

We evaluate our online method with respect to three different datasets in a zero-shot setting: KITTI [12] (outdoor), Bonn [30] (indoor), and Sintel [4] (synthetic). In contrast to other online depth prediction approaches we evaluate results by computing the optimal scale and shift *only* for the first frame and then align the entire depth video using these two parameters. Our evaluation is motivated by a typical, real-world use case where scale and shift is computed from external measurements or a calibration object. This evaluation is more challenging, as drifts in scale and/or shift can no longer be partially mitigated by global alignment. Also, in contrast to previous works such as VDA [6], FlashDepth [7], or DepthCrafter [15], we evaluate video sequences in full length without limiting the number of frames, which is more realistic for an online setting. Note, an evaluation with global alignment is in the supplement.

4.1. Zero-Shot Results

After resizing to match the native resolution of each dataset, we compute the absolute relative error, defined as AbsRel $= \frac{1}{N}\sum_i \frac{|D_i - \hat{D}_i'|}{D_i}$ and the inlier ratio $\delta_1 = \frac{1}{N}\sum_i \mathbb{I}\left(\max\left(\frac{D_i}{\hat{D}_i'}, \frac{\hat{D}_i'}{D_i}\right) < 1.25\right)$, where D_i is the ground truth depth, $\hat{D}_i'$ the aligned predicted depth, $\mathbb{I}(\cdot)$ the indicator function, and N the number of valid pixels. Aligning only with respect to the first frame makes temporal consistency strongly correlated with the overall performance of each method, as a drifting scale and shift automatically lead to larger errors. As in prior work [6, 7], the following analysis focuses on the inlier δ_1 ratio. Results are shown in Tab. 1.

We compare our approach against ChronoDepth [36], a diffusion-based method, NVDS [43], a method which smooths single-frame depth predictions (details in the supplement), FlashDepth [7] which augments a single-frame method with temporal blocks and CUT3R [40], a persistent state model. Depth Anything v2 [47] is shown in grey, as it is a single-frame method with no temporal context but strong generalisation capabilities. It is still often applied to video depth prediction even though it can produce depths with inconsistent scale and shift.

In the outdoor driving scenario (KITTI), our oVDA method outperforms the best competitor, FlashDepth-s, by more than 3 pp in the δ_1 inlier ratio, demonstrating that our sliding window approach is able to produce high accuracy, temporally consistent video depth predictions. As well, in the indoor setting (Bonn), we outperform the best competitor, CUT3R, with a 2 pp δ_1 gap. Only for the synthetic Sintel datasets our method is the runner-up after FlashDepth. However, as we see next, FlashDepth needs more memory than our oVDA, since it has ten times more parameters, runs also slower and can still produce flickering depth (see Fig. 3). Overall, the δ_1-Rank clearly shows the superiority of our oVDA method.

Tab. 2 presents the FPS and VRAM consumption of all evaluated methods. Since both metrics are highly dependent on the input resolution, we use the same resolution for all methods, even if they were not trained at that resolution. All measurements were done on an NVIDIA A100 GPU.

FlashDepth-s achieves the highest frame rate, followed by our oVDA approach. Furthermore, oVDA achieves the lowest VRAM usage. Please note that, apart from the high frame rate, FlashDepth-s is inferior to oVDA in all other as-

Method	KITTI [12] (375 × 1242)			Bonn [30] (480 × 640)			Sintel [4] (436 × 1024)			δ_1-Rank
	Res.	AbsRel (↓)	δ_1 (↑)	Res.	AbsRel (↓)	δ_1 (↑)	Res.	AbsRel (↓)	δ_1 (↑)	
DAv2-s [47]	378×1246	0.216	0.655	490×644	0.202	0.712	378×1246	0.390	0.514	5.3
DAv2-l [47]	378×1246	0.243	0.610	490×644	0.196	0.721	378×1246	0.403	0.521	5.0
FlashDepth [7]	378×1246	0.165	0.760	490×658	0.131	0.789	434×1022	0.370	**0.564**	2.6
FlashDepth-s [7]	378×1246	0.156	0.774	490×658	**0.116**	0.848	434×1022	**0.355**	0.532	2.6
NVDS [43]**	288×896	0.490	0.380	480×896	0.259	0.602	448×896	0.440	0.413	6.6
CUT3R [40]	144×512	0.238	0.690	384×512	**0.116**	0.850	208×512	0.635	0.400	4.3
ChronoDepth [36]*	256×896	0.515	0.289	480×640	0.262	0.587	448×1024	0.563	0.355	8.0
Our oVDA	280×924	**0.140**	**0.809**	480×640	0.118	**0.871**	329×924	0.380	0.548	**1.3**

Table 1. Quantitative results for outdoor (KITTI [12]), indoor (Bonn [30]) and synthetic datasets (Sintel [4]). The maximum depth is clipped to 80m. We evaluate full sequences without a maximum frame cap. The average sequence length is 311 for KITTI, 1118 for Bonn, and 46 for Sintel. Predictions are resized to the original resolution before evaluation. Depths are aligned only with respect to the first frame and the respective scale and shift are then applied to the entire video sequence. For each method and dataset we show the processing resolution alongside the Absolute Relative Error (AbsRel) and the inlier ratio (δ_1). Depth Anything v2 small (DAv2-s) and DAv2-l are highlighted grey since they are single-frame methods and not designed to predict temporally consistent videos. *For online usage of ChronoDepth, we predict a single new frame with a context window of four. **We use the online version (no backwards refinement).

pects: i) lower accuracy (δ_1-Rank 1.3 versus 2.6), ii) higher VRAM usage, iii) temporally flickering depth (see Fig. 3), and iv) lower temporal stability (see Sec. 5.3)

4.2. Qualitative Comparison

Fig. 3 presents results for an in-the-wild video from DAVIS [32]. We show four uniformly sampled frames, as well as a stitched image of 24-pixel-wide vertical slices (red column in each image). This helps to assess temporal consistency, where, ideally, the stitched image shows a smooth and stable depth/RGB gradient over time. Note that ChronoDepth [36] and NVDS [43] are run in their online setting. Resolution is kept very near native resolution. Our oVDA method achieves the highest temporal consistency, as seen in the stable depth gradient of the longboard rider (last row), followed by CUT3R [40]. FlashDepth and FlashDepth-s [7] in particular produce noticeable flickering artifacts, making it less suitable for downstream applications [22]. Additionally, our oVDA approach, as well as FlashDepth and FlashDepth-s, preserves fine details in the depth prediction, such as individual tree leaves. In contrast, CUT3R produces coarse and blurry depth maps. These results highlight the ability of oVDA to deliver high-quality, temporally stable depth predictions suitable for demanding applications like augmented reality (AR). For more visualisations we refer to the supplement and the supplementary videos.

4.3. Edge Device

To demonstrate oVDA's versatility and suitability for edge device deployment, we run it on an NVIDIA Jetson Orin NX, which is a compact device with limited VRAM and limited GPU power. As shown in Tab. 3, oVDA reaches 9 FPS with 32-bit floating point precision (FP32), sufficient for tasks such as robotic indoor navigation or obstacle avoidance. When using FP16, the runtime increases to 20 FPS without a major drop in accuracy, enabling more demanding applications like real-time AR or low-latency drone control. This stability across precisions likely stems from the fact that a frozen encoder is used and no numerically extreme activations occur in the Motion Module, making oVDA ideal for resource-constrained hardware.

It is worth noting that oVDA outperforms methods like

Res.	Method	FPS (↑)	VRAM [GB] (↓)
280×924	DAv2-s	72	0.25
	DAv2-l	12	2.04
	FlashDepth-s	**60**	0.69
	FlashDepth	30	2.72
	NVDS*	7	9.09
	CUT3R*	10	6.70
	ChronoDepth*	2	10.27
	Our oVDA	42	**0.45**

Table 2. Speed Comparison. All experiments were conducted on an NVIDIA A100 graphics card, processing 5,000 frames. We round the FPS to the closest integer and the VRAM to 10 MB. We try to keep the processing resolution as close as possible to 280×924. * marks methods that do not support this resolution and use a slightly adjusted resolution.

Precision	AbsRel (↓)	δ_1 (↑)	FPS (↑)	VRAM [GB] (↓)
FP32	0.1399	0.8094	9	0.64
FP16	0.1398	0.8093	20	0.49

Table 3. Results of oVDA on an NVIDIA Jetson Orin NX. We achieve 20 FPS with 16-bit floating-point (FP16) precision. We evaluated two precisions for KITTI [12] and observe that FP16 does not significantly degrade performance. FPS and VRAM usage were measured at a resolution of 280×924 over 1,000 frames.

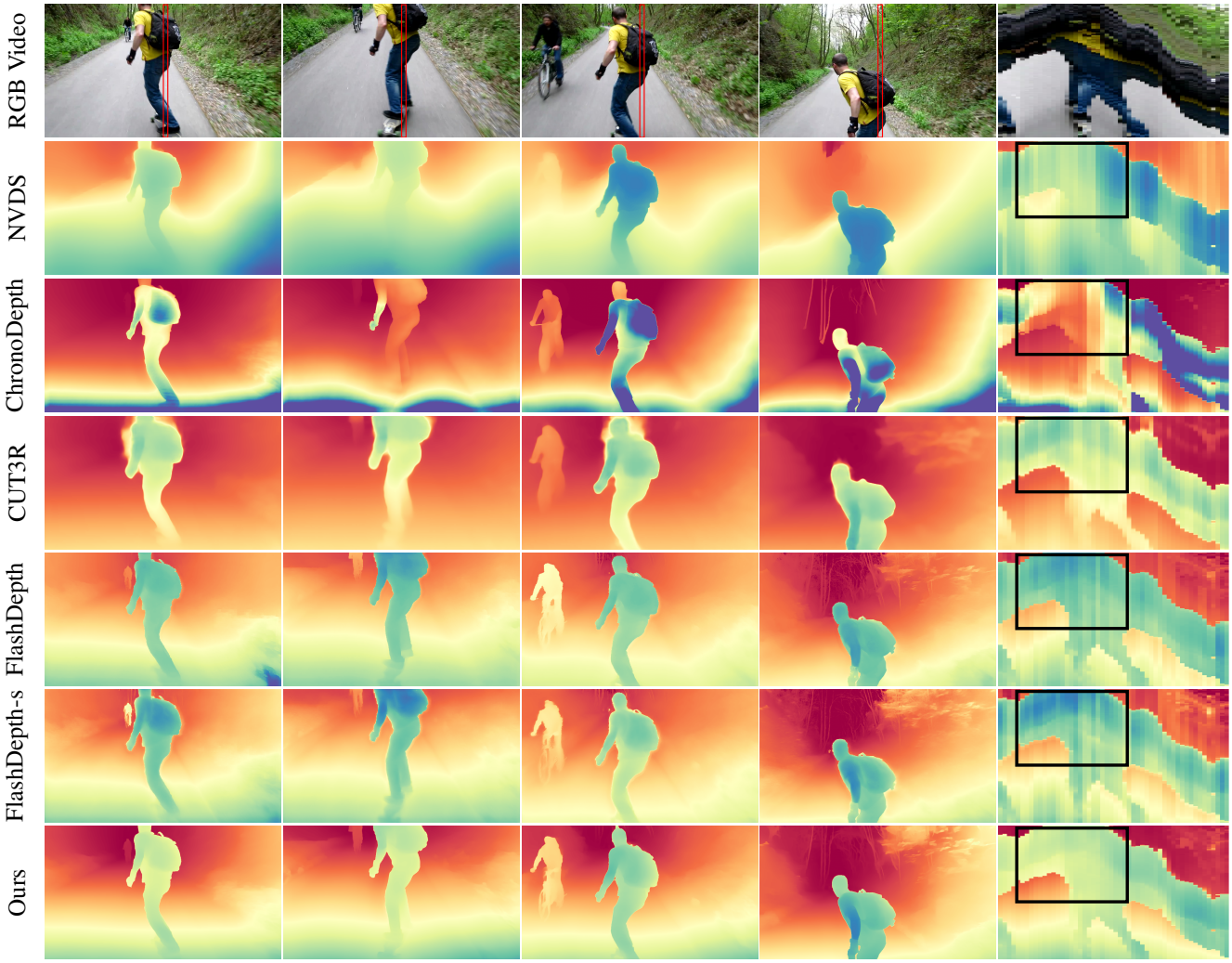

Figure 3. Visual comparison for an in-the-wild-video from DAVIS [32]. We show four frames, equally spaced in time throughout the entire video sequence. The last row gives a stitched image of vertical slices (indicated by the red column in each RGB image, where the column is 24 pixels wide).The rightmost image is again the stitched version of the predicted depths to inspect temporal consistency. Note that both ChronoDepth [36] and NVDS [43] are run in their online setting. We see that oVDA produce the temporally most consistent results followed by CUT3R [40], in contrast to e.g., FlashDepth-s [7] (black box). However, the predictions of CUT3R are more blurry compared to ours. In summary, our approach is visually best in terms of temporal consistency and on par with FlashDepth in terms of details.

CUT3R, which has 26× more parameters, in terms of accuracy (see Tab. 1), runtime, and VRAM (see Tab. 2). In particular, oVDA achieves twice the FPS (20 versus 10) on an NVIDIA Jetson Orin NX compared to CUT3R on an NVIDIA A100 graphics card.

We will release model weights, code, and a compilation script for edge device deployment and integration into downstream applications.

5. Ablation Studies

We demonstrate in Sec. 5.1 the benefits of our proposed SaSCon loss. Sec. 5.2 compares our oVDA method to the offline base model Video Depth Anything (VDA) [6]. Finally, an ablation study with respect to temporal stability is given in Sec. 5.3. Note, an ablation of different context lengths is available in the supplement. In the following, all results are based on the KITTI dataset [12]. As in Sec. 4, the alignment of the predicted depth sequence is performed with respect to the first frame.

5.1. SaSCon Loss

Tab. 4 presents results for various loss configurations. Applying the oVDA inference scheme to VDA without fine-tuning produces poor performance, as VDA is not designed for online inference and its attention mechanisms assume

that past frames can be modified. Fine-tuning with the original $\mathcal{L}_{\text{VDA}}$ loss (see Eq. (1)) improves results by nearly 10 pp over the untrained baseline. The frame augmentation strategy described in Sec. 3.3 yields a further gain of about 1 pp. Adding our proposed $\mathcal{L}_{\text{SaSCon}}$ loss (see Eq. (2)) on top of $\mathcal{L}_{\text{VDA}}$ yields an additional improvement of approximately 1.7 pp. Finally, our $\mathcal{L}_{\text{oVDA}}$ loss (see Eq. (3)) combined with frame augmentation delivers the highest overall performance, defining our proposed oVDA method.

Loss	AbsRel (↓)	δ_1 (↑)
No Training	0.201	0.693
$\mathcal{L}_{\text{VDA}}$	0.155	0.778
$\mathcal{L}_{\text{VDA}}$ + Frame Aug.	0.149	0.788
$\mathcal{L}_{\text{VDA}}$ + $\mathcal{L}_{\text{SaSCon}}$	0.148	0.795
$\mathcal{L}_{\text{VDA}}$ + $\mathcal{L}_{\text{SaSCon}}$ + Frame Aug.	0.140	0.809

Table 4. Accuracy with respect to varying loss configurations. We observe that our proposed $\mathcal{L}_{\text{SaSCon}}$ loss improves accuracy. Our final oVDA method with $\mathcal{L}_{\text{VDA}}$ and $\mathcal{L}_{\text{SaSCon}}$ loss in conjunction with frame augmentation yields best performance.

5.2. Comparison to VDA

The offline Video Depth Anything (VDA) [6] approach defines an upper bound for our method, illustrated in Tab. 5 Since VDA uses a fixed batch size (i.e. context window) of 32, we compare it to our $c = 32$ model. VDA exists in two variants, i.e. -small and -large, where the latter has about 13× more parameters. Note that our oVDA model builds upon VDA-s. oVDA lags behind VDA-s by only 2.2 pp in accuracy (δ_1), which can be expected since it operates without access to future frames. The difference between an offline and an online version can be seen in the latency. The latency of VDA-s is about 9× higher than that of oVDA.

5.3. Temporal Stability

Maintaining a temporal consistent scale is essential for online video depth estimation, as alignment typically requires either costly external measurements or heavy computations [6, 19, 26]. Fig. 4 visualises the scale drift over time by plotting the absolute relative difference between the optimal scale s_0 for the first frame 0 and the optimal scale s_i computed for each subsequent frame i: AbsRel = $\frac{1}{N_j}\sum_j \frac{|s_0 - s_j|}{s_0}$. We refer to N_j as data support (grey area in Fig. 4), defined as the number of sequences that contain at least j frames. We observe that our oVDA method consistently exhibits lower scale drift than all competitors, particularly for up to 300 frames. Beyond this point, FlashDepth and CUT3R approach the performance of oVDA, while NVDS and ChronoDepth continue to show a substantially higher drift. After 1000 frames, data support drops below 20 sequences, making further analysis less reliable.

Method	AbsRel (↓)	δ_1 (↑)	Latency [ms] (↓)
VDA-l [6]	0.127	0.860	579
VDA-s [6]	0.137	0.832	232
Ours ($c = 32$)	0.136	0.811	26

Table 5. Comparison to Video Depth Anything [6] at a resolution of 280×924. In contrast to VDA, our oVDA method has no access to future frames which reduces accuracy slightly. Since VDA predicts in batches, we provide the batch-latency of VDA. The latency of online oVDA is about 9× less than that of offline VDA-s.

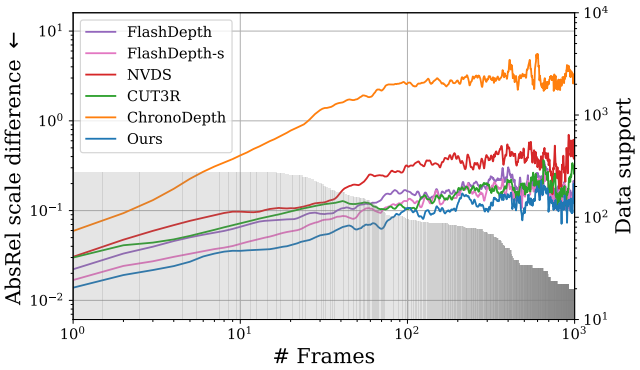

Figure 4. Scale drift over time for KITTI [12]. We plot the absolute relative difference between the optimal scale for the first frame and the optimal scale for each subsequent frame. The grey histogram gives the data support, i.e. the number of data points used to calculate the scale error, which decreases for later frames due to varying sequence lengths. We smoothed the scale error with a window size of 4 for better visualisation. Our oVDA method has lowest scale drift.

6. Conclusion and Limitations

We presented oVDA, a transformation of the offline VDA technique to the online setting, inspired by inference and training strategies used in LLMs. Extensive experiments demonstrate that oVDA achieves state-of-the-art accuracy and temporal consistency across both indoor and outdoor datasets, while significantly reducing memory usage. Furthermore, it achieves 42 FPS on an NVIDIA A100 and 20 FPS on an NVIDIA Jetson edge device. Nevertheless, oVDA still struggles with scale drift for extremely long video sequences. Moreover, fast-moving objects can produce so-called trailing artefacts, highlighting the need for further research, especially for safety-critical applications.

7. Acknowledgements

This project has been supported by HARMAN International.

References

[1] Andreas Blattmann, Tim Dockhorn, Sumith Kulal, Daniel Mendelevitch, Maciej Kilian, Dominik Lorenz, Yam Levi, Zion English, Vikram Voleti, Adam Letts, et al. Stable video diffusion: Scaling latent video diffusion models to large datasets. *arXiv preprint arXiv:2311.15127*, 2023. 2

[2] Andreas Blattmann, Robin Rombach, Huan Ling, Tim Dockhorn, Seung Wook Kim, Sanja Fidler, and Karsten Kreis. Align your latents: High-resolution video synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 22563–22575, 2023. 2

[3] Aleksei Bochkovskii, Amaël Delaunoy, Hugo Germain, Marcel Santos, Yichao Zhou, Stephan R. Richter, and Vladlen Koltun. Depth pro: Sharp monocular metric depth in less than a second. In *International Conference on Learning Representations*, 2025. 1

[4] D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black. A naturalistic open source movie for optical flow evaluation. In *European Conf. on Computer Vision (ECCV)*, pages 611–625. Springer-Verlag, 2012. 5, 6

[5] Yohann Cabon, Naila Murray, and Martin Humenberger. Virtual kitti 2, 2020. 4

[6] Sili Chen, Hengkai Guo, Shengnan Zhu, Feihu Zhang, Zilong Huang, Jiashi Feng, and Bingyi Kang. Video depth anything: Consistent depth estimation for super-long videos. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 22831–22840, 2025. 2, 3, 5, 7, 8

[7] Gene Chou, Wenqi Xian, Guandao Yang, Mohamed Abdelfattah, Bharath Hariharan, Noah Snavely, Ning Yu, and Paul Debevec. Flashdepth: Real-time streaming video depth estimation at 2k resolution. 2025. 2, 3, 5, 6, 7

[8] Tri Dao and Albert Gu. Transformers are ssms: generalized models and efficient algorithms through structured state space duality. In *Proceedings of the 41st International Conference on Machine Learning*. JMLR.org, 2024. 3

[9] Yue-Jiang Dong, Wang Zhao, Jiale Xu, Ying Shan, and Song-Hai Zhang. Depthsync: Diffusion guidance-based depth synchronization for scale-and geometry-consistent video depth estimation. *arXiv preprint arXiv:2507.01603*, 2025. 2

[10] David Eigen, Christian Puhrsch, and Rob Fergus. Depth map prediction from a single image using a multi-scale deep network. *Advances in neural information processing systems*, 27, 2014. 1, 2

[11] Xiao Fu, Wei Yin, Mu Hu, Kaixuan Wang, Yuexin Ma, Ping Tan, Shaojie Shen, Dahua Lin, and Xiaoxiao Long. Geowizard: Unleashing the diffusion priors for 3d geometry estimation from a single image. In *European Conference on Computer Vision*, pages 241–258. Springer, 2024. 2

[12] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012. 1, 5, 6, 7, 8

[13] Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. In *First Conference on Language Modeling*, 2024. 3

[14] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997. 2

[15] Wenbo Hu, Xiangjun Gao, Xiaoyu Li, Sijie Zhao, Xiaodong Cun, Yong Zhang, Long Quan, and Ying Shan. Depthcrafter: Generating consistent long depth sequences for open-world videos. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 2005–2015, 2025. 2, 5

[16] Bingxin Ke, Anton Obukhov, Shengyu Huang, Nando Metzger, Rodrigo Caye Daudt, and Konrad Schindler. Repurposing diffusion-based image generators for monocular depth estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024. 1, 2

[17] Bingxin Ke, Dominik Narnhofer, Shengyu Huang, Lei Ke, Torben Peters, Katerina Fragkiadaki, Anton Obukhov, and Konrad Schindler. Video depth without video models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025. 2

[18] Faisal Khan, Saqib Salahuddin, and Hossein Javidnia. Deep learning-based monocular depth estimation methods—a state-of-the-art review. *Sensors*, 20(8), 2020. 1

[19] Numair Khan, Eric Penner, Douglas Lanman, and Lei Xiao. Temporally consistent online depth estimation using point-based fusion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9119–9129, 2023. 2, 8

[20] Johannes Kopf, Xuejian Rong, and Jia-Bin Huang. Robust consistent video depth estimation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021. 2

[21] Vincent Leroy, Yohann Cabon, and Jerome Revaud. Grounding image matching in 3d with mast3r, 2024. 2

[22] Zhaoshuo Li, Wei Ye, Dilin Wang, Francis X. Creighton, Russell H. Taylor, Ganesh Venkatesh, and Mathias Unberath. Temporally consistent online depth estimation in dynamic scenes. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 3018–3027, 2023. 2, 6

[23] Hanxue Liang, Jiawei Ren, Ashkan Mirzaei, Antonio Torralba, Ziwei Liu, Igor Gilitschenski, Sanja Fidler, Cengiz Oztireli, Huan Ling, Zan Gojcic, and Jiahui Huang. Feed-forward bullet-time reconstruction of dynamic scenes from monocular videos. 2024. 2

[24] Akide Liu, Jing Liu, Zizheng Pan, Yefei He, Gholamreza Haffari, and Bohan Zhuang. Minicache: Kv cache compression in depth dimension for large language models. *Advances in Neural Information Processing Systems*, 37: 139997–140031, 2024. 2

[25] Fayao Liu, Chunhua Shen, and Guosheng Lin. Deep convolutional neural fields for depth estimation from a single image. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5162–5170, 2015. 2

[26] Jiahao Lu, Tianyu Huang, Peng Li, Zhiyang Dou, Cheng Lin, Zhiming Cui, Zhen Dong, Sai-Kit Yeung, Wenping Wang, and Yuan Liu. Align3r: Aligned monocular depth estimation for dynamic videos. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 22820–22830, 2025. 8

[27] Xuan Luo, Jia-Bin Huang, Richard Szeliski, Kevin Matzen, and Johannes Kopf. Consistent video depth estimation. 39 (4), 2020. 2

[28] Alican Mertan, Damien Jade Duff, and Gozde Unal. Single image depth estimation: An overview. *Digital Signal Processing*, 123:103441, 2022. 2

[29] Maxime Oquab, Timothée Darcet, Theo Moutakanni, Huy V. Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Russell Howes, Po-Yao Huang, Hu Xu, Vasu Sharma, Shang-Wen Li, Wojciech Galuba, Mike Rabbat, Mido Assran, Nicolas Ballas, Gabriel Synnaeve, Ishan Misra, Herve Jegou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. Dinov2: Learning robust visual features without supervision, 2023. 2

[30] E. Palazzolo, J. Behley, P. Lottes, P. Giguère, and C. Stachniss. ReFusion: 3D Reconstruction in Dynamic Environments for RGB-D Cameras Exploiting Residuals. In *iros*, 2019. 5, 6

[31] Vaishakh Patil, Wouter Van Gansbeke, Dengxin Dai, and Luc Van Gool. Don't forget the past: Recurrent depth estimation from monocular video. *IEEE Robotics and Automation Letters*, 5(4):6813–6820, 2020. 2

[32] Jordi Pont-Tuset, Federico Perazzi, Sergi Caelles, Pablo Arbeláez, Alexander Sorkine-Hornung, and Luc Van Gool. The 2017 davis challenge on video object segmentation. *arXiv:1704.00675*, 2017. 6, 7

[33] Reiner Pope, Sholto Douglas, Aakanksha Chowdhery, Jacob Devlin, James Bradbury, Jonathan Heek, Kefan Xiao, Shivani Agrawal, and Jeff Dean. Efficiently scaling transformer inference. *Proceedings of machine learning and systems*, 5: 606–624, 2023. 2

[34] René Ranftl, Katrin Lasinger, David Hafner, Konrad Schindler, and Vladlen Koltun. Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(3), 2022. 1, 2, 5

[35] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022. 2

[36] Jiahao Shao, Yuanbo Yang, Hongyu Zhou, Youmin Zhang, Yujun Shen, Vitor Guizilini, Yue Wang, Matteo Poggi, and Yiyi Liao. Learning temporally consistent video depth from video diffusion priors, 2024. 2, 3, 5, 6, 7

[37] Zachary Teed and Jia Deng. Droid-slam: Deep visual slam for monocular, stereo, and rgb-d cameras. *Advances in neural information processing systems*, 34:16558–16569, 2021. 2

[38] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017. 2, 3, 4

[39] Qiang Wang, Shi gang Zheng, Qingsong Yan, Fei Deng, Kaiyong Zhao, and Xiaowen Chu. Irs: A large synthetic indoor robotics stereo dataset for disparity and surface normal estimation. *ArXiv*, abs/1912.09678, 2019. 4

[40] Qianqian Wang, Yifei Zhang, Aleksander Holynski, Alexei A Efros, and Angjoo Kanazawa. Continuous 3d perception model with persistent state. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 10510–10522, 2025. 2, 3, 5, 6, 7

[41] Shuzhe Wang, Vincent Leroy, Yohann Cabon, Boris Chidlovskii, and Jerome Revaud. Dust3r: Geometric 3d vision made easy. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 20697–20709, 2024. 2

[42] Wenshan Wang, Delong Zhu, Xiangwei Wang, Yaoyu Hu, Yuheng Qiu, Chen Wang, Yafei Hu, Ashish Kapoor, and Sebastian Scherer. Tartanair: A dataset to push the limits of visual slam. 2020. 4

[43] Yiran Wang, Min Shi, Jiaqi Li, Zihao Huang, Zhiguo Cao, Jianming Zhang, Ke Xian, and Guosheng Lin. Neural video depth stabilizer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9466–9476, 2023. 2, 3, 5, 6, 7

[44] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online, 2020. Association for Computational Linguistics. 2, 3

[45] Honghui Yang, Di Huang, Wei Yin, Chunhua Shen, Haifeng Liu, Xiaofei He, Binbin Lin, Wanli Ouyang, and Tong He. Depth any video with scalable synthetic data. In *The Thirteenth International Conference on Learning Representations*, 2025. 2

[46] Lihe Yang, Bingyi Kang, Zilong Huang, Xiaogang Xu, Jiashi Feng, and Hengshuang Zhao. Depth anything: Unleashing the power of large-scale unlabeled data. In *CVPR*, 2024. 1, 2

[47] Lihe Yang, Bingyi Kang, Zilong Huang, Zhen Zhao, Xiaogang Xu, Jiashi Feng, and Hengshuang Zhao. Depth anything v2. *Advances in Neural Information Processing Systems*, 37:21875–21911, 2024. 1, 3, 5, 6

[48] Nan Yang, Lukas von Stumberg, Rui Wang, and Daniel Cremers. D3vo: Deep depth, deep pose and deep uncertainty for monocular visual odometry. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1281–1292, 2020. 2

[49] Rajeev Yasarla, Hong Cai, Jisoo Jeong, Yunxiao Shi, Risheek Garrepalli, and Fatih Porikli. Mamo: Leveraging memory and attention for monocular video depth estimation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8754–8764, 2023. 3

[50] Denis Zavadski, Damjan Kalšan, and Carsten Rother. Primedepth: Efficient monocular depth estimation with a stable diffusion preimage. In *Proceedings of the Asian Conference on Computer Vision*, pages 922–940, 2024. 1, 2

[51] Haokui Zhang, Chunhua Shen, Ying Li, Yuanzhouhan Cao, Yu Liu, and Youliang Yan. Exploiting temporal consistency for real-time video depth estimation. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 1725–1734, 2019. 2

[52] Junyi Zhang, Charles Herrmann, Junhwa Hur, Varun Jampani, Trevor Darrell, Forrester Cole, Deqing Sun, and Ming-Hsuan Yang. Monst3r: A simple approach for estimating geometry in the presence of motion. *arXiv preprint arxiv:2410.03825*, 2024. 2

[53] Zhoutong Zhang, Forrester Cole, Richard Tucker, William T Freeman, and Tali Dekel. Consistent depth of moving objects in video. *ACM Transactions on Graphics (TOG)*, 40(4):1–12, 2021. 2

[54] Yang Zheng, Adam W. Harley, Bokui Shen, Gordon Wetzstein, and Leonidas J. Guibas. Pointodyssey: A large-scale synthetic dataset for long-term point tracking. In *ICCV*, 2023. 4

[55] Dong Zhuo, Wenzhao Zheng, Jiahe Guo, Yuqi Wu, Jie Zhou, and Jiwen Lu. Streaming 4d visual geometry transformer. *arXiv preprint arXiv:2507.11539*, 2025. 2