# DINO-VO: Learning Where to Focus for Enhanced State Estimation

Qi Chen[1,*] Guanghao Li[1,2,*] Sijia Hu[1] Xin Gao[1] Junpeng Ma[1]
Xiangyang Xue[1] Jian Pu[1,✉]
[1]Fudan University [2]Shanghai Innovation Institute
{qichen21, ghli22, sjhu23, gaoxin23, jpma24}@m.fudan.edu.cn
{xyxue, jianpu}@fudan.edu.cn
*Equal contribution

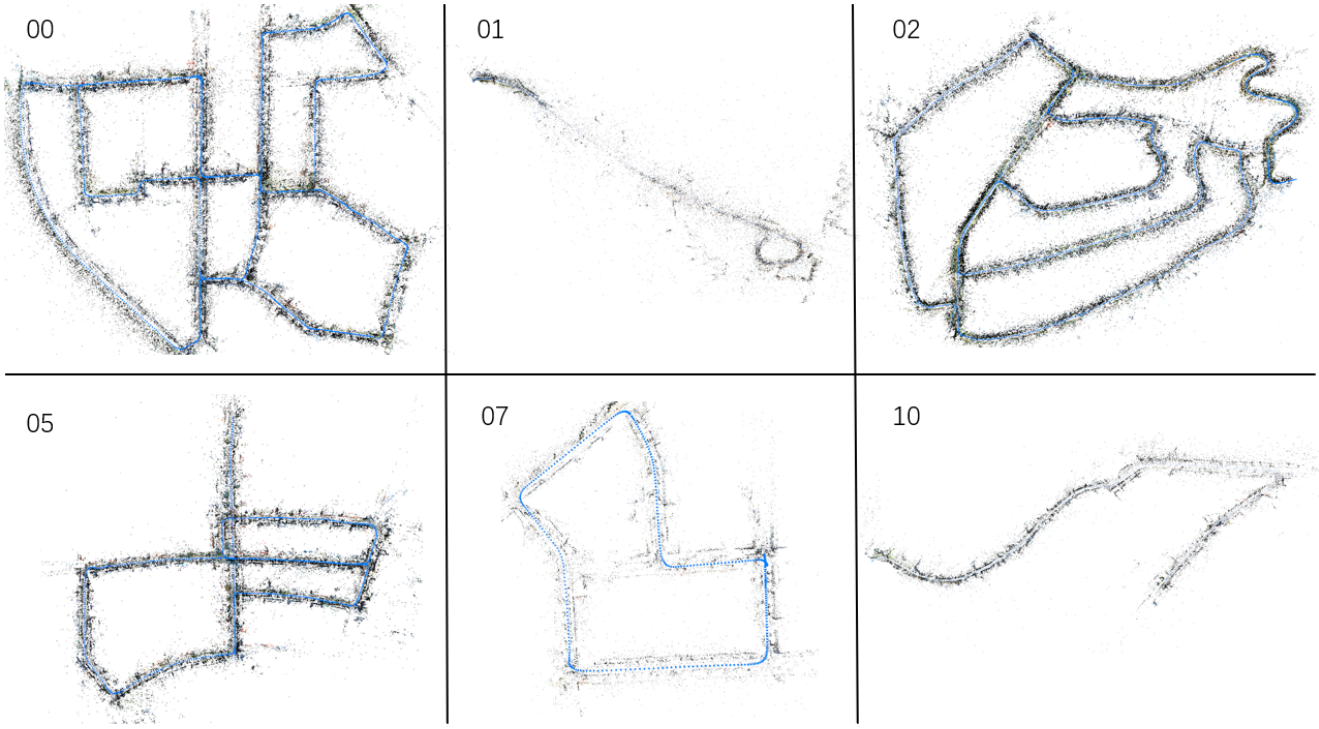


Figure 1. Visualization results on the KITTI dataset.

## 1. Visualization on KITTI

Several mapping visualization results on the KITTI dataset of DINO-VO with loop closure mechanism are shown in Fig.1.

## 2. Efficiency

### 2.1. Memory Usage

To assess the memory usage of our system, we measure the VRAM consumption of DINO-VO and compare it to DPVO[7] using the command $nvidia-smi$. The results are shown in Fig.2 on the EuRoC [1] dataset with the default configuration introduced in the paper. The findings reveal that our 4-head multi-task feature extractor does not consume more VRAM than the 2-backbone feature extractor model proposed in DPVO [7], demonstrating the efficiency of our design.

### 2.2. Running Time

We also evaluate the Frames Per Second (FPS) performance of DINO-VO and compare it to DPVO[7] on a machine equipped with 64GB RAM, an Intel Core i7-12700KF CPU, and an NVIDIA RTX 3090 GPU. Using the EuRoC[1] dataset, DROID [6] achieves a frame rate of 21 FPS, while

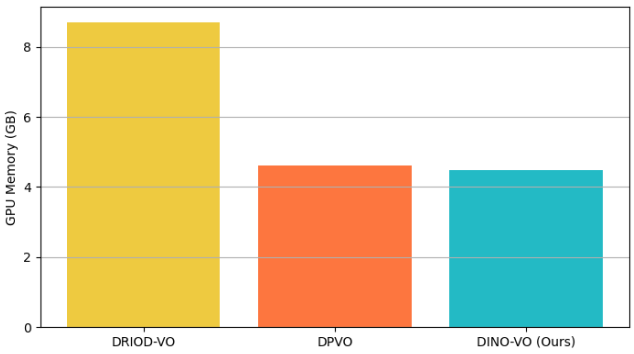


Figure 2. Memory usage of DINO-VO compared to DROID-VO[6] and DPVO[7], measured on the EuRoC [1] dataset. DROID-VO uses 8.7GB, DPVO (Default) uses 4.6GB and our system uses 4.4GB.

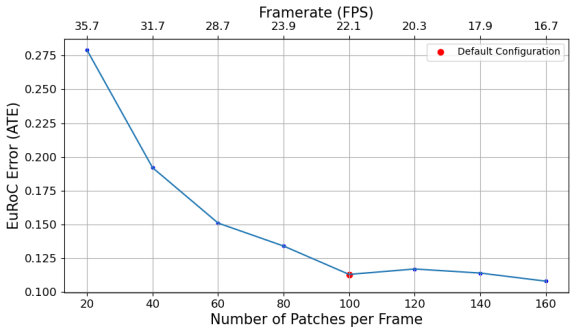


Figure 3. Accuracy on the EuRoC [1] dataset as a function of the number of patches. To balance the accuracy and efficiency, we set 100 patches per image as our default configuration.

DPVO runs at 25 FPS. In contrast, our method operates at 22 FPS, demonstrating that while DINO-VO is slightly slower than DPVO, it remains highly competitive for real-time applications.

## 3. Additional Details

Similar to DPVO [7], our model is trained exclusively on the TartanAir dataset. The feature maps created by heads are segmented during training into m-by-m patches, with m set to 14. This choice aligns with the design of our backbone model, DINOv2 [5], which divides each image frame into 14-by-14 patches. The feature dimension in the residual blocks of the frame fusion layers is set to 64. We balance performance and efficiency for the prior weight head by setting the feature dimension of residual blocks to 32.

During the inference phase, $\boldsymbol{W}_j^{p'}$ is divided into $n \times n$ pixel regions, and the position of the maximum prior weight within each region is identified. These positions are then ranked by their corresponding weights, and the top $N$ positions are selected as the final patches. By default, we set $N = 100$. If the number of regions is fewer than $N$ due to a small input image size, $n$ is reduced to ensure the number of regions exceeds $N$. Conversely, $n$ is increased if the number of regions is too large. In summary, we aim to keep the number of regions as close as possible to $N$.

### 3.1. Limitations

DINO-VO performs well in many scenarios, achieving a good balance between efficiency and accuracy. However, several challenges remain unresolved. The first issue concerns the influence of the number of patches extracted from each image frame on the system's performance. As shown in Fig. 3, our method achieves optimal results with the default configuration on the EuRoC dataset [1], but the performance degrades when the number of selected patches is reduced. In future work, we aim to improve the patch selection strategy during inference to reduce the number of extracted patches while preserving optimal accuracy. The second issue involves integrating a traditional loop closure mechanism, as proposed in [4], to support large-scale scenario mapping. This mechanism relies on a bag-of-words approach and requires additional ORB feature points for loop detection. It also uses off-the-shelf keypoint detectors [2] and matchers [3] to estimate poses between image pairs. While effective, incorporating this module significantly slows down the system during mapping tasks. In future work, we plan to simplify the loop closure mechanism by consolidating it into a unified deep-learning model to enhance efficiency.

## References

[1] Michael Burri, Janosch Nikolic, Pascal Gohl, Thomas Schneider, Joern Rehder, Sammy Omari, Markus W Achtelik, and Roland Siegwart. The euroc micro aerial vehicle datasets. *The International Journal of Robotics Research*, 35(10):1157–1163, 2016. 1, 2

[2] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superpoint: Self-supervised interest point detection and description. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 224–236, 2018. 2

[3] Philipp Lindenberger, Paul-Edouard Sarlin, and Marc Pollefeys. Lightglue: Local feature matching at light speed. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 17627–17638, 2023. 2

[4] Lahav Lipson, Zachary Teed, and Jia Deng. Deep patch visual slam. In *European Conference on Computer Vision (ECCV)*, pages 424–440. Springer, 2025. 2

[5] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *Transactions on Machine Learning Research Journal*, pages 1–31, 2024. 2

[6] 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. 1, 2

[7] Zachary Teed, Lahav Lipson, and Jia Deng. Deep patch visual odometry. *Advances in Neural Information Processing Systems*, 36, 2024. 1, 2