# Improving DINOv2 Monocular Metric Depth Estimates by Smoothing Over Consecutive Frames

Erik Jagnandan
*University of Pennsylvania*
ejag@seas.upenn.edu

Keyush Shah
*University of Pennsylvania*
keyush06@seas.upenn.edu

## I. Abstract

We introduce a method for improving the metric depth estimates produced by Meta's vision foundation model, DINOv2, as applied to video stream data. This approach involves combining the DINO features (produced by the DINO encoder) from consecutive video frames using a zero-initialized, two-layer Convolutional Neural Network (CNN) placed between the DINO encoder and the downstream depth adapter. This CNN receives the DINO features from the previous and current frames as input, along with the horizontal and vertical pixel shift between the two frames (to make the task of aligning and subsequently combining the features easier). The resulting set of DINO features are added to the original DINO features from the current frame via a skip connection and then input to the downstream depth adapter, which produces the output depth map. We find that our approach reduces the MSE incurred by DINOv2 by 23.8 percent on the NYUv2 Depth dataset, a widely used dataset for depth estimation which contains images of indoor scenes and their corresponding depth maps. Additionally, through visualizations of the DINO features both before and after our two-layer CNN, as well as comparisons of the output depth maps with and without applying our approach, we observe that our approach is not altering the fundamental structure of the DINO features or of the output depth map. However, we do observe that the depth map with our approach is more accurately scaled to the ground truth, which our approach accomplishes by smoothing over the errors in depth estimation across consecutive frames. We believe that this improvement in scaling is primarily responsible for the reductions in MSE achieved by our approach.

## II. Introduction and Related Work

Following the transformative impact provided by large pretrained models in natural language processing (NLP), similar advancements have been achieved by applying large-scale models to computer vision tasks. One such model is DINOv2, a vision transformer introduced by Meta AI Research, which has achieved state-of-the-art performance in image classification, semantic segmentation, instance retrieval, and monocular depth estimation. In order to achieve each of these downstream tasks, a task specific adapter is placed downstream of the pretrained DINOv2 encoder, and trained on task-specific data [1].

In this paper, we examine DINOv2's ability to perform monocular depth estimation. While alternative methods like LiDAR and multi-view stereopsis offer reliable depth estimation, they come with significant drawbacks. LiDAR systems, for instance, are substantially more expensive than standard cameras [2] and can consume up to ten times more power when integrated into autonomous vehicles [3]. Multi-view stereopsis, which requires a second camera, increases cost, engineering complexity, and weight, making it impractical for lightweight platforms such as aerial drones. Accurate monocular depth estimation has the potential to enhance navigation capabilities in robotics and autonomous driving without these limitations.

In addition to DINOv2, models such as DepthAnything and MIDAS have been created for monocular depth estimation. DepthAnything employs a vision transformer architecture and is trained using a data engine that automatically labels a large-scale, initially unlabeled dataset. It also incorporates auxiliary supervision to provide its model with semantically rich priors from its pretrained encoders [4]. MIDAS, which originally relied on a convolutional-based design, has transitioned to using vision transformers such as BEiT, Swin, and Next-ViT as its backbone [5]. A common issue across these models is that, like DINOv2, while their relative depth performance (distinguishing foreground from background) is strong, their metric depth performance (predicting the exact depth at each pixel) is comparatively weak [4] [5], leaving room for improvement.

Several models have specifically targeted improvements in metric depth estimation. For example, ZeroDepth employs a large transformer-based encoder-decoder architecture, achieving strong results across several depth datasets including KITTI and NYUv2 [6]. However, its reliance on a large network significantly slows inference, making it unsuitable for real-time applications such as robotics and autonomous driving. While matching ZeroDepth's accuracy is challenging, our objective is to develop an approach that significantly improves the accuracy of DINOv2's metric depth estimates, while remaining lightweight enough for real-time deployment.

There also exist recent approaches similar to ours which aim to modify the pipeline of an existing depth estimation model to improve its performance, rather than develop an entirely new model. One such approach is Radar Meets Vision [7], which improves the absolute relative error of DepthAnythingV2 by 9 to 64 percent (depending on the

type of environment) by incorporating radar data into the DepthAnythingV2 pipeline. While this improvement in error is substantial and the mmWave radar used in this approach can be cheaply collected on a single-chip, a major motivation of monocular depth estimation is to estimate depth strictly from the visual data provided by a single camera. Thus, we aim to improve the performance of DINOv2 using an approach which does not rely on any additional input data streams.

We summarize our contributions as follows:

1) Identifying that the errors made by the DINOv2 depth estimation pipeline can be reduced by combining either the output depth estimates (Iterations 0 and 1) or the DINO features (Iteration 2) from consecutive frames
2) Developing a simple approach based on phase correlation for aligning and averaging the output depth maps of the DINOv2 depth estimation pipeline, which reduces MSE on the NYUv2 dataset by 3.8 percent while incurring minimal additional latency ($\sim$3 ms)
3) Extending our approach by inserting a small CNN between the DINO encoder and depth adapter which receives the DINO features for the current and previous frame, as well as the horizontal and vertical shift between these frames, to produce an improved set of DINO features. Applying this CNN-based approach resulted in a 23.8 percent reduction in the MSE incurred by DINOv2

## III. Background

### A. Phase Correlation

Phase correlation is a signal processing technique used to estimate the displacement between two image signals by analyzing the difference in their phase in the frequency domain. It first involves transforming the two images into the frequency domain using the Fourier Transform. In the frequency domain, spatial shifts between the images appear as linear phase differences between the Fourier coefficients of the two images. Then the normalized cross-power spectrum of the two Fourier Transforms is computed, in which the Fourier Transform of one image is multiplied by the complex conjugate of the Fourier Transform of the other, and the result is divided by its magnitude to normalize it. Finally, the Inverse Fourier Transform is applied to the normalized cross-power spectrum to convert it back to the spatial domain. In this spatial domain image, a sharp peak in the signal will be observed, and the pixel location of this peak is indicative of the shift between the two images. For example, if the peak occurs at pixel (5, 10), this corresponds to a shift of 5 pixels vertically and 10 pixels horizontally.

## IV. Approach

### A. Dataset

The train split of the NYUv2 Depth dataset contains $\sim$400k images of indoor scenes and corresponding depth maps. Due to memory space limitations, we select 50k of these images for our work. Notably, the images in the NYUv2 dataset are frames recorded from video streams taken in various kinds of indoor environments, such as bedrooms, basements, or cafes, and the images from each stream are provided in order. This allows us to leverage our approach in which we combine images from consecutive frames. However, the test split contains a selection of unordered image frames in which the frames from different scenes are randomly interleaved (e.g. a random frame from an office is followed by a random frame from a kitchen). Therefore, we cannot apply our method to the test dataset since we cannot identify consecutive frames.

To resolve this, we treat our 50k frames from the train split of the NYUv2 Depth dataset as our entire dataset, and perform a training-validation split in which we allocate $\sim$40k image frames to the training set and $\sim$10k image frames to the validation set. This may arouse concern that for each frame allocated to the validation set, the subsequent frame cannot be used as training data, as this would require using the preceding validation frame during training, which would amount to data leakage between the training and validation sets. However, note that this does not require using the ground truth label corresponding to the validation frame, as only the ground truth label for the current training frame is used. Since the model is not exposed to the validation labels during training, it has no opportunity to fit to those labels, and thus validation frames may be used as the preceding frame during training without concern of data leakage. This same reasoning can be applied in reverse to justify usage of training frames as the preceding frame during validation, as only the previously unseen ground truth validation labels are used during validation, and not the training labels which the model previously fit to.

### B. Broad Approach

Note that all of our experiments apply our methods on top of DINOv2 with the small backbone (DINO encoder) size and the DPT decoder.

Our approach is broadly guided by the concept that for a camera recording video at real time speeds (25-30 FPS), the camera moves minimally between consecutive frames and thus most parts of the environment which are seen at one frame are also seen at the previous frame. DINOv2 does not leverage this fact, as it predicts a depth map for each input image independently without incorporating any information seen previously. Thus, for objects seen in each of several consecutive frames, DINOv2 independently predicts their depth multiple times. We can treat these independent predictions as if they came from an ensemble of models, and average over them to compute a depth estimate which, on average, achieves a lower MSE than any individual depth estimate.

### C. Iteration 0

A basic way of smoothing over consecutive frames would be to average the output depth map at each frame with the depth map at the previous frame. However, the two depth maps cannot be averaged directly, as their content is slightly

shifted due to the movement of the camera. Therefore, the shift between the two images must be computed. Our first method of accomplishing this involved computing correspondences between consecutive images using SIFT features, and calculating the average pixel displacement in x and y between the features at the current frame with the corresponding features at the previous frame. This allowed us to shift the previous frame to align its content with the current frame and then average the predicted depth maps.

As we experimented with this approach, we found that our SIFT-based pipeline was too slow, taking $\sim$1 second per frame. Since the full DINO depth estimation pipeline with the small backbone size and DPT decoder requires $\sim$150 ms per frame, an additional $\sim$1000 ms (equivalent to 1000 validation samples at the inference time) is an unacceptable amount of additional latency for a small improvement in MSE. As a result, we experimented with other feature algorithms, ultimately finding that all algorithms provided similar improvement in MSE over the vanilla DINO model, but that ORB features achieved the fastest performance at approximately $\sim$290 ms of additional latency. Our ORB correspondence-based approach reduced the MSE of DINOv2 from $0.1865$ to $0.1770$, an improvement of $5.1$ percent. We further optimized the performance of our ORB-based pipeline by introducing a threshold on the confidence score for each feature correspondence, which significantly sped up our pipeline (reduced additional latency to $\sim$85 ms) by eliminating extra computation while only worsening performance slightly. Specifically, the MSE rose from $0.1770$ to $0.1796$ when introducing this threshold, improving on the MSE of vanilla DINOv2 by 3.7 percent as opposed to $5.1$ percent. We refer to these two approaches as "ORB Correspondence Accuracy" and "ORB Correspondence Speed".

Notably, the alignment provided by averaging the pixel displacement between correspondences in the two images is imperfect, as the depth and perspective distortion vary throughout the image, causing the locations of different points in the environment (and thus the corresponding ORB features) to move differently in pixel space as the camera translates. For example, points at more distant objects will move less in the pixel space than points at nearby objects, given the same translation of the camera. Moreover, when aligning the frames at the pixel level, we must round the pixel shift to the nearest whole pixel, so the content will not align perfectly. Because of this imperfect alignment, the aligned depth map from the previous frame is less reliable than the original depth map from the current frame, so the original depth map should be weighted more heavily in the average. Accordingly, we weight the original depth map for the current frame by a parameter $\rho$ and the aligned depth map for the previous frame by $1-\rho$. Through manual tuning on training data, we selected a value of $\rho = 0.83$.

### D. Iteration 1

After establishing a baseline approach in Iteration 0, we considered alternative methods of aligning consecutive frames and discovered that phase correlation can provide very fast results ($\sim$3 ms of additional latency) while also achieving similar performance to our speed-optimized approach from Iteration 0. Specifically, our approach using phase correlation achieves an MSE of $0.1794$, a $3.8$ percent reduction in MSE relative to vanilla DINOv2. As phase correlation operates on grayscale images, we converted the current and previous frames into grayscale and used OpenCV's phase_correlation() function to compute the vertical and horizontal shift between the two frames, allowing us to align them. As with our ORB correspondence-based approach from Iteration 0, the alignment of the depth map from the previous frame with the depth map from the current frame is imperfect, largely due to the need to round the depth map to the nearest whole for pixel-level alignment. Therefore, the aligned depth map from the previous frame is less reliable and consequently downweighted in the average by multiplying by $1-\rho$, while the depth map from the current frame should be upweighted by multiplying by $\rho$. As before, we tuned $\rho$ via manual tuning and selected a value of $\rho = 0.83$.

### E. Iteration 2

Due to the inaccuracies resulting from perspective distortion and rounding pixel shifts which arise when aligning and manually averaging the depth maps in Iterations 0 and 1, we considered that using a machine learning-based approach to combining the depth maps might provide superior performance. While such an approach could be applied on the pixel-level depth maps output by the depth adapter, we chose to apply our model in the DINO feature space in order to operate on compressed features and therefore achieve greater computational efficiency. Moreover, since the original DINO features at the current frame are already a good starting point for accurate depth estimation, we implemented our model as a zero-initialized CNN with a skip connection propagating the original DINO features of the current frame to the output of the CNN. Thus, at the beginning of training, our CNN would output a tensor of all zeros, which would be added to the original DINO features of the current frame. Thus, the same original DINO features would be input to the depth adapter. As training progresses, the model would learn to compute the appropriate modification to be added to the current DINO features to produce an improved set of combined features to be passed to the depth adapter.

Note that if our model only received the original DINO features at the current and past frames, the task of aligning them correctly would be very difficult, as the model would have to identify the shift in the DINO features and then combine them accordingly. To make the task much easier and thus avoid requiring a large, computationally slower model, we use phase correlation to compute the the vertical and horizontal shift between the previous and current frame and provide it as additional input to our model. Note that the DINO features for a single frame are of shape H/patch size $\times$ W/patch size $\times$ channel dim, so we achieve this by creating a feature map of shape H/patch size $\times$ W/patch size in which

every value is equal to the horizontal shift computed by the phase correlation, creating an identically-shaped feature map in which every value is equal to the vertical shift computed by phase correlation, and concatenating these feature maps with the DINO features along the channel dimension to create an input with dimensions H/patch size $\times$ W/patch size $\times$ 2*channel dim+2. Our model, which we implement as a 2-layer CNN with $3\times3$ kernels and stride of 1, receives this input and produces an output of shape H/patch size $\times$ W/patch size $\times$ channel dim which is added to the original DINO features via skip connection before being input to the depth adapter.

## V. Results

### A. Iterations 0-1

We summarize the performance of our approaches involving manual averaging of the output depth maps in Figure 1. The speed-optimized version of our ORB correspondence-based approach ("ORB Correspondence Speed") improves substantially on the accuracy-optimized version ("ORB Correspondence Accuracy") in terms of speed, reducing additional latency from $\sim$290 ms to $\sim$85 ms, while sacrificing some performance, as the reduction in MSE goes from 5.1 percent to 3.7 percent. However, our phase correlation-based approach (Iteration 1) achieves the best combination of speed and accuracy, incurring minimal additional latency ($\sim$3 ms) over vanilla DINOv2 while providing 3.8 percent reduction in MSE. We also provide the speed and accuracy performance of vanilla DINOv2 with the base backbone size (0.1479 MSE at $\sim$240 ms inference time) as a reference point. Ideally, we would like to provide an approach building upon vanilla DINOv2 with the small backbone size which provides a better tradeoff of speed and accuracy than simply increasing to the base backbone size. Specifically, such an approach would achieve equal or lower MSE than vanilla DINOv2 with the base backbone size while achieving faster inference. However, the small improvement in MSE currently provided by phase correlation is far less than that achieved by increasing to the base backbone size, motivating further improvement in Iteration 2.

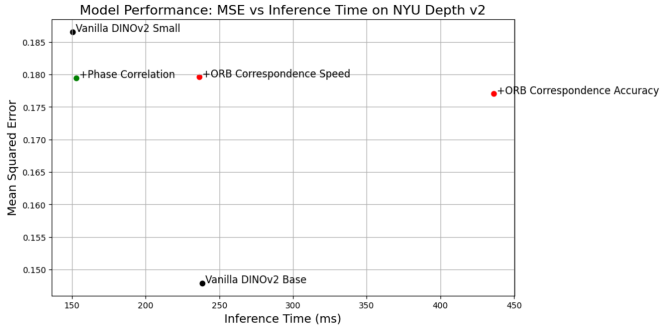


Fig. 1: Inference Time for Processing One Image versus MSE for Each Approach

### B. Iteration 2

After 10 epochs of training, our model for Iteration 2 achieves an MSE of 0.1421, a $\sim$23.8 percent reduction in MSE relative to the vanilla DINOv2 model with small backbone size. Notably, this performance also exceeds that of the vanilla DINOv2 model with base backbone size, constituting a $\sim$3.9 percent reduction in MSE relative to this model. We have yet to evaluate the speed of our approach relative to vanilla DINOv2, so we present the plot of the validation MSE achieved by our model over the training epochs in Figure 2, with the validation MSE achieved by vanilla DINOv2 with small and base backbone size plotted for reference.

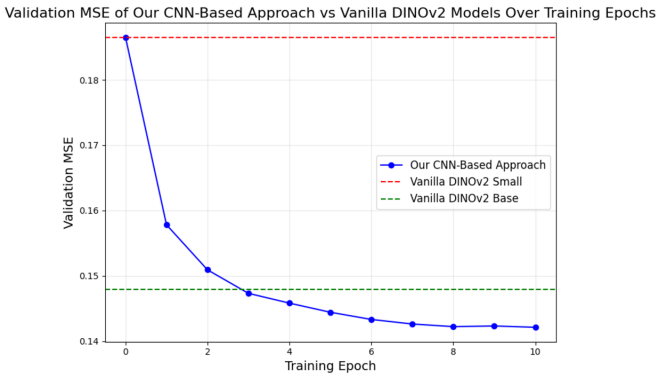


Fig. 2: Validation MSE Achieved by Our CNN-Based Approach over Training Epochs vs Vanilla DINOv2 Small and Base

### C. Feature Visualizations

Since our approach in Iterations 0 and 1 uses a non-ML-based method applied only at the output, it does not alter any of the intermediate features of the DINO encoder or depth adapter. However, as our approach in Iteration 2 does alter the DINO features between the DINO encoder and depth adapter, we can use feature visualization to examine how the DINO features output by our CNN have been changed relative to the DINO features originally produced by the DINO encoder. Figures 4 and 5 display NCUT visualizations of the DINO features for the same input image (shown in Figure 3) before and after being modified by our CNN. While there has been a change to the features, as expected, the underlying structure depicted in the features is consistent before and after being passed through our CNN. Thus, it appears that the reduction in MSE achieved by our approach cannot be primarily attributed any changes in structure. Our visualizations of the corresponding output depth maps for both vanilla DINOv2 and DINOv2 with our CNN are shown in Figures 6 and 7, respectively. Similarly, we observe that the overall structure of the output depth maps remains consistent, but, as indicated by the colorbar to the right of each depth map, the scale of the updated depth map has been shifted closer to the scale of the ground truth depth map. This suggests that our approach is reducing MSE because it has made the scale of each depth map more accurate to the ground truth by smoothing across the errors in depth estimation at each frame.

Fig. 3: Input RGB Image from NYUv2 Depth Dataset

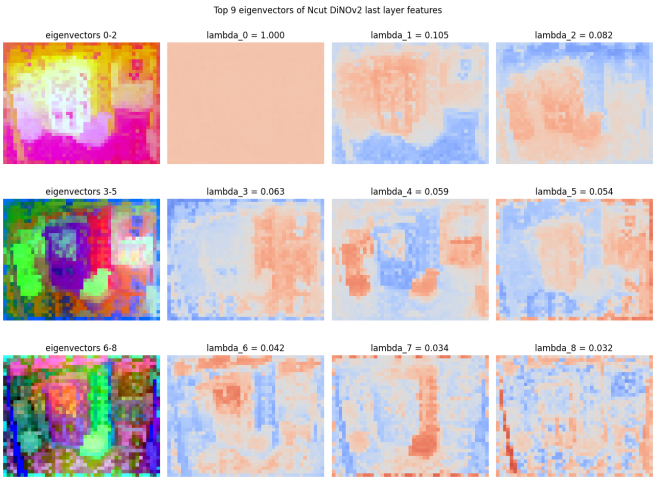


Fig. 4: NCUT Visualizations of DINO Features Originally Produced by the DINO Encoder for the Above Image

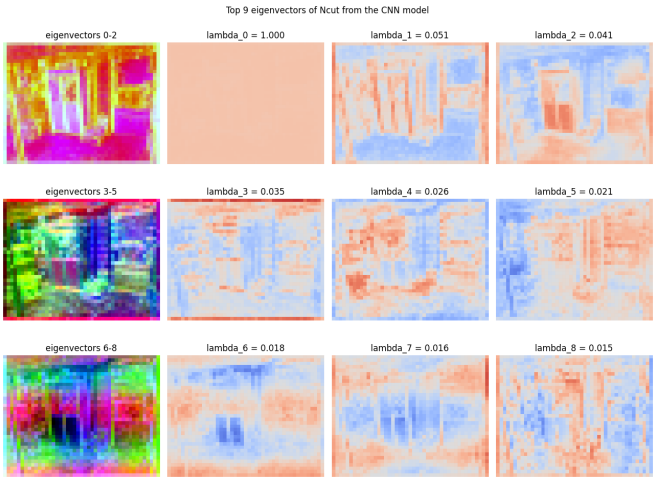


Fig. 5: NCUT Visualizations of DINO Features After Being Modified by our CNN

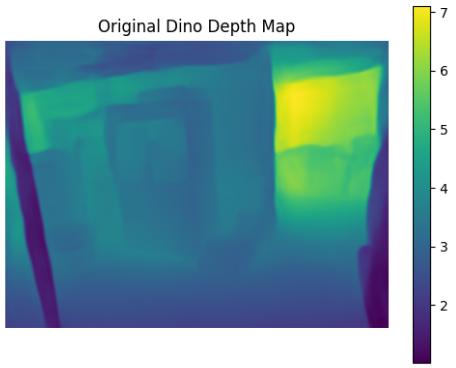


Fig. 6: Original Output Depth Map Produced by Vanilla DINOv2 with Small Backbone Size, Depth in Meters Indicated by Colorbar to the Right

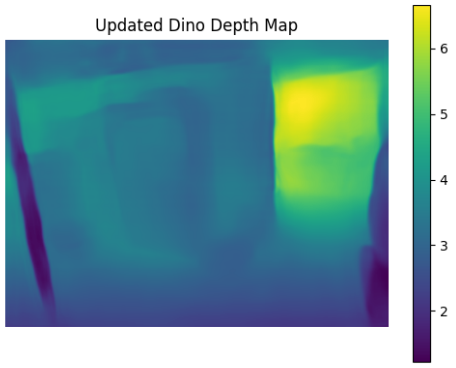


Fig. 7: Updated Output Depth Map as Produced with our CNN when Applied to DINOv2 with Small Backbone Size, Depth in Meters Indicated by Colorbar to the Right

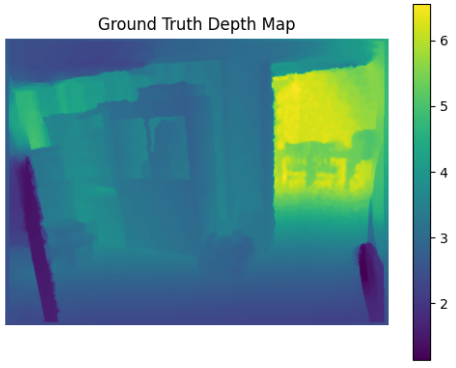


Fig. 8: Ground Truth Depth Map, Depth in Meters Indicated by Colorbar to the Right

## VI. Conclusions and Future Iterations

Our work demonstrates that the DINOv2 depth estimator can be improved by combining the depth estimates at consecutive frames and thus smoothing over the errors at each frame. We show that a non-ML-based approach leveraging phase correlation to align and average consecutive predicted depth maps can achieve a small reduction in MSE with very little additional latency. However, the highest performing approach we have identified thus far is to apply a zero-initialized CNN between the DINO encoder and depth adapter which computes one set of DINO features given the DINO features for two consecutive frames and the pixel shift between them, as computed by phase correlation.

As we continue our work, we aim to first perform a speed evaluation for our approach for Iteration 2, and then attempt Iteration 3, in which we will replace the CNN used in Iteration 2 with a transformer block. We anticipate that the transformer may provide additional reduction in the MSE, but will likely introduce additional latency, as compared to our CNN in Iteration 2. If so, we will analyze the tradeoff between speed and accuracy between our approaches for Iterations 2 and 3.

One limitation which we are presently aware of is that in order for our approach to provide reduction in the MSE by smoothing over consecutive frames, the underlying model must make varied estimates of the depth to each object at each frame. If the model makes identical mistakes in its depth estimates at each frame, then combining the depth estimates will not lead to any improvement. Additionally, we observe a slight reduction in resolution in the output depth maps when applying our CNN-based approach. This may be because our model, being trained on MSE, focuses primarily on making large depth estimation errors smaller, and low resolution around the edges of objects may only lead to errors which are small in magnitude and contribute minimally to the MSE.

## References


[1] M. Oquab, T. Darcet, T. Moutakanni, H. Vo, M. Szafraniec, V. Khalidov, P. Fernandez, D. Haziza, F. Massa, A. El-Nouby, M. Assran, N. Ballas, W. Galuba, R. Howes, P.-Y. Huang, S.-W. Li, I. Misra, M. Rabbat, V. Sharma, G. Synnaeve, H. Xu, H. Jegou, J. Mairal, P. Labatut, A. Joulin, and P. Bojanowski, "Dinov2: Learning robust visual features without supervision," 2024.

[2] A. Broggi, P. Grisleri, and P. Zani, "Sensors technologies for intelligent vehicles perception systems: A comparison between vision and 3d-lidar," in *16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013)*, pp. 887–892, 2013.

[3] Z. Liu, H. Tan, X. Kuang, H. Hao, and F. Zhao, "The negative impact of vehicular intelligence on energy consumption," *Journal of Advanced Transportation*, vol. 2019, 2019.

[4] L. Yang, B. Kang, Z. Huang, X. Xu, J. Feng, and H. Zhao, "Depth anything: Unleashing the power of large-scale unlabeled data," 2024.

[5] R. Birkl, D. Wofk, and M. Müller, "Midas v3.1 – a model zoo for robust monocular relative depth estimation," 2023.

[6] V. Guizilini, I. Vasiljevic, D. Chen, R. Ambruș, and A. Gaidon, "Towards zero-shot scale-aware monocular depth estimation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 9233–9243, October 2023.

[7] M. Job, T. Stastny, T. Kazik, R. Siegwart, and M. Pantic, "Radar meets vision: Robustifying monocular metric depth prediction for mobile robotics," 2024.